

Continuous Compliance Home

Continuous Compliance

Exported on 08/20/2024

Table of Contents

1	Welcome to the Delphix Continuous Compliance documentation!	41
2	Quick references	45
3	Release notes	46
3.1	New features	46
3.1.1	Release 26.0.0.0.....	46
3.1.2	Release 25.0.0.0.....	46
3.1.3	Release 24.0.0.0.....	47
3.1.4	Release 23.0.0.0.....	47
3.1.5	Release 22.0.0.0.....	47
3.1.6	Release 21.0.0.0.....	48
3.1.7	Release 20.0.0.0.....	49
3.1.8	Release 19.0.0.0.....	49
3.1.9	Release 18.0.0.0.....	49
3.1.10	Release 17.0.0.0.....	50
3.1.11	Release 16.0.0.0.....	50
3.1.12	Release 15.0.0.0.....	51
3.1.13	Release 14.0.0.0.....	51
3.1.14	Release 13.0.0.0.....	53
3.1.15	Release 12.0.0.0.....	53
3.1.16	Release 11.0.0.0.....	53
3.1.17	Release 10.0.0.0.....	54
3.1.18	Release 9.0.0.0.....	54
3.1.19	Release 8.0.0.0.....	55
3.1.20	Release 7.0.0.0.....	55
3.1.21	Release 6.0.17.....	55
3.1.22	Release 6.0.16.0.....	56
3.1.23	Release 6.0.15.0.....	56

3.1.24	Release 6.0.14.0.....	56
3.1.25	Release 6.0.13.0.....	57
3.1.26	Release 6.0.12.0.....	58
3.1.27	Release 6.0.11.0.....	59
3.1.28	Release 6.0.10.0.....	61
3.1.29	Release 6.0.9.0.....	62
3.1.30	Release 6.0.8.0.....	62
3.1.31	Release 6.0.7.0.....	63
3.1.32	Release 6.0.6.0.....	65
3.1.33	Release 6.0.5.0.....	65
3.1.34	Release 6.0.4.0.....	66
3.1.35	Release 6.0.3.0.....	67
3.1.36	Release 6.0.2.0.....	68
3.1.37	Release 6.0.1.0.....	69
3.1.38	Release 6.0.0.0.....	70
3.1.39	Release 5.3.....	72
3.2	Fixed issues.....	76
3.2.1	Release 26.0.0.0.....	76
3.2.2	Release 25.0.0.0.....	77
3.2.3	Release 24.0.0.0.....	77
3.2.4	Release 23.0.0.0.....	78
3.2.5	Release 22.0.0.1.....	79
3.2.6	Release 22.0.0.0.....	79
3.2.7	Release 21.0.0.0.....	80
3.2.8	Release 20.0.0.0.....	81
3.2.9	Release 19.0.0.0.....	82
3.2.10	Release 18.0.0.0.....	82
3.2.11	Release 17.0.0.0.....	83
3.2.12	Release 16.0.0.0.....	84

3.2.13	Release 15.0.0.0.....	84
3.2.14	Release 14.0.0.0.....	85
3.2.15	Release 13.0.0.0.....	86
3.2.16	Security Fixes	87
3.2.17	Release 12.0.0.0.....	87
3.2.18	Security fixes.....	88
3.2.19	Release 11.0.0.0.....	88
3.2.20	Security fixes.....	89
3.2.21	Release 10.0.0.0.....	89
3.2.22	Release 9.0.0.0.....	90
3.2.23	Release 8.0.0.0.....	91
3.2.24	Release 7.0.0.0.....	91
3.2.25	Release 6.0.17.0.....	93
3.2.26	Release 6.0.16.0.....	95
3.2.27	Release 6.0.15.0.....	97
3.2.28	Release 6.0.14.0.....	98
3.2.29	Release 6.0.13.0.....	100
3.2.30	Release 6.0.12.0.....	101
3.2.30.1	Log4j updates.....	101
3.2.30.2	Fixed Issues	102
3.2.31	Release 6.0.11.0.....	105
3.2.32	Release 6.0.10.0.....	107
3.2.33	Release 6.0.9.0.....	109
3.2.34	Release 6.0.8.0.....	111
3.2.35	Release 6.0.7.0.....	112
3.2.36	Release 6.0.6.0.....	115
3.2.37	Release 6.0.5.0.....	116
3.2.38	Release 6.0.4.0.....	117
3.2.39	Release 6.0.3.0.....	119

3.2.40	Release 6.0.2.0.....	122
3.2.41	Release 6.0.1.0.....	123
3.2.42	Release 6.0.0.0.....	125
3.3	Known issues	127
3.3.1	Release 26.0.0.0.....	127
3.3.2	Release 25.0.0.0.....	128
3.3.3	Release 24.0.0.0.....	128
3.3.4	Release 23.0.0.0.....	128
3.3.5	Release 22.0.0.0.....	128
3.3.6	Release 21.0.0.0.....	129
3.3.7	Release 20.0.0.0.....	129
3.3.8	Release 19.0.0.0.....	130
3.3.9	Release 18.0.0.0.....	130
3.3.10	Release 17.0.0.0.....	131
3.3.11	Release 16.0.0.0.....	132
3.3.12	Release 15.0.0.0.....	134
3.3.13	Release 14.0.0.0.....	135
3.3.14	Release 13.0.0.0.....	135
3.3.15	Release 12.0.0.0.....	136
3.3.16	Release 11.0.0.0.....	136
3.3.17	Release 10.0.0.0.....	137
3.3.18	Release 9.0.0.0.....	137
3.3.19	Release 8.0.0.0.....	138
3.3.20	Release 7.0.0.0.....	138
3.3.21	Release 6.0.17.0.....	138
3.3.22	Release 6.0.16.0.....	139
3.3.23	Release 6.0.15.0.....	139
3.3.24	Release 6.0.14.0.....	139
3.3.25	Release 6.0.13.0.....	139

3.3.26	Release 6.0.12.0.....	140
3.3.27	Release 6.0.11.0.....	140
3.3.28	Release 6.0.10.0.....	141
3.3.29	Release 6.0.9.0.....	141
3.3.30	Release 6.0.8.0.....	141
3.3.31	Release 6.0.7.0.....	141
3.3.32	Release 6.0.6.0.....	141
3.3.33	Release 6.0.5.0.....	141
3.3.34	Release 6.0.4.0.....	141
3.3.35	Release 6.0.3.0.....	141
3.3.36	Release 6.0.2.0.....	142
3.3.37	Release 6.0.1.0.....	142
3.3.38	Release 6.0.0.0.....	142
3.4	Deprecated and end-of-life features	143
3.4.1	Release 26.0.0.0.....	143
3.4.1.1	End-of-life features	143
3.4.2	Release 24.0.0.0.....	143
3.4.2.1	Deprecated features	143
3.4.3	Release 20.0.0.0.....	144
3.4.3.1	End-of-life features	144
3.4.4	Release 15.0.0.0.....	144
3.4.4.1	Deprecated features	144
3.4.5	Release 11.0.0.0.....	144
3.4.5.1	Deprecated features	144
3.4.5.2	End-of-Life features	144
3.4.6	Release 10.0.0.0.....	144
3.4.6.1	Deprecated features	144
3.4.7	Release 6.0.17.0.....	145
3.4.7.1	End-of-life features	145

3.4.8	Release 6.0.15.0.....	145
3.4.8.1	End-of-life features	145
3.4.9	Release 6.0.12.0.....	145
3.4.9.1	End-of-life features	145
3.4.10	Release 6.0.11.0.....	145
3.4.10.1	Deprecated features	145
3.4.10.2	End-of-life features	146
3.4.11	Release 6.0.10.0.....	146
3.4.11.1	End-of-life features	146
3.4.12	Release 6.0.9.0.....	146
3.4.12.1	Deprecated features	146
3.4.13	Release 6.0.8.0.....	147
3.4.13.1	End-of-life features	147
3.4.14	Release 6.0.7.0.....	147
3.4.14.1	End-of-life features	147
3.4.15	Release 6.0.4.0.....	147
3.4.15.1	Deprecated features	147
3.4.15.2	End-of-life features	147
3.4.16	Release 6.0.3.0.....	148
3.4.16.1	End-of-life features	148
3.4.17	Release 6.0.0.0.....	148
3.4.17.1	End-of-life features	148
3.5	Licenses and notices	148
4	Getting started	150
4.1	Introduction to Delphix Masking.....	150
4.1.1	Challenge.....	150
4.1.2	Solution.....	151
4.1.3	High-level platform architecture	152
4.1.4	How Delphix identifies sensitive data.....	152

4.1.5	How Delphix secures your sensitive data.....	153
4.2	Data source support	154
4.2.1	Standard connectors	154
4.2.2	Select connectors	154
4.2.3	Db2 LUW connector	154
4.2.3.1	Introduction	154
4.2.3.2	Support matrix.....	155
4.2.3.3	TLS/SSL setup	156
4.2.4	Oracle connector.....	156
4.2.4.1	Introduction	156
4.2.4.2	Support matrix.....	156
4.2.4.3	TLS/SSL setup	157
4.2.5	Microsoft SQL Server connector.....	158
4.2.5.1	Introduction	158
4.2.5.2	Support matrix.....	158
4.2.5.3	TLS/SSL setup	159
4.2.5.4	Google Cloud SQL IAM Authorization setup	160
4.2.6	PostgreSQL connector	160
4.2.6.1	Introduction	160
4.2.6.2	Support matrix.....	160
4.2.6.3	TLS/SSL setup	161
4.2.6.4	Google Cloud SQL IAM Authorization setup	162
4.2.7	Yugabyte connector.....	162
4.2.7.1	Introduction	162
4.2.7.2	Support matrix.....	163
4.2.7.3	TLS/SSL setup	164
4.2.8	CockroachDB connector.....	164
4.2.8.1	Introduction	164
4.2.8.2	Support matrix.....	165

4.2.8.3	TLS/SSL setup	166
4.2.9	MySQL / MariaDB connector.....	167
4.2.9.1	Introduction	167
4.2.9.2	MySQL support matrix.....	167
4.2.9.3	Google Cloud SQL IAM Authorization setup	168
4.2.9.4	MariaDB support matrix	168
4.2.9.5	TLS/SSL setup	170
4.2.10	SAP ASE (Sybase) connector.....	170
4.2.10.1	Introduction	170
4.2.10.2	Support matrix.....	170
4.2.10.3	TLS/SSL setup	171
4.2.11	Db2 z/OS and iSeries connectors	172
4.2.11.1	Introduction	172
4.2.11.2	Support matrix.....	172
4.2.11.3	TLS/SSL setup	173
4.2.12	Files connector.....	173
4.2.12.1	Introduction	173
4.2.12.2	Support matrix.....	173
4.2.13	Mainframe data set connector.....	174
4.2.13.1	Introduction	174
4.2.13.2	Support matrix.....	174
4.2.14	On-The-Fly masking jobs	174
4.3	Installation.....	175
4.3.1	Prerequisites	175
4.3.1.1	VM-based Continuous Compliance Engines.....	175
4.3.1.2	Container (Kubernetes) based Continuous Compliance Engine	176
4.3.1.3	Differences between VM-based and Container-based Engines.....	176
4.3.2	Network connectivity requirements.....	177

4.3.2.1	General outbound connections from the virtual machine Delphix Continuous Compliance Engine.....	178
4.3.2.2	General inbound connections to the virtual machine Delphix Continuous Compliance Engine	178
4.3.2.3	General outbound connections from the containerized Delphix Continuous Compliance Engine.....	179
4.3.2.4	General inbound connections to the containerized Delphix Continuous Compliance Engine	179
4.3.2.5	Firewalls and Intrusion Detection Systems (IDS)	180
4.3.3	First time setup	180
4.3.3.1	Setting up network access to the Delphix Engine.....	180
4.3.3.2	Setting up the Delphix Engine	181
4.3.3.3	Logging in to the Delphix Continuous Compliance Engine	182
4.3.4	AWS EC2 installation	183
4.3.4.1	Instance types	183
4.3.4.2	Network configuration.....	184
4.3.4.3	Storage configurations	184
4.3.4.4	Additional AWS configuration notes.....	186
4.3.4.5	Installing AMI on AWS EC2	186
4.3.5	Azure installation	187
4.3.5.1	Instance types	187
4.3.5.2	Network configuration.....	188
4.3.5.3	Storage configuration	189
4.3.5.4	Extensions	189
4.3.5.5	Installing VHD on AZURE.....	189
4.3.6	Google Cloud Platform installation.....	190
4.3.6.1	Machine types	190
4.3.6.2	Network configuration.....	190
4.3.6.3	Storage configuration	191
4.3.6.4	Additional GCP configuration notes	191

4.3.6.5	Installing on Google Cloud Platform.....	191
4.3.7	Hyper-V installation.....	192
4.3.7.1	Supported versions.....	192
4.3.7.2	Virtual CPUs	192
4.3.7.3	Memory.....	193
4.3.7.4	Network	193
4.3.7.5	SCSI Controller.....	194
4.3.7.6	Storage configuration.....	194
4.3.7.7	Installing Hyper-V.....	196
4.3.8	IBM Cloud Platform installation.....	197
4.3.8.1	Supported profiles	197
4.3.8.2	Network configuration.....	198
4.3.8.3	Storage configuration.....	198
4.3.8.4	Additional IBM configuration notes.....	199
4.3.8.5	Procedure for deploying in the IBM Cloud.....	199
4.3.9	KVM installation.....	204
4.3.9.1	Overview	204
4.3.9.2	Pre-requisites	205
4.3.9.3	Virtual CPUs	205
4.3.9.4	Memory.....	206
4.3.9.5	Network	206
4.3.9.6	SCSI controller	207
4.3.9.7	General storage.....	207
4.3.10	OCI installation.....	208
4.3.10.1	Supported databases	208
4.3.10.2	Compute image types.....	208
4.3.10.3	Supported shapes.....	208
4.3.10.4	Network configuration.....	209
4.3.10.5	Storage configuration.....	209

4.3.10.6	Additional OCI configuration notes.....	210
4.3.10.7	Installing OCI	210
4.3.11	VMware installation	214
4.3.11.1	Supported ESX versions	215
4.3.11.2	Virtual CPUs	215
4.3.11.3	Memory.....	216
4.3.11.4	Network	216
4.3.11.5	Storage	217
4.3.11.6	Additional VMware configuration notes.....	218
4.3.11.7	Installing OVA on VMware.....	218
4.3.12	Containerized masking installation	219
4.3.12.1	Obtaining the images.....	220
4.3.12.2	Setup.....	220
4.3.12.3	Debugging	224
4.3.12.4	Managing LDAP over TLS/SSL for containerized masking.....	225
4.4	Naming requirements	228
4.4.1	Affected configurable objects.....	228
4.4.2	Upgrade	230
4.4.3	Maximum name lengths.....	231
4.4.4	Create/rename	232
4.4.5	Environment export/import	232
4.4.6	Sync	232
4.5	Graphical user interface	232
4.5.1	Grid UI	232
4.5.1.1	Sorting	232
4.5.1.2	Filtering.....	234
4.5.1.3	Pinning columns	238
4.5.1.4	Auto-sizing columns	239
4.5.1.5	View more columns	239

4.5.1.6	Actions.....	240
4.5.1.7	Copy cell data.....	240
4.5.2	Select/Deselect all the records.....	241
4.5.3	List Editor UI.....	242
4.6	Managing application settings.....	246
4.6.1	The Application Settings screen.....	246
4.6.2	Modify Application Settings values.....	247
4.7	Users and roles.....	248
4.7.1	What are roles?.....	248
4.7.1.1	Role Type.....	249
4.7.1.2	Actions.....	249
4.7.1.3	Objects.....	250
4.7.1.4	Adding a role.....	251
4.7.1.5	Recommended roles.....	253
4.7.1.6	Modifying Roles.....	254
4.7.2	What are users?.....	257
4.7.2.1	Adding a user.....	258
4.7.2.2	Modifying User.....	260
4.7.2.3	View User.....	261
4.7.2.4	Edit User.....	262
4.7.2.5	Delete User.....	264
4.7.3	Sample JSON.....	265
4.7.3.1	All privileges.....	266
4.7.3.2	Create new user.....	269
4.7.3.3	User update.....	270
4.8	Best practices for defining masking roles.....	270
4.8.1	Introduction.....	270
4.8.2	Sample RACI.....	271
4.8.3	Sample roles for Masking.....	272

4.9	Audit logs.....	273
4.9.1	Audit logs UI page.....	273
4.9.2	Audit log APIs.....	274
4.9.3	What gets logged?	274
4.9.4	Retention policy.....	275
4.9.5	Recommendation.....	275
4.10	Monitor Async Task Status	275
4.10.1	Async task status.....	276
4.11	Kerberos configuration.....	277
4.11.1	Introduction	277
4.11.2	Terminology	277
4.11.3	Configuring Kerberos on the appliance.....	277
4.11.4	Creating masking database connectors using Kerberos	279
4.11.5	Reference database configurations	281
4.11.6	Oracle and Kerberos	281
4.11.6.1	Overview	281
4.11.6.2	Prerequisites	281
4.11.6.3	Creating the Oracle service principal.....	282
4.11.6.4	Configuring the Oracle database for Kerberos	282
4.11.6.5	Creating a database User Identified via Kerberos	283
4.11.6.6	Troubleshooting.....	284
4.11.7	MSSQL Server and Kerberos	284
4.11.7.1	Overview	284
4.11.7.2	Prerequisites	285
4.11.7.3	Creating SPNs for the Database Service.....	285
4.11.7.4	Creating the Database Connector on the Continuous Compliance Engine.....	285
4.11.7.5	Creating a keytab file for an Active Directory user	286
4.11.7.6	Troubleshooting tips.....	286

4.11.7.7	The SPN for the SQL Server is registered to the incorrect Active Directory account.....	287
4.11.8	Sybase and Kerberos.....	288
4.12	Password vault configuration.....	293
4.12.1	Introduction.....	293
4.12.2	Configuring a password vault on the appliance.....	293
4.12.2.1	Setting up a password vault.....	293
4.12.2.2	Setting up a credential path.....	296
4.12.3	Configuring the database connector.....	297
4.12.4	UI configuration.....	298
4.12.5	API configuration.....	298
4.13	Db2 connector license installation.....	298
4.13.1	Applying Db2 connector for mainframe.....	299
4.13.2	Applying Db2 connector for iSeries.....	299
4.14	Continuous Compliance Engine icon reference.....	300
4.15	Delphix masking terminology.....	303
4.15.1	Products.....	303
4.15.2	Interfaces.....	304
4.15.3	Core concepts.....	304
4.15.4	Data Source Connectors.....	305
4.15.5	Continuous Compliance.....	306
4.15.6	Masking algorithms.....	307
4.15.7	Profile job concepts.....	309
4.15.8	Masking job concepts.....	310
4.16	Changing the IP address of the Delphix Engine.....	312
4.16.1	Pre-requisites.....	312
4.16.2	Changing the IP address from the user interface.....	312
4.16.3	Changing the IP address using CLI.....	313
4.17	Stopping and starting the containerized Continuous Compliance Engine.....	314

4.17.1	Overview	314
4.17.2	Starting the containerized Masking Engine.....	315
4.17.3	Stopping the containerized Masking Engine.....	315
4.17.4	Removing persistent volumes / persistent volume claims.....	316
4.18	Stopping, starting, and restarting the continuous compliance engine...	317
4.18.1	Overview	317
4.18.2	Use cases examples	317
4.18.3	Troubleshooting before a restart	317
4.18.4	Using the Command-Line Interface (CLI).....	318
4.18.4.1	Restarting the Masking Engine example	318
4.19	Upgrading the Continuous Compliance Engine	319
4.19.1	Upgrades for virtual Compliance Engines	319
4.19.2	Upgrades for containerized Compliance Engines	319
4.20	Utilization.....	320
4.20.1	Overview	320
4.20.2	Utilization UI page	320
4.20.3	The Jobs Utilization Report	320
4.20.4	The Database Size Report	321
4.20.4.1	Support matrix.....	321
5	Preparing data.....	322
5.1	Database user permissions for executing masking and profiling job....	322
5.1.1	Introduction.....	322
5.1.2	List of database entitlements required to run masking jobs	322
5.1.3	List of database entitlements required to run profiling jobs.....	323
5.2	Preparing Oracle database for profiling/masking	323
5.2.1	Overview	323
5.2.2	Archive logging	323
5.2.3	DB/VDB memory allocation	323
5.2.4	Undo tablespace size and undo retention time:	324

5.2.5	Redo logs are optimally sized	324
5.2.6	Change PCTFREE to 40-50:	325
5.2.7	Change primary Key To ROWID:.....	325
5.2.8	Masking user privileges:.....	325
5.3	Preparing SQL server database for profiling and masking	326
5.3.1	Logging.....	326
5.3.2	DB/VDB memory allocation	326
5.3.3	Primary/Foreign/DMS_ROW_ID Keys	327
5.3.4	Creating a masking user and privileges	327
5.4	Preparing Sybase database for profiling and masking	328
5.4.1	Determining the amount of memory SAP ASE needs	328
5.4.2	Determining SAP ASE memory configuration.....	328
5.4.3	In-place masking and table keys.....	328
5.4.4	Transaction logs	329
5.4.4.1	On-the-Fly.....	329
5.4.4.2	Sizing Transaction Logs.....	329
5.4.5	Creating a masking user and privileges	329
6	Connecting data.....	331
6.1	Managing environments.....	331
6.1.1	Adding an application.....	332
6.1.2	Creating an environment	333
6.1.3	Exporting settings	334
6.1.4	Importing settings.....	334
6.1.5	Exporting an environment	335
6.1.6	Importing an environment.....	336
6.1.7	Editing an environment.....	338
6.1.8	Copying an environment.....	339
6.1.9	Deleting an environment.....	340
6.2	Managing remote mounts for VM continuous compliance engines	340

6.2.1	Mount filesystem API	341
6.2.1.1	Mount information	341
6.2.2	Mount options	341
6.2.2.1	Enforced options.....	341
6.2.2.2	Minimal options	342
6.2.2.3	Version options	342
6.2.2.4	Generic options	342
6.2.3	CRUD operations.....	343
6.2.3.1	Create	343
6.2.3.2	Read.....	343
6.2.3.3	Update	343
6.2.3.4	Delete.....	343
6.2.4	Mount operations.....	343
6.2.4.1	Connect	343
6.2.4.2	Disconnect.....	343
6.2.4.3	Remount	344
6.2.4.4	Resolve mount consistency	344
6.2.5	Using mounts	344
6.2.5.1	File connector.....	344
6.2.5.2	Sync mounts.....	347
6.2.6	Recommended mount server configuration	347
6.2.6.1	CIFS server	347
6.2.6.2	NFS server	347
6.3	Managing remote mounts for containerized masking	348
6.3.1	Creating the mountpoint connection in Kubernetes.....	348
6.3.1.1	NFS Persistent Volume YAML	348
6.3.1.2	NFS persistent volume claim YAML	349
6.3.2	Using the mountpoint in the pod configuration	349
6.3.2.1	Excerpt of kubernetes-config.yaml to show support for NFS volumes..	349

6.3.3	Using the mountpoint in the UI.....	350
6.3.4	Other types of filesystem mountpoint.....	351
6.3.5	Known limitations	351
6.3.6	Local file masking troubleshooting	351
6.4	Managing SSL/TLS over JDBC for containerized masking.....	352
6.4.1	Prerequisites	352
6.4.2	Create configmap entry based on database provided SSL/TLS certificate.....	352
6.4.3	Mount the configured configmap as volume.....	353
6.4.4	Create trust store and upload all mounted SSL/TLS certificates.....	353
6.4.5	Configure SSL/TLS over JDBC connector.....	354
6.4.6	SSL/TLS over JDBC troubleshooting.....	354
6.5	Managing connectors.....	355
6.5.1	Database connectors.....	356
6.5.2	Database connector properties	357
6.5.2.1	Getting properties	357
6.5.2.2	Setting properties.....	358
6.5.2.3	Security considerations.....	360
6.5.3	File connectors.....	360
6.5.4	Create, test, edit, and delete a connector	364
6.5.4.1	Create a connector	364
6.5.4.2	Editing a connector.....	368
6.5.4.3	Testing a connector.....	371
6.5.4.4	Deleting a connector.....	373
6.5.5	Authentication types.....	375
6.5.5.1	Username/Password.....	375
6.5.5.2	Kerberos authentication	375
6.5.5.3	Password Vault.....	376
6.5.6	Mainframe MVS storage access	376

6.5.6.1	Valid connection with Mainframe	377
6.5.6.2	Invalid connection with Mainframe	378
6.5.6.3	FTPS Connector for Mainframe storage	379
6.5.6.4	Invalid/no certificate.....	381
6.5.7	Amazon Simple Storage Service (S3) connector for files.....	382
6.5.7.1	Configuring an AWS S3 Connector	383
6.5.7.2	Configuring Other S3 Compatible storage Connector.....	385
6.5.7.3	S3 upload sizing.....	388
6.6	Managing extended connectors	389
6.6.1	Limitations.....	389
6.6.2	Installing a new driver.....	389
6.6.3	Driver permissions	392
6.6.4	Extended logging	393
6.6.5	Creating an extended connector.....	393
6.6.6	Synchronization	397
6.6.7	License entitlement for commercial JDBC drivers	397
6.6.7.1	Installing a license	398
6.6.7.2	Managing licenses.....	399
6.7	Managing rule sets	400
6.7.1	The rule set screen	400
6.7.2	Creating a new rule set.....	401
6.7.3	Editing/Modifying a rule set.....	402
6.7.4	The Add/Edit rule set wizard.....	403
6.7.4.1	Step 1: Details	403
6.7.4.2	Step 2: Data Tables / Files	404
6.7.4.3	Step 3: Summary.....	414
6.7.5	Removing a table or file from the ruleset.....	414
6.7.6	Removing or Modifying All records in a rule set	415
6.7.7	Modifying table properties in a rule set.....	416

6.7.7.1	Logical key.....	416
6.7.7.2	Edit filter	417
6.7.7.3	Custom SQL	418
6.7.8	Control character support for delimited files.....	420
6.7.8.1	Control character as a delimiter.....	420
6.7.8.2	Control character as an end of record.....	421
6.7.8.3	Control character as a value	422
6.7.9	Define enclosure escaping strategy for delimited files.....	423
6.7.9.1	Double enclosure	423
6.7.9.2	Custom	424
6.7.9.3	Escape "enclosure escape character"	425
6.7.9.4	Configure enclosure escape character for a large rule set.....	425
6.7.10	Refreshing a rule set.....	425
6.7.11	Copying a rule set	426
6.7.12	Deleting a rule set	426
6.8	Managing file formats	427
6.8.1	Assigning file formats to files.....	428
6.8.2	Record types	429
6.8.3	Redefine conditions	430
6.8.3.1	Entering a Redefine condition	431
6.8.3.2	Format for redefine conditions	432
6.8.4	Constructing file formats for upload	433
6.8.4.1	Delimited files	433
6.8.4.2	Fixed-width files	434
6.8.4.3	XML files.....	435
6.8.4.4	JSON files.....	435
6.8.4.5	Mainframe files	435
6.8.5	Import, edit, and delete formats.....	436
6.8.5.1	Import file formats	436

6.8.5.2	Import Mainframe formats.....	438
6.8.5.3	Edit a file format.....	440
6.8.5.4	Delete a file format	441
6.8.6	Add, view, edit, and delete fields for file formats.....	442
6.8.6.1	Add file fields	442
6.8.6.2	View, edit, or delete a file field.....	445
6.8.7	File structure for masking	446
6.8.7.1	XML structure.....	446
6.8.7.2	JSON structure.....	448
6.9	Managing inventories	451
6.9.1	The inventory screen	451
6.9.2	Assigning algorithms.....	453
6.9.3	Managing database inventory settings	457
6.9.4	Managing a fixed-width or delimited file inventory settings	460
6.9.5	Managing a JSON file inventory settings	461
6.9.6	Managing an XML file inventory settings.....	462
6.9.7	Managing Mainframe inventory settings	463
6.9.8	Importing and exporting an inventory.....	464
6.9.8.1	To export an inventory	465
6.9.8.2	To import an inventory.....	465
6.9.9	Document Store Type masking.....	466
6.9.9.1	Multi-column algorithm support for document store type masking.....	469
6.9.10	Inventory Approval Workflow (database rule sets only).....	472
6.9.10.1	Enabling Inventory Approval Workflow for an environment.....	472
6.9.10.2	Workflow stages	473
6.10	Managing record types and header/footer records	475
6.10.1	Records types.....	475
6.10.2	Header and footer records	476
6.10.2.1	Adding record types.....	477

6.10.2.2	Editing record types	478
6.10.2.3	Deleting record types	479
6.10.2.4	Managing qualifiers	479
6.10.2.5	Configure header and footer	482
6.11	Managing jobs.....	482
6.11.1	Jobs list	482
6.11.1.1	Jobs - Masking Environment.....	482
6.11.1.2	Jobs - Tokenize Environment.....	483
6.11.2	Export Environment	484
6.11.3	Running and Stopping Jobs	485
6.11.3.1	Submitting a job	485
6.11.3.2	Stopping a Job	488
6.11.3.3	Enabling and disabling database constraints in Jobs.....	490
6.11.3.4	Creating SQL statements to run before and after jobs.....	490
6.11.4	Create, view, edit, and delete jobs.....	491
6.11.4.1	Create Masking Job.....	491
6.11.4.2	Create Tokenization Job	501
6.11.4.3	Create Re-Identification Job.....	501
6.11.4.4	Edit job	502
6.11.4.5	View job	503
6.11.4.6	Delete job.....	504
6.11.5	Managing profiling jobs.....	505
6.11.5.1	Create Profile Job	506
6.11.5.2	Edit Profile Job.....	509
6.11.5.3	View Profile Job	511
6.11.5.4	Delete Profile Job.....	512
6.11.6	Job monitoring.....	513
6.11.6.1	Monitoring jobs	513
6.11.6.2	Monitoring a single job	515

6.11.6.3	Displaying non-conformant data.....	519
6.11.6.4	Interpreting samples of non-conformant data patterns.....	520
6.11.6.5	Tracking Non-conformant Data	521
6.12	Whole file masking	522
6.12.1	Pre-requisites	522
6.12.2	Masking a whole file	523
7	Identifying sensitive data	526
7.1	Discovering your sensitive data	526
7.1.1	Overview	526
7.1.2	Concepts	527
7.1.2.1	Profile set	527
7.1.2.2	Domain	527
7.1.2.3	Level - column or data	527
7.1.2.4	Classifier.....	527
7.1.2.5	Search expression	528
7.1.2.6	Type expression.....	528
7.2	Out of the box profiling settings	528
7.2.1	ASDD standard profile set.....	528
7.2.2	Standard profile set	528
7.2.3	Legacy profile sets.....	529
7.3	ASDD standard profile set.....	529
7.4	Standard profile set expressions	586
7.4.1	Column level expressions	586
7.4.2	Type expressions	589
7.5	Legacy profile set expressions	592
7.5.1	Account numbers.....	592
7.5.2	Physical addresses.....	593
7.5.3	Beneficiary ID	594
7.5.4	Biometrics	594

7.5.5	Certificate ID.....	594
7.5.6	City.....	595
7.5.7	Country.....	595
7.5.8	Credit card.....	595
7.5.9	Customer number.....	596
7.5.10	Date of birth.....	596
7.5.11	Driver license number.....	597
7.5.12	Email.....	597
7.5.13	First name.....	598
7.5.14	IP address.....	598
7.5.15	Last name.....	599
7.5.16	Plate number.....	599
7.5.17	PO Box numbers.....	599
7.5.18	Precinct.....	599
7.5.19	Record number.....	600
7.5.20	School name.....	600
7.5.21	Security code.....	600
7.5.22	Serial number.....	600
7.5.23	Signature.....	601
7.5.24	Social security number.....	601
7.5.25	Tax ID.....	601
7.5.26	Telephone number.....	602
7.5.27	Vin number.....	602
7.5.28	Web address.....	602
7.5.29	ZIP code.....	603
7.6	Configuring profile sets.....	603
7.6.1	Creating Profile Sets.....	604
7.6.1.1	Adding a Profiler set for the legacy profiler.....	604
7.6.1.2	Adding a Profiler set for the ASDD profiler.....	606

7.6.2	Modifying Profile Set	607
7.6.2.1	View Profile Set.....	608
7.6.2.2	Edit Profile Set.....	609
7.6.2.3	Duplicate Profile Set	609
7.6.2.4	Delete Profile Set	610
7.7	Managing domains	611
7.7.1	Overview	611
7.7.2	Domains	611
7.7.3	Adding a new domain	612
7.7.4	Modifying Domains.....	614
7.7.4.1	View Domain	615
7.7.4.2	Edit Domain	615
7.7.4.3	Delete Domain.....	616
7.8	Managing classifiers	616
7.8.1	Adding a new Classifier.....	617
7.8.2	Modifying Classifiers	623
7.8.2.1	View Classifier.....	624
7.8.2.2	Edit Classifier	625
7.8.2.3	Delete Classifier	626
7.8.3	Configuration considerations for classifiers.....	627
7.8.3.1	Strength values	627
7.8.3.2	Regex configuration.....	629
7.8.3.3	Type classifiers	630
7.8.3.4	Tokenization in list classifiers.....	630
7.9	Managing expressions	630
7.9.1	Profile expressions	631
7.9.2	Managing search expressions	632
7.9.2.1	Adding new Search Expression.....	633
7.9.2.2	Modifying Search Expressions.....	635

7.9.3	Managing type expressions	639
7.9.3.1	Adding new Type Expression	640
7.9.3.2	Modifying Type Expressions	641
7.10	ASDD profile set import and export	646
7.10.1	ASDD profile set import	646
7.10.1.1	Limitations	647
7.10.2	ASDD profile set export	647
7.11	Reporting profiling results	647
7.11.1	Job Execution page	647
7.11.2	PDF report	648
7.11.3	Rule set page	649
7.11.3.1	Database Rule Set	649
7.11.3.2	File Rule Set	650
7.11.3.3	Mainframe Rule Set	650
7.11.4	CSV	651
7.11.5	API endpoint	652
7.12	ASDD features and support	653
7.12.1	ASDD features	653
7.12.2	ASDD limitations	654
8	Securing sensitive data	656
8.1	Algorithms	656
8.1.1	Introduction to Masking Algorithms	656
8.1.2	Algorithm options	656
8.1.2.1	Out-of-the-box algorithm instances	656
8.1.2.2	Algorithm frameworks	660
8.1.3	Configuring your algorithms	662
8.1.3.1	Algorithm settings	662
8.1.3.2	Creating new algorithms	663
8.1.3.3	Editing algorithms	666

8.1.3.4	Viewing algorithms	667
8.1.3.5	Deleting algorithms.....	668
8.1.4	Algorithms Keys	669
8.1.5	Multi-column algorithms	669
8.1.5.1	Overview	669
8.1.5.2	Usage	670
8.1.6	Limitations.....	670
8.1.7	Algorithm frameworks overview	670
8.1.7.1	Choosing an algorithm framework	670
8.1.7.2	Choosing between character and segment mapping frameworks	671
8.1.8	Out of the box algorithm instances	671
8.1.8.1	dlpx-core: CM Alpha-Numeric	672
8.1.8.2	dlpx-core: CM Digits.....	673
8.1.8.3	dlpx-core: CM Numeric	673
8.1.8.4	Credit Card.....	674
8.1.8.5	Date Shift Discrete	674
8.1.8.6	Date Shift Fixed	675
8.1.8.7	Date Shift Variable	675
8.1.8.8	dlpx-core: Email SL	676
8.1.8.9	dlpx-core: Email Unique.....	676
8.1.8.10	dlpx-core: FirstName	677
8.1.8.11	dlpx-core: FullName	678
8.1.8.12	dlpx-core: LastName.....	680
8.1.8.13	dlpx-core:Lat_Long Coordinates	681
8.1.8.14	NullValueLookup.....	683
8.1.8.15	dlpx-core: Phone Unique.....	684
8.1.8.16	dlpx-core: Phone US.....	684
8.1.8.17	dlpx-core:Redact Digits-Zero	685
8.1.8.18	RepeatFirstDigit	685

8.1.8.19	Secure Lookup (Out of the box algorithm instances).....	686
8.1.8.20	dlpx-core:TimeRange.....	702
8.1.8.21	dlpx-core: IBAN	702
8.1.8.22	SecureShuffle.....	704
8.1.9	Algorithm frameworks.....	704
8.1.9.1	Binary Lookup	705
8.1.9.2	Character Mapping (Algorithm frameworks).....	707
8.1.9.3	Character Replacement (Algorithm frameworks).....	712
8.1.9.4	Data Cleansing (Algorithm frameworks).....	714
8.1.9.5	Date Replacement (Algorithm frameworks)	717
8.1.9.6	Date Shift (Algorithm frameworks).....	720
8.1.9.7	Dependent Date Shift (Algorithm frameworks).....	722
8.1.9.8	Email (Algorithm frameworks).....	726
8.1.9.9	Free Text Redaction (Algorithm frameworks).....	730
8.1.9.10	Full Name (Algorithm frameworks)	735
8.1.9.11	Mapping (Algorithm frameworks)	740
8.1.9.12	Min Max Number (Algorithm frameworks)	754
8.1.9.13	Min Max Date (Algorithm frameworks)	757
8.1.9.14	Multi Column Address (Algorithm frameworks).....	759
8.1.9.15	Multi Column Condition (Algorithm frameworks).....	767
8.1.9.16	Name (Algorithm frameworks)	773
8.1.9.17	Numeric Expression (Algorithm frameworks)	777
8.1.9.18	Payment Card (Algorithm frameworks)	786
8.1.9.19	Regex Decompose (Algorithm frameworks).....	788
8.1.9.20	Secure Lookup (Algorithm frameworks)	793
8.1.9.21	Segment Mapping (Algorithm frameworks)	797
8.1.9.22	String Algorithm Chain (Algorithm Frameworks)	804
8.1.9.23	Tokenization (Algorithm frameworks)	806
8.1.10	General UI for extended algorithms.....	813

8.1.10.1	Overview	813
8.1.10.2	GUI steps	813
8.2	Builtin Driver Supports	820
8.2.1	Introduction	820
8.2.2	Oracle.....	821
8.2.3	MSSQL	821
8.2.4	PostgreSQL	821
8.2.5	Db2 LUW	821
8.2.6	Db2 z/OS	821
8.2.7	Db2 iSeries	822
8.2.8	Built-in Oracle driver support plugin	822
8.2.8.1	Optimizations	822
8.2.8.2	Task execution order	822
8.2.8.3	Enabling tasks on a job.....	823
8.2.8.4	Known limitations	823
8.2.9	Built-in MSSQL driver support plugin.....	823
8.2.9.1	Handling Primary Keys (PK).....	824
8.2.9.2	Handling Foreign Keys (FK).....	824
8.2.9.3	Handling Unique Constraints	824
8.2.9.4	Handling Unique Indexes.....	825
8.2.9.5	Handling Check Constraints (untested)	826
8.2.10	Built-in PostgreSQL driver support plugin	826
8.2.10.1	Tasks	826
8.2.10.2	Task Execution Order.....	827
8.2.10.3	Important Considerations	827
8.2.10.4	Known Limitations	827
8.2.11	Built-in DB2 LUW driver support plugin.....	828
8.2.11.1	Tasks	828
8.2.11.2	Task Execution Order.....	828

8.2.11.3	Important Considerations	829
8.2.11.4	Known Limitations	829
8.2.12	Built-in DB2 z/OS driver support plugin	829
8.2.12.1	Tasks	829
8.2.12.2	Task execution order	830
8.2.12.3	Important considerations	830
8.2.12.4	Known limitations	830
8.2.13	Built-in DB2 iSeries driver support plugin	831
8.2.13.1	Tasks	831
8.2.13.2	Task Execution Order.....	831
8.2.13.3	Important Considerations	832
8.2.13.4	Known Limitations	832
9	Masked provisioning.....	833
9.1	Configuring virtualization service for masked provisioning.....	833
9.1.1	Introduction	833
9.1.2	Instructions	833
9.2	Provision masked VDBs	834
9.2.1	Prerequisites	834
9.2.2	Restrictions	834
9.2.3	Identifying and navigating to masked VDBs	835
9.2.4	Provisioning masked VDBs	835
9.2.4.1	Associating a masking job with the dSource.....	835
9.2.4.2	Provisioning a masked VDB using the dSource provisioning wizard	837
9.2.5	Refresh a masked VDB.....	840
9.2.6	Disassociating a masking operation on a dSource	840
9.2.7	Masked VDB data operations	840
9.2.8	Continuous Data and Continuous Compliance Engine compatibility matrix.....	841
10	Managing multiple engines for masking	842

10.1	Introduction (Managing multiple engines for masking)	842
10.1.1	Software Development Life Cycle (SDLC)	842
10.1.2	Horizontal scale	842
10.1.3	Best practice guide and example architectures for synchronizing	843
10.1.3.1	SDLC	843
10.1.3.2	Horizontal Scale	844
10.2	Sync concepts	844
10.2.1	Syncable object	844
10.2.2	Object identifiers and types	845
10.2.3	Dependencies	846
10.2.3.1	Syncable Object dependencies relationship	847
10.2.4	Object revision tracking	848
10.2.5	Export document	849
10.2.6	Security	849
10.2.6.1	Access control	850
10.2.6.2	Transmission security	850
10.2.6.3	Storage security	850
10.2.7	Digital signature	850
10.2.8	Overwrite	850
10.2.8.1	Attempting to import identical objects	851
10.2.8.2	Overwrite of the encryption key	852
10.2.9	Error handling	852
10.2.10	Concurrent sync operations	853
10.2.11	Global objects	853
10.2.11.1	Global KEY	853
10.2.12	Reference objects	853
10.2.13	On-the-fly masking jobs	854
10.2.14	Circular dependencies	854
10.3	Sync endpoints	854

10.3.1	Export endpoints	854
10.3.1.1	GET /syncable-objects.....	854
10.3.1.2	POST /export.....	855
10.3.1.3	POST /export-async.....	856
10.3.2	Import endpoints.....	857
10.3.2.1	POST /import	857
10.3.2.2	POST /import-async	858
10.4	Algorithm syncability	859
10.4.1	Overview	859
10.4.2	Non-deterministic Algorithms	859
10.4.3	Fixed Algorithms	860
10.5	User workflow examples	860
10.5.1	Syncing all global objects.....	860
10.5.1.1	Source masking engine steps.....	860
10.5.1.2	Destination Masking Engine steps	864
10.5.2	Syncing a masking job.....	867
10.5.2.1	1. Export the job	867
10.5.2.2	2. Import the job.....	869
10.5.3	Syncing an environment	870
10.5.3.1	1. Export the environment	870
10.5.3.2	2. Create a new environment on the target engine.....	872
10.5.3.3	3. Import the environment into the newly created environment	873
10.6	Change log	875
10.6.1	Changes in 6.0.....	875
10.6.1.1	New syncable objects.....	875
10.6.2	Changes in 5.3.....	875
10.6.2.1	New syncable objects.....	875
10.6.2.2	Key per algorithm.....	876
10.6.2.3	Changed model of import status reporting.....	876

10.6.2.4	Changed granularity of transactions for import	877
10.6.2.5	Filter for /syncable-objects	877
10.6.2.6	Async endpoints	877
11	Delphix masking APIs	878
11.1	API client	878
11.1.1	Continuous Compliance API client	878
11.1.1.1	Introduction	878
11.1.1.2	Application settings APIs	882
11.1.2	Backwards compatibility API usage	890
11.1.2.1	API versioning context.....	890
11.1.2.2	Pinning down a version number to guarantee backwards- compatibility.....	891
11.1.2.3	Omitted version numbers	891
11.1.3	API response escaping.....	892
11.2	API workflows	892
11.2.1	API calls for masking administration	892
11.2.2	API calls for managing algorithms	893
11.2.2.1	Configuring algorithms.....	893
11.2.2.2	Managing algorithm usage	894
11.2.2.3	Migrating algorithms	898
11.2.2.4	Binary lookup.....	900
11.2.2.5	Character mapping	902
11.2.2.6	Data cleansing.....	904
11.2.2.7	Date replacement.....	905
11.2.2.8	Date shift	907
11.2.2.9	Dependent date shift	908
11.2.2.10	Email	910
11.2.2.11	Free text redaction	912
11.2.2.12	Full name	914

11.2.2.13 Mapping.....	916
11.2.2.14 Min Max.....	918
11.2.2.15 Name	921
11.2.2.16 Numeric expression.....	923
11.2.2.17 Payment card	925
11.2.2.18 Regex decompose	926
11.2.2.19 Secure lookup.....	929
11.2.2.20 Segment mapping.....	932
11.2.2.21 Tokenization.....	934
11.2.3 API calls for managing extended connectors.....	936
11.2.3.1 Introduction.....	936
11.2.3.2 Installing a driver support plugin.....	936
11.2.3.3 Installing a JDBC driver	939
11.2.3.4 Creating an extended database connector	940
11.2.3.5 Managing masking job driver support tasks.....	941
11.2.4 API calls for ASDD profile set import and export.....	941
11.2.4.1 ASDD profile set import.....	941
11.2.4.2 ASDD profile set export	944
11.2.5 API calls for managing classifiers	945
11.2.5.1 Retrieving classifier framework configurations.....	945
11.2.5.2 Example: Creating a new PATH classifier	946
11.2.5.3 Downloading files associated with classifiers.....	948
11.2.5.4 Searching and Filtering Classifiers	948
11.2.6 API calls for managing profile set usage	949
11.2.6.1 Overview	949
11.2.6.2 Viewing profile set usage	949
11.2.6.3 Examples	949
11.2.7 API calls for searching and filtering.....	952
11.2.7.1 Comparison operators.....	952

11.2.7.2	Search operator.....	953
11.2.7.3	Logical operators	953
11.2.7.4	Grouping	953
11.2.7.5	Literal Values.....	953
11.2.7.6	Limitations.....	954
11.2.7.7	Filtering usage examples	954
11.2.8	API calls for managing masking job driver support tasks	958
11.2.8.1	View the tasks implemented by driver support plugin	958
11.2.8.2	Create masking Job that enables tasks.....	960
11.2.8.3	Disable tasks.....	961
11.2.9	API calls for creating an inventory	963
11.2.9.1	Fetch table names from database connector.....	963
11.2.9.2	Create table metadata	964
11.2.9.3	Get All column metadata belonging to table metadata	964
11.2.9.4	Update column metadata with algorithm assignment.....	965
11.2.10	API calls for creating and running masking jobs	966
11.2.10.1	Creating a masking job.....	967
11.2.10.2	Running a masking job	968
11.2.10.3	Checking the status of a masking job	968
11.2.10.4	Retrieving execution events related to a masking job.....	969
11.2.10.5	Retrieving non-conformant data samples associated with an execution Event	970
11.2.11	API calls involving file upload and download.....	972
11.2.11.1	File download	972
11.2.11.2	File upload	973
11.2.11.3	Creating a file format.....	973
11.2.11.4	Creating an SSH Key.....	974
11.2.12	API call for generating support bundle	974
11.2.12.1	Generating a support bundle.....	975

11.2.12.2	Reading the async task result	975
11.2.12.3	Retrieving the generated support bundle file	976
11.3	API examples	978
11.3.1	loginCredentials	979
11.3.2	helpers	979
11.3.3	apiHostInfo.....	982
11.3.4	Configure enclosure escape character	983
11.3.5	createApplication.....	986
11.3.6	createEnvironment.....	986
11.3.7	createInventory	987
11.3.8	create DatabaseConnector	988
11.3.9	create DatabaseRuleset	989
11.3.10	getBillingUsage	989
11.3.11	getAuditLogs	990
11.3.12	getSyncableObjects	991
11.3.13	getSyncableObjectsExport	991
11.3.14	profileTypeExpressions.....	993
11.3.14.1	Add a new type expression	993
11.3.14.2	Delete a type expression	994
11.3.15	runMaskingJob	994
11.3.16	getDatabaseUsage	995
12	Authoring extensible plugins.....	996
12.1	Introduction (Authoring extensible plugins).....	996
12.1.1	Before getting started.....	996
12.1.2	SDK features	997
12.1.3	Versions Compatibility	997
12.2	General plugin structure	999
12.2.1	Introduction (General plugin structure)	999
12.2.2	Dependency management	1000

12.2.2.1	How to properly use and embed external libraries	1000
12.2.2.2	Example build file.....	1001
12.2.2.3	Using a shadow JAR to resolve dependency issues	1002
12.2.3	Plugin Metadata	1002
12.2.3.1	Manifest Attributes	1003
12.2.4	Versioning	1003
12.2.4.1	Table of Versioned Objects	1004
12.2.4.2	Ensuring Plugin Compatibility	1004
12.2.4.3	Plugin Naming.....	1005
12.3	Setting up your development environment	1005
12.3.1	Downloading and installing tools.....	1006
12.3.2	Creating a new project.....	1006
12.3.3	Enabling remote debugging	1006
12.4	Algorithms (Authoring extensible plugins).....	1007
12.4.1	SDK Features.....	1007
12.4.2	Getting more information	1007
12.4.3	The MaskingAlgorithm Java interface	1008
12.4.3.1	Core data types	1008
12.4.3.2	Special case values	1009
12.4.3.3	Method overview.....	1009
12.4.3.4	The life cycles of algorithm objects.....	1010
12.4.3.5	Multi-column masking	1011
12.4.3.6	Batch masking	1012
12.4.4	SDK Workflows (Algorithms)	1013
12.4.4.1	Outline for a guided tour.....	1013
12.4.4.2	Building the sample plugin (SDK workflows/Algorithms).....	1014
12.4.4.3	Creating a New Project (SDK workflows/Algorithms).....	1014
12.4.4.4	Service discovery (SDK workflows/Algorithms)	1017

12.4.4.5	Running an Algorithm using the SDK tools (SDK workflows/ Algorithms).....	1017
12.4.4.6	Installing multiple plugins onto the Delphix Masking engine (SDK workflows/Algorithms).....	1020
12.4.4.7	Retrieving information about installed plugins (SDK workflows/ Algorithms).....	1022
12.4.5	Configurability	1024
12.4.5.1	Introduction	1024
12.4.5.2	Making an algorithm configurable.....	1024
12.4.5.3	Using an Algorithm Framework	1028
12.4.5.4	Using Multi-Column Algorithms.....	1031
12.4.6	Service interfaces (Algorithms)	1036
12.4.6.1	Introduction	1036
12.4.6.2	Accessing Files	1037
12.4.6.3	Accessing Database Servers (JDBC)	1039
12.4.6.4	Algorithm chaining.....	1041
12.4.6.5	Using cryptographic keys	1044
12.4.6.6	Logging.....	1046
12.4.7	Security considerations.....	1048
12.4.7.1	Algorithm implementation.....	1048
12.5	Driver supports.....	1050
12.5.1	Introduction.....	1050
12.5.1.1	SDK features	1051
12.5.1.2	Getting more information	1051
12.5.2	The DriverSupport Java interface	1051
12.5.2.1	Method overview.....	1051
12.5.2.2	The life cycles of driver support objects	1051
12.5.3	SDK workflows (Driver supports).....	1052
12.5.3.1	Introduction.....	1052
12.5.3.2	Outline for a guided tour.....	1052

12.5.3.3	Building the sample plugin (SDK workflows/Driver supports).....	1053
12.5.3.4	Creating a new project (SDK workflows/Driver supports)	1053
12.5.3.5	Service discovery (SDK workflows/Driver supports).....	1056
12.5.3.6	Executing a driver support task using the SDK (SDK workflows/Driver supports)	1057
12.5.3.7	Retrieving information about installed plugins (SDK workflows/Driver supports)	1058
12.5.4	Service Interface (Driver supports)	1059
12.5.4.1	Introduction	1059
12.5.4.2	Accessing masking engine rulesets	1060
12.5.4.3	Accessing database server (JDBC)	1061
12.5.4.4	Logging (Service interfaces)	1062
12.6	Managing plugins using the API client	1065
12.6.1	Displaying information about installed plugins	1065
12.6.2	Other plugin endpoint operations	1065
12.7	Installing a plugin onto the Delphix masking engine	1065
12.8	Secure plugin deployment.....	1066
12.8.1	Using roles to restrict plugin installation.....	1066
12.8.2	Verifying the SHA256 hash of installed plugins.....	1067
12.9	Terminology	1068
12.9.1	Terminology	1068

1 Welcome to the Delphix Continuous Compliance documentation!

This information explains how to deploy Continuous Compliance Engines for data masking, use its features, or tune its configurations for optimal performance. The content has been organized into several categories, available from the lefthand navigation.

List of Continuous Compliance documentation versions in PDF format.

- [25.0.0.0_ContinuousCompliance.pdf](#)¹
- [24.0.0.0_ContinuousCompliance.pdf](#)²
- [23.0.0.0_ContinuousCompliance.pdf](#)³
- [22.0.0.0_ContinuousCompliance.pdf](#)⁴
- [21.0.0.0_ContinuousCompliance.pdf](#)⁵
- [20.0.0.0_ContinuousCompliance.pdf](#)⁶
- [19.0.0.0_ContinuousCompliance.pdf](#)⁷
- [18.0.0.0_ContinuousCompliance.pdf](#)⁸
- [17.0.0.0_ContinuousCompliance.pdf](#)⁹
- [16.0.0.0_ContinuousCompliance.pdf](#)¹⁰
- [15.0.0.0_ContinuousCompliance.pdf](#)¹¹
- [14.0.0.0_ContinuousCompliance.pdf](#)¹²

1 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/25.0.0.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058453793&version=1

2 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/24.0.0.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058454046&version=1

3 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/23.0.0.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058454283&version=1

4 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/22.0.0.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058454549&version=1

5 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/21.0.0.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058455365&version=1

6 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/20.0.0.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058456215&version=1

7 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/19.0.0.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058457051&version=1

8 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/18.0.0.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058457987&version=1

9 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/17.0.0.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058458957&version=1

10 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/16.0.0.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058460107&version=1

11 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/15.0.0.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058461022&version=1

12 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/14.0.0.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058462068&version=1

- [13.0.0.0_ContinuousCompliance.pdf](#)¹³
- [12.0.0.0_ContinuousCompliance.pdf](#)¹⁴
- [11.0.0.0_ContinuousCompliance.pdf](#)¹⁵
- [10.0.0.0_ContinuousCompliance.pdf](#)¹⁶
- [9.0.0.0_ContinuousCompliance.pdf](#)¹⁷
- [8.0.0.0_ContinuousCompliance.pdf](#)¹⁸
- [7.0.0.0_ContinuousCompliance.pdf](#)¹⁹
- [6.0.17.0_ContinuousCompliance.pdf](#)²⁰
- [6.0.16.0_ContinuousCompliance.pdf](#)²¹
- [6.0.15.0_ContinuousCompliance.pdf](#)²²
- [6.0.14.0_ContinuousCompliance.pdf](#)²³
- [6.0.13.0_ContinuousCompliance.pdf](#)²⁴
- [6.0.12.0_ContinuousCompliance.pdf](#)²⁵
- [6.0.11.0_ContinuousCompliance.pdf](#)²⁶
- [6.0.10.0_ContinuousCompliance.pdf](#)²⁷

13 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/13.0.0.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058489985&version=1

14 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/12.0.0.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058489109&version=1

15 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/11.0.0.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058488397&version=1

16 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/10.0.0.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058487426&version=1

17 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/9.0.0.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058486442&version=1

18 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/8.0.0.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058485680&version=1

19 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/7.0.0.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058484854&version=1

20 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/6.0.17.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058484099&version=1

21 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/6.0.16.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058483116&version=1

22 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/6.0.15.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058482271&version=1

23 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/6.0.14.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058481531&version=1

24 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/6.0.13.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058480836&version=1

25 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/6.0.12.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058479997&version=1

26 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/6.0.11.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058479335&version=1

27 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/6.0.10.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058478644&version=1

- [6.0.9.0_ContinuousCompliance.pdf](#)²⁸
- [6.0.8.0_ContinuousCompliance.pdf](#)²⁹
- [6.0.7.0_ContinuousCompliance.pdf](#)³⁰
- [6.0.6.0_ContinuousCompliance.pdf](#)³¹
- [6.0.5.0_ContinuousCompliance.pdf](#)³²
- [6.0.4.0_ContinuousCompliance.pdf](#)³³
- [6.0.3.0_ContinuousCompliance.pdf](#)³⁴
- [6.0.2.0_ContinuousCompliance.pdf](#)³⁵
- [6.0.1.0_ContinuousCompliance.pdf](#)³⁶
- [6.0.0.0_ContinuousCompliance.pdf](#)³⁷
- [5.3.9_ContinuousCompliance.pdf](#)³⁸
- [5.3.8_ContinuousCompliance.pdf](#)³⁹
- [5.3.7_ContinuousCompliance.pdf](#)⁴⁰
- [5.3.6_ContinuousCompliance.pdf](#)⁴¹
- [5.3.5_ContinuousCompliance.pdf](#)⁴²

28 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/6.0.9.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058477701&version=1

29 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/6.0.8.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058476952&version=1

30 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/6.0.7.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058476265&version=1

31 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/6.0.6.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058475430&version=1

32 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/6.0.5.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058474367&version=1

33 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/6.0.4.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058473500&version=1

34 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/6.0.3.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058472769&version=1

35 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/6.0.2.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058471645&version=1

36 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/6.0.1.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058470939&version=1

37 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/6.0.0.0_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058470129&version=1

38 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/5.3.9_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058469444&version=1

39 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/5.3.8_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058468845&version=1

40 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/5.3.7_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058468081&version=1

41 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/5.3.6_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058466879&version=1

42 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/5.3.5_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058466072&version=1

- [5.3.4_ContinuousCompliance.pdf](#)⁴³
- [5.3.3_ContinuousCompliance.pdf](#)⁴⁴
- [5.3.2_ContinuousCompliance.pdf](#)⁴⁵
- [5.3.1_ContinuousCompliance.pdf](#)⁴⁶

43 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/5.3.4_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058465310&version=1

44 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/5.3.3_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058464671&version=1

45 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/5.3.2_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058463901&version=1

46 https://delphixdocs.atlassian.net/wiki/download/attachments/205324526/5.3.1_ContinuousCompliance.pdf?api=v2&cacheVersion=1&modificationDate=1724058463086&version=1

2 Quick references

- [Data masking overview](#)⁴⁷
- [Installation](#)⁴⁸
- [Identifying sensitive data](#)⁴⁹
- [New features](#)⁵⁰
- [Fixed issues](#)⁵¹

⁴⁷ <https://masking.delphix.com/docs/latest/introduction-to-delphix-masking>

⁴⁸ <https://masking.delphix.com/docs/latest/installation>

⁴⁹ <https://masking.delphix.com/docs/latest/discovering-your-sensitive-data>

⁵⁰ <https://masking.delphix.com/docs/latest/new-features>

⁵¹ <https://masking.delphix.com/docs/latest/fixes>

3 Release notes

This section covers the following topics:

- [New features](#) (see page 46)
- [Fixed issues](#) (see page 76)
- [Known issues](#) (see page 127)
- [Deprecated and end-of-life features](#) (see page 143)
- [Licenses and notices](#) (see page 148)

3.1 New features

3.1.1 Release 26.0.0.0

- **Google Cloud SQL IAM authorization for MySQL**
Google Cloud SQL is Google Cloud's managed database service. This release adds built-in IAM Authorization support for Google Cloud SQL MySQL, similar to the support added in release 24.0.0.0 for Google Cloud SQL Microsoft SQL Server and Google Cloud SQL PostgreSQL.
- **MySQL and MariaDB JDBC Driver Update**
The MySQL and MariaDB connectors' JDBC driver has been updated. These connectors will automatically use the MariaDB Connector/J 3.4.0 driver. This brings numerous performance and functional improvements. Please note the following:
 - Generic Connectors: Any generic connectors using a JDBC URL with the MySQL schema (`jdbc:mysql:`) must be updated to set the `permitMySQLScheme` option. For example:
`jdbc:mysql://HOST/DATABASE?permitMySQLScheme` .
 - Note, Generic Connectors were [deprecated](#)⁵² in release 24.0.0.0 and will be removed from the product in the future.
 - Aurora MySQL 5.7 is past end-of-life and therefore this release removes support for it.

3.1.2 Release 25.0.0.0

- **Cockroach DB Connector in the picklist**
For enhanced usability, Cockroach DB has been added to the user interface picklist within the Database Connector wizard.
- **Improved automated sensitive data discovery (ASDD) accuracy for files**
ASDD now aggregates the discovery results across files of the same format within a discovery job before making sensitive element assignments.

⁵² <https://delphixdocs.atlassian.net/wiki/spaces/CC/pages/8323113/Deprecated+and+end-of-life+features#Deprecated-features>

3.1.3 Release 24.0.0.0

- **Automated Sensitive Data Discovery (ASDD) for mainframe datasets**

ASDD was introduced last year, bringing a complete set of sensitive data identification techniques to structured data and semi-structured files. Previously, ASDD worked with databases, JSON, delimited, XML, and fixed width files. It now supports mainframe datasets as well.
- **Google Cloud SQL IAM authorization for Microsoft SQL Server and PostgreSQL**

Google Cloud SQL is Google Cloud's managed database service. Google offers Microsoft SQL Server, PostgreSQL, and MySQL versions of the service. Continuous Compliance has supported all three versions since release 6.0.5.0.

 - A Google Cloud SQL instance authorizes connections, including connections from Continuous Compliance, based on one of the following: (a) Google's Identity and Access Management (IAM) feature, (b) self-managed SSL/TLS certificates, or (c) authorized networks.
 - While all three options were supported in the past, IAM Authorization required an external proxy. In this release, we've added built-in support for IAM Authorization for both Microsoft SQL Server or PostgreSQL instances. IAM Authorization for MySQL instances will be available in a future release. This built-in support simplifies the configuration of Continuous Compliance by eliminating the need for an additional proxy.
- **Configurable user session timeout**

When a user logs into the UI or API, a user session is created. If the user session is idle for a certain amount of time, it will be terminated due to inactivity. This timeout is now configurable via an application setting called `UserSessionTimeoutMinutes`.

3.1.4 Release 23.0.0.0

- **Valid address masking**

A new algorithm framework has been introduced to allow for valid address masking. Prior to this, users relied on a services-provided extended algorithm.
- **S3-compatible storage enablement**

File masking support for S3-compatible storage has been introduced to augment the existing file masking support for AWS S3. Users can now connect to GCP using this approach.
- **Remaining UI component updates**

The final group of UI components, including the Login and Algorithm Setting pages, have been updated and enhanced. Also, administrators can now add a custom message to the login screen.
- **Legacy UI removal**

As part of the completion of the UI refresh, legacy UI components that are no longer needed have been removed, resulting in improved security and UI performance.

3.1.5 Release 22.0.0.0

- **User experience**

The user experience has been dramatically improved across all remaining admin sub-pages, including Users, Logs, About, and Email Notifications. Additionally, a new sub-page for Application Settings has

been introduced, enabling users to conveniently adjust application behavior via the UI. The Environments landing page has been upgraded, and the async task is now accessible as a separate page under Monitor.

Moreover, the entire UX has transitioned into a single-page application, dramatically enhancing UI performance and user experience. This transition results in smoother interactions and faster load times, among other improvements.

- **Automated Sensitive Data Discovery (ASDD) for fixed width files**
ASDD was introduced last year, bringing a complete set of sensitive data identification techniques to structured data and semi-structured files. Previously, ASDD worked with databases, JSON, delimited, and XML files. It now supports fixed width files as well.
- **Automated Sensitive Data Discovery (ASDD) improvements**
ASDD now comes with a major time-saver – profile sets are now tunable with a single assignment threshold. This allows us to ship the ASDD Standard profile set with a better tuned assignment threshold that reduces false positives without impacting the default/out-of-the-box, application-wide assignment threshold for all profile sets. Moreover, users can now fine-tune the ASDD profile sets that are created easier.
- **GCP Secrets Manager support for PostgreSQL**
Increasingly, organizations are looking to connect to their databases various secret manager solutions. With this release, support has been added for using GCP Secrets Manager with Continuous Compliance connectors to Postgres databases.
- **SAP Accelerator**
Introducing a new method to provide secure password properties. Removed the `engine` command to reduce redundancy.
- **YugabyteDB**
YugabyteDB is now supported. This is certified with the existing PostgreSQL connector.
- **New Job Status - WARNING**
A new job status, WARNING, has been introduced to indicate that a job completed with warnings. A job that contains one or more execution components with status warning, such as execution components with non conformant data, will show this status. Previously, these jobs had a status of SUCCEEDED. API code must be updated to interpret this status value correctly (e.g., code that waits for a job to complete must recognize that WARNING, in addition to SUCCEEDED, FAILED, and CANCELLED, indicates job completion).

3.1.6 Release 21.0.0.0

- **Automated Sensitive Data Discovery (ASDD) for XML files**
ASDD was introduced last year, bringing a much more complete set of sensitive data identification techniques to structured data and semi-structured files. Previously, ASDD worked with databases, JSON, and delimited files. Now ASDD also supports XML files.
- **New profiling jobs use ASDD by default**
When creating a new profiling job on an ASDD supported data source, the [ASDD Standard \(see page 528\)](#) profile set will be select by default.

- **AWS S3 connector UI**

AWS S3 connectors can now be created in the UI, in addition to the API. This connector allows you to connect to S3 buckets and mask supported file types.

- **Improved user experience**

This release introduces completely overhauled Environments Jobs and Connectors pages. These improvements include easy to use wizards for both connectors and jobs, as well as new grid views.

3.1.7 Release 20.0.0.0

- **Automated Sensitive Data Discovery for JSON and delimited files**

Automated Sensitive Data Discovery (ASDD) was introduced last year, bringing a more complete set of sensitive data identification to structured data. This has now been expanded to operate against JSON and delimited files.

- **File masking in S3 buckets**

Increasingly, organizations store sensitive application data in object storage buckets. To ensure that these buckets can be masked alongside other application data, connecting directly to S3 buckets and masking supported file types is now available.

- **Improved user experience**

The Add and Edit Ruleset pages in the Environments page have been updated.

3.1.8 Release 19.0.0.0

- **Character Replacement Algorithm Framework**

This framework allows for the creation of rules to replace specific characters in a string with other characters. As an example, you can use this framework to remove inconsistent punctuation in data to maintain referential integrity across data sources.

3.1.9 Release 18.0.0.0

- **New String Chaining Framework**

Introducing a new framework, StringAlgorithmChain, designed to make it easier to build chained algorithms for String types. This framework allows for combining various algorithms to produce a specific output. A scenario where this would be helpful is if an instance of the Data Cleansing framework should be used before another masking algorithm, or if highly unique data needs to be normalized or formatted before or after masking with any type of algorithm.

- **Expanded Classifiers**

Automated Sensitive Data Discovery has been expanded to add data discovery for medical codes (CPT, ICD-9, ICD-10), IBAN TYPE & PATH classifiers, swift codes and bank account routing numbers. Further, we've added a type classifier for license plates.

- **Expanded algorithms**

New masking algorithms have been added for swift codes and bank account routing numbers.

- **UI improvements**

Continued improvement on the user interface, upgrading the monitoring functionality in this release.

- **Certifications**
Certified SAP HANA SPS 07.

3.1.10 Release 17.0.0.0

- **Password Vault support for MariaDB/MySQL**
Supported password vault platforms may now be used to authenticate the connections to MariaDB and MySQL databases.
- **Email algorithm enhancement to include ExtendedEmail plugin capabilities**
The Email algorithm framework has been updated to allow the chaining of existing first name and last name algorithms, expanding on the current Email algorithm capabilities. Options to remove accent characters, specify error handling actions, and preserve original domains are also added, all to facilitate the creation of a realistic masked email value. More details can be found in the [Email \(algorithm frameworks\)](#)⁵³ page.
- **Continuous Compliance user experience improvements**
The Continuous Compliance user experience overhaul continues with an update to the Environments Rule Set page in the UI.
- **New Classifiers**
In Automated Sensitive Data Discovery (ASDD), classifiers are used to identify different types of data. New classifiers have been added for gender, sexual orientation, house number, marital status, language, ethnicity, blood type, prescription drugs, companies, job titles, and department.
- **New Secure Lookup algorithms**
To pair with the new classifiers, new Secure Lookup algorithms have been added for marital status, language, ethnicity, blood type, prescription drugs, job titles, and department.

3.1.11 Release 16.0.0.0

- **New Multi-Column Conditional Algorithm Framework**
A new algorithm framework has been added that allows users to conditionally mask a column. This is an extremely handy new capability that will eliminate scripting or custom plug-ins that were required to satisfy conditional masking use cases in the past.
- **FTPS support for Mainframe data sets**
Users masking files on the Mainframe can now establish direct connections using the built-in FTPS protocol support on the Mainframe.
- **Automated index, constraint, and trigger control for Db2 iSeries**
Users masking Db2 iSeries databases can now take advantage of our automated support for managing indexes, constraints, and triggers that are impacted by masked columns. This eliminates custom pre and post scripting that has otherwise been required.
- **Automated constraint and trigger control for Db2 z/OS**
Users masking Db2 z/OS databases can now take advantage of our automated support for managing constraints and triggers that are impacted by masked columns. This eliminates some of the custom

⁵³ <https://masking.delphix.com/docs/latest/email-algorithm-frameworks>

pre and post scripting that has otherwise been required. Index automation is not available for Db2 z/OS.

- **Additional classifiers and algorithms**

Additional classifiers, domains, and algorithms have been added to allow users an easier experience finding and masking age and location data.

- **Password vault for SAP ASE**

Supported password vaults may now be used with SAP ASE (Sybase) databases.

- **Updated user experience**

Continued overhauls of the user interface have been implemented to provide better utility, scalability, and stability. In this release, the Inventory page for XML and Mainframe file formats has been updated.

- **Redeploy Support (Repave)**

Continuous Compliance Engines may now be disconnected from their storage and redeployed, maintaining the previous configuration and data. Redeployment support only works with the same Delphix Engine version.

3.1.12 Release 15.0.0.0

- **Document type masking support for Delimited File Fields (JSON and XML)**

Increasingly, XML and JSON data are stored in delimited file fields, often as the result of exporting a database table to a delimited file. Now, these fields can be assigned an appropriate file format so fine grained masking of XML or JSON can be performed.

- **Masking UI Revamp - Inventory screens for databases and JSON**

This release introduces completely overhauled database and JSON inventory pages, showcasing a range of user-friendly improvements. These improvements include filtering, sorting, column resizing, and overall performance optimizations. For those using databases and working with JSON data, this upgrade is a must-have.

- **Improved auditing of cascading deletes**

In some cases, deleting an object (e.g., a Connector) results in a cascading deletion of dependent objects (e.g., rule sets, jobs). Previously the audit log only recorded the initial object deletion. Now, the audit log will also record an entry for all other deleted objects.

3.1.13 Release 14.0.0.0

- **Updated Classifiers**

A new tranche of classifiers has been added to discover additional sensitive data. These new classifiers include elements like medical record number, full name, age, and IP address. Several existing classifiers have been improved to better discover sensitive information, including credit card numbers and bank account numbers.

- **Data Discovery and Authentication Support**

Automated Sensitive Data Discovery now supports OAuth for Salesforce or Kerberos for Oracle Database, Microsoft SQL Server, and SAP ASE.

- **FTP Support for Mainframe MVS Storage**

The Compliance Engine now offers enhanced functionality with FTP support, enabling direct access to the mainframe MVS storage environment.

- **Updated IBM Db2 LUW Connector**

The connector will now provide more automated control of indexes, constraints, and triggers as well as deliver automated identity column support. This will automatically upgrade and allow you to choose these features.

- **IBM Db2 LUW, iSeries and z/OS Temporary Identity Column Support**

The connector will now create an identity column and index for Db2 tables where none exists. These will be removed and the table returned to its original state after masking concludes.

- **Password Vault for IBM DB2 on LUW, iSeries and z/OS**

Supported password vaults may now be used with DB2 databases on LUW, iSeries, and z/OS systems.

- **Updated User Experience**

The process of overhauling the user interface to provide you better utility, scalability, and stability continues. In this release, the inventory page has been updated for fixed-length and delimited file types, as well as the file format settings page.

- **ESXi 8.0 U1**

Continuous Compliance may now be run on VMware ESXi 8.0 U1.



The product's REST API schema specification has been upgraded from Swagger 2.0 to OpenAPI 3.0.1. This may impact API consumers in different ways, depending on what tools are used to access the API. The short summary is: unless your API client is *dynamically* generating API access classes (aka. stubs), you should not notice any disruption.

Usage that will see **limited or no** Impact:

- **Using Curl / HttpRequest / other Third party HTTP libraries**

These tools do not consume the REST API schema. The only known, observable difference regards the error response returned when a request is sent with an empty body (when body content is required). Valid API usage should never encounter this error.

- **Using Swagger 2.0 static stubs**

API client stub methods generated previously from an API schema taken from earlier product versions should continue to function normally. The REST API version used to construct the API stubs should be specified in the request path to maintain compatibility; this a general requirement for API compatibility not resulting from this change.

Usage that will see **significant** impact:

- **Generating API client stubs**

This approach dynamically generates API client access classes using the REST API's swagger schema specification. These classes are sometimes referred to as client-stubs. Third-party libraries like `swagger-codegen` or `openapi-generator` can be used to generate API helper classes. To

continue with dynamic client generation, the third-party library in use will need to be upgraded to a version compatible with the OpenAPI 3 schema format.

- After the library upgrade, the auto-generated OpenAPI 3 client stubs will likely differ from the Swagger 2 specification's client stubs, requiring updates to any code that consumes the stub classes. These changes include the following, though other variations may exist that were not encountered during testing:
 - Change in default date-time libraries (can be configured through generator library arguments).
 - Change in the order of request parameters and request body within the method arguments of stubs classes.
 - Change in getter functions of boolean variables (ex: variable-"valid" getter function in 2.0 specification is `getValid()`, whereas in 3.0 it is `isValid()`).

3.1.14 Release 13.0.0.0

- **Password Vault Support for Microsoft SQL Server**
In this release, the SQL Server connector joins the growing list of connectors that support credential management systems.
- **SQL Server**
SQL Server 2022 is now supported.
- **Microsoft Azure SQL Data Warehouse**
Azure SQL Data Warehouse is now supported.
- **Updated User Experience**
The Roles page has been updated, with a number of handy templates now provided as well.

3.1.15 Release 12.0.0.0

- **ASDD extended drivers**
Extended drivers allow you to provide your driver with additional data sources. Automated Sensitive Data Discovery, which aims to identify common sensitive and personally identifiable data elements, now works with these sources.
- **Password vault support for Oracle**
You can now leverage supported password vaults to manage credentials for Oracle databases.

3.1.16 Release 11.0.0.0

- **IBAN discovery and masking**
Supports automatically discovering and masking International Bank Account Numbers.
- **New US driver's license classifier**
Supports automatically discovering and masking US Driver's Licenses.

- **Oracle source sizing**
Measures the size of connected Oracle databases. This information will be available via a new utilization PDF report, as well as through the GET /billing-usage API.
- **New features for the Postgres connector**
The connector will now provide more automated control of indexes, constraints, and triggers. This will automatically upgrade and allow you to choose these features.
- **User-controlled automated sensitive data discovery (ASDD) set upgrade**
With the 11.0 release, Delphix will begin shipping the standard ASDD discovery set independent of Continuous Compliance. You can use a new API endpoint to control when to adopt new ASDD discovery sets. This will allow you to upgrade to newer releases of the software without impacting existing workflows, as well as easily creating and distributing your own classifier sets.

3.1.17 Release 10.0.0.0

- **Password vault support for PostgreSQL connections**
This release adds HashiCorp Vault and CyberArk support for storing and accessing secrets, keys, and certificates necessary for PostgreSQL operations.
- **More automated sensitive data discovery classifiers**
Classifiers improve our ability to automatically discover sensitive information and accurately recommend masking algorithms. This update adds classifiers for identifying Credit Cards, Addresses, Telephone numbers, US Passports, US SSNs, and Bank accounts. With ASDD Classifiers, organizations can easily mask new data sources for downstream teams to start working with compliant data quickly.
- **Classifiers user interface**
This update adds a new user interface for the Automated Sensitive Data Discovery classifiers introduced in 9. Previously, classifier definitions were only accessible via the API.
- **Enhanced document type support for JSON and XML**
This update expands our support for the JSON and XML Document types. Applying the multi-column algorithms to JSON and XML structures is now possible. Further, it is now possible to mask JSON and XML structures stored in BLOBs.
- **CData license upload**
Our partnership with CData allows customers to purchase and use CData JDBC drivers from Delphix to connect to additional data sources with Continuous Compliance. This release enables users to install the required license files on their engines. Customers download the CData drivers directly from CData but download the corresponding license files for these drivers from Delphix. Consult your account representative for purchasing and using a specific CData driver.

3.1.18 Release 9.0.0.0

- **New Automated Sensitive Data Discovery (ASDD) profiler implementation**
The new ASDD profiler has been introduced with a number of [new features \(see page 653\)](#), including new data-level profiling logic and improved database sampling capabilities. A new profile set, [ASDD Standard \(see page 528\)](#) has been added that leverages the new implementation.
- **UI Updates: Audit and Classifiers**

Audit: This page allows for a quick review of historical user activity with global audit logs, to help ensure compliance. Updated for speed and ease-of-use, with significantly better filtering, searching, and performance.

Classifiers: This section of the Settings page contains a list of Classifiers with relevant information and actions. The Profiler can be customized for domain and data level (name, column metadata, data value) using standard regex expressions. Improved performance, data level inspection, and accuracy for matching data types (using data classifiers).

3.1.19 Release 8.0.0.0

- **Improved JSON document store performance**
Upon upgrade, JSON Document Store masking is significantly faster. For more information, visit [Document Store Type Masking \(see page 0\)](#).
- This release contains bug fixes for the Continuous Compliance Engine.

3.1.20 Release 7.0.0.0

- **JSON and XML: tokenization/re-identification and chained algorithms**
Easily leverage tokenization/re-identification algorithms and chained algorithms with JSON and XML data. Data tokenization/re-identification provides reversible data anonymization to protect data in non-prod environments. Chain algorithms enable complex multistep algorithms to be run on separate values, such as Full Name algorithms.

3.1.21 Release 6.0.17

- **JSON; XML field masking**
This release introduces the ability to mask JSON; XML objects nested in string/text fields, allowing Continuous Compliance to be maintained for semi-structured JSON & XML data in non-production environments. With this, the Continuous Compliance library can be leveraged, or customized algorithms can be created to meet required data schemas.
- **Microsoft Intelligent Data Platform Integration (Azure Data Factory; Azure Synapse Pipelines)**
Accelerate data compliance using Microsoft Intelligent Data Platform's ETL tools with Delphix Continuous Compliance. This allows users to quickly mask data while moving between 100+ Azure Synapse Analytics and Azure Data Factory connections. Quickly identify sensitive data in ETL pipelines and mask using Delphix algorithms.
- **Containerized masking**
This feature allows users to efficiently spin up and tear down Continuous Compliance containers. Easily orchestrate scaling out Continuous Compliance using a container orchestrator, allowing for quick parallelized masking jobs in the self-managed container cluster to multiple instances.
- **Hyperscale Compliance masking job sync**
Introducing fast migration for masking jobs from existing Continuous Compliance Engines to the Hyperscale Compliance Orchestrator. This accelerates the masking of massive Oracle databases for compliance and greatly improves masking speed for databases with billions of rows containing PII, PHI, or sensitive data fields.

3.1.22 Release 6.0.16.0

- **Strict content security policy**

This release adds a new application setting group that drives strict content security policy. Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks. Including (but not limited to) Cross Site Scripting (XSS), and data injection attacks.

3.1.23 Release 6.0.15.0

- **JSON file masking**

This update introduces support for JSON File Masking, a popular human-readable data exchange format. Teams can leverage Delphix's existing library of pre-built algorithms to mask sensitive information. For more information, see [JSON File Masking](#).⁵⁴

- **Segmented mapping algorithm V2**

This update enhances the Segmented Mapping Algorithm for improved performance, extensibility, security, and portability. It produces new masking results from the legacy algorithm. Segmented Mapping allows a user to create unique masked values by dividing data into segments that are masked piecewise. For more information, see [Segmented Mapping](#). (see page 932)

- **Numeric expression algorithm**

This new algorithm enables formula transformations to values, so that teams can run common math operators to scale or shift numbers.

- **New API endpoints**

This release introduced a new API Support Bundle Generation endpoint:

The new API endpoint is:

Group	Endpoints	Description
supportBundle	POST /support-bundle	Generates support bundle

For more information, see [API for generating support bundle](#). (see page 974)

3.1.24 Release 6.0.14.0

- **Certifications**

- SAP HANA 2.0 SP 05
- CockroachDB
- VMware ESXi 7.0 U3c

- **Data cleansing**

The Data Cleansing algorithm has been updated to standardize spellings, misspellings, and convert

⁵⁴ <https://delphixdocs.atlassian.net/wiki/spaces/CC/pages/8456938>

abbreviations. Algorithm based data cleansing eliminates slow and manual processes prior to masking the data. For more information, see [Data Cleansing](#). (see page 904)

- **Min Max**

The MinMax Number and MinMax Date algorithms have been updated for normalizing outlier data in a table column by masking numbers and dates that could give context of records based on values.

- **Continuous Compliance UI improvements**

- Monitoring interface now details which job step is currently underway
- Improved Continuous Compliance job management for intermediate steps
- Redesigned Execution Monitor interface
- For more information, see [Monitoring Masking Job](#). (see page 513)

- **“Optional” columns for multi-column algorithms**

The Multi-column algorithm now allows for some fields to be marked as optional for more run-time flexibility. Concurrent Continuous Compliance operations can now occur using multiple data columns in a single operation. For more information, see [Using Multi-Column Algorithms](#). (see page 1031)

- **Automated sensitive data discovery**

A new default profile set is being implemented to improve the accuracy and speed of column level profiling. 40 new profile expressions utilizing type constraints are introduced for domain and algorithm assignment to help reduce false positives associated with data type mismatches during inspection. The default profiler set upgrade has no change or impact on existing users.

- **New API endpoints**

This release extends the list of API-endpoints by adding the following task-based progress monitoring endpoints.

The new API endpoints are :

Group	Endpoints	Description
monitoring	GET /monitor-task	get the status of Execution or async task
monitoring	GET /monitor-task/	get the status of Execution or async task by the id

3.1.25 Release 6.0.13.0

- **Certifications**

This release adds support for VMware ESX/ESXi 7.0 U3c and DB2 12.0 on z/OS.

- **New tokenization algorithm**

In this release, Delphix introduces a new Tokenization algorithm framework to replace the legacy Tokenization algorithm. This new Tokenization algorithm framework includes additional configuration options for increased security. For more information, see [Tokenization](#) (see page 806). Legacy Tokenization algorithm instances will remain in place and function the same until their planned EOL in version 6.0.15.0, migration to the new Tokenization algorithm is recommended.

- **Zip+4 algorithm**

A new version of the Zip+4 algorithm is now available that is used for full-length (nine-digit) zip codes. This new version is built upon the Masking Algorithm SDK and offers the same benefits as other new algorithms, including greater performance.

- **Improved masking monitoring**

This release improves usability and diagnosability of the Masking Engine by allowing users to search for past jobs and filter the results based on job type and status. For more information, see the **Search** section at [Monitoring Masking Job](#). (see page 513)

- **New API endpoints**

This release extends the list of API-endpoints by adding the following execution logs endpoints. These API endpoints will return file download ID that can be used to download execution logs under GET /file-downloads/{fileDownloadId}.

The new API endpoints are :

Group	Endpoints	Description
executions	GET /execution-logs	get all execution logs of all jobs.
executions	GET /executions/	get a particular execution log by using execution ID.
execution-component-log	GET /execution-component-log	get all the execution component logs of all jobs, execution ID is a mandatory parameter.
execution-component-log	GET /execution-component-log/	get a particular execution log by using componentId.

3.1.26 Release 6.0.12.0

- **New Microsoft SQL Server implementation to disable constraints/triggers and drop indexes**

In this release, Delphix adds default driver support for Microsoft SQL Server database masking options of Disable Constraints, Drop Indexes, and Disable Triggers as job tasks. These changes apply to masking, reidentification, and tokenization jobs where enabled.

For details on the usage and known limitations of the Microsoft SQL Server Disable Constraints, Drop Indexes, and Disable Triggers driver support tasks, see [Microsoft SQL Server Built-in Driver Support Plugin](#) (see page 823). Upon engine upgrade, any existing jobs on built-in Microsoft SQL Server connectors where these options were selected will be upgraded to these enabled driver support plugin tasks.

- **Improved user experience diagnosability**

This release improves user experience by ensuring time is displayed in a consistent fashion. It prevents users from running parallel update threads against incompatible databases and provides

per-job log information. The job monitoring view now displays the total time taken in the hours:minutes:seconds format.

- **Free text redaction**

This release updates the free text redaction algorithm to the new extensible Algorithm framework to improve performance and allow chaining of algorithm instances. For more information, see [Free Text Redaction](#). (see page 912)

- **Updated secure lookup instances**

This release updates the legacy Secure Lookup algorithms to the extensible Secure Lookup framework. Masked results will remain the same other than whitespace handling.

- **New API endpoint for define fields**

This release extends the list of API-endpoints by adding the following file-field-metadata endpoint to create field metadata for a file format. This field allows users to add a file field that you want to mask in a format. After the user uploads a format, all the fields from the uploaded file format are displayed at the inventory screen.

The new API endpoint is :

Group	Endpoints	Description
fileFieldMetadata	POST /file-field-metadata	Creates field metadata for a file format.

- **Masking whole file**

You can now configure the masking engine to mask the complete file and pass the content of that file as a single input to an algorithm. For more information, see [Masking Whole File](#). (see page 522)

- **Character mapping algorithm support for tokenization/reidentification jobs**

The character mapping algorithm can now be used for tokenization and reidentification jobs.

3.1.27 Release 6.0.11.0

- **Certifications**

This release adds support for Oracle database 21c.

- **General UI for extended algorithms**

In this release, Delphix continues to improve the experience of creating and using new extended algorithms. These algorithms may include configuration information stored in JSON format. The configurations are now editable via the UI. For more information, see [General UI for Extended Algorithms](#). (see page 813)

- **OAuth2 API support**

The Virtualization and Masking engine APIs are now accessible via OAuth2 tokens that improve Delphix's security offerings. For more information, see [Configuring OAuth2 Authentication for API Access](#).⁵⁵

- **New Oracle optimizations to disable constraints/triggers and drop indexes**

⁵⁵ <https://delphixdocs.atlassian.net/wiki/spaces/CD/pages/6295763/Configuring+OAuth2+authentication+for+API+access>

In this release, Delphix has re-implemented the Oracle database masking options of Disable Constraints, Drop Indexes, and Disable Triggers as job tasks, using the [Driver Support Plugin Framework](#) (see page 820), improving both functionality and performance. These optimizations apply to masking, reidentification, and tokenization jobs where these tasks are enabled.

For details on the optimizations, usage, and known limitations of the Oracle Disable Constraints, Drop Indexes, and Disable Triggers driver support tasks, see [Oracle Built-in Driver Support Plugin](#) (see page 822). Upon engine upgrade, any existing jobs on built-in Oracle connectors where these options were selected will be upgraded to these enabled driver support plugin tasks.

- **New export secure lookup values API**

This release extends the list of API-endpoints by adding a new API for exporting the values from a secure lookup algorithm instance.

The new API endpoint is:

Group	Endpoints	Description
algorithm	POST /algorithms/	Export lookup values form secure lookup algorithm.

For more information, see [Secure Lookup - Exporting Secure Lookup Values via API](#) (see page 793).

- **New copy ruleset API**

This release extends the list of API-endpoints by adding three new APIs for copying rulesets under databaseRuleset, fileRuleset, and mainframeDatasetRuleset.

The new API endpoints are :

Group	Endpoints	Description
databaseRuleset	PUT /database-rulesets/	Copy ruleset objects in the same database environment.
fileRuleset	PUT /file-rulesets/	Copy ruleset objects in the same file environment.
mainframeDatasetRuleset	PUT /mainframe-dataset-rulesets/	Copy ruleset objects in the same dataset environment.

- **New binary lookup algorithm**

This release introduces a new binary lookup algorithm framework in the masking extensibility SDK that supports advanced features such as algorithm chaining. Legacy binary lookup algorithm instances will be automatically and seamlessly migrated to the new binary lookup framework when you upgrade the masking engine. For more information, see [Binary Lookup](#) (see page 705).

- **UI/UX enhancements**

This release introduces substantial improvements to the user interface that gives a new look and feel to the masking engine.

3.1.28 Release 6.0.10.0

- **Masking Salesforce data**

There has been an increasing demand for an easy way to manage and utilize the highly sensitive data stored in Salesforce. With this new Select Connector offering, sensitive data discovery and masking algorithm assignment is automatically handled for the Salesforce default schema; this is not only unique in the market, but also the first time Delphix is delivering this solution as an addition to its product suite. This is the top compliance solution for Salesforce on the market and provides a dramatically simpler deployment option to manage and secure this business-critical data. For more information, see [Application Solutions documentation](#).⁵⁶

- **New mapping algorithm**

A more powerful and faster mapping algorithm is now available. This allows running the same mapping algorithm across multiple jobs and across multiple engines. Running the same mapping algorithm across multiple engines requires a compatible external database. New APIs now support migrating mappings from existing mapping algorithms to the new mapping algorithms.

- **Algorithm replacement APIs**

APIs are now being introduced to list and replace algorithms.

Group	Endpoints	Description
algorithm	GET /algorithms/	Retrieves all usage of the algorithm specified in the request path.
algorithm	PUT /algorithms/	Updates all usage of the algorithm specified in the request path to use the new algorithm name supplied as a query parameter.

For more information, see [Managing Algorithm Usage](#) (see page 894).

Group	Endpoints	Description
algorithm	GET /algorithms/migration	Returns a list of result objects describing each possible migration. One object is returned for every algorithm on the engine that can be migrated.
algorithm	POST /algorithms/	Creates a new algorithm named <code>newAlgorithmName</code> (from the API query parameters), by migrating from the algorithm named in the query path.

⁵⁶ <https://ecosystem.delphix.com/docs>

For more information, see [Migrating algorithms \(see page 898\)](#).

- **New phone masking algorithm**

A new masking algorithm for the phone number framework for US and international numbers is now available. Migration from the old phone masking algorithm to the new one is required. For more information on transition, see [Delphix Community Post](#)⁵⁷.

- **New custom SQL API**

In this release, Delphix has extended the list of API-endpoints by adding a new table-metadata endpoint for generating custom SQL for the given tableMetadataId.

The API endpoint is :

Group	Endpoints	Description
tableMetadata	GET /table-metadata/	Generates a custom SQL.

3.1.29 Release 6.0.9.0

- **Masking SDK driver support plugins**

The Masking SDK functionality is extended with the ability to develop a new kind of plugin, called driver support plugins. These allow the execution of developer-defined tasks as part of a masking job.

- **Masking SFTP connector is extended with a new flag UserDirIsRoot**

Delphix introduces a new flag, setting whether the SFTP Connector configured Path is relative or absolute.

- **New Email framework**

Delphix introduces a new Email Framework along with two default algorithm instances. This functionality allows for more customization in masking email addresses.

- **New copy environment API**

In this release, Delphix has extended the list of API-endpoints by adding a new API for copying environments.

The API endpoint is :

Group	Endpoints	Description
environment	POST /environments/	Copy environment objects in the same or a different application

3.1.30 Release 6.0.8.0

- **New name and full name frameworks**

⁵⁷ <https://community.delphix.com/blogs/michael-torok/2021/08/12/delphix-end-of-life-notice-legacy-masking-algorithm>

Delphix introduces new Name and Full Name Frameworks, as well as their default algorithms instances. That functionality adds flexibility and more sophisticated ways for name masking.

- **Masking SDK multiple plugins capacity**

Masking SDK functionality is extended with an option of loading multiple plugins and chaining extensible algorithms based on different plugins. The dlpx-core plugin is uploaded by default.

- **New regex decompose algorithm chaining framework**

Delphix introduces the Regex Decompose extensible algorithm framework, which allows the capability to build new algorithms from a combination of predefined actions and existing algorithms.

- **Enclosure escape character support for delimited file masking**

In this release, Delphix has added escape character support for delimited file masking. Specifically the following were added:

- **Enclosure escaping strategy:** The user can configure the enclosure escape character from the UI/API to escape the enclosure. To configure the enclosure escape character from the UI, the user must select the "Enclosure Escaping Strategy" dropdown value as per the below options on the edit Rule Set popup window.
 - Double Enclosure:** Double enclosure option will set the escape character value same as enclosure value.
 - Custom:** By selecting custom option, the user can specify any single character as an enclosure escape character, except the "escape sequences" and "control characters".

- **Escape "Enclosure Escape Character"**

The user can escape the "enclosure escape character" itself by clicking on the Escape "Enclosure Escape Character" checkbox on the edit RuleSet popup window.

For more detailed information, see [Managing rule sets \(see page 400\)](#).

3.1.31 Release 6.0.7.0

- **New date masking frameworks**

Delphix introduces new date masking frameworks, which includes date replacement, date shift, and multi-column dates. These new frameworks obviate the need for many of the custom date algorithms that were required in the past. Delphix also introduces new default implementations of common date-masking functionality. The new date masking frameworks are briefly described below.

- **Date Replacement:** Selects a replacement value from a configurable date range.
- **Date Shift:** Produces a replacement value by randomly shifting the input date by a configurable increment range.
- **Multi-column Date:** Masks date values that have a dependency, such as admission and discharge date using the same algorithm as Date Shift. This allows masking of both the initial date and the difference between the dates.

- **New credit card masking algorithms**

Delphix introduces a robust payment-card masking framework, as well as a default algorithm implementation for credit card data. The legacy credit card algorithm (that produced random values) is being replaced by the new default instance, which provides consistent masking results, a unique output for every valid input, always changes a valid input value, and preserves all non-digit portions of the input value.

- **Masking Engine changes for users and groups**

This enhancement adds stronger on-Masking Engine safeguards to the Users and Groups experience delivered in Central Management ,in which the access to a Masking Engine’s objects is determined by assigning authorization via global access groups. Specifically, when an engine opts into the global model, it relinquishes local control of object access. With this, the local enforcement of global (Central Management) settings is strengthened by deactivating local object access in the UI, thus ensuring the local values will not be overridden via frequent, periodic scans from Central Management.

- **New forgot and reset password APIs**

In this release, Delphix has extended the list of API-endpoints by adding two new API's related to the existing Forgot and Reset password feature for a user, which was available via GUI only till now.

The two new sets of API endpoints are :

Group	Endpoints	Description
user	POST /users/forgot-password	Send reset password mail to the user
	POST /users/reset-password	Reset new password for the user

The forgot-password API will generate and send a password reset link to the registered email id of the user, for which the password has to be reset.

The reset-password API will use the token sent via the password reset link, to set the new password.

- **Control character support for delimited file masking**

In this release, Delphix has added the control character support for delimited file masking.

Specifically the following were added:

- Control character as a delimiter:** The user can specify a control character as the delimiter from UI/API.
- Control character as an end of record:** The user can specify a control character as the end of record from UI/API.
- Control character as a value:** Delimited files containing values with control characters are now supported.

- **Date-Time format change for the API response**

In this release, the date-time format for API responses is changed

From: yyyy-MM-dd'T'HH:mm:ss.SSSZ e.g. 2021-03-17T17:35:39.352+0000

To: yyyy-MM-dd'T'HH:mm:ss.SSSXX e.g. 2021-03-17T17: 35:39.352+00:00.

The API endpoints below will be affected by this change:

- GET /system-information
- GET /plugin
- GET /profile-jobs
- GET /profile-sets
- GET /execution-events
- GET /async-tasks

- GET /audit-logs
- GET /algorithms in algorithm extension object
- GET /execution-components
- GET /jdbc-drivers
- GET /masking-jobs
- GET /reidentification-jobs
- GET /tokenization-jobs

3.1.32 Release 6.0.6.0

- **Certifications**

This release adds support for DB2 iSeries v7.4.

- **Multi-column algorithm**

In this release, Delphix has introduced a Multi-Column Extensible Algorithm mechanism, which allows masking multiple columns of the same table conditional to their values (or using any other logic needed by the customer). To use the Multi-Column Algorithm Framework, users first create an algorithm via the Masking SDK and then install their algorithm on a Masking Engine via the Extensible Algorithm Plugin interface.

- **Latest API version**

The latest masking API version supported on the engine will be included in the `GET /system-information` API response.

- **Custom database connection properties**

There is now a way to specify custom connection properties for all of our database connector types by uploading a properties file. For more information, see [Database Connection Properties \(see page 355\)](#).

3.1.33 Release 6.0.5.0

- **Certifications**

This release adds support for the following certificates:

- MySQL 8
- Postgres SQL 12
- DB2 LUW 11.5
- Oracle Database Cloud Services on Virtual Machines
- Oracle Database Cloud Services on Bare Metal
- Google Cloud SQL for PostgreSQL
- Google Cloud SQL for MySQL
- Google Cloud SQL for SQL Server

- **Character mapping algorithm**

Delphix is introducing a replacement for the Segment Mapping Algorithm, the Character Mapping Algorithm. The new Character Mapping Algorithm is built using the recently released algorithm SDK,

and in most common configurations this new algorithm will be faster and require less memory than the existing segment mapping algorithm. In addition, this new version does not have a length limitation for the input string and can handle non-ASCII characters.

- **Default API version**

Introducing the ability to specify the Masking API version to be used when the version is omitted from the base path of the Masking API request's URL.

- **New API version**

To reflect the API improvements mentioned above, the API version increased to 5.1.5 in this release. For a complete listing of version 5.1.5, see [Masking API Client \(see page 878\)](#).

3.1.34 Release 6.0.4.0

- **Certifications**

This release adds support for SQL Server 2017 and 2019.

- **Masking job memory improvements**

Memory management has been dramatically improved. Not only can jobs run with less memory, but the Masking Engine will also now ensure that jobs can only run if enough memory is available and that the engine cannot run out of memory.

Along with these changes, there are two new execution statuses: `CANCELLED` and `QUEUED`.

- **Extensible connector permissions change**

The first iteration of the Masking Extensible Connectors, supporting the ability to upload and use JDBC drivers, required that the permissions for each driver be enumerated at install time. Delphix has now replaced this mechanism with a fixed security policy blocking only the most dangerous permissions (specifically those that could inflict harm to the Masking Engine), removing the need for user management of permissions. It remains the case that the engine administrator must ensure that only trusted JDBC driver software is installed.

- **File masking performance**

The performance of file masking has been significantly improved.

- **Builtin extensible secure lookup framework**

Delphix has added a builtin, configurable Secure Lookup Algorithm Framework, based on the Extensible Algorithms feature (introduced in 6.0.3.0 release).

This framework provides better performance and new features when compared with the Legacy Secure Lookup Algorithms.

It allows configuring the case sensitivity of input values (true/false), and the case configuration of the output values:

```
Preserve Lookup File Case // i.e. as found in Lookup File Preserve Input
Case // i.e. preserve case of input value - UpperCase /
LowerCase / Mixed Force all Lowercase // forces output to
LowerCase Force all Uppercase // forces output to UpperCase
```

The algorithm instance (based on the new Secure Lookup Algorithm Framework) might be managed via the existing Algorithm API, similar to any other plugin algorithm. The GUI has been changed for configuring/editing Secure Lookup Algorithm. For more information, see [Secure Lookup Algorithm Framework \(see page 793\)](#).

- **Job scheduler removed**

As of this release, we have removed the Job Scheduler feature. The introduction of Masking's REST API several releases ago allowed customers to schedule job executions using their preferred job scheduler. As a result, the integrated scheduler is seldom used.

- **Free text redaction algorithm**

The redaction strategies used in a free text redaction algorithm have been renamed to "Allowlist" and "Denylist".

- **New API version**

To reflect the API improvements mentioned above, the API version increased to 5.1.4 in this release. For a complete listing of version 5.1.4, see [Masking API Client \(see page 878\)](#).

3.1.35 Release 6.0.3.0

- **Extensible algorithms**

We introduced a new, radically simpler, method to create new masking algorithms. With the new framework, Delphix partners and customers can create and share new algorithms.

Extensible algorithms and their related algorithm plugins can be managed through the following APIs:

Group	Endpoints	Description
plugin	GET /plugin	Get all plugins
	POST /plugin	Install plugin
	DELETE /plugin/	Delete plugin
	GET /plugin/	Get plugin detail by pluginId
	PUT /plugin/	Update plugin

Existing *algorithm* API is extended with the following endpoints:

Group	Endpoints	Description
algorithm	GET /algorithm/frameworks	Get all algorithm frameworks
	GET /algorithm/frameworks/id/	Get algorithm framework by frameworkId

- **UI-based environment sync**

Over the past several releases Delphix has introduced and refined the ability to synchronize objects between Masking Engines via the API. In 6.0.3, Delphix now supports importing and exporting environments via the UI.

Note: In this release, the deprecated XML import/export functionality has been removed. If you used the XML import/export feature in previous releases, you'll find the new Sync Environment feature to be a more robust and complete solution with complete API support in addition to being available in the UI.

- **New SQL Server JDBC driver**

The product switched from the jTDS JDBC driver to Microsoft's official open-source JDBC driver. This was done to obtain improved support for recent versions of SQL Server.

All SQL Server basic connectors will be converted transparently. If you used a SQL Server Advanced connector or a Generic connector using the jTDS driver, you will need to manually convert your JDBC URL to the Microsoft JDBC driver's format. To perform this conversion, see the references for the [jTDS parameters](#)⁵⁸ and the [Microsoft JDBC parameters](#)⁵⁹. Delphix Customer Support's upgrade validation checks will detect any SQL Server Advanced connectors and Generic connectors using the jTDS driver in your installation and they will notify you of the need to manually convert those connectors.

- **AzureSQL managed databases**

This release is certified to be compatible with the following Azure SQL Managed Databases:

- Azure Database for PostgreSQL service
- Azure Database for MySQL service
- Azure Database for MariaDB service
- Azure Database for SQL

Note: You must enable support for non-TLS connections.

- **File masking performance**

This release contains significant performance improvements for delimited and XML file masking.

- **New API version**

To reflect the API improvements mentioned above, the API version increased to 5.1.3 in this release. For a complete listing of version 5.1.3, see [Masking API Client](#) (see page 878).

3.1.36 Release 6.0.2.0

- **Certifications**

This release adds support for Oracle 19c.

- **Mainframe data set improvements for masking**

This release delivers multiple quality-of-experience enhancements around mainframe masking workflows:

- **Mainframe masking performance:** Anyone masking mainframe data sets may see a large improvement in performance.

⁵⁸ <http://jtds.sourceforge.net/faq.html>

⁵⁹ <https://docs.microsoft.com/en-us/sql/connect/jdbc/building-the-connection-url?view=sql-server-ver15>

- **Engine sync support for mainframe:** The Sync APIs and workflows now support mainframe objects: connectors, rule sets, jobs, and formats.
- **Mainframe data set record type APIs:** This enhancement builds upon the recent release of Record Type APIs to include mainframe support. You will now be able to manage Mainframe data set record types via REST API, including redefine conditions. When masking a mainframe data set, the Masking Engine uses a mainframe data set format to interpret the data set's contents. A mainframe data set format has one default record type "All Record". If a mainframe data set format contains redefined fields, each redefined and redefines field will have a corresponding record type that holds the redefined condition for the redefined and redefines fields. Specifically, the following APIs were added:

Group	Endpoints	Description
mainframeDatasetRecordType	GET /mainframe-dataset-record-types	Get all Mainframe Dataset record type
	GET /mainframe-dataset-record-types/	Get Mainframe Dataset record type by ID
	PUT /mainframe-dataset-record-types/	Update Mainframe Dataset record type by ID

- For more information on redefine conditions, see [Managing a mainframe inventory \(see page 451\)](#).
- **JDBC to delimited files support**
On-the-fly masking jobs with a JDBC source and delimited file target are now supported. This is targeted at users with data lake applications. This is targeted at users with data lake applications who wish to extract unmasked data using a JDBC connection and insert masked data back using a bulk file load mechanism.
- **Environment sync support for masking**
With this release, an entire environment is now syncable with a single operation via the Sync REST APIs. Previously, Sync users would have to export/import objects on an individual basis, the process now is far more streamlined. Note: Environment Sync APIs are the preferred way of handling environment export/import versus XML-based transfer.
- **New API version**
To reflect the API improvements mentioned above, the API version increased to 5.1.2 in this release. For a complete listing of version 5.1.2, see [Masking API Client \(see page 878\)](#).

3.1.37 Release 6.0.1.0

- **Extended connectors**
Extended connectors is a new feature that allows you to upload additional JDBC Drivers to the Continuous Compliance engine. This enables masking data sources that are not natively supported by Continuous Compliance. For more information, see [Managing Extended Connectors \(see page 389\)](#).

- **Sync for tokenization and reidentification jobs**

The Sync feature allows you to coordinate the operation of multiple engines. This release adds Sync support for Tokenization and Reidentification Jobs. For more information on the Sync feature, see [Managing Multiple Engines for Masking \(see page 842\)](#).

- **File record type APIs**

When masking a delimited or fixed length file, the Masking Engine uses a file format to interpret the file's contents. Each format has one or more record types. In previous releases, these record types could only be created and managed through the graphical user interface. This release adds the ability to also create and manage file record types through the APIs. Specifically, the following APIs were added:

Group	Endpoints	Description
recordType	GET /record-types	Get all record type
	POST /record-types	Create record type
	DELETE /record-types/	Delete record type by ID
	GET /record-types/	Get record type by ID
	PUT /record-types/	Update record type
recordTypeQualifier	GET /record-type-qualifiers	Get all record type qualifiers
	POST /record-type-qualifiers	Create record type qualifier
	DELETE /record-type-qualifiers/	Delete record type qualifier by ID
	GET /record-type-qualifiers/	Get record type qualifier by ID
	PUT /record-type-qualifiers/	Update record type qualifier by ID

Note that record types are only used for delimited and fixed-length file formats. For more information on record types, see [Adding Record Types for Files \(see page 475\)](#).

3.1.38 Release 6.0.0.0

- **Objects names requirements**

Delphix 6.0 adds validations for object names that can be created/renamed manually. For more information, see [Naming Requirements \(see page 228\)](#).

Please note that enforcing these requirements might fail the import, sync, or upgrade from pre-6.0 release. For resolving those failures, see [Knowledge Base Article KBA5096](https://support.delphix.com/Delphix_Masking_Engine/Object_Naming_Requirements_(KBA5096)).

- **Versioning framework**

6.0 marks the release of version 5.1 of the Masking API. For information on how the Masking API is versioned, see [Masking API Versioning Documentation \(see page 890\)](#).

- **New API endpoints**

In 6.0 we have expanded the list of API endpoints to include:

Group	Endpoints	Description
Application	DELETE /applications/	Delete application by ID
Mount Filesystem	GET /mount-filesystem	Get all mounts
	POST /mount-filesystem	Create a mount
	GET /mount-filesystem/	Get a mount by ID
	DELETE /mount-filesystem/	Delete a mount by ID
	PUT /mount-filesystem/	Update a mount by ID
	PUT /mount-filesystem/	Connect a mount by ID
	PUT /mount-filesystem/	Disconnect a mount by ID
	PUT /mount-filesystem/	Remount a mount by ID

In addition to the new API endpoints, we have improved existing API endpoints. These improvements include:

- Addition of the applicationId field to the application model
- Replacement of the application field with an applicationId field in the Environment model
- Removal of the classification field from the domain model
- Addition of the rulesetType field to the Masking, Profiling, Reidentification, and Tokenization job models.
- Addition of mountName in the ConnectionInfo of a file connector and a mainframe dataset connector to use a filesystem mount point.

- For more information on Continuous Compliance APIs, see [API documentation \(see page 878\)](#).
- **NFS and CIFS mounts**
In previous releases, the Masking Engine has supported masking files via FTP or SFTP. In this release, we have added the ability for users to directly mount and mask a file system over NFS and CIFS. This should dramatically simplify the process of file masking. As with other Masking Engine objects, the Sync feature can be used to coordinate mount objects across multiple engines. For more information on the mount feature, see [Managing Remote Mounts \(see page 348\)](#).

3.1.39 Release 5.3

- **Synchronizing masking jobs and universal settings across Engines**
In 5.2 we introduced the ability to synchronize Masking Algorithms between engines to ensure consistent masking, regardless of the engine executing the masking. In 5.3 we are expanding the list of syncable objects to include:
 - Masking Jobs
 - Connectors
 - Rulesets
 - Domains
 - File Formats

The sync of objects is possible through improvements to several sync API endpoints, including:

- GET /syncable-objects[?object_type=
- POST /export
- POST /export-async
- POST /import
- POST/import-async

This expansion of syncable objects ensures that users can sync their Masking Jobs and all the objects necessary for that masking job to execute successfully - regardless of the masking engine it lives on, allowing for easier scaling of Continuous Compliance across the enterprise. For more information, see [Managing Multiple Masking Engines \(see page 842\)](#).

- **Support for Kerberized connections**
In 5.2.4 we added support for Kerberos for our Oracle Masking Connector. In 5.3 we have expanded the list of connectors that support Kerberos to:
 - SQL Server
 - Sybase
 - To enable Kerberized connectors your engine must be configured properly and you must configure your masking Connectors for Kerberos. Kerberos can be enabled by going to the Advanced mode on Oracle, SQL Server and Sybase. For more information, see [Managing Connectors \(see page 355\)](#).

- **New API endpoints**

In 5.2 we released an all-new set of API endpoints allowing for the automation of many masking workflows. In 5.3, we have expanded this list of API endpoints around Algorithms, Users, Roles, File Upload, System Information, Login, Rulesets, and Connector. Below are the net new API endpoints:

Group	Endpoints	Description
Algorithms	POST /algorithms	Create algorithm
	DELETE /algorithms/	Delete algorithm by name
	GET /algorithms/	Get algorithm by name
	PUT /algorithms/	Update algorithm by name
	PUT /algorithms/	Randomize key by name
Users	GET /users	Get all users
	POST /users	Create user
	DELETE /users/	Delete user by ID
	GET /users/	Get user by ID
	PUT /users/	Update user by ID
Roles	GET /roles	Get all roles

Group	Endpoints	Description
	POST /roles	Create role
	DELETE /roles/	Delete role by ID
	GET /roles/	Get role by ID
	PUT /roles/	Update role by ID
Rulesets	PUT /database-rulesets/	Update the rule set's tables
	PUT /database-rulesets/	Refresh the rule set
Connectors	POST /database-connectors/	Test a database connector
	POST /database-connectors/test	Test an unsaved database connector
	POST /file-connectors/	Test a file connector
	POST /file-connectors/test	Test an unsaved file connector
Async Tasks	GET /async-tasks	Get all asyncTasks
	GET /async-tasks/	Get asyncTask by ID
	PUT /async-tasks/	Cancel asyncTask by ID
File Upload/Download	DELETE /file-uploads	Delete all file uploads
	POST /file-uploads	Upload file
	GET /file-downloads/	Download file
System Information	GET /system-information	Get version, etc.

Group	Endpoints	Description
Login/Logout	PUT /logout	User logout
Executions	GET /execution-components	Status for a table, file, or Mainframe data set
Tokenization Job	GET /tokenization-jobs	Get all tokenization jobs
	POST /tokenization-jobs	Create tokenization job
	DELETE /tokenization-jobs/	Delete tokenization job by ID
	GET /tokenization-jobs/	Get tokenization job by ID
	PUT /tokenization-jobs/	Update tokenization job by ID
Re-identification Job	GET /reidentification-jobs	Get all re-identification jobs
	POST /reidentification-jobs	Create re-identification job
	DELETE /reidentification-jobs/	Delete re-identification job by ID
	GET /reidentification-jobs/	Get re-identification job by ID
	PUT /reidentification-jobs/	Update re-identification job by ID
Database Rulesets	PUT	Update Database Ruleset by ID

In addition to the net new API endpoints, we have improved pre-existing API endpoints. Some of the improvements include:

- Addition of DB2 iSeries and Mainframe to connector endpoints.
- Addition of Kerberos configuration on Oracle, SQL Server, and Sybase connectors
- Ability to have ruleset refresh drop tables
- Support for XML file types
- Addition of dataType to column metadata

- Addition of `isProfilerWritable` field to file-field-metadata endpoints. This is now represented in the API as a new `isProfilerWritable` boolean field in the body of a file-field-metadata. When the `isProfilerWritable` field is set to true, the algorithm/domain assignment on a column can be overwritten by the profiler. When the field is false, it may not be overwritten.
- Addition of `multipleProfilerCheck` field to Profile Job endpoints. This feature is turned on using the boolean field in the body of a profile job. The job profiler normally stops profiling a column as soon as it flags a field as sensitive. If `multipleProfilerCheck` is true, the profiler will continue to scan the column for additional sensitive patterns. In the event that it finds more than one pattern, it will tag all the data domains found and apply 'one' standard algorithm for all those domains. The standard algorithm is 'Null SL' as of 5.3.4.0. This feature was formerly called 'multi PHI'.

For more information on Continuous Compliance APIs, see [API documentation \(see page 878\)](#). Please note that the previous generation of Masking APIs (commonly referred to as V4) is EOL and no longer supported in this release. All users are encouraged to migrate to the V5 APIs.

3.2 Fixed issues

3.2.1 Release 26.0.0.0

Bug Number	Description
DLPX-69342	The job monitor and execution component API should no longer intermittently show incomplete or -1 row counts for tables after masking is complete.
DLPX-72404	Total row counts for database tables should no longer be shown as "Counting", even after masking jobs have completed.
DLPX-90379	Fixed an issue where copybook file formats overwrite copybooks with the same file name.
DLPX-90631	Fixed a race condition while running a job consecutively that lead to job update reverts and inconsistent job status in the Environment-Jobs UI page.
DLPX-91381	Fixed an issue where delimited file masking may add an extra character to a field when a multi-character delimiter is used.
DLPX-91429	Fixed an issue where incorrect connector details were sometimes shown in the Rule Set wizard.

DLPX-91524	Fixed an issue where masking primary key columns resulted in unexpected duplicate masked values.
DLPX-91740	Decreased the amount of time it takes for an Inventory CSV export.
DLPX-91755	Multiple AD domains can be specified for LDAP authentication using the <code>MsadDomain</code> application settings with comma-separated values.

3.2.2 Release 25.0.0.0

Bug Number	Description
DLPX-91087	Users can now see the suggestions while creating character groups for the Character Mapping algorithm framework.
DLPX-91303	The masking engine now supports masking columns with commas in their names, in PostgreSQL.
DLPX-91430	Fixed an issue where the Rule Set CSV import response message at the bottom of the sheet was closing automatically when errors/warnings occurred while importing.
DLPX-91443	Updated the masking report to display the breakdown of the total masked count by Success and Warning Job statuses.

3.2.3 Release 24.0.0.0

Bug Number	Description
DLPX-79997	Fixed an issue where audit logs were being flooded with the <code>Viewed all Execution Event</code> by replacing long polling with server-side events.
DLPX-90918	Fixed an issue where masking jobs failed for MSSQL database tables having a BIT data type column.
DLPX-91112	Users can now edit file patterns after adding them to a ruleset.

DLPX-91201	File masking jobs with empty body record types (no fields) will no longer be allowed to run; previously such configurations might leave data unmasked or drop rows from the output file.
DLPX-91215	The permission required to select a rule set and view the inventory have been documented.
DLPX-91222	Improved resiliency of background row counting to ensure that unexpected states cannot block row counts indefinitely.
DLPX-91240	Fixed an issue where modifying an extended algorithm was not working.
DLPX-91243	Fixed an issue where logical name and group number field did not show up on editing a ruleset property, if it was previously saved with 0 as group number.
DLPX-91331	Fixed an issue when a non-admin user added environments to another non-admin user, all environments that are not in common were removed.
DLPX-91363	Fixed an issue causing omission of non-conformant data tables in the masking report due to the addition of a new JOB status.

3.2.4 Release 23.0.0.0

Bug Number	Description
DLPX-78386	Fixed an issue where all frameworks did not appear under the framework dropdown in the Extended Algorithm UI.
DLPX-82977	Removed vulnerable JavaScript library, jQuery v1.12.4, by replacing legacy UI with a new SPA.
DLPX-82978	Removed vulnerable JavaScript library, jQuery-ui v1.12.1, by replacing legacy UI with a new SPA.
DLPX-87134	The framework dropdown in the Extended Algorithm UI during create/update algorithms UI now shows framework name with plugin name.
DLPX-87883	Fixed an issue where creating and updating algorithms did not provide any error message in the UI.

DLPX-88639	Fixed an issue that sometimes prevented the masking service from correctly restarting after an unsafe shutdown.
DLPX-88774	In-place Oracle masking jobs will no longer include unmasked primary key columns in the input SELECT query.
DLPX-90039	Fixed a database masking job failure issue when a Drop Index option is enabled for the job.
DLPX-90384	Updated the edit Rule Set wizard to have edit/delete all tables/files with filters.
DLPX-90687	Lack of execution queue slots for background table row counts no longer cause Rule Set refresh to fail.
DLPX-90693	Added EC (elliptic curve) certificate/private key authentication for password vaults.
DLPX-90952	Fixed an issue where job performance decreased if post-job emails failed to send.
DLPX-91008	Fixed a significant performance regression when masking VARCHAR or text fields across multiple database platforms
DLPX-91016	Fixed ASDD failure for XML files when some fields containing sensitive information are excluded from inventory.

3.2.5 Release 22.0.0.1

Bug Number	Description
DLPX-91029	Fixed a significant performance regression when masking varchar or text fields across multiple database platforms.

3.2.6 Release 22.0.0.0

Bug Number	Description
------------	-------------

DLPX-55643	On the Environments UI page, users can now see when a copy environment operation is in progress.
DLPX-56003	Fixed the performance of the Environments UI page when displaying a large number of environments.
DLPX-75217	Fixed an issue with larger environments or large settings exports, the file download popup in the GUI would appear to hang without showing any progress information.
DLPX-75795	Fixed the File Format listing to show the correct file-format IDs in the UI.
DLPX-86719	Fixed the Jobs UI to only show supported tasks for SAP ASE jobs (e.g., Drop Indexes).
DLPX-89299	Adds ASDD support for Yugabyte database.
DLPX-90170	The Job status icon that was shown in the Job Monitor UI if an execution event type of <code>FILE_PATTERN_NO_MATCH</code> is raised is now a warning icon.
DLPX-90225	Fixed an issue where masking a database to a file would fail if the file format had either header or trailer records.
DLPX-90506	Driver support plugin's tasks were updated to compensate for Db2 LUW internally padding trailing spaces (which are considered insignificant) to schema names shorter than 8 bytes, which prevents error.
DLPX-90728	Fixed an issue where too many running row count tasks in the background should no longer cause ruleset Create or Refresh operations to report a failure.

3.2.7 Release 21.0.0.0

Bug Number	Description
DLPX-69181	In the new UI, a Db2 connector's network port and database name are optional
DLPX-71997	Fixed an issue where the UI incorrectly indicated that synced NFS mount connectors were missing their password

DLPX-80803	With the ongoing UI improvements, the New Job page addresses issues related to scalability and performance of the grid.
DLPX-81579	A new execution event type <code>FILE_PATTERN_NO_MATCH</code> will be raised when a file name pattern does not have any match for XML file masking jobs.
DLPX-84106	Fixed an issue in the UI with scrolling a long jobs list using the Firefox browser
DLPX-84517	Fixed an issue in the UI that prevented users from configuring the source connector for an on-the-fly (OTF) MariaDB job.
DLPX-87773	Fixed an issue where the connector UI was not accepting hostnames with underscores.
DLPX-89316	Multicolumn algorithms can no longer be assigned to a domain for profiling and an error message is shown.
DLPX-89601	In the the rule set inventory UI, opening the dialogue box for a column or field now correctly shows the current algorithm assignment
DLPX-89668	Fixed an issue that caused the Drop Index feature to fail on a SAP ASE (Sybase) table with a NULL partitiontype.
DLPX-90075	Fixed a rare issue in the UI framework that caused startup of the Continuous Compliance service to fail

3.2.8 Release 20.0.0.0

Bug Number	Description
DLPX-82812	Added a notification to not use <code>\r\n</code> or <code>\n</code> for a custom end of record, <code>^t</code> or <code>\t</code> for a delimiter, and to use the appropriate CTRL characters.
DLPX-82813	The dialog box for entering a control characters as a delimited file's end-of-record or delimiter was redesigned to improve the usability.
DLPX-89175	Batch masking is now possible for delimited and fixed-width files with multiple record types.

DLPX-89208	File masking will no longer fail when a mix of wildcard patterns and named files are present in the ruleset.
DLPX-89227	Added an application setting for the database masking text field maximum size.
DLPX-89660	Fixed the cluster environment creation error when the cluster environment name is longer than 15 characters.
DLPX-89797	Locking that prevents engine sync import and export from occurring simultaneously has been removed.
DLPX-89803	Fixed an issue where storage space recovery may not happen right after a failover commit operation.

3.2.9 Release 19.0.0.0

Bug Number	Description
DLPX-85261	Fixed an issue related to GraphQL service stability, where the service was not recovering from a kill.
DLPX-89037	Updated Db2 license installation script to display helpful error messages when restarting the masking service fails.
DLPX-89205	Postgres SQL masking jobs no longer fail when an algorithm is assigned to a column with type INT8.

3.2.10 Release 18.0.0.0

Bug Number	Description
DLPX-86874	Fixed an issue that sometimes showed an incorrect timestamp in the UI.
DLPX-87707	In very rare cases, the Continuous Compliance Engine's storage can be filled with compliance jobs and inventory PDF reports. This issue might be encountered after generating hundreds of thousands of reports over multiple years.

DLPX-88335	Fixed an issue where rapid scrolling through the inventory tables in the presence of a sluggish API response could lead to an infinite loop of API calls, resulting in the grid becoming unresponsive and stuck in a perpetual loading state.
DLPX-88654	Fixed an issue where instead of 'NaN' for the Start Time column in the queued job grid, it now shows empty.
DLPX-89105	Fixed the dropping of views that exists in schemas other than the user's default schema
DLPX-89149	Fixed ASDD profiling 'CAST' errors with Db2 ZOS, LUW.

3.2.11 Release 17.0.0.0

Bug Number	Description
DLPX-66872	Fixed an internal server error caused by import of subtype connectors Maria DB, RDS Postgres, and Aurora Postgres.
DLPX-82740	Kerberos authentication for SQL Server advanced connector is now working as expected.
DLPX-87025	Fixed an issue with the algorithm plugin file not being found after a sync import fails.
DLPX-87291	ASDD profiling jobs now have an option to clear previous profiling-inventory and domain assignments before a new run of the job.
DLPX-87576	Fixed an issue with compliance job status reverting to Running after masking is completed, if the row count returns at the same time.
DLPX-88132	Fixed an issue for SQL Sever where jobs were failing if the column name contains double quotes.
DLPX-88220	Fixed an issue causing failures on import of large size sync bundles.
DLPX-88575	Fixed column CAST issues for Sybase database that occurred during ASDD profiling.

DLPX-88658	Fixed the error in Redact Digits-Zero masking algorithm, which also fails the dependent Lat_Long coordinates algorithm.
DLPX-83390	Improved SFTP protocol interoperability by adding support for the <code>ssh-ed25519</code> host key algorithm. This enables SFTP connections with Amazon Linux 2023 and other recent SFTP servers.

3.2.12 Release 16.0.0.0

Bug Number	Description
DLPX-52296	Fixed an issue where masking a MSSQL table without primary key column(s) using custom SQL fails.
DLPX-84493	Fixed an issue where an Oracle table with primary key column(s) fails if primary key column(s) are not included in custom SQL.
DLPX-87311	Fixed an issue related to uploaded copybook formats that have a group item with a picture node.
DLPX-87535	Fixed an issue where copy privileges for custom roles are not getting updated via masking UI and API.
DLPX-87800	Improved performance of saving profiling results for non-ASDD database profiling.
DLPX-87875	Improved performance of Mainframe dataset files with redefined conditions.
DLPX-88027	Fixed an issue where a masking job could override JDBC driver default fetch size, which caused memory issues for MySQL and MariaDB.
DLPX-88182	Removed the unused libwebp library in Containerized Masking.

3.2.13 Release 15.0.0.0

Bug Number	Description
------------	-------------

DLPX-86773	Improved performance of the Environment-Jobs overview UI page.
DLPX-87009	Fixed an issue regarding profile job errors due to duplicate billing periods.
DLPX-87054	Masking Teradata TIMESTAMP columns no longer fails when data has null values.
DLPX-87131	Added a new search and filter API for fetching executions.
DLPX-87183	Fixed an issue where an extended connector's uploaded JDBC driver passed an extra property named "URL".
DLPX-87185	Masking Teradata INTEGER and BIGINT columns no longer fails when data has null values.
DLPX-87349	Sync import of the Free Text Redaction algorithm will no longer fail when the lookup file name has spaces.
DLPX-87363	Fixed an issue in the Continuous Compliance UI where inventory or edit format did not have an option to filter by Masked Fields.
DLPX-87514	The presences of header and/or trailer record types will no longer cause algorithm batching to be disabled for delimited file masking.
DLPX-87604	The secure shuffle algorithm should now function normally when the Row Limit for a job is set to 0 (unlimited).
DLPX-87697	Fixed an issue where sorting the <code>monitorJobsDBCompleted</code> grid by 'Status/Logs' column did not sort the rows that have non-conforming data with the "success" status.
DLPX-49075 DLPX-87767	Improved performance of database inventory screen having a large number of tables and columns.

3.2.14 Release 14.0.0.0

Bug Number	Description
------------	-------------

DLPX-85469	For JSON Masking, Algorithm assignment to JSON multi-dimensional array field (\$ ['sample'][*][*] type of paths) and multi-column algorithm assignment are allowed when they are under the same parent level.
DLPX-86322	Columns that are set with ID method = USER are excluded from ASDD profiling, including data fetch.
DLPX-86877	Masking OTF job with Sybase and MSSQL database will no longer fail if the table name starts with a number and the table contains an identity column.
DLPX-86955	Email Unique algorithm no longer strips beginning or ending whitespace from input rows.
DLPX-87189	Fixed an issue for PostgreSQL where masking a large value in NUMERIC(0) column was causing precision loss.
DLPX-86865	Clarified that only database rule sets support the approval workflow feature.

3.2.15 Release 13.0.0.0

Bug Number	Description
DLPX-79868	Fixed an issue with Inventory export/import functionality to avoid CSV injection.
DLPX-82217	Fixed an issue where dlpX-core:FirstName would not mask as expected.
DLPX-82912	Fixed an issue showing HTML code instead of an actual success/error GIF when performing an inventory import from the UI.
DLPX-84783	Fixed an issue with permissions ambiguity when viewing Jobs logs vs Application logs. Now, a user who can view a job on the Monitor page will be able to view logs as well. In addition, a non-admin user with permission to view diagnostics will be able to view application logs.
DLPX-84875	Fixed an issue where a user with create/edit permissions was not able to create or edit the classifier.

DLPX-85564	Fixed an issue where non-admin user access was restricted to only the tasks they started (API methods: GET /async-tasks and GET /async-tasks/{asyncTaskId}).
DLPX-86134	Fixed an issue where a masking algorithm migration failure during engine upgrade causes the Algorithm UI to not load and the dlpX-core plugin to not upgrade.
DLPX-86254	Added Audit log entries for rulesets and jobs in case of the deletion of a related connector.
DLPX-86715	Secured GraphQL APIs by adding Content-Security-Policy headers and removing unsafe CPRS headers.
DLPX-86717	Fixed an issue where ASDD jobs on Maria DB will no longer fail with SQL exceptions due to incorrect identifier quoting.

3.2.16 Security Fixes

Bug Number	Introduced	Description	Security Bulletin
DLPX-86715	13.0.0.0	Secured GraphQL APIs by adding Content-Security-Policy headers and removing unsafe CORS headers.	TB110

3.2.17 Release 12.0.0.0

Bug Number	Description
DLPX-85600	Fixed the DROP INDEX feature for MySQL databases.
DLPX-85864	Fixed an issue related to uninformative masking progress whenever the row count is more than 2.1bn for SQL server.

Bug Number	Description
DLPX-8597 1	Fixed an issue where a masking job fails if column name contains special characters and custom SQL is used.
DLPX-8604 9	Fixed an issue related to the GraphQL service returning Http-502 in certain scenarios.
DLPX-8619 6	Fixed an issue where segment mapping algorithms exported using Engine Synchronization (from releases 6.0.15.0 -> 10.0) were not validated upon import to 11.0.
DLPX-8624 9	Updated protobuf-java dependency version to 3.23.2.
DLPX-8625 9	ASDD job execution will now correctly show execution event(s) in the UI when failures occur.

3.2.18 Security fixes

Bug Number	Introduced	Description	Security Bulletin
DLPX-86329	6.0.13.0	Sysadmin can execute shell commands on the underlying Operating System.	TB109

3.2.19 Release 11.0.0.0

Bug Number	Description
DLPX-85961	Fixed an issue where the Filter, Logical Key, and Custom sql for DB tables could not be updated when using the table name search feature inside DB rulesets.
DLPX-85867	The non-conformant data will no longer get copied to all the tables.
DLPX-85862	Fixed a CSS issue causing the Inventory UI to slide to the bottom of the page.

Bug Number	Description
DLPX-85743	Fixed missing tables in the UI-Job Monitor-Waiting grid page.
DLPX-85514	Fixed an issue in the ASDD profiler where the count of rows profiled was not displayed in the job monitor.
DLPX-85459	Fixed an issue that sometimes caused long delays before log messages related to jobs would show in the main logs.
DLPX-86112	ASDD profiling fails when multiple tables with similar names containing an underscore are present.
DLPX-82007	The deletion of a user-created mapping algorithm now deletes associated mappings from MDS.
DLPX-85289	Added changes for ensuring JDBC parameters are handled properly with DB2 mainframe connectors.

3.2.20 Security fixes

Bug Number	Introduced	Description	Security Bulletin
DLPX-85414	6.0.15.0	Unmasked Data or Masking Job Failure When Using a Migrated Version of the Segment Mapping Algorithm	TB108

3.2.21 Release 10.0.0.0

Bug Number	Description
DLPX-79335	Fix for Oracle databases masking job failing while masking XMLType columns, having masked data more than 2000 characters.
DLPX-84025	If a mapping algorithm is used on a field that contains only characters set to be ignored, a non-conforming data error will no longer be thrown.

Bug Number	Description
DLPX-84303	Phone Unique and Phone US algorithms no longer report non-conforming data as an error, since masking can continue and it can obscure true breaking errors.
DLPX-84874	Fixed an issue that causes long load times for the algorithm and domain settings screens.
DLPX-84966	Prevents masking job failures by adding extra validation when defining alphanumeric segments with Segment Mapping algorithms.
DLPX-85382	Fixed an issue with the ASDD profiler that can make data-level profiling take much longer than expected on large tables in a Microsoft SQL Server DB.
DLPX-85437	Fixed an issue that can make deletion of a database ruleset very time consuming when many profiler results exist.
DLPX-85961	

3.2.22 Release 9.0.0.0

Bug Number	Description
DLPX-82535	Added an application setting to change the default API page size for all GET APIs from 1 to 5000.
DLPX-84110	Fixed an issue where the drop index checkbox reset to an unchecked state after the masking dialog box is closed.
DLPX-84336	The search options on the Environment and Ruleset page now allow for "-" character matching.
DLPX-84488	Phone Unique and Phone US algorithms saw a failure on zero width and formatting characters included with a phone number in a database row. Phone algorithms can now pass over these non-numeric characters as expected.

3.2.23 Release 8.0.0.0

Bug Number	Description
DLPX-73334	Profiling jobs in tokenization environments where domains lack a tokenization algorithm assignment no longer fail.
DLPX-82175	Fixed an issue where editing a DB ruleset was taking a long time to modify the logical key of a table with a massive amount of data, by moving the validation from the UI to a backend check for null values in a conditional column.
DLPX-83340	Encryption algorithm upgrade for Engine sync bundles.
DLPX-83536	The city column level profiling expression has been updated to not match "address".
DLPX-84046	Added a flag 'trimWhitespaceFromInput' to the Secure Lookup algorithm that can be used to restore pre-6.0.12.0 trim behavior.
DLPX-84081	Fixed an issue where the Continuous Compliance Engine environment revisionHash changes on profile job execution, even though there was no change in inventory.
DLPX-84230	Fixed an issue where support for moving data from DB tables to delimited files was inoperable.
DLPX-84463	Swagger-UI minor version upgrade for bug fixes done in library.
DLPX-84525	Inventory page fails to load on Continuous Compliance Engine when there are more than 32,767 masked columns on the engine.

3.2.24 Release 7.0.0.0

Bug Number	Description
DLPX-76693	Added an execution-event and a log error message showing the missing column names, when custom SQL is used in the ruleset and some columns are not specified in the query.

Bug Number	Description
DLPX-79530	Continuous Compliance Engine environment revisionHash changes frequently due to execution of masking and tokenization jobs.
DLPX-80124	Column name is checked if it contains a space, if a true rename of the column name is set to false, so that the column name is not renamed. Note, this change is only for column names having whitespace and not for columns having special characters.
DLPX-81503	Fixed an issue when wrong environment link was getting generated on Monitor page.
DLPX-82950	Fixed an issue where Filter By in the Inventory page on the UI was intermittently working for File inventories.
DLPX-83035	In case of Job failures, Error details will be displayed in Status/Logs dialog on the Execution details page.
DLPX-83566	Fixed an issue when connection to NFS mount failed after the NFS server was restarted.
DLPX-83593	Fixed an issue where unnecessary locking can cause slowdowns and possibly deadlocks.
DLPX-83659	An update was made to the DB2 license upload script to make it compatible with all Continuous Compliance versions 6.0.14.0 and later.
DLPX-83804	Fixed an issue where assigning Dataset File Formats using the API was not working for some formats.
DLPX-83809	Fixed an issue causing random job failures when masking SQL Server tables that have columns of date/time data type as part of the PK.
DLPX-83817	Fixed an issue where non-admin users can submit a inventory change for approval workflow without approval inventory permission.
DLPX-83930	Fixed an issue that caused some environment sync import operations to fail with PersistentObjectException.

3.2.25 Release 6.0.17.0

Bug Number	Description
DLPX-46230	Fixed an issue where tables with a Primary Key column of datatype RAW would cause an error when selected for masking.
DLPX-55224	Search using wildcards and substrings is now allowed on search boxes across multiple pages.
DLPX-69096	Email addresses are case sensitive when SSO is enabled.
DLPX-69501	User defined domains cannot be deleted when assigned to profiler expressions.
DLPX-76315	Display of field level redefine fields have been applied in the Inventory screen.
DLPX-76696	Audit logs being displayed is not limited to 1000 entries, all the logs recorded will be displayed with a pagination of 50 rows per page. API of audit logs is modified to include all search and filter parameters used in UI.
DLPX-77069	Fixed an issue where using regex for File Name Patterns in xmlfile Rule Sets would succeed but give an error.
DLPX-80496	Removed extraneous link to the job's execution log from the monitor screen for profiling jobs.
DLPX-80539	Fixed an issue where some failures detected during file masking job generation are not correctly reported via an execution event.
DLPX-80540	Fixed an issue where some failures during file masking job generation leave a job in running status indefinitely.
DLPX-80984	Fixed negative row counts when custom SQL includes column name that contains SQL reserved word.
DLPX-81311	A generic DB error message will now be shown to users for fetching and creating rule set.
DLPX-81725	Fixed a bug that could cause the masking engine to run out of memory processing results from large jobs.

Bug Number	Description
DLPX-82064	Fixed MSSQL masking performance issue when using Kerberos authentication and masked columns that are unicode, but primary keys are non-unicode.
DLPX-82289	Fix provided to support backslashes and other special characters that might be required to form a valid regular expression.
DLPX-82310	User's first and last names are now redacted in the support bundle.
DLPX-82579	Improved performance to prevent GUI lag when navigating across pages.
DLPX-82864	Error message for invalid LDAP authentication attempt has been updated to prevent username harvesting.
DLPX-82925	Fixed an issue where the Profile Results screen breaks due to a database error.
DLPX-83026	Fixed an issue causing the DateShiftDiscrete algorithm to not be assigned a new random key when an engine is deployed.
DLPX-83086	Upgraded Swagger UI to enhance the security and usability of the swagger API Client.
DLPX-83200	Fixed scale issues related to storing job logs in the product's internal database.
DLPX-83232	Added an option to the Secure Lookup algorithm to disable whitespace cleanup when the lookup file is loaded.
DLPX-83354	Updated the profile expression 'Full_Name_V2' to reduce the number of false positive results. This change only applies to newly deployed Continuous Compliance Engines.
DLPX-83427	Fixed Directory Travel vulnerability for the export inventory UI.
DLPX-83431	Upgraded Apache Commons to 1.10.0.
DLPX-83576	Fixed an issue where the Support bundle process did not collect the /etc/hotfix file.

3.2.26 Release 6.0.16.0

Bug Number	Description
DLPX-49116	Data truncation when masking CHAR(n) using Segment Mapping to mask short numeric segments
DLPX-73539	Pdf for Audit log does not contain Status column
DLPX-77777	View only connector privilege user will be able to see connection details and test connection using Masking UI
DLPX-77929	Algorithm API: Add mask_type attribute in the Api response for component type algorithm.
DLPX-78474	Able to validate algorithm from: Algorithms > edit extended instance > "validate configuration" button
DLPX-79358	Monitor page UI Start and End Date Filters do not work as expected
DLPX-79607	Show a better user friendly error message when creating full name algorithm with invalid input
DLPX-79743	User will not be logged out, after performing an operation for which they have insufficient privilege. API will be throwing 403 status code, instead of 401.
DLPX-80304	isIdentity has been added to the GET /column-metadata API response that indicates whether the table column is an identity column
DLPX-80668	For Job execution steps, icon for last event step "Job completed" will be aligned to final status of the job. (1) If job is success - Green tick icon (2) If job has failure - Red failed icon, (3) If job was cancelled - Red cancelled icon
DLPX-80784	Addition of new application setting group to drive strict content security policy
DLPX-81058	Saving File/Copybook field properties on Inventory screen will now retain the ruleset selected
DLPX-81422	Updated Job Monitor page with small UI fixes. Page position will remain same after page refresh. Renamed job execution events.

Bug Number	Description
DLPX-81730	Segment Mapping pads short numeric input even when not masked
DLPX-81807	Fixed an issue that can very rarely cause the masking service to fail with a stack overflow during startup
DLPX-81895	Data profiling results are not shown under job -> monitor page -> results tab for delimited, fixed and XML files
DLPX-81897	Internal error when GET syncable-objects API is called with object_type=LOOKUP
DLPX-81935	Profiling regex with escaped double-quotes fails in compiling javascript
DLPX-81946	Allow masking of dates containing month values in all-upper character case
DLPX-82025	Setting pseudo column as ROWID in the Logical Key of the Ruleset is possible
DLPX-82057	GET /file-uploads API endpoint returns a 500 error when plugins exist on the engine
DLPX-82086	From this change, all the timestamps on Masking UI will be as per user timezone. Logs, search filter on Job monitor and Audit page will be in UTC Timezone. User won't be able to change timezone on UI. It will be by default as per their browser timezone.
DLPX-82166	Segment Mapping v2 UI cannot specify SPACE as an ignore character
DLPX-82186	Clicking outside connector dialog will not dismiss the dialog
DLPX-82337	Segment Mapping algorithm with CONSTANT segment should not be allowed in Tokenization job
DLPX-82355	Fixes an internal server error that occurred when attempting to import a legacy Secure Lookup algorithm of type LOOKUP.
DLPX-82491	From this change, User can hover on the time displayed on UI and will be able to identify timezone offset related information.
DLPX-82627	Fix an incorrect check preventing the Segment Mapping Algorithm from running in REIDENTIFY mode

3.2.27 Release 6.0.15.0

Bug Number	Description
DLPX-77088	Fixed an issue causing the Full Name Algorithm to fail masking when a single non-alphanumeric char is present, one of the input words is in the particles file, and the 'Last-First-Middle' convention is used.
DLPX-79547	Fixed an issue causing: MSSQL Masking Job XmlArtistFailureException: Input has a cycle and cannot be used with XML Artist.
DLPX-80123	Non-conforming Data is now reported when a mapping algorithm's available mappings are exhausted.
DLPX-80225	A new interface has been developed in the masking SDK for SingleOperationTask.
DLPX-80603	Fixed an issue causing read-only multi-column fields to be upgraded incorrectly.
DLPX-80604	Fixed an issue where records are truncated if the field length is not '0' in the fixed width file format, and 'whole file masking' selected.
DLPX-80732	Fixed incorrect text in an error message that occurs while exporting a profile_typed_expression.
DLPX-80788	Added a check to validate same site cookies transfer.
DLPX-80837	Fixed an issue where a copybook with a file name more than 24 characters fails loading: "String field too long".
DLPX-80842	Fixed an issue where the "All Fields" button in database inventory unexpectedly refreshes inventory for the displayed table.
DLPX-80922	Changed the version for the extensibility API to 1.10.0.
DLPX-81067	Cleaned up some stale build properties.
DLPX-81110	Fixed an issue where the filter on the Job Wizard Inventory screen was missing.

Bug Number	Description
DLPX-81128	Fixed an issue of being unable to create/edit a profile set from the GUI.
DLPX-81175	Fixed an issue with Tokenization v2 instances where character mapping fallback and the same character groups produced the same results within a job, and inconsistent results across jobs.
DLPX-81259	Removed the 256 character limit on masking copybook redefine conditions.
DLPX-81261	Fixed an issue where masking VSAM with different Algorithms causes: IndexOutOfBoundsException: Index: 0, Size: 0 .
DLPX-81330	Permanent file upload is now available for files that are not explicitly associated with a JDBC driver, algorithm, driver support plugin, or connection properties.
DLPX-81397	Fixed an issue where the JOB ID was not displayed in Job Monitor Processing or Waiting tabs.
DLPX-81512	Fixed an issue where sync bundles from releases prior to 6.0.15.0 could not be imported into 6.0.15+ releases.
DLPX-81666	Fixed an issue where sync exports of environments/jobs/connectors using non-default driver support settings were unusable. This happened due to failures of jobs/edits on the system where they are imported, where the non-default driver support settings are used.

3.2.28 Release 6.0.14.0

Bug Number	Description
DLPX-56200	Stopped execution.job.jobId logs from spamming the log files.
DLPX-77999	The Legal disclaimer and description of the framework on the Algorithm GUI now appear.
DLPX-78560	The issue with sync compatibility for multiple headers and footers is now fixed.
DLPX-78671	Added pagination for Syncable objects with the object_type LOOKUP.

Bug Number	Description
DLPX-78850	Fixed an issue where the mainframe file format delete fails with NoSuchFileException if the format file is not present on disk.
DLPX-79160	Fixed an issue that could cause sync import to fail due to inconsistent multi-column algorithm assignments in the sync document.
DLPX-79472	Fixed an issue that prevented the saving of multi-column algorithm masking assignments in file inventories.
DLPX-79608	Fixed an issue causing SLv2 to fail when the lookup file contains just spaces.
DLPX-79720	Fixed an issue where DriverSupport logs stopped printing after switching to the next log file due to file size limit.
DLPX-79865	Improvements have been made to the Forgotten Password API error message.
DLPX-79966	PostgreSQL driver updated from 42.2.23 to 42.3.2 version.
DLPX-79986	The expression_name field in the profile-type-expression endpoint has been renamed type_expression_name.
DLPX-80386	Column and data level profiler expressions are now tested in a predictable order - alphabetically, by expression name.
DLPX-80411	Upgraded Spring framework version from 5.2.5 to 5.2.20.
DLPX-80078	The issue while removing files with complex file permissions on EBS is now fixed.
DLPX-81071	HTML escape the inventory notes field.
DLPX-81082	Addresses the issue described in TB098 ⁶⁰ .

⁶⁰ https://support.delphix.com/Support_Policies_and_Technical_Bulletins/Technical_Bulletins/TB098_Arbitrary_Code_Execution_May_Be_Performed_When_Configuring_Masking_Environments

3.2.29 Release 6.0.13.0



This release renames Delphix Masking to Continuous Compliance.

Bug Number	Description
DLPX-77075	The issue with masking MSSQL date field that was causing the error "Conversion failed when converting date and/or time from character string" is now fixed.
DLPX-78363	This release adds support for API to get execution logs.
DLPX-78366	This release adds the new API endpoints to get execution component logs.
DLPX-78472	This release adds the total job time to the Masking job reports.
DLPX-78755	The issue with the failure of async export (if data is huge) with the OOM(Requested array size exceeds VM limit)error message is now fixed.
DLPX-78948	Previously, the edit user dialog was not opening for non-admin users if SSO is enabled. This issue is now resolved.
DLPX-79152	Extended algorithm enclosure handling was throwing an NPE when there are too few fields in a delimited file. This issue is now fixed.
DLPX-79177	The mapping Algorithm was failing in some cases(such as delimited files and BigInt Column) with error Mapping output value exceeding maximum value length of 0 characters. This issue is now fixed.
DLPX-79567	Previously drop index was failing if an index with the same name existed on the masked columns across multiple tables for MSSQL databases. This issue is now fixed.
DLPX-79627	The issue with the failure of the Masking SQL Server when special characters(such as] [or ') are used in the table column is now fixed.
DLPX-79632	The issue with the failure of the Masking Oracle DB when special characters are used in the table or column name is now fixed.

Bug Number	Description
DLPX-79803	The issue with the failure of the Masking with MSSQL database if table name contains '[' is now fixed.
DLPX-79804	The issue with the failure of the Masking with MSSQL database if table name contains '\' is now fixed.

3.2.30 Release 6.0.12.0

3.2.30.1 Log4j updates

Based on detailed testing and analysis, all the currently supported products are not susceptible to known log4j vulnerabilities. Please refer to [TB095 Technical Bulletin](#)⁶¹ for more information. All instances of log4j in currently supported Delphix products are updated to **log4j 2.17.1** as of this release.

Delphix keeps you updated on the latest developments and keeps releasing hotfixes, procedures, and workarounds for such critical vulnerabilities. For more information on how Delphix supports our product and customers in such cases, see [Delphix Product Security](#).⁶²

For more information, refer to the following pages:

- [TB095 log4j Vulnerabilities](#)⁶³
- [Uninstalling the Delphix Connector Service from the Target Database Servers](#)⁶⁴
- [Delphix Product Lifecycle Policies](#)⁶⁵
- [Product Security](#)⁶⁶

⁶¹ [https://support.delphix.com/Support_Policies_and_Technical_Bulletins/Technical_Bulletins/TB095_Log4j_Vulnerabilities_\(CVE-2021-44228%2C_CVE-2021-45046%2C_CVE-2021-45105%2C_CVE-2019-17571%2C_CVE-2021-4104\)](https://support.delphix.com/Support_Policies_and_Technical_Bulletins/Technical_Bulletins/TB095_Log4j_Vulnerabilities_(CVE-2021-44228%2C_CVE-2021-45046%2C_CVE-2021-45105%2C_CVE-2019-17571%2C_CVE-2021-4104))

⁶² <https://delphixdocs.atlassian.net/wiki/spaces/CD/pages/4227213/Delphix+product+security>

⁶³ [https://support.delphix.com/Support_Policies_and_Technical_Bulletins/Technical_Bulletins/TB095_Log4j_Vulnerabilities_\(CVE-2021-44228%2C_CVE-2021-45046%2C_CVE-2021-45105%2C_CVE-2019-17571%2C_CVE-2021-4104\)](https://support.delphix.com/Support_Policies_and_Technical_Bulletins/Technical_Bulletins/TB095_Log4j_Vulnerabilities_(CVE-2021-44228%2C_CVE-2021-45046%2C_CVE-2021-45105%2C_CVE-2019-17571%2C_CVE-2021-4104))

⁶⁴ <https://delphixdocs.atlassian.net/wiki/spaces/CD/pages/4819106/Uninstalling+the+Delphix+connector+service+from+the+target+database+servers>

⁶⁵ [https://support.delphix.com/Support_Policies_and_Technical_Bulletins/Support_Policies/Product_Lifecycle_Policies_\(KBA1003\)](https://support.delphix.com/Support_Policies_and_Technical_Bulletins/Support_Policies/Product_Lifecycle_Policies_(KBA1003))

⁶⁶ <https://delphixdocs.atlassian.net/wiki/spaces/CD/pages/4227974/Product+security>

3.2.30.2 Fixed Issues

Bug Number	Description
DLPX-48506	The issue with the VSAM masking job failing with an error message, "Multiple entries with the same key: FILLER" is now fixed.
DLPX-64060	For the "Define Fields" popup in File Inventory, the previously saved algorithm is now displayed as selected. If domain and algorithm were not assigned, then selecting a domain will not select the respective default algorithm in the algorithm field.
DLPX-67419	The issue with the generation of the Generic Security Services API exception when performing data-level profiling on a Kerberized database is now fixed.
DLPX-69263	The issue with the failure of masking Hana DB using an extended connector when binary columns are masked or present for OTF jobs is now fixed.
DLPX-75726	The issue with the clearing of the file format configurations when modifying the file masking pattern is now fixed.
DLPX-76752	Time format now includes seconds on the Monitor page for a better user experience.
DLPX-77036	The issue with setting Null for owner_id on referenced objects when deleting a user and resulting in NPEs is now fixed.
DLPX-77145	The issue with being unable to run any jobs - NPE in getTotalXmxOfRunningExecutions is now fixed.
DLPX-77166	Extended algorithms that support tokenization are now available to assign as the tokenization algorithm in domains.
DLPX-77233	PostgreSQL JDBC driver is upgraded to version 42.2.23.
DLPX-77258	This release fixes a bug in Data Level profiling when the specified schema is not the user's default schema.
DLPX-77401	The issue with not being able to extract the unmasked fields using API is now fixed.

Bug Number	Description
DLPX-77502	This release now adds an end-point (POST) for file-field-metadata API.
DLPX-77503	Inventory GUI now uses a POST API end-point.
DLPX-77506	The issue with the failure of Data level profiling if the EnableDataLevelCount application is set to True is now fixed.
DLPX-77521	The masking engine now bars multiple headers and trailers for the record type.
DLPX-77524	This release adds filters to the table-metadata API.
DLPX-77594	The issue with a regular user not being able to submit an inventory change is now fixed.
DLPX-77629	This release changes the field labels from 'Prescript' and 'Postscript' to 'Pre SQL Script' and 'Post SQL Script' respectively in the Masking Job UI.
DLPX-77636	Job execution API now provides a job status filter to enhance the user experience.
DLPX-77688	The Character Mapping Algorithm's non-editable preserve range when editing the algorithm is now fixed.
DLPX-77718	Users will now be able to associate a new parameter 'Whole File Masking' for any files listed on the Fixed File Rule Set page.
DLPX-77720	The issue with the displaying of an error message, "java.lang.NumberFormatException" when using Save & View option during the environment copy operation is now fixed.
DLPX-77767	Previously, the Delphix Masking engine used the incorrect HTML response code of 400 (Bad Request) for objects that could not be manipulated because they were currently in use. This release changes that to code 409 (Conflict).
DLPX-77786	This release blocks the creation of multiple header/trailer record types.
DLPX-77869	The issue where DESC order indexes were not being dropped and re-created as part of the Oracle Drop Indexes task has now been resolved. Functional indexes, including DESC order indexes, are now dropped and re-created on Oracle tables that contain any masked columns.

Bug Number	Description
DLPX-77931	This release adds a translator to support Backward compatibility for PUT /file-field-metadata/
DLPX-77962	Users will now be unable to update fields like position and length for a fixed-width file if the 'Whole File Masking' feature is enabled.
DLPX-77963	For any Fixed-Width file, if the 'Whole file masking' option is selected, then Kettle Reads the complete content of the file and passes it as one single record to the configured algorithm.
DLPX-77976	This release replaces all the "NULL" values for the user_id column of the algorithm table by the ID of a Delphix internal user called 'deleted-user'.
DLPX-78105	Users will now see a proper error message when creating/updating the mainframe field if the provided date format is invalid.
DLPX-78116	This release adds an 'istokenizationSupported' flag in the Algorithm API response.
DLPX-78161	Created a function that uploads files bypassing tomcat's /tmp directory.
DLPX-78422	The issue with the logical key not being added to the table in Rule Set via GUI if the user is not the schema owner is now fixed.
DLPX-78615	The issue with masking job throwing an exception while logging certain messages from plugin algorithms or driver support modules (This issue resulted in job deadlock during cleanup) is now fixed.
DLPX-78680	This release performs a clean-up of an obsolete lookup file attachment after making the import of an FTR-v2 algorithm.
DLPX-78740	The issue with the changing of an algorithm key when making the import of an FTR-v2 algorithm is now fixed. This release keeps the algorithm key unchanged.
DLPX-78743	This release updates all the masking dependencies on the Apache log4j library to version 2.15.0.
DLPX-78864	This release updates Log4j to version 2.0.17.
DLPX-78943	This release updates the log4j version to 2.17.1.

3.2.31 Release 6.0.11.0

Bug Number	Description
DLPX-55595	The issue where the Edit job dialog closes and leaves the screen greyed out with no errors while jobs are running has now been resolved.
DLPX-55595	The issue where the Edit job dialog closes and leaves the screen greyed out with no errors while jobs are running has now been resolved.
DLPX-65971	The issue where Cancel Masking job fails with "Execution status must be RUNNING, but is SUCCEEDED" has now been resolved.
DLPX-67558	The issue where a Masking job appears to hang when masked columns are unicode, but the primary keys are non-unicode, has now been resolved.
DLPX-69778	SAML response should no longer be logged on successful SSO login.
DLPX-70104	Enhanced the date format validation for file-field-metadata and mainframe-dataset-field-metadata API.
DLPX-70499	The Monitor Page will now show an informative message if no jobs are returned.
DLPX-72196	The issue when editing column properties for a file based inventory with no value selected for the ID Method field causing no validation to show has now been resolved.
DLPX-73326	There was an issue where when copying an environment, a dialog box shows a message that passwords will not be saved for connectors, but in the copied environment, the password information is present and Test Connection succeeds without any change. This issue has now been resolved.
DLPX-74245	The issue with inconsistent deletion behavior for a referenced database, file, and dataset connectors has now been resolved.
DLPX-74745	The <code>DEFAULT_MULTIPHI_ALGORITHM</code> application setting has been renamed to <code>DEFAULT_MULTIPLE_PROFILER_EXPRESSION_ALGORITHM</code> .
DLPX-75948	The issue showing inconsistent breadcrumbs for the VSAM/Mainframe Inventory screen has now been resolved.

Bug Number	Description
DLPX-76365	The issue where the Trans Level Info table grows without bound has now been resolved.
DLPX-76574	The issue causing a failure to retrieve the ERROR or Warning column type has now been resolved.
DLPX-76678	Added validation to disallow null values in the logical key columns at the time of create or update.
DLPX-76707	The issue where update algorithm shows an error with, "installed by the plugin [plugin name], cannot be modified independently" has now been resolved.
DLPX-76847	The issue where Masking PK on Oracle adds ROWID to SELECT but uses PK in UPDATE has now been resolved.
DLPX-76931	The issue where the Masking UI strips extra characters from connector hostname when hostname exceeds max character limit has now been resolved.
DLPX-77056	The ruleset deletion validation message has been updated.
DLPX-77075	The issue where masking an MSSQL date field caused the error, "conversion failed when converting date and/or time from character string" has now been resolved.
DLPX-77103	The issue where mixing extensible algorithms and mapplets in a VSAM jobs causes the job to crash has now been resolved.
DLPX-77138	The issue where the use of Carriage return \r breaks the inventory page when used in mainframe redefine condition has now been resolved.
DLPX-77139	The issue where V2021_04_05_2__fix_algorithm_plugin_metadata migration may fail with a "FileNotFoundException" exception has now been resolved.
DLPX-77159	The issue with VSAM Unmasked fields being truncated when redefines are present and an algorithm returns non-null results for null input has now been resolved.
DLPX-77267	The issue where an XML masking job can hang when GSSAPIAuthentication is enabled on the sftp server has now been resolved.

Bug Number	Description
DLPX-77542	The issue where an extended connector SQL count fails when the column name contains the word 'FROM_DATA' in custom SQL has now been resolved.
DLPX-77544	The issue where deleting a masking user causes the deletion of the masking users' objects (meaning potential loss of important information, including historical information) has now been resolved.
DLPX-77710	The issue with a missing index on an Oracle DB after a successful masking job run has now been resolved.

3.2.32 Release 6.0.10.0

Bug Number	Description
DLPX-59886	You can now set a timeout for the FTP connections.
DLPX-70680	The issue with the increasing of the JobLogs without bounds has now been resolved.
DLPX-71259	Masking Oracle LONG RAW length is now set to 0 characters.
DLPX-71993	The need for the 'Repository' on the Masking Monitor page is now removed.
DLPX-73059	The issue with the Masking Engine throwing the 'Unsupported Property Error' in application logs for properties that differ in the case from the actual properties' has now been resolved.
DLPX-74740	Masking File Format Import error now shows the list of invalid special characters present in the file name.
DLPX-74760	The issue with the failure of the POST /import with "Unknown document version UNRECOGNIZED" when the source engine version is newer than the destination engine version has now been resolved.
DLPX-75441	The issue with maskedObjectName not populating the execution events when masking files have now been resolved.

Bug Number	Description
DLPX-75487	The issue with DMS_ROW_ID as a column name in the Masking Rule Set causing jobs to fail has now been fixed.
DLPX-75712	The "About" page now lists the correct patent number.
DLPX-75868	The issue with the DataLevel Profiling resulting in an abort with "TypeError: Cannot find function getInteger in object false" has now been resolved.
DLPX-76009	The issue with the failure of the 'File format id greater than a specific number' when trying to update the file format ruleset via the API only has now been resolved.
DLPX-76063	The issue with the failure of the DateShift Algorithm when masking the VSAM (Mainframe) numeric data type has now been resolved.
DLPX-76068	Masking now allows passwords that are longer than 12 characters.
DLPX-76134	The issue with the Welcome screen displaying "User can launch 'Create Job' wizard" when they are not able to have now been resolved.
DLPX-76352	Delimited File masking no longer truncates white-space only fields.
DLPX-76405	Multi-column algorithms now display a better error message when logical fields are missing.
DLPX-76428	For masking operation, the Advanced Oracle Connector now rounds decimal numbers to integers.
DLPX-76450	The Payment Card framework UI now permits configuring minimumMaskedPositions to 0.
DLPX-76493	The issue with the MSSQL instance name property not being passed by default when connecting has now been resolved.
DLPX-76541	The issue with the file masking job failure using a pattern with a Windows-based FTP server has now been resolved.
DLPX-76566	The issue with the profiling Job failure with the 'Couldn't get row from result set' error due to conversion unsupported has now been resolved.

Bug Number	Description
DLPX-76608	The plugin's authorization to delete files in the temp directory is now granted.
DLPX-76610	The issue with the IP SFTP Masking failure to delete the file has now been resolved.
DLPX-76670	The issue with the masking Job failure with the 'Conversion failed from string to uniqueidentifier data type' error has now been resolved.
DLPX-76821	The issue with the throwing of JSchException for pattern-based SFTP masking with file count > 10 has now been resolved.

3.2.33 Release 6.0.9.0

Bug Number	Description
DLPX-57961	Inventory export fails silently when a dataFile has fileFormats = NULL.
DLPX-64329	v5 API: Create an endpoint to copy environment objects in the same/different environment.
DLPX-68807	DateShift algorithm example should exclude invalid entries in the UI pop-up.
DLPX-69728	The active CIFS/NFS mount is getting disconnected after the upgrade.
DLPX-72383	Masking job hangs due to "Unable to acquire lock for job removal before timeout."
DLPX-73344	Internal server error when importing invalid delimited or fixed-width file format.
DLPX-74409	Masking Engine: Upgrade slf4j-ext-1.7.25.jar to slf4j-ext-1.7.30.jar.
DLPX-74415	Masking Engine: Upgrade Guava version to 30.1-jre.
DLPX-74882	Masking's SFTP client no longer compatible with SolarWinds and Goanyware SFTP servers.

Bug Number	Description
DLPX-74913	Inventory exports do not include the notes field.
DLPX-74941	Create a sync state on export for syncable objects that have null sync states.
DLPX-75005	Importing the COMPONENT type algorithm does not change the sync state object type.
DLPX-75202	Batch Masking and Failed kettle jobs may fail to terminate.
DLPX-75235	Secure lookup GUI: Add support to specify remote file URI.
DLPX-75244	Extensible driver test fails "Parameter 'directory' is not a directory" for the removed driver.
DLPX-75296	Sync import fails for an object having files with space in the filename.
DLPX-75307	Multi-column algorithm assignment details are missing from CSV inventory export.
DLPX-75308	SQLFeatureNotSupportException method not supported ...getSchema().
DLPX-75311	Debug message with %s logged when using Regex Decomposition Algorithm.
DLPX-75437	LastNameSeparator text box is not disabled for default dlp-core:FullName algorithm.
DLPX-75440	XML masking job fails with "Sequencer step still had unwritten rows!".
DLPX-75468	Upgrade MySQL driver org.mariadb.jdbc:mariadb-java-client from 2.4.1 to latest available version 2.7.2.
DLPX-75516	Updated Masking Web API version to 5.1.9.
DLPX-75520	Fixed an issue that could cause XML masking jobs to stall or fail with the error "Sequencer step still had unwritten rows!".
DLPX-75644	Added new "UserDirIsRoot" flag to the SFTP type connector.

Bug Number	Description
DLPX-75768	Row limiter can still deadlock jobs in some failure cases.
DLPX-76267	Sync export fails with insufficient memory available in JVM error.

3.2.34 Release 6.0.8.0

Bug Number	Description
DLPX-66147	Environment errors occur after deleting a referenced Mainframe connector.
DLPX-71318	Transformation - SQL check for CREATE and DROP IDENTITY Column is not using Schema.
DLPX-71489	Masking plugin API does not include the plugin author from Jar metadata.
DLPX-72581	Masking usernames and emails not redacted in support bundles.
DLPX-72653	Masking Job "Row Limit" UI shows 20 to be the lowest limit - This has been fixed to reflect 100 as the lowest.
DLPX-73207	Table name for MSSQL with single quote appears incorrectly on inventory page.
DLPX-73328	Incorrect tooltip text displayed for Admin link in footer.
DLPX-74152	Unable to edit ruleset from UI after adding tab (4 space) as an "End Of Record" in file ruleset.
DLPX-74190	Sync import of global settings fails with NullPointerException in an extended algorithms tearDown method.
DLPX-74426	PostgreSQL driver got updated from 42.2.10 to 42.2.19 version.
DLPX-74612	Oracle Masking Job fail with FanManager - unable to create ONS subscriber.
DLPX-74638	Bad example format in Date Algorithm GUI.

Bug Number	Description
DLPX-74844	Algorithm UI breaks with JSON special characters in the algorithm extension JSON.
DLPX-74849	Adding a new field to a record type via the GUI incorrectly always sets the field to be masked.
DLPX-74875	Importing pre/post script into the same environment with the same file name and job name deletes the file.
DLPX-74881	Certain algorithm plugins causes minor breakage in Algorithm Settings Screen.
DLPX-74967	New Date Shift algorithms do not allow for any time zone specifiers in the date format.
DLPX-74974	InvalidKeyException "No installed provider supports this key: (null)".
DLPX-74990	Specifying Backspace character("\b") as enclosure for delimited files via API does not throw an error, but crashes UI.
DLPX-75246	Mask Value Range for Segment Mapping (legacy) not getting saved from GUI.
DLPX-75290	Cannot use MSSQL or JTDS driver in SDK as extensible framework.

3.2.35 Release 6.0.7.0

Bug Number	Description
DLPX-45399	Improve masking test connector errors.
DLPX-57910	Control character field delimiters are replaced incorrectly in delimited file masking.
DLPX-67246	The UI and the API should have the possibility to LOCK a user account.
DLPX-70837	Update MDS "All Privileges" role to have correct privileges.

Bug Number	Description
DLPX-70844	End of Record options for file masking is misleading.
DLPX-70885	Masking API to submit update password request with forgot password token.
DLPX-71125	Masking Bundle generation is very slow.
DLPX-72036	UI sync operations initiate but fail; no evidence in MDS or logs.
DLPX-72121	Algorithm description field limit on UI should be same as new API limit i.e 8192.
DLPX-72424	String masking algorithm results in null values when masking oracle LONG(0) columns.
DLPX-72501	Regression in delimited file allowed Delimiters.
DLPX-72509	DateShift cast of DATE to DATETIME is not range cognizant.
DLPX-72551	FreeTextRedactionExtension translator does not properly set profileSetId when API version is v5.1.3 or less.
DLPX-72731	Incorrect handling end-of-record (EOR) character embedded in an enclosure.
DLPX-72734	The plugin VIEW privilege is no longer required to add, update, or delete a plugin.
DLPX-72878	Migration V2019.04.11.0 wrongly assumes role with role_id==1 always present.
DLPX-72879	Extensible algorithm numeric to string conversion is inconsistently producing input String with scientific notation.
DLPX-73068	Fixed an issue that causes numeric algorithms using the extensibility framework to fail when applied to fixed-width files.
DLPX-73157	Masking job queued failing immediately as unable to get the execution ID.
DLPX-73187	Custom sql inside the ruleset is not getting auto-generated in case the custom property file is used.

Bug Number	Description
DLPX-73302	Remove GUI validation to support multiple characters for the delimiter.
DLPX-73327	Job with multiple tables/files that differs only by case run indefinitely.
DLPX-73384	Special characters in mysql database instance names are not properly escaped.
DLPX-73441	Masking IP on DB2 using 'Direct Row Access' with ROWID is failing with conversion error.
DLPX-73477	Prevent locked user accounts from logging in when SSO is enabled.
DLPX-73599	Fixed an issue that causes loss of sub-millisecond precision when processing MS SQL Server datetime types.
DLPX-73671	Uploading Hive driver on the masking engine is failing with InsufficientJvmPermissionException.
DLPX-73702	Extended Connector Profile Job fails with FilePermission required for "target": "/tmp/jtlds2094637632459524041.tmp" with "action": "write".
DLPX-73805	Masking UI: SM editor spins when create 4 * alpha-numeric segments.
DLPX-73886	Upgrade Masking API version to v5.1.7.
DLPX-74055	Allow masking admin users to have api access rights revoked.
DLPX-74135	Empty string delimited inside of enclosures results in masking job failure.
DLPX-74185	Character Mapping algorithms with more than 3 characterGroups do not display correctly in UI.
DLPX-74188	Masking connector properties API/UI needs to redact passwords.
DLPX-74292	Custom property file is getting ignored for the source connector in case of OTF job resulting in job failure.

3.2.36 Release 6.0.6.0

Bug Number	Description
DLPX-59842	Fixed an issue causing jobs to fail with out of memory or stack overflow exceptions when the number of tables exceeded a threshold of approximately 800 per stream. It should no longer be necessary to set job streams greater than 1 to avoid this issue.
DLPX-64493	The Roles API is missing elements for the following categories: Custom Algorithms, Diagnostic, Inventory Report, and Approve Inventories.
DLPX-71396	Settings link is missing from footer for user without setting permissions
DLPX-71397	Settings link in footer redirects to profilerSettings.do instead of default jdbcDriver.do
DLPX-71830	Database Tokenize/re-identify job's commit size is not set to default post-upgrade
DLPX-72079	MSSQL JDBC Urls should accept 'database' as a valid parameter
DLPX-72095	Some extended connectors db drivers - throw errors for connection properties they don't understand
DLPX-72311	Exposed DEFAULT_MULTIPHI_ALGORITHM setting via API.
DLPX-72385	Edit Custom Algorithm - Name of Previously Uploaded File No longer Visible.
DLPX-72460	Large environment export hangs.
DLPX-72564	"Add Application" option should be on top inside the action dropdown list.
DLPX-72704	Expanded LK table text limit 1024 characters.
DLPX-72867	Mssql driver is not working with the extended connector in case the instanceName is given in the JDBC url.
DLPX-73082	Unable to assign algorithm to XML fields which contain special characters.

Bug Number	Description
DLPX-73212	Copying an environment that contains a profile or tokenization job causes the environment export to fail with NullPointerException.
DLPX-73338	XSS attack is getting executed on the environment overview page.
DLPX-73502	OTF job with generic connector is failing.

3.2.37 Release 6.0.5.0

Bug Number	Description
DLPX-62372	API authorization token used by the UI expires before the UI login session.
DLPX-70685	Removal of format installation via FTP, SFTP, and mount for XML and Mainframe File Format.
DLPX-71387	Editing recordType to change recordTypeQualifier results in empty JSON.
DLPX-71540	Added Application option is not displayed to the user without copy environment permission.
DLPX-71686	Deleting all mountFilesystem objects nor rebooting does not stop the running portmapper and auxiliary NFS RPC services.
DLPX-50282	Masking support for Oracle XMLType.
DLPX-71666	Characters in Ignore Characters causes Non-Conforming error in Segment Mapping.
DLPX-71758	Propagated SSL related system properties set in Tomcat to Kettle.
DLPX-71734	Masking SQL Server datatype datetime2 generate conversion error.
DLPX-71824	DB-To-File masking job failure.
DLPX-71159	Uploading copybook file format fails if a filename contains multiple full stops.

Bug Number	Description
DLPX-71915	Segment mapping doesn't mask and reports success when positions are misconfigured.
DLPX-71531	Extended algorithm internal conversion of numeric to string types produces unexpected results.
DLPX-72003	Newline characters in the description of an extended algorithm break the Algorithm Settings UI.
DLPX-72028	Using Algm-SDK 1.1 on Windows, algm builds fail w/ 'Illegal char <:> at index 2:'.
DLPX-72128	Overly aggressive quoting of Oracle usernames breaks proxy users.
DLPX-72194	Upgraded MSSQL driver to latest version 8.4.1.
DLPX-72267	Made default API version configurable through application settings.
DLPX-72263	Domain value is not retained on defining a file field causing NPE while job execution.
DLPX-72308	RPC serviceUser can delete an active mount which resulted in active RPC services.
DLPX-72367	Null Pointer Exception when applying a String type extended algorithm or non-legacy Secure Lookup to numeric type columns.

3.2.38 Release 6.0.4.0

Bug Number	Description
DLPX-69407	Hybrid jobs are not syncable.
DLPX-69476	File connector sync throws an error for missing passwords.
DLPX-69834	The user without permission can access UI components using a direct URL.

Bug Number	Description
DLPX-70053	VSAM job performance still poor when file wildcards are used due to flaw in DLPX-68780 fix.
DLPX-70265	NPE along with 'problem-saving mapplet' pop-up is displayed for invalid filereferenceld.
DLPX-70412	OTF Masking SYBASE could not mask 2 tables with the same name but different owners.
DLPX-67886	Updated the SAP ASE (Sybase) JDBC Driver.
DLPX-70567	Implemented a job queue to regulate memory consumption.
DLPX-70642	Copy Ruleset performance improvement.
DLPX-69699	VSAM Masking - Inventory blank after Copy Rule Set fails to copy and corrupts Rule Set and File Format.
DLPX-67501	Fixed an issue that caused Delimited and Fixed-width data level profiling jobs with an FTP or SFTP connector to hang on large files.
DLPX-63065	Updated jquery.js library for Masking to 1.12.0d.
DLPX-69124	Fixed an issue discovering column metadata for Oracle databases that could result in incorrect column lengths and masking jobs failing on update because values are not trimmed correctly.
DLPX-70651	application_nm is not trimmed automatically during an upgrade.
DLPX-70878	Fixed an issue where an on-the-fly Masking job with the disable constraints feature on attempted to use null as the database password.
DLPX-63491	File Masking OTF jobs create the file at the end of the job instead of continuously writing masked rows.
DLPX-59952	OutOfMemory in File Masking when masking large or many files.
DLPX-70395	Renamed Delphix FT algorithm properties "Blacklist" and "Whitelist" to "Denylist" and "Allowlist".

Bug Number	Description
DLPX-70807	Removed Row Types for Database Inventory.
DLPX-70662	Removed Scheduler from Masking.
DLPX-71000	Fixed an issue where CLOB and NCLOB masked values were being incorrectly truncated on Oracle. Refresh the ruleset for the fix to take effect.
DLPX-70982	Masking LDAP user is locked locally when LDAP auth fails.
DLPX-71235	In the monitor screen, all tables show failed if any tables are failed.
DLPX-71320	Removed/hid the environment export checkbox from the roles page.
DLPX-71310	The profiling job fails if a profiler set matches all columns of a table using column profiling.
DLPX-71424	Disable triggers, drop constraints, drop indexes, prescripts and postscripts target source database with OTF jobs and advanced connectors.
DLPX-71530	Unmasked values with only spaces result in (null) masked value.

3.2.39 Release 6.0.3.0

Bug Number	Description
DLPX-63874	ExecutionComponent status for unwritable files was incorrect when masking over SFTP.
DLPX-68123	Masking Engine does not re-read Kerberos config dynamically.
DLPX-68725	Upgraded tomcat to 9.0.31 or later.
DLPX-69655	loginid did not support '@' when creating connectors.
DLPX-69492	MSSQL driver requires java.net.socketpermission to accept permission which is not present in MDS.

Bug Number	Description
DLPX-69493	Execution event is not getting generated for profile job in case of missing permission.
DLPX-69761	Masking Jobs, fail to save added Pre-Scripts.
DLPX-69766	Masking GUI: Remove any script from masking job dialog removes both the scripts.
DLPX-69782	Export/Import Environment using engine sync API.
DLPX-69780	UI based Export Global Object using engine sync API.
DLPX-46853	Switch from jTDS to Microsoft SQL Server JDBC driver.
DLPX-65380	Masking Jobs with commit size>=340 are getting failed on Azure Managed SQL instance.
DLPX-69815	Secure_shuffle algorithm fails for decimal data type using extended connector.
DLPX-69806	Inventory UI is susceptible to URL based XSS attack.
DLPX-69779	Mapplet's input and output fields are susceptible to XSS attack.
DLPX-69832	Import Environment using sync API.
DLPX-69833	UI: Import Global Object using sync API.
DLPX-69861	Define Fields 'Field Name' input is susceptible to XSS attack.
DLPX-69888	XSS script in file pattern is getting executed.
DLPX-69960	Unable to Edit File format if the Enclosure is set to " (double quote).
DLPX-69671	Delimited File Masking with delimiter inside enclosure is handled incorrectly.
DLPX-69922	Inventory UI is susceptible to XSS attack using malicious column names.

Bug Number	Description
DLPX-69941	Error report on job monitor page is susceptible to XSS attack.
DLPX-69989	dateFormat field of date algorithms is susceptible to XSS attack.
DLPX-69920	Import/Upload file UI is susceptible to iframe based XSS attack, throughout the application.
DLPX-69919	Redaction value input field of Free Text Redaction algorithm is vulnerable to XSS attack.
DLPX-69917	Export Inventory UI is susceptible to URL based XSS attack.
DLPX-70055	Masking - Inventory for oracle always picking up NUMBER (22) instead of real NUMBER definition.
DLPX-70046	OTF job with decimal data type and secure shuffle algorithm is changing the last digit after the decimal point of the unmasked column in case of Hana database.
DLPX-70050	CSV and XML file masking performance improvements.
DLPX-70074	Copying an environment does not create a sync state.
DLPX-69851	Masking jobs fail to set fetch size large enough in the input step query.
DLPX-69672	Delimited File Masking and Segment Mapping is not ignoring delimiter if specified as ignore character.
DLPX-69954	Delimited file masking row parsing incorrect when a field contains multiple enclosure characters and a delimiter.
DLPX-70178	Delimited Files: Improve validation for delimiter and enclosure from API.
DLPX-70182	Improved validation for delimiter and enclosure from GUI.
DLPX-70217	"Max number of jobs" Setting on masking engine should be API accessible.

Bug Number	Description
DLPX-70379	For the multi-tenant job, the source connector dropdown doesn't show the connector in the list if the connector instance name contains the space in between.
DLPX-70558	searchEnvironment parameter in URL is vulnerable to XSS attack.
DLPX-70557	Copy Ruleset has a scale performance issue with a large number of tables/ columns.
DLPX-70641	Unmasked data logged in the support bundle logs when using extended connector with enable_logger functionality on

3.2.40 Release 6.0.2.0

Bug Number	Description
DLPX-65833	Removed unnecessary error out on passwords being provided for file connectors using the mount mode.
DLPX-65319	New API endpoint for mainframe-dataset-record-type.
DLPX-68153	If creating a mapping algorithm in the Masking UI fails, the failure is now properly reported to the user.
DLPX-67882	Upgrade the PostgreSQL JDBC driver to version 42.2.10.
DLPX-58184	List rule sets alphabetically on the inventory page.
ES-662	Added Sync support for data set connectors.
ES-664	Added Sync support for mainframe data set formats
ES-671	Added Sync support for Mainframe data set jobs
ES-665	Added Sync support for Mainframe data set rule sets.

Bug Number	Description
DLPX-68786	Masking job misreported successful tables as 0 rows masked.
DLPX-67517	Added support for on-the-fly jobs from a database to a delimited file.
DLPX-68842	Jobs slowed down over time - after running many jobs.
DLPX-68985	A memory leak occurred for Informix/oracle database on every test connection using an extended connector.
DLPX-68780	VSAM Input step performance was negatively affected by the number of unmasked fields.
DLPX-67886	Sybase jConnect driver failed when a batch contains string parameters of different sizes and HOMEGENOUS_BATCH=true.
DLPX-65841	Fixed an issue where a REST API call to GET /syncable-objects? object_type=MASKING_JOB would fail after environment copy.
DLPX-69156	Test Connection always returned connection succeeded in case of wrong jdbc url with extended connector.
DLPX-69238	Secure Shuffle algorithm, when used with extended connectors, left data unmasked but reports success.
DLPX-69244	Importing a 5.3.x Masking Environments into 6.0.1 ME, the Application Name is converted to numeric.
DLPX-69154	Fixed an issue where setup could fail if the DNS Domain is empty.
DLPX-69622	Data level profiling jobs fail with "Couldn't find field 'XYZ' in row!"

3.2.41 Release 6.0.1.0

Bug Number	Description
DLPX-64530	Allow a JDBC URL to contain a single quote (') character.

Bug Number	Description
DLPX-65302	Add a status column to the audit log page to report each recorded action's result (success/failure).
DLPX-65622	Fix an issue where an in-place, multi-tenant XML file masking job that used file patterns did not have an execution component.
DLPX-65974	Updated log statements in the file masking job logs to reflect that file connectors may use mounts in addition to FTP and SFTP.
DLPX-66127	Fixed a job monitoring issue when counting the rows in table with more than 2+ billion (2,147,483,647) rows.
DLPX-62130	Fixed an issue with the XML file inventory GUI that prevented users from assigning algorithms to both a tag and its attribute(s).
DLPX-66272	Fixed an issue where an on-the-fly job using generic connectors used an incorrect database password.
DLPX-66600	Removed the requirement to restart the Masking service after changing email settings.
DLPX-66328	Fixed an issue with file masking jobs using multiple record types that could cause the job to fail or corrupt the output.
DLPX-66557	Added support to the Date Shift algorithm for numeric data types.
DLPX-66517	Enhanced the GET /file-field-metadata endpoint to return the full XML XPath for an XML field.
DLPX-66102	"Drop Indexes" checkbox now handles compound indices correctly for Sybase.
DLPX-66967	Fixed a Job Scheduler issue that caused a periodic job to only running once.
DLPX-67318	Prevent reordering of the XML file inventory GUI when an algorithm is assigned
DLPX-67317	On the XML file inventory GUI, open the algorithm assignment dialogue box with a single mouse click
DLPX-66076	Added API endpoints for file recordTypes and recordTypeQualifiers

Bug Number	Description
DLPX-65855	Optimize the performance of EngineSync import, export, and get syncable object for large database rule sets.
DLPX-65987	Fixed an issue that caused data level profiling of a database to fail when a column name was a special JavaScript word.
DLPX-67747	Fixed an issue that caused some delimited or fixed file masking jobs with multiple record types of different lengths to fail.
DLPX-67470	Fixed delimited file masking to treat double quote (") characters in fields as normal characters.
DLPX-67765	Updated the Sybase JDBC driver.
DLPX-67838	Fixed an issue that prevented XML File masking jobs from scaling above a few thousand files.
DLPX-67832	Non-administrators can no longer regenerate the engine encryption key.
DLPX-67960	Make username searches on the Audit page case insensitive.
DLPX-68148	Fix an issue that caused an XML file masking job to run out of memory when masking very large XML input files.
DLPX-46220	Import of extremely large object sets via the GUI XML feature is handled inefficiently.

3.2.42 Release 6.0.0.0

Bug Number	Description
DLPX-42385	Added a job execution event with information on how to resolve an Oracle deadlock error (ORA-00060), see https://www.delphix.com/masking-help/knowledge-base/KBA1853 .

Bug Number	Description
DLPX-47004	Added a job execution event with information on how to resolve an Oracle snapshot too old error (ORA-01555), see https://www.delphix.com/masking-help/knowledge-base/KBA1827 .
DLPX-47662	Test connector detects that a file/mainframe connector targets a single file instead of a directory and fails.
DLPX-52151	Fixed copy rule set to prevent leading/trailing spaces in a new rule set's name.
DLPX-55478	Correctly display file patterns, including escape characters, throughout the user interface.
DLPX-55739	Fixed the disable constraint feature to support an Oracle constraint (a) created by a different database user than the Masking job's database user and (b) using a validation setting of "NOT VALIDATED".
DLPX-58958	Added support for LDAPS (LDAP over TLS/SSL).
DLPX-59060	Attach the correct PDF report to all job execution emails.
DLPX-59111	When editing a large rule set in the GUI, do not reset to the first page after editing and saving a modification to a rule set component.
DLPX-59807	If a failure occurs during job generation, do not attempt to execute the job.
DLPX-60200	When uploading an SSH key, return an error if the name contains one of the following restricted characters: \ (backslash), ; (semi-colon), % (percent), ? (question mark), or : (colon).
DLPX-61630	Improved the performance for appending new mapping values to a mapping algorithm.
DLPX-62214	Fixed PDF report download URLs.
DLPX-62593	Fixed creation of a PDF audit report on the Audit tab of the user interface.
DLPX-63365	Removed leading/trailing spaces from Masking object names on upgrade. For naming rules, see the Getting Started > Naming Requirements section in the documentation.

Bug Number	Description
DLPX-63706	Fixed the XML file inventory GUI to show an algorithm edit button for a tag with the same name as its parent.
DLPX-64691	Added support in the user interface for Cobol copybooks with a redefine condition at level 01.
DLPX-64707	Improved the file record types user interface to (a) remove the unnecessary length input and (b) clarify that the qualifier may be a regular expression.
DLPX-65274	Improved the performance of the copy environment feature.
DLPX-65314	Fixed an issue in the copy environment feature that removed file format assignments from the source environment.
DLPX-65632	Fixed an issue in the segment mapping algorithm that caused duplicate mappings if a minimum value was specified for the real values range.
DLPX-65860	For mainframe file masking, add support for a redefine condition on a field name that contains a - (dash) followed by a digit.
DLPX-65866	Fixed an issue with the rule set GUI when displaying table names longer than 50 multi-byte characters.

3.3 Known issues

3.3.1 Release 26.0.0.0

Key	Summary	Workaround
DLPX-81694	When masking two columns with the same name (but with mixed case), it can cause the same algorithm to be applied.	None

3.3.2 Release 25.0.0.0

Key	Summary	Workaround
DLPX-81694	When masking two columns with the same name (but in mixed case), it can cause the same algorithm to be applied.	None

3.3.3 Release 24.0.0.0

Key	Summary	Workaround
DLPX-81694	When masking two columns with the same name (but in mixed case), it can cause the same algorithm to be applied.	None

3.3.4 Release 23.0.0.0

Key	Summary	Workaround
DLPX-81694	When masking two columns with the same name (but in mixed case), it can cause the same algorithm to be applied.	None

3.3.5 Release 22.0.0.0

Key	Summary	Workaround
-----	---------	------------

DLPX-87283	The Continuous Compliance Engine is unable to mask a large dataset present in Mainframe MVS storage.	In the case of in-place jobs with large size datasets (i.e. more size than the configured system default size), users should manually create the dataset with the same name as the input dataset, with a .msk extension at the end. Allocate the newly created dataset with more size than the input dataset, before the job.
------------	--	---

3.3.6 Release 21.0.0.0

Key	Summary	Workaround
DLPX-87283	The Compliance Engine is unable to mask a large dataset present in Mainframe MVS storage.	In the case of in-place jobs with large size data sets (i.e. more size than the configured system default size), users should manually create the dataset with the same name as the input dataset, with a .msk extension at the end. Allocate the newly created dataset with more size than the input dataset, before the job execution, to resolve the issue.
DLPX-90170	The job status icon displayed in the Job Monitor UI, if the execution event type of <code>FILE_PATTERN_NO_MATCH</code> is raised, should be a warning icon.	None

3.3.7 Release 20.0.0.0

Key	Summary	Workaround
DLPX-81579	Job monitor UI should display when a file name pattern does not have any match.	None

DLPX-87283	The Compliance Engine is unable to mask a large dataset present in Mainframe MVS storage.	In the case of in-place jobs with large size data sets (i.e. more size than the configured system default size), users should manually create the dataset with the same name as the input dataset with a .msk extension at the end. Allocate the newly created dataset with more size than the input dataset before the job execution to resolve the issue.
------------	---	---

3.3.8 Release 19.0.0.0

Key	Summary	Workaround
DLPX-87283	Unable to mask a large dataset on the mainframe storage.	In the case of in-place jobs with large size datasets, i.e. more size than the configured system default size, users should manually create the dataset with the same name as the input dataset with a .msk extension at the end. Allocate the newly created dataset with more size than the input dataset before the job execution to resolve the issue.
DLPX-87852	In the Monitor job UI completed grid, sort on the 'status' field will not sort the 'success' rows with non-conforming data when the user setting, 'Nonconforming Data behavior', is set to 'Mark job as Succeeded'.	Set the Nonconforming Data behavior to Mark Job as Failed in the UI or the application setting <code>DefaultNonConformantDataHandling</code> to FAIL would allow the sorting to work with success and failure records.

3.3.9 Release 18.0.0.0

Key	Summary	Workaround
DLPX-83946	The Swagger UI does not work while a user is already logged into the Compliance Engine in another tab.	Use an incognito window for the API client.

DLPX-87283	The Compliance Engine is unable to mask a large dataset present in Mainframe MVS storage.	In the case of in-place jobs with large sized datasets (i.e. more size than the configured system default size), users should manually create the dataset using the same name as the input dataset with a .msk extension at the end. Allocate the newly created dataset with more size than the input dataset before the job execution to resolve the issue.
DLPX-87852	Sorting the <code>monitorJobsDBCompleted</code> grid by the 'Status/Logs' column does not sort the rows that have non-conforming data with the 'success' status.	Setting the non-conforming data behavior to 'Mark Job as Failed' in the UI or using the application setting <code>DefaultNonConformantDataHandling</code> to FAIL would allow the sorting to work with success and failure records.

3.3.10 Release 17.0.0.0

Key	Summary	Workaround
DLPX-87283	The Compliance Engine is unable to mask a large dataset present in Mainframe MVS storage.	<p>In the case of in-place jobs with large size datasets (i.e. more size than the configured system default size), users should manually create the dataset with the same name as the input dataset with a .msk extension at the end.</p> <p>To resolve the issue, allocate the newly created dataset with more size than the input dataset before the job execution.</p>
DLPX-87707	In very rare cases, the Compliance Engine's storage can be filled with compliance job and inventory PDF reports. This issue might be encountered after generating hundreds of thousands of reports over multiple years.	None

DLPX-87852	Sorting the <code>monitorJobsDBCompleted</code> grid by the 'Status/Logs' column does not sort the rows that have non-conforming data with status- 'success'.	Setting the Nonconforming Data behavior to Mark Job as Failed in the UI or the application setting- <code>DefaultNonConformantDataHandling</code> to FAIL would allow the sorting to work with success & failure records.
DLPX-88335	Rapid scrolling through the inventory tables in the presence of a sluggish API response can lead to an infinite loop of API calls, resulting in the grid becoming unresponsive and stuck in a perpetual loading state.	The problem occurs exclusively during high-speed scrolling within the grid. Upon encountering this issue, triggering a manual refresh will restore the grid to the most recently successfully loaded page. However, in scenarios where users continue to scroll even after the page has become unresponsive, a grid refresh may not recover the previous state. In such instances, a complete page refresh is required. To mitigate these issues, users are encouraged to mitigate the problem by using filters to narrow down the grid data.
DLPX-89056	Data-level profiling with ASDD for Oracle database might not return any classification results for a column that has large number of null or empty values.	None
DLPX-89149	ASDD Profiling job for Db2 zOS/LUW databases might fail when profiling a table with a column that has large number of null or empty values.	Update the asdd application setting- <code>DefaultMaximumColumnSize</code> to <code>250</code>

3.3.11 Release 16.0.0.0

Key	Summary	Workaround
-----	---------	------------

DLPX-87283	Compliance Engine is unable to mask a large dataset present in Mainframe MVS storage.	In the case of in-place jobs with large size datasets (i.e. more size than the configured system default size), users should manually create the dataset with the same name as the input dataset with a .msk extension at the end. Allocate the newly created dataset with more size than the input dataset before the job execution to resolve the issue.
DLPX-87707	In very rare cases, the Continuous Compliance Engine's storage can be filled with masking job and inventory PDF reports. This issue might be encountered after generating hundreds of thousands of reports over multiple years.	None
DLPX-87852	Sorting the <code>monitorJobsDBCompleted</code> grid by the 'Status/Logs' column does not sort the rows that have non-conforming data with status- 'success'.	Setting the Nonconforming Data behavior to Mark Job as Failed in the UI or the application setting- <code>DefaultNonConformantDataHandling</code> to FAIL would allow the sorting to work with success & failure records.
DLPX-88335	Rapid scrolling through the inventory tables in the presence of a sluggish API response can lead to an infinite loop of API calls, resulting in the grid becoming unresponsive and stuck in a perpetual loading state.	The problem occurs exclusively during high-speed scrolling within the grid. Upon encountering this issue, triggering a manual refresh will restore the grid to the most recently successfully loaded page. However, in scenarios where users continue to scroll even after the page has become unresponsive, a grid refresh may not recover the previous state. In such instances, a complete page refresh is required. To mitigate these issues, users are encouraged to mitigate the problem by using filters to narrow down the grid data.

DLPX-88658	Masking job abruptly fails when using the 'Lat_Long Coordinates' algorithm and the data doesn't conform to the regexes of this extended algorithm.	The error is due to an issue with the default fallback action which gets triggered when data doesn't match the regexes specified by this algorithm. So add/modify the regexes within the algorithm such that fallback action is not triggered or replace the fallback action with a custom algorithm.
DLPX-89056	Data-level profiling with ASDD for Oracle database might not return any classification results for a column that has large number of null or empty values.	None

3.3.12 Release 15.0.0.0

Key	Summary	Workaround
DLPX-87283	Compliance Engine is unable to mask a large dataset present in Mainframe MVS storage.	In the case of in-place jobs with large size datasets (i.e. more size than the configured system default size), users should manually create the dataset with the same name as the input dataset, with a .msk extension at the end. Allocate the newly created dataset with more size than the input dataset before the job execution to resolve the issue.
DLPX-87535	Copy privileges for custom roles are not getting updated via the Continuous Compliance UI and API.	Users can duplicate this role, uncheck copy privileges, and save it via the UI. Or create a new role via the API/UI. Updating the role with API for copy privilege will also give the same issue.

DLPX-87707	In very rare cases, the Compliance Engine storage can be filled with masking job and inventory PDF reports. This issue might be encountered after generating hundreds of thousands of reports over multiple years.	None
DLPX-87852	Sorting the <code>monitorJobsDBCompleted</code> grid by 'Status/Logs' column does not sort the rows that have non-conforming data with the "success" status.	The user can set to fail with non-conformant data, then sorting works fine for success and failure records.

3.3.13 Release 14.0.0.0

Key	Summary	Workaround
DLPX-87283	Compliance Engine is unable to mask a large dataset present in Mainframe MVS storage.	In the case of In-place jobs with large size data sets i.e. more size than the configured system default size, Users should manually create the dataset with the same name as the input dataset with a .msk extension at the end. Allocate the newly created dataset with more size than the input dataset before the job execution to resolve the issue.

3.3.14 Release 13.0.0.0

Key	Summary	Workaround
DLPX-85469	JSON file masking does not support the use of multi-column algorithms on (a) Fields in two or more different arrays (b) Fields at different levels in a single multi-dimensional array.	None

3.3.15 Release 12.0.0.0

Key	Summary	Workaround
DLPX-85469	<p>JSON file masking does not support the use of a multi-column algorithms on</p> <ul style="list-style-type: none"> • Fields in two or more different arrays. • Fields at different levels in a single multi-dimensional array. 	No workaround.

3.3.16 Release 11.0.0.0

Key	Summary	Workaround
DLPX-86196	Segment Mapping algorithms exported using Engine Synchronization from releases 6.0.15.0 to 10.0.0.0 are not validated upon import to 11.0.0.0.	View the imported algorithm configuration in the UI and click save.
DLPX-84875	User with create/edit permissions is not able to create or edit the classifier	Assign domain view role to user.
DLPX-85469	<p>JSON file masking does not support the use of a multi-column algorithms on</p> <ul style="list-style-type: none"> • Fields in two or more different arrays. • Fields at different levels in a single multi-dimensional array. 	No workaround.
DLPX-85560	Salesforce on the fly jobs fail with latest CDATA driver, with a license error.	No workaround.

3.3.17 Release 10.0.0.0

Key	Summary	Workaround
DLPX-85469	JSON file masking does not support the use of a multi-column algorithms on <ul style="list-style-type: none"> Fields in two or more different arrays. Fields at different levels in a single multi-dimensional array. 	No workaround.
DLPX-85560	Salesforce on the fly jobs fail with latest CDATA driver, with a license error.	No workaround.
DLPX-84875	Unable to create/edit the classifier/ expression using the user with create/edit permission of profiler role	Assign domain view role to user.
DLPX-85867	Incorrect non-conforming data count across all the tables of a job, if at least one table has non-conforming data.	No workaround.

3.3.18 Release 9.0.0.0

Key	Summary	Workaround
DLPX-84875	Unable to create or edit classifiers/ expressions as a user with create/edit permission of the Profiler role.	Assign the domain view role to user.
DLPX-85867	Incorrect non-conforming data count across all the tables of a job, if at least one table has non-conforming data.	No workaround.

3.3.19 Release 8.0.0.0

Key	Summary	Workaround
DLPX-84141	Masking Engine environment revisionHash changes on profile job execution even though there is no change in inventory	No workaround.
DLPX-85867	Incorrect non-conforming data count across all the tables of a job, if at least one table has non-conforming data.	No workaround.

3.3.20 Release 7.0.0.0

Key	Summary	Workaround
DLPX-84141	Continuous Compliance Engine environment revisionHash changes on every profile job execution even if there is no inventory change.	No workaround.
DLPX-84525	Inventory Page fails to load on Continuous Compliance Engines when there are more than 32,767 masked columns.	A possible workaround is to reduce the number of masked columns on the Engine to less than 32,765.
DLPX-85867	Incorrect non-conforming data count across all the tables of a job, if at least one table has non-conforming data.	No workaround.

3.3.21 Release 6.0.17.0

No known issues in this release.

3.3.22 Release 6.0.16.0

Bug Number	Description	Workaround
DLPX-82517	Issues w/ DB2 iSeries Connector License Installation	No workaround.

3.3.23 Release 6.0.15.0

Bug Number	Description	Workaround
DLPX-81895	Data Profiling results are not shown at Job -> Monitor page -> Results tab, in the case of delimited, fixed, and XML files.	No workaround.
DLPX-82517	Issues w/ DB2 iSeries Connector License Installation	No workaround.

3.3.24 Release 6.0.14.0

Bug Number	Description	Workaround
DLPX-82517	Issues w/ DB2 iSeries Connector License Installation	No workaround.

3.3.25 Release 6.0.13.0

No known issues in this release.

3.3.26 Release 6.0.12.0

Bug Number	Description	Workaround
DLPX-78478	Reidentification of CM numeric algorithm on decimal data is failing.	When the field is long enough, use the Tokenization algorithm instead of CM tokenization.
DLPX-78659	CM Numeric is not producing unique results for the floating-point numbers.	Use an algorithm other than CM Numeric algorithm for masking floating-point numbers stored in a numeric field.
DLPX-79567	Drop index fails if an index with the same name exists on the masked columns across multiple tables for MSSQL databases.	No workaround.
DLPX-79803	Masking with MSSQL database fails if table name contains '['.	No workaround.
DLPX-79804	Masking with MSSQL database fails if table name contains '\\	No workaround.

3.3.27 Release 6.0.11.0

Bug Number	Description	Workaround
DLPX-78009	The masking job fails when masking the primary key column if Drop Indexes are not enabled along with the Disable Constraints.	In addition to Drop Indexes, you must enable Disable Constraints when masking primary keys using built-in driver support functionality. Advanced users who are not satisfied by some limitations of the built-in Oracle support for masking primary keys may also create custom pre and post-scripts to perform both drop indexes and disable constraints operations.
DLPX-79803	Masking with MSSQL database fails if table name contains '['.	No workaround.

Bug Number	Description	Workaround
DLPX-79804	Masking with MSSQL database fails if table name contains '\.	No workaround.

3.3.28 Release 6.0.10.0

No known issues in this release.

3.3.29 Release 6.0.9.0

No known issues in this release.

3.3.30 Release 6.0.8.0

Bug Number	Description	Workaround
DLPX-74882	Masking's SFTP client no longer compatible with SolarWinds and Goanyware SFTP servers	No workaround.

3.3.31 Release 6.0.7.0

No known issues in this release.

3.3.32 Release 6.0.6.0

No known issues in this release.

3.3.33 Release 6.0.5.0

No known issues in this release.

3.3.34 Release 6.0.4.0

No known issues in this release.

3.3.35 Release 6.0.3.0

No known issues in this release.

3.3.36 Release 6.0.2.0

Bug Number	Description	Workaround
DLPX-69638	Masking job created on engine 6.0.1.1 or prior is failing after the upgrade to version 6.0.2.0 or later	Masking jobs created in 6.0.1.x using a HANA JDBC driver will need to be updated to grant the following permission

3.3.37 Release 6.0.1.0

No known issues in this release.

3.3.38 Release 6.0.0.0

Bug Number	Description	Workaround
DLPX-60397	If a mapping algorithm is included in multiple jobs, only one job should be run at a time. If multiple jobs are run at the same time, then the mapping algorithm might contain multiple mappings to the same value or the jobs might deadlock.	Only run one job at a time.
DLPX-61405	Masking operation should wait for zfs delete queue to drain	Replication may send more data than expected if masking involves dropping large DBF files.
DLPX-74882	Masking's SFTP client no longer compatible with SolarWinds and Goanyware SFTP servers	No workaround.
DLPX-64493	V5 API /roles endpoint missing certain items	View and set these privileges through the GUI
DLPX-66973	Date format is changed after importing the environment	Either (a) use the GUI import feature and then review the imported date formats for correctness or (b) use EngineSync to export/import jobs, which will not alter the date format.

3.4 Deprecated and end-of-life features

3.4.1 Release 26.0.0.0

3.4.1.1 End-of-life features

Support for the following have reached EOL in this release:

- Aurora MySQL 5.7

3.4.2 Release 24.0.0.0

3.4.2.1 Deprecated features

- **Generic connectors**

Generic connectors are now deprecated. This connector type will reach End-of-Life and be removed from the product in six months.

- To determine if a Continuous Compliance Engine contains a generic connector, users should review the engine's list of database connectors through either the graphical UI or API. The value `Generic` in the connector type field indicates the presence of a generic connector.
- All generic connectors should be replaced by creating an equivalent, built-in connector. For example, a generic connector using an Oracle JDBC URL should be replaced with a new, built-in Oracle connector. If the generic connector set properties in the JDBC URL, those can be set for the built-in connector using the connector properties feature that was introduced in release 6.0.6.0. For more information, see [Database Connection Properties \(see page 355\)](#).

- **Legacy profiler**

The legacy profiler for files and mainframe datasets is now deprecated. It will reach End-of-Life and be removed from Delphix Continuous Compliance with the release of version 27.0.0.0 (same release as the legacy database profiler). Legacy profiling jobs should be updated to use ASDD profile sets. For more information, see this [Delphix Community Post](#)⁶⁷.

⁶⁷ <https://community.delphix.com/blogs/claire-mcmanus/2024/06/18/legacy-file-and-mainframe-profiling-end-of-life>

3.4.3 Release 20.0.0.0

3.4.3.1 End-of-life features

- **Job Wizard UI**

The Job Wizard UI feature has reached End-of-Life (EoL) and has been entirely removed from the product.

3.4.4 Release 15.0.0.0

3.4.4.1 Deprecated features

- **Legacy Profiler**

The legacy profiler for databases is now deprecated. It will reach End of Life and be removed from Delphix Continuous Compliance in twelve months.

3.4.5 Release 11.0.0.0

3.4.5.1 Deprecated features

- **Job Wizard UI**

The Job Wizard is deprecated as of this release. The Job Wizard offers a simplified job creation experience, but only for a limited set of data platforms. We are incrementally upgrading the existing Continuous Compliance UI pages with new, reactive interfaces and improved user experiences. As part of that process, the Job Wizard's capabilities will be incorporated into the product's standard workflows, which makes them available for all data sources.

3.4.5.2 End-of-Life features

Support for the following have reached EOL in this release:

- **ESXi 6.0 support**

3.4.6 Release 10.0.0.0

3.4.6.1 Deprecated features

Support for the following have been deprecated in this release:

- **ESXi 6.5 and 6.7** VMware's end of technical guidance for ESXi is on 11/15/2023. To align our support, Delphix will end support of ESXi 6.5 and 6.7 in version 15.0.0.0 (November 2023 release).

3.4.7 Release 6.0.17.0

3.4.7.1 End-of-life features

Support for the following have reached EOL in this release:

- **Oracle 11.1 and 12.1** Details of the Oracle database end-of-life can be found in the [Oracle Lifetime Support Policy](#)⁶⁸.

3.4.8 Release 6.0.15.0

3.4.8.1 End-of-life features

- **Legacy algorithms & mapplets:** The last algorithm migrations have completed in this version (6.0.15), Legacy Algorithms are now EOL. For more information, see this [Delphix Community Post](#)⁶⁹.

3.4.9 Release 6.0.12.0

3.4.9.1 End-of-life features

- **Legacy Secure Lookup:** Legacy Secure Lookup has been removed and only the extensible version is supported. Previous secure lookup instances have been moved to the extensibility framework.
- **Internet Explorer 11 support:** Internet Explorer 11 is no longer supported by Delphix. Users are requested to refer to the list of [Supported browsers](#)⁷⁰.

3.4.10 Release 6.0.11.0

3.4.10.1 Deprecated features

Support for the following have been deprecated in this release:

- **Oracle 11.1 and 12.1:** Details of the Oracle database end-of-life can be found in the [Oracle Lifetime Support Policy](#)⁷¹.

68 <https://www.oracle.com/us/assets/lifetime-support-technology-069183.pdf>

69 <https://community.delphix.com/blogs/david-wells/2022/06/10/delphix-6015-continuous-compliance-important-infor>

70 <https://delphixdocs.atlassian.net/wiki/spaces/CD/pages/3480605/Tested+browser+and+operating+systems>

71 <https://www.oracle.com/us/assets/lifetime-support-technology-069183.pdf>

- **TLS 1.0 and 1.1** These versions of TLS are known to be vulnerable, enterprise use is heavily discouraged.

3.4.10.2 End-of-life features

Support for the following OS version have reached EOL in this release:

- **Oracle 10g support:** 6.0.10.0 is the last release supporting Oracle 10.1 and 10.2.
- **Create/update of legacy secure lookup algorithms via UI:** The ability to create and update legacy secure lookup algorithms has been removed from the UI. This feature is still accessible through the API endpoints.

3.4.11 Release 6.0.10.0

3.4.11.1 End-of-life features

- **Ruleset Edit:** The Table Suffix, Add Column, Join Table, and List options were deprecated in the 6.0.3.0 release. These options have reached the end of life in the 6.0.10.0 release and have been completely removed from the product. These options are the rarely used feature that can be achieved using the following alternatives:
 - If you were using the Table Suffix functionality, you can achieve the same results with a series of API calls (/table-metadata and /column-metadata endpoints).
 - For Add Column, Join Table, and List, you need to convert these settings to the equivalent Custom SQL configuration before upgrading to 6.0.10.0 release.

3.4.12 Release 6.0.9.0

3.4.12.1 Deprecated features

Delphix has been creating new and improved versions of our existing algorithms, thus, Delphix would like to provide formal notice of deprecation and planned End-of-Life (EoL) for the older algorithm versions. This is to inform our customers that planning should start for their transition to these updated algorithms. For more information and details on the transition, see [Delphix Community Post - Legacy Mapping Algorithm](https://community.delphix.com/blogs/michael-torok/2021/08/12/delphix-end-of-life-notice-legacy-masking-algorithm)⁷².

⁷² <https://community.delphix.com/blogs/michael-torok/2021/08/12/delphix-end-of-life-notice-legacy-masking-algorithm>

3.4.13 Release 6.0.8.0

3.4.13.1 End-of-life features

- Legacy Custom Algorithm (Mapplet). For more information, see [Delphix Community Post - Mapplet EoL](#)⁷³.
- SAP ASE (Sybase) 15.0.3 support

3.4.14 Release 6.0.7.0

3.4.14.1 End-of-life features

- ESX 5.5 support
- Masking Connectors: Db2 LUW and zOS v9, Db2 LUW and zOS v10, SQL Server 2005, 2008, 2008 R2

3.4.15 Release 6.0.4.0

3.4.15.1 Deprecated features

- **FTP, SFTP, and mount upload for XML and Cobol formats** FTP/SFTP/Mount-based format import were the original modes for XML and Cobol files, since then, Delphix has added the ability to upload a format file, which is simpler to set up. After the introduction of “upload”, there has been a dramatic shift away from the legacy import modes in favor of the simplicity of “upload”.
- **Row type feature** Originally geared for limiting masking to subsets of rows within a column, this feature was seldom used. The functionality, if desired, can still be replicated via the Custom SQL feature.
- **Redundant settings for ‘edit table’ under rule sets** Table Suffix, Add Column, Join Table, and List - These settings are redundant and can be replicated with the Custom SQL setting.
- **‘HAVING’ clause from Masking API** Deprecating due to low use. This feature, if desired, can be replicated with Custom SQL.

3.4.15.2 End-of-life features

- **Job Scheduler** As of this release, Delphix has removed the Job Scheduler feature. The introduction of Masking’s REST API several releases ago allowed customers to schedule job executions using their preferred job scheduler. As a result, the integrated scheduler is seldom used.

⁷³ <https://community.delphix.com/blogs/sonali-sharma/2021/06/30/legacy-custom-algorithm-mapplet-end-of-life>

3.4.16 Release 6.0.3.0

3.4.16.1 End-of-life features

In this release, the deprecated XML import/export functionality has been removed. If you used the XML import/export feature in previous releases, you'll find the new Sync Environment feature to be a more robust and complete solution with complete API support in addition to being available in the UI.

3.4.17 Release 6.0.0.0

3.4.17.1 End-of-life features

- Native XML CLOB masking: After the upgrade, columns masked as XML CLOBs will have the NULL SL algorithm assigned.
- Excel files can still be masked by first converting them to Delphix-supported file types (CSV, etc). Also, XML CLOBs can be masked by extracting their values into a table (for example - using `extractValue` in Oracle).
- DB2 9.1, 9.5, and other 9.x versions of LUW & Z/OS
- "Create target" job option: After upgrading jobs using "create target" will be removed.
- "Bulk data" job option: After the upgrade, jobs using "bulk data" will be turned into non-bulk data jobs.
- Native Microsoft Excel Masking: After the upgrade, MS Excel connectors, rulesets, and jobs will be removed.

3.5 Licenses and notices

The Delphix Dynamic Data Platform includes licensed, third-party products from the following companies. These products are copyrighted and all rights are reserved by the respective companies:

- Highcharts, © Highsoft

The Delphix Masking engine includes licensed, third-party products from the following companies. These products are copyrighted and all rights are reserved by the respective companies:

- Kendo UI, © Telerik

Starting with 6.0.3.0, the license info is available via a CLI/API on the engine when logged in as a system administrator.

```
engine> cd license
engine license> getLicense
engine license getLicense *> commit
```

Access to the source code of such third party open source components may be permitted or required in certain instances under the applicable open source licenses by sending an email to <mailto:open-source@delphix.com>.

4 Getting started

This section covers the following topics:

- [Introduction to Delphix Masking \(see page 150\)](#)
- [Data source support \(see page 154\)](#)
- [Installation \(see page 175\)](#)
- [Naming requirements \(see page 228\)](#)
- [Graphical user interface \(see page 232\)](#)
- [Managing application settings \(see page 246\)](#)
- [Users and roles \(see page 248\)](#)
- [Best practices for defining masking roles \(see page 270\)](#)
- [Audit logs \(see page 273\)](#)
- [Monitor Async Task Status \(see page 275\)](#)
- [Kerberos configuration \(see page 277\)](#)
- [Password vault configuration \(see page 293\)](#)
- [Db2 connector license installation \(see page 298\)](#)
- [Continuous Compliance Engine icon reference \(see page 300\)](#)
- [Delphix masking terminology \(see page 303\)](#)
- [Changing the IP address of the Delphix Engine \(see page 312\)](#)
- [Stopping and starting the containerized Continuous Compliance Engine \(see page 314\)](#)
- [Stopping, starting, and restarting the continuous compliance engine \(see page 317\)](#)
- [Upgrading the Continuous Compliance Engine \(see page 319\)](#)
- [Utilization \(see page 320\)](#)

4.1 Introduction to Delphix Masking

4.1.1 Challenge

With data breach incidents regularly making the news and increasing pressure from regulatory bodies and consumers alike, organizations must protect sensitive data across the enterprise. Contending with insider and outsider threats while staying compliant with mandates such as HIPAA, PCI, and GDPR is no easy task—especially as teams simultaneously try to make their organizations more agile.

To tackle the problem of protecting sensitive information, companies are increasingly scrutinizing the tools they've deployed. Instead of reactive perimeter defenses, security-minded organizations must focus on proactively protecting the interior of their systems: their data. Moreover, while mainstay approaches such as encryption may be effective for securing data-in-motion or data resident in hard drives, they are ill-suited for protecting non-production environments for development, testing, and reporting.

4.1.2 Solution

The masking capability of the Delphix DevOps Data Platform represents an automated approach to protecting non-production environments, replacing confidential information such as social security numbers, patient records, and credit card information with fictitious, yet realistic data.

Unlike encryption measures that can be bypassed through schemes to obtain user credentials, masking irreversibly protects data in downstream environments. Consistent masking of data while maintaining referential integrity across heterogeneous data sources enables Delphix masking to provide superior coverage compared to other solutions—all without the need for programming expertise. Moreover, the Delphix DevOps Data Platform seamlessly integrates masking with data delivery capabilities, ensuring the security of sensitive data before it is made available for development and testing, or sent to an offsite data center or the public cloud.

Delphix Masking is a multi-user, browser-based web application that provides complete, secure, and scalable software for your sensitive data discovery, masking, and tokenization needs while meeting enterprise-class infrastructure requirements. The Delphix DevOps Data Platform has several key characteristics to enable your organization to successfully protect sensitive data across the enterprise:

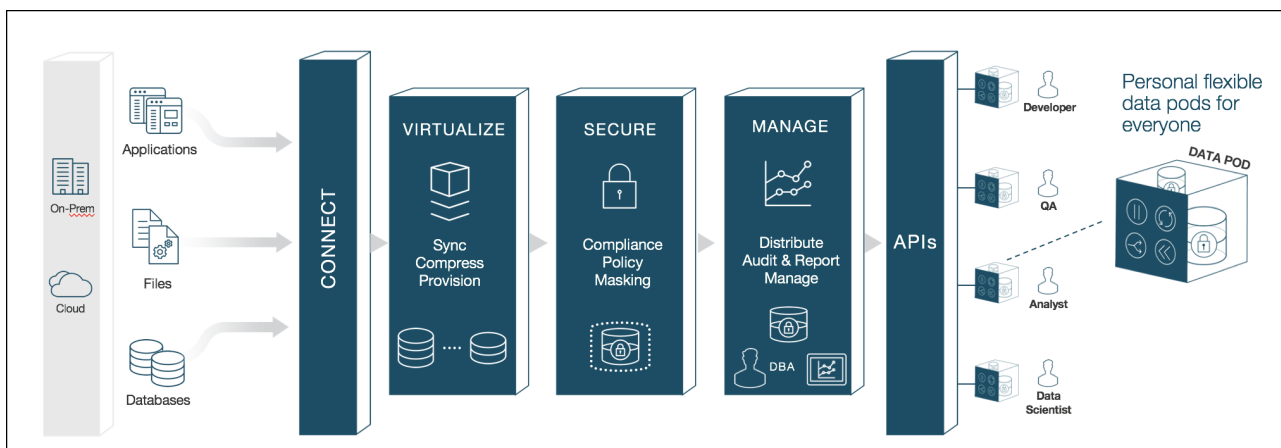
- **End-to-End masking** – The Delphix platform automatically detects confidential information, irreversibly masks data values, then generates reports and email notifications to confirm that all sensitive data has been masked.
- **Realistic data** – Data masked with the Delphix platform is production-like in quality. Masked application data in non-production environments remain fully functional and realistic, enabling the development of higher-quality code.
- **Masking integrated with Virtualization** – Most masking solutions fail due to the need for repeated, lengthy batch jobs for extracting and masking data and a lack delivery capabilities for downstream environments. The Delphix DevOps Data Platform seamlessly integrates data masking with [data virtualization](#)⁷⁴, allowing teams to quickly deliver masked, virtual data copies on-premises or into private, public, and hybrid cloud environments.
- **Referential integrity** – Delphix masks consistently across heterogeneous data sources. To do so, metadata and data are scanned to identify and preserve the primary/foreign key relationships between elements so that data is masked the same way across different tables and databases.
- **Algorithms/Frameworks** – Eighteen algorithm frameworks allow users to create and configure algorithms to match specific security policies. Over twenty-five out-of-the-box, preconfigured algorithms help businesses mask everything from names and addresses to credit card numbers and text fields. Moreover, the Delphix platform includes prepackaged profiling sets for healthcare and financial information, as well as the ability to perform tokenization: a process that can be used to obfuscate data sent for processing, then reversed when the processed data set is returned.
- **Ease of use** – With a single solution, Delphix customers can mask data across a variety of platforms. Moreover, businesses are not required to program their own masking algorithms or rely on extensive administrator involvement. Our web-based UI enables masking with a few mouse clicks and little training.
- **Automated discovery of sensitive data** – The Delphix Profiler automatically identifies sensitive data across databases and files, and the time-consuming work associated with a data masking project is reduced significantly.

⁷⁴ <https://delphixdocs.atlassian.net/wiki/spaces/CD>

4.1.3 High-level platform architecture

The Delphix DevOps Data Platform is made up of 4 main services each of which plays a very important part in delivering fresh secure data to anybody that needs it. These include:

- **Virtualize** – Delphix compresses the data that it gathers, often to one-third or more of the original size. From that compressed data footprint, Delphix virtualizes the data and allows operators to create lightweight, virtual data copies. Virtual copies are fully readable/writable and independent. They can be spun up or torn down in just minutes. And they take up a fraction of the storage space of physical copies – 10 virtual copies can fit into the space of one physical copy.
- **Identify and secure** – The Delphix platform continuously protects sensitive information with integrated data masking. Masking secures confidential data – names, email addresses, patient records, SSNs – by replacing sensitive values with fictitious, yet realistic equivalents. Delphix automatically identifies sensitive values and then applies custom or predefined masking algorithms. By seamlessly integrating data masking and provisioning into a single platform, Delphix ensures that secure data delivery is effortless and repeatable.
- **Manage** – Data operators can now quickly provision secure data copies – in minutes – to users in their target environments. The Delphix platform serves as a single point of control to manage those copies. Data operators maintain full control and visibility into downstream environments. They can easily audit, monitor, and report against access and usage.
- **Self-service** – Provides developers, testers, analysts, data scientists, or other users with controls to manipulate data at will. Users can refresh data to reflect the latest state of production, rewind environments to a prior point in time, bookmark data copies for later use, branch data copies to work across multiple releases, or easily share data with other users.



4.1.4 How Delphix identifies sensitive data

Our platform helps you quickly identify your organization's sensitive data. This sensitive data identification is done using two different methods, column-level profiling, and data-level profiling.

Column-level profiling

Column-level profiling uses REGEX expressions to scan the column names (metadata) of the selected data sources. There are several dozen pre-configured profile expressions (like the one below) designed to identify

common sensitive data types (SSN, Name, Addresses, etc). You also have the ability to write/import your own profile expressions.

First Name Expression	<code><([A-Z][A-Z0-9]*)\b[^>]*>(.*?)</1></code>
-----------------------	--

Data-level profiling

Data level profiling also uses REGEX expressions, but to scan the actual data instead of the metadata. Similar to column-level profiling, there are several dozen pre-configured expressions (like the one below) and you can write/import your own.

Social Security Number Expression	<code><([A-Z][A-Z0-9]*)\b[^>]*>(.*?)</1></code>
-----------------------------------	--

For both column and data level profiling, when data is identified as sensitive, Delphix recommends/assigns particular algorithms to be used when securing the data. The platform comes with several dozen pre-configured algorithms which are recommended when the profiler finds certain sensitive data.

4.1.5 How Delphix secures your sensitive data

Delphix strives to make available multiple methods for securing your data, depending on your needs. The two secure methods Delphix currently supports are masking (anonymization) and tokenization (pseudonymization).

Masking

Data masking secures your data by replacing values with realistic yet fictitious data. Seven out-of-the-box algorithm frameworks help businesses mask everything from names and social security numbers to images and text fields. Algorithms can also be configured or customized to match specific security policies.

Before Masking	After Masking
Elon Musk	Jeff Bezos

Tokenization

Tokenization uses reversible algorithms so that the data can be returned to its original state. Tokenization is a form of encryption where the actual data – such as names and addresses – are converted into tokens that do not convey any meaning (with regard to appearance and formatting).

Before Tokenizing	After Tokenizing
226-74-3756	asdfkajsfdaja

4.2 Data source support

The Continuous Compliance service supports profiling, masking, and tokenizing a variety of different data sources including distributed databases, mainframes, PaaS databases, and files. At a high level, Continuous Compliance breaks up support for data sources into two categories:

- **Delphix connectors:** These are data sources that the Delphix Engine can connect to directly using built-in connectors that have been optimized to perform masking, profiling, and tokenization. Delphix Connectors are available as Standard Connectors and Select Connectors. Standard Connectors are bundled with the Continuous Compliance Engine. Select Connectors are an add-on to the Delphix engine and require a separate installation and configuration process.
- **FEML sources:** FEML (File Extract Mask and Load) is a method used to mask and tokenize data sources that do not have dedicated Delphix Connectors. FEML uses existing APIs from data sources to extract the data to a file, masks the file, and then uses APIs to load the masked file back into the database.

4.2.1 Standard connectors

The Delphix Engine has standard masking connectors for the following data sources:

- **Distributed database:** Db2 LUW, Oracle, MS SQL, MySQL, SAP ASE (Sybase), PostgreSQL, MariaDB
- **Mainframe/Midrange:** Db2 Z/OS, Db2 iSeries, Mainframe data sets
- **Files:** Fixed Width, Delimited, XML

For a detailed view of all the versions, features, etc. Delphix supports each data source - see the sections below.

4.2.2 Select connectors

The Delphix Engine has Select masking connectors for the following data sources:

- **Distributed database:** Salesforce ([Compliance Accelerator](#)⁷⁵ documentation), YugabyteDB, CockroachDB, and SAP HANA 2.0 ([SAP Compliance](#)⁷⁶ documentation)

4.2.3 Db2 LUW connector

4.2.3.1 Introduction

Db2 for Linux, UNIX, and Windows is a database server product developed by IBM. Sometimes called Db2 LUW for brevity, it is part of the Db2 family of database products. Db2 LUW is the "Common Server" product member of the Db2 family, designed to run on the most popular operating systems. By contrast, all other Db2 products are specific to a single platform.

⁷⁵ <https://ecosystem.delphix.com/docs/main/compliance-accelerator-salesforce>

⁷⁶ <https://ecosystem.delphix.com/docs/main/sap-compliance-accelerator>

4.2.3.2 Support matrix

Platforms	Versions	Feature	Availability
Unix	11.1	TLS/SSL	Available
Linux	11.5	Password Vault	Available
Windows		Kerberos	Unavailable
		In-place Masking Mode	
		Multi-tenant	Available
		Streams/Threads	Available
		Batch Update	Available
		Drop Indexes	Available
		Drop Triggers	Available
		Drop Constraints	Available
		Identity Column Support	Available
		On-the-fly Masking Mode	
		Truncate	Available
		Drop Triggers	Available
		Drop Constraints	Available
		Profiling	
		Multi-tenant	Available

Platforms	Versions	Feature	Availability
		Streams	Available

4.2.3.3 TLS/SSL setup

1. Add the database's certificate in the setup application using instructions in the **Adding a certificate** section, in the [TrustStore settings](#)⁷⁷ page.
2. Restart the Compliance engine.
3. Create a Db2 connector in Continuous Compliance with the relevant parameters. Upload a properties file for the connector with the following:

```
sslConnection = True
```

4.2.4 Oracle connector

4.2.4.1 Introduction

Oracle Database (commonly referred to as Oracle RDBMS or simply as Oracle) is a multi-model database management system produced and marketed by Oracle Corporation.

4.2.4.2 Support matrix

Platforms	Versions	Feature	Availability
Unix	11.2	TLS/SSL	Available
Linux	12c	Password Vault	Available
Windows	12cR	Kerberos	Available
AWS RDS	18c	In-place Masking Mode	
OCI DBaaS on Bare Metal	19c	Multi-tenant	Available

⁷⁷ <https://cd.delphix.com/docs/latest/truststore-settings>

OCI DBaaS on VM	21c	Streams/Threads	Available
		Batch Update	Available
		Drop Indexes	Available
		Disable Triggers	Available
		Disable Constraints	Available
		Identity Column Support	Available
		On-the-fly Masking Mode	
		Truncate	Available
		Disable Triggers	Available
		Disable Constraints	Available
		Profiling	
		Multi-tenant	Available
		Streams	Available

4.2.4.3 TLS/SSL setup

1. Add the database’s certificate in the setup application using instructions in the **Adding a certificate** section, in the [TrustStore settings](https://cd.delphix.com/docs/latest/truststore-settings)⁷⁸ article.
2. Restart the Compliance engine.
3. Create an Oracle connector in Continuous Compliance with JDBC URL, similar to below examples, depending upon the Oracle database and certificate configuration.
Note: SSL connection with BASIC Oracle connector is not supported.

⁷⁸ <https://cd.delphix.com/docs/latest/truststore-settings>

```

jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=servername)
(PORT=2484))(CONNECT_DATA=(SERVICE_NAME=servicename)))
jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=servername)
(PORT=2484))(CONNECT_DATA=(SID=SID_NAME)))
jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=servername)
(PORT=2484))(CONNECT_DATA=(SID=SID_NAME))
(SEcurity=(SSL_SERVER_CERT_DN="CN=<certificate cn_name>")))

```

4.2.5 Microsoft SQL Server connector

4.2.5.1 Introduction

Microsoft SQL Server is a relational database management system developed by Microsoft. As a database server, it is a software product with the primary function of storing and retrieving data as requested by other software applications—which may run either on the same computer or on another computer across a network (including the Internet).

4.2.5.2 Support matrix

Platforms	Versions	Feature	Availability
Unix	2012	TLS/SSL	Available
Linux	2014	Password Vault	Available
Windows	2016	Kerberos	Available
AWS RDS	2017	In-place Masking Mode	
Azure SQL	2019	Multi-tenant	Available
Azure Managed Instance	2022	Streams/Threads	Available
Azure SQL Data Warehouse		Batch Update	Available
Google Cloud SQL		Drop Indexes	Available
		Disable Triggers	Available

Platforms	Versions	Feature	Availability
		Disable Constraints	Available
		Identity Column Support	Available
		On-the-fly Masking Mode	
		Truncate	Available
		Disable Triggers	Available
		Disable Constraints	Available
		Profiling	
		Multi-tenant	Available
		Streams	Available

4.2.5.3 TLS/SSL setup

1. Add the database's certificate in the setup application using instructions in the **Adding a certificate** section, in the [TrustStore settings](#)⁷⁹ article.
2. Restart the Compliance engine.
3. Create a Microsoft SQL Server connector in Continuous Compliance with the relevant parameters. Upload a properties file for the connector with the following:

```

encrypt = true
trustServerCertificate = false / true <----- Depending upon whether to directly
accept the database certificate without checking certificate common-name
hostNameInCertificate = 10-110-229-143.qa-ad.delphix.com <----- This property
is only required in case trustServerCertificate is set to false

```

⁷⁹ <https://cd.delphix.com/docs/latest/truststore-settings>

4.2.5.4 Google Cloud SQL IAM Authorization setup

To authorize connections from Continuous Compliance to a Google Cloud SQL Microsoft SQL Server instance, do the following:

1. Provision a Google Compute Engine running Continuous Compliance. In the compute engine's settings, enable *Cloud SQL* in the *Identity and API access* section.
2. Create a built-in Microsoft SQL Server connector with the following settings:
 - a. Advanced Connector with JDBC URL:

```
jdbc:sqlserver://localhost;databaseName=<your-database>;encrypt=false
```

- b. Upload a property file with the following:

```
socketFactoryClass=com.google.cloud.sql.sqlserver.SocketFactory
socketFactoryConstructorArg=<connection name of the SQL Server instance
from the Google Cloud web console>
```

4.2.6 PostgreSQL connector

4.2.6.1 Introduction

PostgreSQL, often simply Postgres, is an object-relational database management system (ORDBMS) with an emphasis on extensibility and standards compliance. PostgreSQL is developed by the PostgreSQL Global Development Group, a diverse group of many companies and individual contributors. It is free and open-source, released under the terms of the PostgreSQL License, a permissive software license.

4.2.6.2 Support matrix

Platforms	Versions	Feature	Availability
Unix	9.2	TLS/SSL	Available
Linux	9.3	Password Vault	Available
Windows	9.4	Kerberos	Unavailable
AWS RDS	9.5	In-place Masking Mode	

Platforms	Versions	Feature	Availability
AWS Aurora	9.6	Multi-tenant	Available
Azure Database for PostgreSQL	10	Streams/Threads	Available
Google Cloud SQL	11	Batch Update	Available
	12	Drop Indexes	Available
	13	Disable Triggers	Available
	14	Drop Constraints	Available
	Enterprise DB	Identity Column Support	Available
		On-the-fly Masking Mode	
		Truncate	Available
		Disable Triggers	Available
		Drop Constraints	Available
		Profiling	
		Multi-tenant	Available
		Streams	Available

4.2.6.3 TLS/SSL setup

1. Add the database's certificate in the setup application using instructions in the **Adding a certificate** section, in the [TrustStore settings](#)⁸⁰ article.
2. Restart the Compliance engine.

⁸⁰ <https://cd.delphix.com/docs/latest/truststore-settings>

3. Create a PostgreSQL connector in Continuous Compliance with the relevant parameters. Upload a properties file for the connector with the following:

```
ssl=true
sslmode=verify-full
sslfactory=org.postgresql.ssl.DefaultJavaSSLFactory
```

Note: To use the verify-full setting (highest security), the PostgreSQL database certificate's Common Name (CN) field must match the hostname. To check the CN value in the certificate: `openssl x509 -in server.crt -text -noout`

4.2.6.4 Google Cloud SQL IAM Authorization setup

To authorize connections from Continuous Compliance to a Google Cloud SQL PostgreSQL instance, do the following:

1. Provision a Google Compute Engine running Continuous Compliance. In the compute engine's settings, enable *Cloud SQL* in the *Identity and API access* section.
2. Create a built-in PostgreSQL connector with the following settings:
 - a. Host: 127.0.0.1
 - b. Port: 12345
 - c. Upload a property file with the following:

```
cloudSqlInstance=<connection name of the PostgreSQL instance from the
Google Cloud web console>
socketFactory=com.google.cloud.sql.postgres.SocketFactory
```

4.2.7 Yugabyte connector

4.2.7.1 Introduction

YugabyteDB, often simply referred to as Yugabyte, is a distributed SQL database management system with a focus on scalability, performance, and high availability. It is designed to be compatible with PostgreSQL and offers distributed ACID transactions, automatic sharding, and fault tolerance. Yugabyte is developed by Yugabyte Inc., with contributions from various companies and individual contributors. It is free and open-source software, released under the YugabyteDB License.

4.2.7.2 Support matrix

Platforms	Versions	Feature	Availability
Unix	2.18	TLS/SSL	Available
Linux		Password Vault	Available
		Kerberos	Unavailable
		In-place Masking Mode	
		Multi-tenant	Available
		Streams/Threads	Available
		Batch Update	Available
		Drop Indexes	Available
		Disable Triggers	Available
		Disable Constraints	Available
		Identity Column Support	Unavailable
		On-the-fly Masking Mode	
		Truncate	Available
		Disable Triggers	Available
		Disable Constraints	Available
		Profiling	
		Multi-tenant	Available

Platforms	Versions	Feature	Availability
		Streams	Available

4.2.7.3 TLS/SSL setup

1. Add the database's certificate in the setup application using instructions in the **Adding a certificate** section, in the [TrustStore settings](#)⁸¹ article.
2. Restart the Delphix Continuous Compliance Engine.
3. Create a YugabyteDB connector in the Delphix Continuous Compliance Engine with the relevant parameters. Upload the properties file for the connector with the following:

```
ssl=true
sslmode=verify-full
sslfactory=org.postgresql.ssl.DefaultJavaSSLFactory
```

- To use the verify-full setting (highest security), the Yugabyte database certificate's Common Name (CN) field must match the hostname. To check the CN value in the certificate, run the following command: `openssl x509 -in server.crt -text -noout`.

4.2.8 CockroachDB connector

4.2.8.1 Introduction

CockroachDB is a distributed SQL database management system with focus on scalability, performance, and high availability. It is designed to be compatible with PostgreSQL and offers distributed ACID transactions, automatic sharding, and fault tolerance. CockroachDB is an open-source software developed by Cockroach Labs.

⁸¹ <https://cd.delphix.com/docs/latest/truststore-settings>

4.2.8.2 Support matrix

Platforms	Versions	Feature	Availability
Unix	23.1	TLS/SSL	Available
Linux		Password Vault	Available
		Kerberos	Unavailable
		In-place Masking Mode	
		Multi-tenant	Available
		Streams/Threads	Available
		Batch Update	Available
		Drop Indexes	Available
		Disable Triggers	Available*
		Disable Constraints	Available*
		Identity Column Support	NA
		On-the-fly Masking Mode	
		Truncate	Available
		Disable Triggers	Available*
		Disable Constraints	Available*
		Profiling	
		Multi-tenant	Available

Platforms	Versions	Feature	Availability
		Streams	Available

= Identity column support is not applicable for CockroachDB because CockroachDB will automatically create a primary key column with the name 'rowid' if you don't define a primary key at table creation time

Triggers are not a supported feature for CockroachDB, request to add this feature is tracked under <https://github.com/cockroachdb/cockroach/issues/28296>

Primary Key and Unique constraints can't be dropped because of the issues tracked under <https://github.com/cockroachdb/cockroach/issues/48026?version=v23.1> and <https://github.com/cockroachdb/cockroach/issues/42840?version=v23.1>

4.2.8.3 TLS/SSL setup

1. Add the database's certificate in the setup application using instructions in the **Adding a certificate** section, in the [TrustStore settings](#)⁸² article.
2. Restart the Delphix Continuous Compliance Engine.
3. Create a CockroachDB connector in the Delphix Continuous Compliance Engine with the relevant parameters. Upload the properties file for the connector with the following:

```
ssl=true
sslmode=verify-full
sslfactory=org.postgresql.ssl.DefaultJavaSSLFactory
```

= To use the verify-full setting (highest security), the Cockroach database certificate's Common Name (CN) field must match the hostname. To check the CN value in the certificate, run the following command: `openssl x509 -in server.crt -text -noout`.

⁸² <https://cd.delphix.com/docs/latest/truststore-settings>

4.2.9 MySQL / MariaDB connector

4.2.9.1 Introduction

MySQL is an open-source relational database management system (RDBMS). MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB. MySQL is now owned by Oracle Corporation.

MariaDB is a community-developed fork of the MySQL relational database management system intended to remain free under the GNU GPL. Development is led by some of the original developers of MySQL, who forked it due to concerns over its acquisition by Oracle Corporation.

A MySQL Connector may be used to connect to either a MySQL or MariaDB database instance.

4.2.9.2 MySQL support matrix

Platforms	Versions	Feature	Availability
Unix	5.5	TLS/SSL	Available
Linux	5.6	Password Vault	Available
Windows	5.7*	Kerberos	Unavailable
AWS RDS	8	In-place Masking Mode	
AWS Aurora		Multi-tenant	Available
Azure Database for MySQL		Streams/Threads	Available
Google Cloud SQL		Batch Update	Available
		Drop Indexes	Available
		Disable Triggers	Unavailable
		Disable Constraints	Unavailable
		Identity Column Support	Available

Platforms	Versions	Feature	Availability
		On-the-fly Masking Mode	
		Truncate	Available
		Disable Triggers	Unavailable
		Disable Constraints	Unavailable
		Profiling	
		Multi-tenant	Available
		Streams	Available

NOTE: AWS Aurora MySQL 5.7 is not supported

4.2.9.3 Google Cloud SQL IAM Authorization setup

To authorize connections from Continuous Compliance to a Google Cloud SQL MySQL instance, do the following:

1. Provision a Google Compute Engine running Continuous Compliance. In the compute engine’s settings, enable *Cloud SQL* in the *Identity and API access* section.
2. Create a built-in MySQL connector with the following settings:
 - a. Host: igoreme
 - b. Port: 123
 - c. Upload a property file with the following:

```
socketFactoryClass=com.google.cloud.sql.mariadb.SocketFactory
socketFactoryConstructorArg=<connection name of the SQL Server instance from the Goog
```

4.2.9.4 MariaDB support matrix

Platforms	Versions	Feature	Availability
Unix	10	TLS/SSL	Available

Platforms	Versions	Feature	Availability
Linux		Password Vault	Available
Window		Kerberos	Unavailable
AWS RDS		In-place Masking Mode	
AWS Aurora		Multi-tenant	Available
Azure Database for MariaDB		Streams/Threads	Available
		Batch Update	Available
		Drop Indexes	Available
		Disable Triggers	Unavailable
		Disable Constraints	Unavailable
		Identity Column Support	Available
		On-the-fly Masking Mode	
		Truncate	Available
		Disable Triggers	Unavailable
		Disable Constraints	Unavailable
		Profiling	
		Multi-tenant	Available
		Streams	Available

4.2.9.5 TLS/SSL setup

1. Add the database's certificate in the setup application using instructions in the **Adding a certificate** section, in the [TrustStore settings](#)⁸³ article.
2. Restart the Compliance engine.
3. Create a MySQL/MariaDB connector in Continuous Compliance with the relevant parameters. Upload a properties file for the connector with the following:

```
useSSL=True
trustServerCertificate=false
keyStore=file:/var/delphix/server/etc/.truststore
keyStoreType=JKS
```

4.2.10 SAP ASE (Sybase) connector

4.2.10.1 Introduction

SAP ASE (Adaptive Server Enterprise), originally known as Sybase SQL Server, and also commonly known as Sybase DB or Sybase ASE, is a relational model database server product for businesses developed by Sybase Corporation which became part of SAP AG.

4.2.10.2 Support matrix

Platforms	Versions	Feature	Availability
Unix	15.5	TLS/SSL	Available
Linux	15.7	Password Vault	Available
Windows	16	Kerberos	Available
		In-place Masking Mode	
		Multi-tenant	Available

⁸³ <https://cd.delphix.com/docs/latest/truststore-settings>

Platforms	Versions	Feature	Availability
		Streams/Threads	Available
		Batch Update	Available
		Drop Indexes	Available
		Disable Triggers	Available
		Disable Constraints	Available
		Identity Column Support	Available
		On-the-fly Masking Mode	
		Truncate	Available
		Disable Triggers	Available
		Disable Constraints	Available
		Profiling	
		Multi-tenant	Available
		Streams	Available

4.2.10.3 TLS/SSL setup

1. Add the database's certificate in the setup application using instructions in the **Adding a certificate** section, in the [TrustStore settings](#)⁸⁴ article.
2. Restart the Compliance engine.
3. Create a Sybase connector in Continuous Compliance with the relevant parameters. Upload a properties file for the connector with the following contents:

⁸⁴ <https://cd.delphix.com/docs/latest/truststore-settings>

```
ENABLE_SSL=true
```

4.2.11 Db2 z/OS and iSeries connectors

4.2.11.1 Introduction

Db2 for z/OS and iSeries are relational database management systems that run on IBM Z (mainframe) and IBM Power Systems.

4.2.11.2 Support matrix

iSeries	z/OS	Feature	Availability
7.1	11	TLS/SSL	Available
7.2	12	Password Vault	Available
7.3		Kerberos	Unavailable
7.4		In-place Masking Mode	
		Multi-tenant	Available
		Streams/Threads	Available
		Batch Update	Available
		Drop Indexes	z/OS: Unavailable iSeries: Available
		Disable/Drop Triggers	z/OS: Available iSeries: Available v7.2+
		Drop Constraints	Available
		Identity Column Support	Available

iSeries	z/OS	Feature	Availability
		On-the-fly Masking Mode	
		Truncate	Available
		Disable/Drop Triggers	z/OS: Available iSeries: Available v7.2+
		Drop Constraints	Available
		Profiling	
		Multi-tenant	Available
		Streams	Available

4.2.11.3 TLS/SSL setup

See the instructions in the Db2 LUW connector section

4.2.12 Files connector

4.2.12.1 Introduction

Data stored in a variety of different formats may be masked using the same algorithms available for other data sources.

4.2.12.2 Support matrix

File type/format	Supported encodings	Support level
Fixed Width	ASCII, UTF-8	Supported
Delimited	ASCII, UTF-8	Supported
XML	ASCII, UTF-8	Supported

File type/format	Supported encodings	Support level
JSON	ASCII, UTF-8	Supported

4.2.13 Mainframe data set connector

4.2.13.1 Introduction

In addition to databases and files, the Continuous Compliance Engine can process data stored in Mainframe data sets commonly found on the IBM z/OS operating system. For more information on data sets, see this [IBM knowledge center article](#)⁸⁵.

4.2.13.2 Support matrix

The Continuous Compliance Engine requires that data be encoded in EBCDIC rather than something like ASCII or UTF-8. EBCDIC is the encoding traditionally used on Mainframes.

4.2.14 On-The-Fly masking jobs

Continuous Compliance supports **On-The-Fly** (OTF) masking jobs where the data is read from a source location and written to a different target location. Only certain combinations of connector types are supported for OTF jobs.

OTF jobs with connectors of the same type are supported. For example, masking data from an Oracle source database to an Oracle target database is supported if both are using the built-in Oracle connector. OTF jobs using Extended Connectors are supported if both the source and target are using the same Extended Driver (the same uploaded JDBC driver). Additionally, OTF jobs with a relational database source and a delimited file target are supported. The following data sources are supported as source connectors for OTF jobs with delimited file targets.

- Oracle
- Db2
- MS SQL
- PostgreSQL
- MySQL / MariaDB
- SAP ASE (Sybase)
- Connectors created as [Extended Connectors](#) (see page 389).

For masking flat files (e.g. XML, delimited, etc) in an on-the-fly masking job, it is no longer required to copy or create empty files on the target. If the file name pattern does not match any file on the source, the execution will reported as success, although no file is masked.

No other combinations of connector types are supported. For example, an Oracle source with a PostgreSQL target, or an MS SQL source with a fixed-width file target, are unsupported.

⁸⁵ https://www.ibm.com/support/knowledgecenter/zosbasics/com.ibm.zos.zconcepts/zconc_datasetintro.htm

4.3 Installation

This section covers the following articles:

- [Containerized installation](#) (see page 219)
- [Network connectivity requirements](#) (see page 177)
- [Prerequisites](#) (see page 175)
- [First time setup](#) (see page 180)
- [AWS EC2 installation](#) (see page 183)
- [Azure installation](#) (see page 187)
- [Google Cloud platform installation](#) (see page 190)
- [IBM Cloud platform installation](#) (see page 197)
- [Hyper-V installation](#) (see page 192)
- [OCI installation](#) (see page 208)
- [VMware installation](#) (see page 214)
- [Naming requirements](#) (see page 228)

4.3.1 Prerequisites

4.3.1.1 VM-based Continuous Compliance Engines

This page details the hardware and software requirements needed to deploy the Continuous Compliance Engine for data masking. The Continuous Compliance Engine is a self-contained operating environment and application, provided as a Virtual Appliance, which is certified to run on a variety of platforms like VMware, AWS, and Azure.

The Continuous Compliance Engine should be placed on a server where it will not contend with other virtual machines for network, storage, or other computing resources. The Continuous Compliance Engine is a CPU and I/O-intensive application; deploying it in an environment where it must share resources can significantly reduce performance.

To use both Continuous Data (data virtualization) and Continuous Compliance (data masking) Engines, they must be deployed separately. One Delphix Engine is required per service, running both operations on one engine is not supported.

4.3.1.1.1 Web browser support

The Continuous Compliance Engine's graphical interface can be accessed from a variety of different web browsers. Currently, the following web browsers are supported.

- Microsoft Edge 40.x or higher
- Mozilla Firefox 35.0 or higher
- Chrome 40 or higher

4.3.1.1.2 AWS EC2 platform

See [AWS EC2 Installation \(see page 183\)](#) for information about virtual machine requirements to install a dedicated Continuous Compliance Engine on Amazon's Elastic Cloud Compute (EC2) platform.

4.3.1.1.3 Azure platform

See [Azure Installation \(see page 187\)](#) for information about virtual machine requirements to install a dedicated Continuous Compliance Engine on the Azure platform.

4.3.1.1.4 Google cloud platform

See [Google Cloud Platform Installation \(see page 190\)](#) for information about virtual machine requirements to install a dedicated Continuous Compliance Engine on the GCP platform.

4.3.1.1.5 IBM Cloud

See [IBM Cloud Installation \(see page 197\)](#) for information about virtual machine requirements to install a dedicated Continuous Compliance Engine on the IBM Cloud.

4.3.1.1.6 VMware platform

See [VMware installation \(see page 214\)](#) for information about virtual machine requirements to install a dedicated Continuous Compliance Engine on the VMware Virtual platform.

4.3.1.2 Container (Kubernetes) based Continuous Compliance Engine

The Containerized Masking product is delivered as a set of containers that are deployed as a Pod in your Kubernetes infrastructure. The Pod provides a similar set of functionality as the Continuous Compliance VM-based appliance.

Containerized Masking was developed to provide the ability to create ephemeral engines, or small engines that can be spun up quickly for a specific need, then thrown away once that need is fulfilled.

You would need to provide the Kubernetes infrastructure whether it is on-prem (such as MiniKube or MicroK8s) or cloud-based (such as AWS EKS).

Some functionality may require additional software installations on the Kubernetes node systems. For example, if the use of NFS-mounted filesystems is planned, each node would need the NFS client software that allows Kubernetes to perform the desired NFS mounts.

4.3.1.3 Differences between VM-based and Container-based Engines

There are some functional differences between the VM-based engine and the Container-based engine. In some cases, a piece of functionality is available in both, but implemented in different ways. In other cases, the functionality is unavailable. The table below summarizes these differences.

Functionality	VM-based engine	Container-based engine
FTP support for file masking	Yes	No
SSL / TLS for connectors	Yes ¹	Yes ²
Local file masking via NFS	Yes ¹	Yes ²
Local file masking via CIFS	Yes ¹	Yes ²
LDAP over TLS/SSL	Yes ¹	Yes ²
Kerberos	Yes ¹	No
SSO / OAuth	Yes ¹	No
Upgrade / upgrade validation	Yes ¹	No
Db2 z/OS and iSeries Connector	Yes	No



1. Via the Engine Setup app.
2. Via Kubernetes.

4.3.2 Network connectivity requirements

This topic covers the general network and connectivity requirements, including connection requirements, port allocation, and firewall and Intrusion Detection System (IDS) considerations.



A security mechanism exists that does not allow the Masking engine to deploy behind a reverse proxy on the network.

4.3.2.1 General outbound connections from the virtual machine Delphix Continuous Compliance Engine

Protocol	Port Numbers	Use
TCP	25	Connection to a local SMTP server for sending email.
TCP/UDP	53	Connections to local DNS servers.
UDP	123	Connection to an NTP server.
UDP	162	Sending SNMP TRAP messages to an SNMP Manager.
TCP	443	HTTPS connections from the Delphix Engine to the Delphix Support upload server.
TCP/UDP	636	Secure connections to an LDAP server.
TCP/UDP	various	Connections to target environments such as databases (JDBC) and files (FTP, SFTP, NFS, or CIFS).

4.3.2.2 General inbound connections to the virtual machine Delphix Continuous Compliance Engine


Protocol	Port Numbers	Use
TCP	22	SSH connections to the Delphix Engine.
TCP	80	HTTP connections to the Delphix GUI (optional).
UDP	161	Messages from an SNMP Manager to the Delphix Engine.
TCP	443	HTTPS connections to the Delphix GUI.

4.3.2.3 General outbound connections from the containerized Delphix Continuous Compliance Engine

Containerized Masking is deployed as a Pod on a customer Kubernetes infrastructure rather than being a self-contained machine like the VM deployments. There is much underlying infrastructure (NTP, for example) that the VM deployment must manage, which is unnecessary for a containerized deployment. There are many features (again using time as one example) that a containerized deployment requires from the underlying infrastructure, but since they are no longer managed by the Pod itself, they no longer appear in the list of networking requirements.

Protocol	Port Numbers	Use
TCP	25	Connection to a local SMTP server for sending email.
TCP/UDP	53	Connections to local DNS servers.
TCP/UDP	various	Connections to target environments such as databases (JDBC) and files (FTP, SFTP, NFS, or CIFS).

4.3.2.4 General inbound connections to the containerized Delphix Continuous Compliance Engine

 The inbound ports shown in the table below are all internal. The kubernetes config defines a service that routes customer supplied external facing ports to the listed internal ports allowing the customer to choose any ports that work best for their infra. The example config maps external port 30080 to internal port 8080 and external port 30443 to internal port 8443, but that is left entirely to customer discretion.

Protocol	Port Numbers	Use
TCP	8080	HTTP connections to the Delphix GUI (optional).
TCP	8443	HTTPS connections to the Delphix GUI.

4.3.2.5 Firewalls and Intrusion Detection Systems (IDS)

Firewalls can add milliseconds to the latency between servers. Accordingly, for best performance, there should be no firewalls between the Delphix Masking Engine and the target environments. If the Delphix Masking Engine is separated from a target environment by a firewall, the firewall must be configured to permit network connections between the Delphix Masking Engine and the target environments for the application protocols (ports) listed above.

Intrusion detection systems (IDSs) should also be made permissive to the Delphix Masking Engine deployment. IDSs should be made aware of the anticipated high volumes of data transfer between the Delphix Masking Engine and target environments.

4.3.3 First time setup

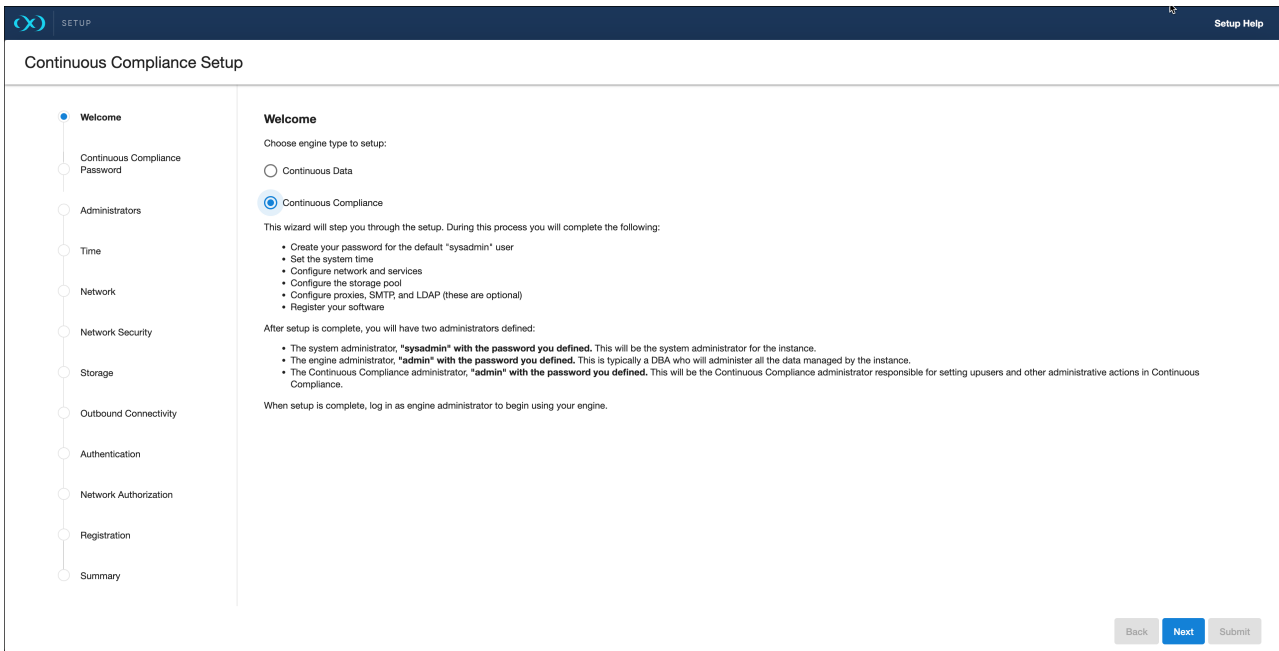
This section walks you step by step on how to download and install the Delphix Engine software onto your infrastructure (VMware, AWS EC2, Azure, or GCP).

4.3.3.1 Setting up network access to the Delphix Engine

1. Power on the Delphix Engine and open the Console.
2. Wait for the Delphix Management Service and Delphix Boot Service to come online. This might take up to 10 minutes during the first boot. Wait for the large orange box to turn green.
3. Press any key to access the sysadmin console.
4. Enter **sysadmin** for the username and **sysadmin** for the password (when installing a new engine via AWS AMI, the initial sysadmin password is the AWS Instance ID).
5. You will be presented with a description of available network settings and instructions for editing.
6. Configure the hostname. Use the same hostname you entered during the server installation. If you are using DHCP, this step can be skipped.
7. Configure DNS. If you are using DHCP, this step can be skipped.
8. Configure either a static or DHCP address. The static IP address must be specified in CIDR notation (for example, 192.168.1.2/24).
9. Configure a default gateway. If you are using DHCP, this step can be skipped.
10. Commit your changes. Note that you can use the get command prior to committing to verify your desired configuration.
11. Check that the Delphix Engine can now be accessed through a Web browser by navigating to the displayed IP address, or hostname if using DNS.
12. Exit setup.

4.3.3.2 Setting up the Delphix Engine

Once you setup the network access for your Delphix Engine, enter the Delphix Engine URL in your browser for server setup. The Unified Setup wizard Welcome screen below will appear for you to begin your Delphix Engine setup.



The Welcome page allows you to setup Masking-specific settings such as Masking admin user's email and password as well as Masking SMTP settings directly from the setup wizard. It will then redirect the customer to the corresponding login page based on the engine type selected.

When Masking is selected, the following will be added to the Welcome screen; "admin" with the password you defined. This will be the Masking administrator responsible for setting up users and other administrative actions in Masking.

There are limitations to this feature:

- Only Masking user settings (email and password) and SMTP settings are supported. Customers will need to use the API to setup LDAP.
- Once set, these settings can only be updated via the Masking API. There are no corresponding sections in the system dashboard.
- Engine Type cannot be modified once set in the Setup Wizard because it has other dependencies such as SSO.



If the wrong password is entered, after 3 times the user will be locked out of the Masking service.

1. On the **Welcome** tab select **Masking** and then click **Next**.

2. In the Masking Password tab enter the current default (out-of-box) password for Masking. (Currently, the default is **Admin-12**)
3. Click **Validate** or **Next**. This causes the engine to validate the entered password with the masking service.
4. In the Administrators tab enter **System Administrator, Masking Administrator, and Engine Administrator** credentials. Then click **Next**.
5. Select an option for maintaining system time. Then click **Next**.
Note: The Masking engine only works with the UTC time zone. Time zone selection at the time of engine setup is not applicable for the Masking engine.
6. Configure your network interfaces and services and then select **Next**.
7. Delphix installs certificates signed by the Engines Certificate Authority. You can replace any certificate. Once you are ready click **Next**.
8. The Delphix Engine automatically discovers and displays storage devices. For each device, set the Usage Assignment to Data and set the Storage Profile to Striped. Then click **Next**.
9. Enter the **Masking SMTP** settings and then click **Next**.
10. The Authentication tab allows users to configure Virtualization LDAP settings. But Masking LDAP settings must be configured via the Masking API.
11. To enable SAML/SSO, set the Audience Restriction (SP entity ID, Partner’s Entity ID) in the identity provider to be the Engine UUID. Select **Use SAML/SSO**. IdP metadata is an XML document which must be exported from the application created in your IdPCopy and pasted in the IdP Metadata field. Click **Next**.
12. If using Kerberos authentication select **Use Kerberos authentication** and complete all fields. Then enter **Next**.
13. If the Delphix Engine has access to the external Internet (either directly or through a web proxy), then you can auto-register the Delphix Engine. If external connectivity is not immediately available, you must perform manual registration. Copy the Delphix Engine registration code.
14. Click **Next**.
15. The final Summary tab will enable you to review your configuration. Click **Submit** to acknowledge the configuration.

4.3.3.3 Logging in to the Delphix Continuous Compliance Engine

1. Login to a web browser that points to `http://masking-engine.example.com/masking`.
2. Enter default username: `admin`.
3. Enter default user password: `Admin-12`.

4.3.4 AWS EC2 installation

This section covers the virtual machine requirements for installation of a dedicated Continuous Compliance Engine on Amazon's Elastic Cloud Compute (EC2) platform.

For best performance, the Continuous Compliance Engine and all database/file servers should be in the same AWS region.


The following topics are covered:

- Instance Types
- Network Configuration
- EBS Configuration
- General Storage Configuration
- Additional AWS Configuration Notes

4.3.4.1 Instance types

The Continuous Compliance Engine can run on a variety of different instances, including large memory instances (preferred) and high I/O instances. We recommend the following large memory and high I/O instances:

Requirements	Notes
Large Memory Instances: r5n.2xlarge r5n.4xlarge r5n.8xlarge r5n.16xlarge r5n.24xlarge r4.2xlarge r4.4xlarge r4.8xlarge r4.16xlarge	- Larger instance types provide more CPU, which can prevent resource shortfalls under high I/O throughput conditions. - Larger instances also provide more memory, which the Delphix Engine uses to cache database blocks. More memory will provide better read performance.
High I/O Instances (supported) i3.2xlarge i3.4xlarge i3.8xlarge	

 On the AWS EC2 platform, the Continuous Compliance Engine must have sufficient memory to operate when multiple masking jobs are running. Our recommendation is to provide 8 GB of memory for the Continuous Compliance Engine in addition to any memory that will be used by running jobs.

4.3.4.2 Network configuration

Requirements	Notes
Virtual Private Cloud	<ul style="list-style-type: none"> - You must deploy the Delphix Engine and all of the source and target environments in a VPC network to ensure that private IP addresses are static and do not change when you restart instances. - When adding environments to the Delphix Engine, you must use the host's VPC (static private) IP addresses.
Static Public IP	The EC2 Delphix instance must be launched with a static IP address; however, the default behavior for VPC instances is to launch with a dynamic public IP address – which can change whenever you restart the instance. If you're using a public IP address for your Delphix Engine, static IP addresses can only be achieved by using assigned AWS Elastic IP Addresses.
Security Group Configuration	The default security group will only open port 22 for SSH access. You must modify the security group to allow access to all of the networking ports used by the Delphix Engine and the various source and target engines.


4.3.4.3 Storage configurations



You must always attach a minimum of 2 storage pools to the Delphix Engine; one for rpool and other for domain0 pool.

4.3.4.3.1 EBS configuration

Deploying Delphix on AWS EC2 requires EBS-provisioned IOPS volumes. Since EBS volumes are connected to EC2 instances via the network, other network activity on the instance can affect throughput to EBS volumes. EBS-optimized instances provide guaranteed throughput to EBS volumes and are required to provide consistent and predictable storage performance.

Requirements	Notes
<p>EBS Provisioned IOPS Volumes</p> <div style="border: 1px solid purple; padding: 10px; margin-top: 10px;">  All attached storage devices must be EBS volumes. </div>	<ul style="list-style-type: none"> - Delphix does not support the use of instance store volumes. - Use EBS volumes with provisioned IOPS in order to provide consistent and predictable performance. The number of provisioned IOPS depends on the estimated IO workload on the Delphix Engine. - Provisioned IOPS volumes must be configured with a volume size to provisioned IOPS per the EBS Volume Types⁸⁶ guidelines. - I/O requests of up to 256 kilobytes (KB) are counted as a single I/O operation (IOP) for provisioned IOPS volumes. Each volume can be configured for up to 4,000 IOPs.

4.3.4.3.1.1 System disk

The minimum recommended storage size for the System Disk is 127 GB.

4.3.4.3.1.2 Metadata disk(s)

The minimum recommended storage size of the Metadata Volume is 50 GB.

4.3.4.3.2 General storage configuration

Requirements	Notes
<ul style="list-style-type: none"> - Allocate initial storage equal to the size of the physical source database storage. - Add storage when storage capacity approaches 30% free. 	<ul style="list-style-type: none"> - For high redo rates and/or high DB change rates, allocate an additional 10-20 %. - Add new storage by provisioning new volumes of the same size. - This enables the Delphix File System (DxFS) to make sure that its file systems are always consistent on disk without additional serialization. This also enables the Delphix Engine to achieve higher I/O rates by queueing more I/O operations to its storage.
<p>EBS Volume Size and Count</p> <ul style="list-style-type: none"> - Keep all EBS volumes the same size. Maximize Delphix Engine RAM for a larger system cache to service reads - Use at least 4 EBS volumes to maximize performance. 	<p>This enables the Delphix File System (DxFS) to make sure that its file systems are always consistent on disk without additional serialization. This also enables the Delphix Engine to achieve higher I/O rates by queueing more I/O operations to its storage.</p>

⁸⁶ <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-volume-types.html>

4.3.4.4 Additional AWS configuration notes

- Using storage other than EBS is not supported.
- Limits on the number of volumes are dictated by the EBS instance type, and is generally advised that over 40 can be expected to cause issue on Linux VMs. More information can be found in the [AWS Volume Limits](#)⁸⁷ and [AWS Volume Constraints](#)⁸⁸ articles. The maximum device limit imposed by AWS can be handled by the Delphix Engine.
- The use of the local SSDs attached to i2 instance types is not supported.
- Using fast storage for EBS volumes is supported and recommended, including (in order of decreasing speed):
 - Provisioned IOPS (io1) volumes (recommended).
 - Virtual Machine Requirements for AWS EC2 Platform
 - General Purpose SSD (gp2) volumes (supported)
 - Throughput Optimized HDD (st1) volumes (supported)
 - Cold HDD (sc1) volumes (not supported due to poor performance)
 - Magnetic (standard) volumes (supported, but use st1 instead where possible)

4.3.4.5 Installing AMI on AWS EC2

The following two methods can be used to install/deploy Continuous Compliance in AWS:

- Access Delphix provided AMI through the Delphix download site
- Subscribe to Continuous Compliance through the Amazon Marketplace

4.3.4.5.1 Using the Delphix download site to deploy masking

1. On the Delphix download site, click the AMI you would like to share and accept the Delphix License agreement. Alternatively, follow a link given by your Delphix solutions architect.
2. On the **Amazon Web Services Account**Details form presented:
 - a. Enter your **AWS Account Identifier**, which can be found here: <https://console.aws.amazon.com/billing/home?#/account>. If you want to use the GovCloud AWS Region, be sure to enter the ID for the AWS Account which has GovCloud enabled.
 - b. Select which **AWS Region** you would like the AMI to be shared in. If you would like the AMI shared in a different region, contact your Delphix account representative to make the proper arrangements.
3. Click **Share**. The Delphix Engine will appear in your list of AMIs in AWS momentarily.
4. Reference the Installation and Configuration Requirements for AWS/EC2 when deploying the AMI.

⁸⁷ https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/volume_limits.html

⁸⁸ https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/volume_constraints.html

- Once you have launched your Continuous Compliance EC2 instance and it is accessible via a web browser (port 80), proceed to [First time setup \(see page 180\)](#) to configure the system.

4.3.4.5.2 Subscribing to Continuous Compliance through the Amazon Marketplace

- Sign in to the AWS Console.
- Navigate to AWS Marketplace.
- Typing Delphix in the search bar will find several Delphix Product offerings. Select **Continuous Compliance for AWS (3TB)**.
- Click **Continue to Subscribe**.
- Click **Accept Terms**.
- Wait for the subscription to be confirmed, then click **Continue to Configuration**.
- Select or verify the correct **Region** for launch/deployment.
- Then click **Continue to Launch**.
- Select either to **Launch from Website** or **Launch through EC2**.
- For either option you will need to enter the following:
 - VPC in which to launch the instance.
 - Subnet on which the instance will reside.
 - Instance Type (Recommended: r4.2xlarge).
 - Security Group (Minimal access required: 22, 80, or 443)
- Once the Delphix EC2 instance is launched proceed to [Setting up the Delphix Engine \(see page 180\)](#) to configure the system.

4.3.5 Azure installation

This topic covers the virtual machine requirements, including memory and data storage, for deploying the Delphix Engine on the Azure public cloud and Government Cloud.

4.3.5.1 Instance types

The Delphix Engine can run on a variety of different Azure instances. We recommend the following instances:

Requirements	Notes
Memory-Optimizes	

Requirements	Notes
DS14v2 E8S_v3 E16S_v3 E32S_v3	16 CPUs, 112GB, 32 devices 8 CPUs, 112GB, 16 devices 16 CPUs, 244GB, 32 devices 32 CPUs, 448GB, 64 devices Network bandwidth and IOPS limits are specific to each instance type: - See DSv2 specifications ⁸⁹ for more details. - See GS specifications ⁹⁰ for more details.
General Purpose	
D16s_v3 D32_v3	Network bandwidth and IOPS limits are specific to each instance type: - See DSv2 specifications ⁹¹ for more details. - See https://docs.microsoft.com/en-us/azure/virtual-machines/dv3-dsv3-series for more details.



On the Azure platform, recommendation is to provide 8 GB of memory for the Delphix Masking Engine in addition to any memory that will be used by running jobs.

4.3.5.2 Network configuration

Requirements	Notes
Azure Virtual Network (VNet)	The Delphix Engine and all the source and target environments must be accessible within the same virtual network.
Network Security Group (NSG)	You must modify the security group to allow access to all of the networking ports used by the Delphix Engine and the various source and target platforms.

See [Network connectivity requirements](#) (see page 177) for information about specific port configurations.

⁸⁹ <https://docs.microsoft.com/en-us/azure/virtual-machines/dv2-dsv2-series>

⁹⁰ <https://docs.microsoft.com/en-us/azure/virtual-machines/sizes-previous-gen>

⁹¹ <https://docs.microsoft.com/en-us/azure/virtual-machines/dv2-dsv2-series>

4.3.5.3 Storage configuration

ⓘ You must always attach a minimum of 2 storage pools to the Delphix Engine; one for rpool and other for domain0 pool.

Delphix recommends using four equally sized storage disks to run the Delphix Engine. This allows the Delphix File System (DxFS) to assure its file systems are always consistent on disk without additional serialization. This also enables the Delphix Engine to achieve higher I/O rates by queueing more I/O operations to its storage.

Requirements	Notes
Azure Premium Storage	<ul style="list-style-type: none"> - Premium storage utilizes solid-state drives (SSDs) - Devices up to 4096GB are supported - Maximum of 256TB is supported - I/O requests of up to 256 kilobytes (KB) are counted as a single I/O operation (IOP) for provisioned IOPS volumes - IOPS vary based on storage size with a maximum of 7,500 IOPS
System Disk	The minimum recommended storage size for the System Disk is 127 GB.
Metadata Disk(s)	The minimum recommended storage size of the Metadata Volume is 50 GB.

4.3.5.4 Extensions

Extensions are not currently supported.

4.3.5.5 Installing VHD on AZURE

Use the following steps to install your VHD:

1. On the [Microsoft Azure Marketplace](https://azuremarketplace.microsoft.com/en-us/marketplace/apps/delphix.delphix_dynamic_data_platform?tab=Overview)⁹², search for Delphix. Click **GET IT NOW**.
2. Reference the Installation and Configuration Requirements for the Delphix Engine in Azure when deploying the VHD.
3. Jump to [Setting up the Delphix Engine \(see page 180\)](#) section to learn how to activate the masking service now that you have the software installed.

⁹² https://azuremarketplace.microsoft.com/en-us/marketplace/apps/delphix.delphix_dynamic_data_platform?tab=Overview

4.3.6 Google Cloud Platform installation

This section covers the virtual machine requirements for the installation of a dedicated Continuous Compliance Engine on Google Cloud Platform (GCP).

4.3.6.1 Machine types

The following is a list of instance types that are supported to deploy Delphix on GCP. Delphix periodically certifies new instance types, which will be added to the list here.

Requirements	Notes
n2-standard-(16, 32, 64)	Larger instance types provide more CPU, which can prevent resource shortfalls under high I/O throughput conditions.
n2-highmem-(8, 16, 32, 64)	Larger instances also provide more memory, which the Delphix Engine uses to cache database blocks. More memory will provide better read performance.

4.3.6.2 Network configuration

Requirements	Notes
Virtual Private Cloud	You must deploy the Delphix Engine and database/file servers in a VPC network to ensure that private IP addresses are static and do not change when you restart instances. When adding connectors to the Masking Engine, you must use the host's VPC (static private) IP addresses.
Static Public IP	The GCP Delphix instance must be launched with a static IP address; however, the default behavior for VPC instances is to launch with a dynamic public IP address – which can change whenever you restart the instance.
Security Group Configuration	The default security group will only open port 22 for SSH access. You must modify the security group to allow access to all of the networking ports used by the Delphix Engine and the various source and target engines.
Premium Networking	It is recommended to use GCP Premium Tier Networking.

4.3.6.3 Storage configuration



You must always attach a minimum of 2 storage pools to the Delphix Engine; one for rpool and other for domain0 pool.

4.3.6.3.1 System disk

The minimum recommended storage size for the System Disk is 127 GB.

4.3.6.3.2 Metadata disk(s)

The minimum recommended storage size of the Metadata Volume is 50 GB.

4.3.6.4 Additional GCP configuration notes

- Delphix supports both Zonal and Regional SSD persistent disks.

4.3.6.5 Installing on Google Cloud Platform

This section covers the requirements, including memory and data storage, for deploying the Delphix Engine on the Google Cloud Platform (GCP).

4.3.6.5.1 Prerequisites to deploying in GCP

- A license is required to use the Delphix software. If you are a new customer contact Delphix to get started.

4.3.6.5.2 Deploying a Delphix Engine in GCP

1. Log into Google Cloud Marketplace with your account.
2. Search for **Delphix**.
3. Click **Launch on Compute Engine**.
 - Machine Type: See the table below for supported configurations.
 - Boot disk type: SSD Persistent Disk
 - Boot disk size in GB: 127
 - Networking interfaces: Configure as appropriate for your environment

- IP forwarding: Configure as appropriate for your environment
4. Click on **Deploy**.
 5. Once deployed, go to [Setting up the Delphix Engine \(see page 180\)](#) section to learn how to activate the masking service now that you have the software installed.

4.3.7 Hyper-V installation

The Delphix Engine is a virtual appliance that runs in a hypervisor. In this section, you'll find requirements to run Delphix on Hyper-V including supported versions and instance configurations as well as recommended configuration parameters for optimal performance.

Contact your Delphix representative to request this capability. Delphix will assist you to review that all Hyper-V requirements are met to successfully run a Delphix Engine with the most appropriate configuration for your Use Cases.

If the Delphix Engine competes with other virtual machines on the same host for resources it will result in increased latency for all operations. As such, it is crucial that your Hyper-V host is not over-subscribed, as this eliminates the possibility of a lack of resources for the Delphix Engine. This includes allowing a percentage of CPU resources for the hypervisor itself as it can de-schedule an entire VM if the hypervisor is needed for managing IO or compute resources.

4.3.7.1 Supported versions

- Hyper-V Version: 10.0 and later
- Gen 1 only is supported

4.3.7.2 Virtual CPUs

Requirements	Notes
8 vCPUs	<ul style="list-style-type: none"> - CPU resource shortfalls can occur both on an over-committed host as well as competition for host resources during high IO utilization. - CPU reservations are strongly recommended for the Delphix VM so that Delphix is guaranteed the full complement of vCPUs even when resources are overcommitted. - It is suggested to use a single core per socket unless there are specific requirements for other VMs on the same Hyper-V host.
Never allocate all available physical CPUs to virtual machines	<ul style="list-style-type: none"> - CPU for the Hyper-V Server to perform hypervisor activities must be set aside before assigning vCPUs to Delphix and other VMs. - We recommend that a minimum of 8-10% of the CPUs available are reserved for hypervisor operation. (e.g. 12 vCPUs on a 128 vCore system).


4.3.7.3 Memory

Requirements	Notes
<p>128 or higher GB vRAM (recommended) 64GB vRAM (minimum)</p>	<ul style="list-style-type: none"> - The masking service on the Delphix Engine uses its memory to process database and file blocks. - Memory reservations are required for the Delphix VM. The performance of the Delphix Engine will be significantly impacted by the over-commitment of memory resources in the Hyper-V Server. - Reservations ensure that the Delphix Engine will not be forced to swap pages during times of memory pressure on the host. A swapped page will require orders of magnitude more time to be brought back to physical memory from the Hyper-V swap device.
<p>Memory for the Hyper-V Server to perform hypervisor activities must be set aside before assigning memory to Delphix and other VMs.</p>	<p>Failure to ensure sufficient memory for the host can result in a hard memory state for all VMs on the host which will result in a block for memory allocations.</p>


4.3.7.4 Network

Requirements	Notes
<p>Virtual ethernet adapter requirements.</p>	<ul style="list-style-type: none"> - SR-IOV recommended for all virtual ethernet adapters that will be used for Delphix data IO. - Jumbo frames recommended. - A 10GbE NIC in the Hyper-V Server is recommended.
<p>If the network load in the Hyper-V Server hosting the Delphix engine VM is high, dedicate one or more physical NICs to the Delphix Engine.</p>	<ul style="list-style-type: none"> - Adding NICs only works if VMs are discovered using different interfaces.

4.3.7.5 SCSI Controller

Requirements	Notes
LSI Logic Parallel	<p>- Per Hyper-V Storage I/O Performance Tuning Guidelines⁹³, it is recommended that you attach multiple disks to a single virtual SCSI controller and create additional controllers only as they are required to scale the number of disks connected to the virtual machine. For example, a VM with 3 virtual disks should distribute the disks across the single SCSI controller as follows:</p> <ul style="list-style-type: none"> - IDE Controller 1 - Boot Drive - SCSI Controllers - Disk 1, Disk 2, Disk 3 <div style="border: 1px solid purple; padding: 10px; margin-top: 10px;"> <p> For load purposes, we generally focus on the Delphix storage (data disks) and ignore the controller placement of the system disk.</p> </div>

4.3.7.6 Storage configuration

 You must always attach a minimum of 2 storage pools to the Delphix Engine; one for rpool and other for domain0 pool.

Requirements	Notes
Storage used for Delphix must be provisioned from storage that provides data protection.	<p>For example, using RAID levels with data protection features, or equivalent technology. The Delphix Engine does not protect against data loss originating at the hypervisor or SAN layers.</p>

4.3.7.6.1 Delphix storage operations

There are three types of data that Delphix stores on disk, which are:

⁹³ <https://docs.microsoft.com/en-us/windows-server/administration/performance-tuning/role/hyper-v-server/storage-io-performance>

1. Delphix VM Configuration Storage: stores data related to the configuration of the Delphix VM. VM Configuration Storage includes the Hyper-V configuration data as well as log files.
2. Delphix Engine System Disk Storage: stores data related to the Delphix Engine system data, such as the Delphix .ova settings.

4.3.7.6.1.1 Delphix VM configuration storage

The Delphix VM configuration must be stored on an NTFS volume(s).

Requirements	Notes
The volumes should have enough available space to hold all Hyper-V configuration and log files associated with the Delphix Engine.	If a memory reservation is not enabled for the Delphix Engine (in violation of memory requirements stated above), then space for a paging area equal to the Delphix Engine's VM memory must be added to the volumes containing the Delphix VM configuration data.

4.3.7.6.1.2 Delphix Engine system disk storage

Requirements	Notes
The Delphix Engine disks must be stored on NTFS volume(s).	The volume for the Delphix Engine System Disk Storage is often created on the same volume as the Delphix VM definition. In that case, the volume must have sufficient space to hold the Delphix VM Configuration, the virtual disk for the system disk, and a paging area if a memory reservation was not enabled for the Delphix Engine.
The Delphix .vhdx file is configured for a 128GB system drive.	The volume where the .vhdx is deployed should, therefore, have at least 128GB of free space prior to deploying the .vhdx.

4.3.7.6.1.3 Metadata disk(s)

In addition to making sure the latest Hyper-V patches have been applied, check with your hardware vendor for updates specific to your hardware configuration. VHDXs (virtual machine disks).

Requirements	Notes
A minimum of 4 VHDXs should be allocated for database storage.	Allocating a minimum of 4 VHDXs for database storage enables the Delphix File System (DxFS) to make sure that its file systems are always consistent on disk without additional serialization. This also enables the Delphix Engine to achieve higher I/O rates by queueing more I/O operations to its storage.
<p>If using VHDXs:</p> <ul style="list-style-type: none"> - Each VHDX should be the only VHDX on its NTFS volume - The VHDX volumes should be assigned to dedicated physical LUNs on redundant storage. - The VHDXs should be created as the Fixed Size type. 	<p>Provisioning VHDXs from isolated volumes on dedicated physical LUNs:</p> <ul style="list-style-type: none"> - Reduces contention for the underlying physical LUNs - Eliminates contention for locks on the volumes from other VMs and/or the Hyper-V Server Console
The quantity and size of VHDXs or RDMs assigned must be identical across all 4 controllers.	If the underlying storage array allocates physical LUNs by carving them from RAID groups, the LUNs should be allocated from different RAID groups. This eliminates contention for the underlying disks in the RAID groups as the Delphix engine distributes IO across its storage devices.
The physical LUNs used for NTFS volumes and RDMs should be of the same type in terms of performance characteristics such as latency, RPMs, and RAID level.	The total number of disk drives that comprise the set of physical LUNs should be capable of providing the desired aggregate I/O throughput (MB/sec) and IOPS (Input/Output Operations per Second) for all virtual databases that will be hosted by the Delphix Engine.
The physical LUNs used for NTFS volumes can be thin-provisioned in the storage array.	If the storage array allocates physical LUNs from storage pools comprising dozens of disk drives, the LUNs should be distributed evenly across the available pools.

4.3.7.7 Installing Hyper-V

1. Download the image from Delphix's Download site and copy it to your VM directory.
2. Start the Hyper-V Manager and specify **Name and Location** and then select **Next**.
3. Specify the **Generation**, configure memory, and then select **Next**. Memory: 64 GB (minimum), 128 GB (recommended)
4. Set up Networking by selecting **vNIC** then select **Next**.
5. Attach the downloaded image as a boot disk. Create a unique boot disk for each image.

Note:

Boot disks cannot be shared.

- Use an existing virtual hard disk.
- Browse to the location of VM.
- Select the Image.

6. Select **Finish**, the VM will appear in the inventory.

7. Customize the VM by selecting Settings:

- Delphix recommends having the IDE be the first device to boot from (under BIOS setting).
- Adjust the number of CPU (min 8)
- Add Hard Drive. Use VHDX formatted disks. Recommend Fixed Size.

Note:

Differencing Disk Types are not supported.

- 128 GB Disk Storage

8. Repeat step 7 as necessary.

9. Connect to the console and start the VM.

10. Once the installation is complete go to [Setting up the Delphix Engine \(see page 180\)](#) section to learn how to activate the masking service now that you have the software installed.

4.3.8 IBM Cloud Platform installation

This topic covers the virtual machine requirements, including memory and data storage, for the deployment of the Delphix Engine on IBM Cloud.

4.3.8.1 Supported profiles


The following is a list of profiles that are supported to deploy Delphix on IBM Cloud.

Requirements	Notes
mx2-8x64 mx2-16x128 mx2-32x256 mx2-48x384	- The Delphix Engine most closely resembles a storage appliance and performs best when provisioned using a storage-optimized profile - Larger profiles provide more CPU, which can prevent resource shortfalls under high I/O throughput conditions. - Larger profiles also provide more memory, which the Delphix Engine uses to cache database blocks. More memory will provide better read performance.

4.3.8.2 Network configuration

Requirements	Notes
Virtual Server Instances	<ul style="list-style-type: none"> - You must deploy the Delphix Engine and all of the source and target environments in the same VPC network. - When adding environments to the Delphix Engine, you must use the host's VPC IP addresses.
Security Configuration	<ul style="list-style-type: none"> - The default security group will only open port 22 for SSH access. You must modify the security group to allow access to all of the networking ports used by the Delphix Engine and the various source and target engines. - See Network Performance Configuration Options for information about network performance tuning. - See General Network and Connectivity Requirements for information about specific port configurations. - Reference: IBM Cloud Security and Compliance documentation⁹⁴

4.3.8.3 Storage configuration

 You must always attach a minimum of 2 storage pools to the Delphix Engine; one for rpool and other for domain0 pool.

Requirements	Notes
<ul style="list-style-type: none"> - Allocate initial storage equal to the size of the physical source database storage. - Add storage when storage capacity approaches 30% free. 	<ul style="list-style-type: none"> - For high redo rates and/or high DB change rates, allocate an additional 10-20 %. - Add new storage by provisioning new volumes of the same size. This enables the Delphix File System (DxFS) to make sure that its file systems are always consistent on disk without additional serialization. This also enables the Delphix Engine to achieve higher I/O rates by queueing more I/O operations to its storage. - A Delphix Engine requires a minimum of three (3) equally sized Block Volumes, in addition to the Boot volume which was automatically created while creating the virtual server instance. - IBM Block Storage Documentation⁹⁵

⁹⁴ <https://cloud.ibm.com/docs/security-compliance?topic=security-compliance-getting-started>

⁹⁵ <https://cloud.ibm.com/docs/vpc?topic=vpc-block-storage-profiles#tiers-beta/>

4.3.8.4 Additional IBM configuration notes

- Resize/expansion of a storage volume
- Expandable volume is a beta feature that is available for evaluation and testing purposes. This feature is available in the US South, US East, London, and France regions. Contact your IBM Sales representative if you are interested in getting access to [Expanding Block Storage](#)⁹⁶.
- After performing an “online” resize/expansion of a storage volume using IBM Cloud tools, then use the Delphix sysadmin interface to “Expand” the storage device; otherwise, the newly allocated storage space, from the resize/expansion, will not be used.
- Resize/expansion of a storage volume using IBM Cloud is not supported while the Delphix Engine is in a stopped state.
- Removing a storage volume
- It should be done while the machine is running.
- First, use the Delphix sysadmin CLI interface to “Unconfigure” the storage device, then remove it from IBM Cloud.

4.3.8.5 Procedure for deploying in the IBM Cloud

4.3.8.5.1 Prerequisites to Deploying in IBM Cloud

1. You require a license to use Delphix software. If you are a new customer, contact [Delphix](#)⁹⁷ to get started.
2. Review [IBM's cloud documentation](#)⁹⁸ for IBM Cloud-specific information.

4.3.8.5.2 Deploying Delphix in the IBM Cloud

There are two methods for deploying a Delphix Engine in the IBM Cloud using the Software Catalog or Manually Uploading the Delphix Image.

4.3.8.5.2.1 Deploying from the IBM Software Catalog

1. Navigate to the [IBM Software Catalog](#)⁹⁹ and search for Delphix.
2. Select the Delphix Data Masking Tile for the Masking product.
3. Scroll down to the Deployment Values section and input the specifics for your environment.

⁹⁶ <https://cloud.ibm.com/docs/vpc?topic=vpc-expanding-block-storage-volumes/>

⁹⁷ <https://www.delphix.com/company/contact>

⁹⁸ <https://cloud.ibm.com/docs/>

⁹⁹ <https://cloud.ibm.com/catalog#software/>

Required Parameters	Description
hostname	The name of the VSI you will use to deploy Delphix.
profile	Compute profile to be used for deploying Delphix (see recommended profiles).
ssh_key	Your public SSH key to be used when provisioning the VSI.
subnet_id	The id of the subnet where the VSI will be provisioned.
volumecount	Number of block storage volumes.
volumeprofile	Block storage profile to use (recommended is >= 10 IOPS/GB)
volumesize	Block storage volume size.
vpcname	The name of your VPC where the VSI is provisioned.
zone	VPC zone to provision your environment.

4.3.8.5.2.2 Manually downloading and deploying the Delphix Image

Downloading the Delphix Image



Contact your account manager to request access to the IBM variant of the Delphix product.

1. Follow the link given to you by your Delphix solutions architect. Download the Delphix_Verson...._Standard_IBM.qcow2 file and the SHA256SUMS file.
2. Once both files have finished downloading and assuming both files were downloaded to the same directory, you can run the following command to verify the download:

```
$ grep -i IBM.qcow2 ./SHA256SUMS | sed -E 's,Appliance_Images/(Controlled_Availability)?, ,g' | sha256sum --check
```

4.3.8.5.2.3 Uploading the Delphix Engine image as an object

1. Authenticate with the IBM Cloud and navigate to the <https://cloud.ibm.com/login/>.
2. Use the navigation menu to reach the **Resource List** page. The Resource List page can be navigated from the Dashboard by clicking on **Storage** within the **Resource summary** pane.
3. Expand Storage from the menu and select the appropriate resource group. You should have [created a resource group](#)¹⁰⁰ depending on your organization's strategy for managing IBM resources.
4. [Create a storage bucket](#)¹⁰¹ or select an existing bucket.
5. Click the blue **Upload** button and select **Files**.
6. A pop-up menu appears to select the transfer type. **Aspera High-Speed Transfer** is required for large files. For this, you will need to install the plugin. It will automatically navigate you through the steps to install the plugin.
7. In the **Upload Files (objects)** window, click on the **Select Files (objects)** button and choose the IBM specific **QCOW2** file that was previously downloaded.
8. Click the **Upload** button.

4.3.8.5.2.4 Creating a custom image

1. Authenticate with IBM Cloud and navigate to the [Dashboard](#)¹⁰².
2. Use the navigation menu to reach the **Custom images** page for VPC within the VPC infrastructure (IBM Cloud pull-down menu, upper left, VPC Infrastructure > Custom images).
3. Click the blue **Create** button.
4. In the **Import custom image** page, specify a unique name for the image.
5. From the **Resource Group** drop-down, select your organization's resource group.
6. Optional: In the **Tags** section, provide appropriate [tags](#)¹⁰³ to organize your resources.
7. Select the appropriate **Region**.
8. Select the **Cloud Object Storage** bucket containing the uploaded image by selecting the appropriate **Cloud Object Storage instances > Location > Bucket** from the drop-down menus. The downloaded QCOW2 image should appear in the pane below the three drop-down menus.
9. Within the **Operating System** section, click on the **Ubuntu Linux** tile and select **ubuntu-18-04-amd64** from the drop-down menu.
10. Once all the parameters are entered, in the right pane click on the blue button to **Import custom image**.

100 https://cloud.ibm.com/docs/account?topic=account-rgs#create_rgs/

101 <https://cloud.ibm.com/docs/cloud-object-storage/getting-started.html#gs-create-buckets/>

102 <https://cloud.ibm.com/login/>

103 <https://cloud.ibm.com/docs/account?topic=account-tag>

4.3.8.5.2.5 Launching the Delphix Engine

1. Authenticate with IBM Cloud and navigate to the [Dashboard](#)¹⁰⁴.
2. Use the navigation menu to reach the **Virtual Server Instances** page within the VPC Infrastructure (IBM Cloud pull-down menu, upper left, VPC Infrastructure > Virtual Server Instances). **Note:** To maximize performance, deploy the Delphix Engine instance in the same VPC/subnet in which you will create your virtual databases (VDBs).
3. Click the blue **Create** button.
4. In the **New Virtual Server for VPC** page, specify a unique name for the VM.
5. From the **Virtual Private Cloud** drop-down, select your organization's VPC.
6. From the **Resource Group** drop-down, select your organization's resource group.
7. Optional: In the **Tags** section, provide appropriate [tags](#)¹⁰⁵ to organize your resources.
8. Select the Location of your IBM Cloud resources.
9. In the **Operating System** section, click on the **Select Custom Image** link within the **Custom Image** block.
10. In the pop-menu, select the IBM-specific image you previously uploaded.
11. Within the **Profile** section, click on **View all profiles**. Select one of the supported instance types and click Save.
12. You can skip the **User Data** section.
13. You can also skip the **Boot Volume** section since it would already have the default values.
14. You can create block storage volumes later, so skip that for now. It will be discussed in the next section.
15. Continue on to the **Network Interfaces** section. If you already have a subnet configured in your zone and VPC, then this section will already have a default network interface. Otherwise, you need to create a subnet with the appropriate security groups. This part is critical, if the network isn't specified correctly, you are likely to run into firewall issues; please consult your IT or DevOps teams. Configure Network Security Groups (NSGs) for your subnet as required; again, please consult your IT or DevOps teams.
16. Click the **Create virtual server instance** button on the right panel. This will take a couple of minutes.

4.3.8.5.2.6 Creating block storage volumes

1. Authenticate with IBM Cloud and navigate to the [Dashboard](#)¹⁰⁶.

¹⁰⁴ <https://cloud.ibm.com/login/>

¹⁰⁵ <https://cloud.ibm.com/docs/account?topic=account-tag>

¹⁰⁶ <https://cloud.ibm.com/login/>

2. Use the navigation menu to reach the **Block Storage Volumes** within VPC Infrastructure (IBM Cloud pull-down menu > VPC Infrastructure > Block Storage Volumes).
3. Click the blue **Create** button.
4. In the **Block Storage Volume for VPC** modal window, specify a unique name for this Block Volume. It can be helpful if this name is descriptive or identifies the VM it is intended to be attached to and ends in a sequence number.
5. From the **Resource Group** drop-down, select your organization's resource group.
6. Optional: In the **Tags** section, provide appropriate **tags**¹⁰⁷ to organize your resources.
7. Select the Location of your IBM Cloud resources.
8. Enter the required IOPS. The recommended supported IOPS is 10/GB.
9. Enter the storage size in GB. Set the size of the volume to be sufficiently large, with room for growth, to support the databases that will be virtualized, or masked, by this Delphix Engine.
10. For **Encryption**, you can let it be the default, e.g. **Provider Managed**.
11. Click the blue **Create** Volume button. A Delphix Engine requires a minimum of three (3) equally sized Block Volumes, in addition to the Boot volume which was automatically created while creating the virtual server instance. Repeat Steps 3-11 as many times as necessary.

4.3.8.5.2.7 Attaching block storage volumes

1. Authenticate with IBM Cloud and navigate to the **Dashboard**¹⁰⁸.
2. Use the navigation menu to reach the **Block Storage Volumes** within VPC Infrastructure (IBM Cloud pull-down menu > VPC Infrastructure > Block Storage Volumes).
3. From the list of pre-existing Block Volumes, identify the volumes you wish to attach to a Delphix Engine and wait until the volume's state becomes Available.
4. Note that the volumes you wish to attach have **Attachment Type** set as a **hyphen**.
5. The right side of the volume row shows an Expandable menu. Click on it and select **Attach to Instance**.
6. In the **Attach Virtual Server Instance** modal window, select your virtual server instance (Delphix Engine) from the drop-down menu.
7. Click on the blue **Attach Volume** button.
8. Repeat Steps 3-7 until all associated Block Volume resources have been attached to the Delphix Engine instance.

¹⁰⁷ <https://cloud.ibm.com/docs/account?topic=account-tag>

¹⁰⁸ <https://cloud.ibm.com/login/>

4.3.8.5.2.8 Configuring the Delphix Engine

1. Connect to your running Delphix Engine instance with a web browser. Use the IP address or DNS name noted in the Instance Description. Upon successful connection, the browser will display a login prompt to enter the Delphix Setup Page.
2. Refer to the standard product deployment instructions to complete your Delphix deployment.

4.3.8.5.2.9 Next Steps

Congratulations! You have successfully deployed a Delphix Engine in IBM Cloud.

4.3.9 KVM installation

4.3.9.1 Overview

This page outlines the requirements for deploying the Continuous Compliance Engine via Linux KVM, as well as recommended configuration parameters.

Contact a Delphix representative to request this capability. Delphix will assist in assuring that all KVM requirements are met to successfully run a Continuous Compliance Engine with the most appropriate configurations for the use case.

If the Delphix Continuous Compliance Engine compete with other virtual machines on the same host for resources, it will result in increased latency for all operations.

With that, it is crucial that your KVM host is not over-subscribed, eliminating the possibility of insufficient resources for the Continuous Compliance Engine. This includes allowing a percentage of CPU resources for the hypervisor itself, as it can de-schedule an entire VM, in a case where the hypervisor is needed for managing I/O or computing resources.

The KVM ecosystem includes many versions/variations. Delphix is announcing general support for KVM, but will forego any formal certification for specific versions, focusing rather on declared support for specific Linux Kernels.



Delphix disk storage capabilities are based on the backend storage provided. Performance, redundancy, and stability characteristics are determined by the hypervisor or Cloud provisioned storage.

4.3.9.2 Pre-requisites

1. The KVM provider must explicitly state that the Ubuntu 20.04 kernel is supported by their variation of KVM.
 - a. Most, if not all, KVM hypervisor providers should have public facing documentation that includes support matrices for guest OS compatibility. If the KVM hypervisor provider does not explicitly state support for Ubuntu kernel version 20.04, Delphix cannot support the KVM provider.
 - i. As an example, RedHat Linux (a KVM hypervisor provider), provides a publicly accessible [guest operating system support matrix](#)¹⁰⁹. As of March 2024, this matrix does not include support for any Ubuntu guest OS, and therefore Delphix would also not support deployment on RedHat Linux KVM.
 - ii. Oracle Linux (a different KVM hypervisor provider) also provides a publicly accessible [guest operating system support matrix](#)¹¹⁰, which declares support for Ubuntu 20.04. In this case, since the hypervisor provider declares support for the Delphix Ubuntu kernel, Delphix will support deployment on Oracle Linux KVM.
2. A check of the Delphix appliance on the variation/version of KVM.
 - a. It is **required** that a Delphix representative works with the organization to check the Delphix appliance (and/or use case) on the specified KVM variation/version. This check will ensure that the software is compatible.

4.3.9.3 Virtual CPUs

Requirements	Notes
8 vCPUs	<ul style="list-style-type: none"> • CPU resource shortfalls can occur on an over-committed host or if facing competition for host resources during high I/O utilization. • CPU reservations are strongly recommended for the Delphix VM, to guarantee the full complement of CPUs, even when resources are overcommitted. • It is suggested to use a single core per socket, unless there are specific requirements for other VMs on the same KVM host.

¹⁰⁹ <https://access.redhat.com/articles/973163>

¹¹⁰ <https://docs.oracle.com/en/operating-systems/oracle-linux/kvm-user/kvm-AboutOracleLinuxKVM.html#kvm-linux-guest>

Requirements	Notes
Never allocate all available physical CPUs to virtual machines.	<ul style="list-style-type: none"> • A CPU for the KVM Server to perform hypervisor activities must be set aside before assigning vCPUs to Delphix and other VMs. • It is recommended that a minimum of 8-10% of the CPUs available are reserved for hypervisor operation. (e.g. 12 vCPUs on a 128 vCore system).

4.3.9.4 Memory

Requirements	Notes
128 GB vRAM (recommended) 64GB vRAM (minimum)	<ul style="list-style-type: none"> • More memory will provide better performance. • Memory reservations are required for the Delphix VM. The performance of the Continuous Compliance Engine will be significantly impacted by the over-commitment of memory resources in the KVM Server. • Reservations ensure that the Continuous Compliance Engine will not be forced to swap pages during times of memory pressure on the host. A swapped page will require more time from orders of magnitude to be brought back to physical memory, from the KVM swap device.
Memory for the KVM Server to perform hypervisor activities must be set aside before assigning memory to Delphix and other VMs.	<ul style="list-style-type: none"> • Failure to ensure sufficient memory for the host can result in a hard memory state for all VMs on the host, which will result in a block for memory allocations.

4.3.9.5 Network


Requirements	Notes
Virtual ethernet adapter requirements.	<ul style="list-style-type: none"> • Jumbo frames are highly recommended to reduce CPU utilization, decrease latency, and increase network throughput (typically 10-20% throughput improvement). • A 10GbE NIC/network is recommended for performance dSource and VDB operations.

Requirements	Notes
If the network load in the KVM Server hosting the Continuous Compliance Engine VM is high, dedicate one or more physical NICs to the Continuous Compliance Engine.	<ul style="list-style-type: none"> • Adding NICs only works if VMs are discovered using different interfaces. The NFS/iSCSI mounts will only use the network associated with the discovery. • See Network connectivity requirements¹¹¹ for information about specific port configurations.

4.3.9.6 SCSI controller

When adding virtual disks, make sure that they are evenly distributing the load across the maximum of four virtual SCSI controllers. Spreading the disks across available SCSI controllers evenly will ensure optimal I/O performance from the disks. For example, a VM with 4 SCSI controllers and 8 virtual disks should distribute the disks across the controllers as follows:

- disk0 = SCSI(0:0) - System Disk on Controller 0 Port 0
 - (ignore for purposes of load balancing)
- disk1 = SCSI(0:1) - Data Disk on Controller 0 Port 1
- disk2 = SCSI(1:1) - Data Disk on Controller 1 Port 1
- disk3 = SCSI(2:1) - Data Disk on Controller 2 Port 1
- disk4 = SCSI(3:1) - Data Disk on Controller 3 Port 1
- disk5 = SCSI(0:2) - Data Disk on Controller 0 Port 2
- disk6 = SCSI(1:2) - Data Disk on Controller 1 Port 2
- disk7 = SCSI(2:2) - Data Disk on Controller 2 Port 2
- disk8 = SCSI(3:2) - Data Disk on Controller 3 Port 2

 For load purposes, we generally focus on the Delphix storage (data disks) and ignore the controller placement of the system disk.

4.3.9.7 General storage

Using a minimum of four disks to run your Continuous Compliance Engine is recommended. One disk is used for the Delphix File System (DxFS), to ensure that its file systems are always consistent on disk without additional serialization. The other three, or more equally, sized disks will be used for metadata storage. This also enables the Continuous Compliance Engine to achieve higher I/O rates by queueing more I/O operations to its storage.

¹¹¹ <https://masking.delphix.com/docs/latest/network-connectivity-requirements>

Requirements	Notes
Storage used for Delphix must be provisioned from storage that provides data protection.	<ul style="list-style-type: none"> For example, using RAID levels with data protection features, or equivalent technology. The Continuous Compliance Engine does not protect against data loss originating at the hypervisor or SAN layers.

4.3.10 OCI installation

This topic covers the virtual machine requirements for deploying the Continuous Compliance Engine on Oracle Cloud Infrastructure (OCI).

4.3.10.1 Supported databases

Oracle databases up to version 19c are supported. Please reference the [Oracle Support Matrix \(see page 154\)](#) for the detailed list.

4.3.10.2 Compute image types

Delphix distributes product images, for OCI, using the QCOW2 image type. Compute Images must be imported into OCI using the Paravirtualized launch mode; currently, images using the Emulated launch mode are not supported.

4.3.10.3 Supported shapes

The following is a list of shapes that are supported to deploy Delphix on OCI.

Requirements	Notes
Large Memory Instances (perferred) VM.Standard2.8 VM_Standard2.16 VM_Standard2.24	The Delphix Engine most closely resembles a storage appliance and performs best when provisioned using a storage-optimized shape. Larger shapes provide more CPU, which can prevent resource shortfalls under high I/O throughput conditions. Larger shapes also provide more memory, which the Delphix Engine uses to cache database blocks. More memory will provide better read performance.

4.3.10.4 Network configuration

Requirements	Notes
Virtual Cloud Network (VCN)	<p>You must deploy the Delphix Engine and all of the source and target environments in a VCN to ensure that private IP addresses are static and do not change when you restart instances.</p> <p>By default, OCI subnets are considered public. When defining a subnet, we encourage configuring it as private. Unless required by your environment, your VCN should not include a Public Subnet.</p> <p>When adding environments to the Delphix Engine, you must use the host's VCN (static private) IP addresses.</p>
Static Private IP	<p>The Delphix instance should be launched with a static private IP address. For security reasons, it is encouraged to avoid configuring your engine with a Public IP address; but, in some cases, it may be ok to use a dynamic Public IP address in addition to a static Private IP address if your environment requires such access.</p>
Security Rules Configuration	<p>OCI allows two firewall features: Network Security Groups (NSGs) and Security Lists. Oracle recommends the use of NSGs over Security Lists because “NSGs let you separate the VCN's subnet architecture from your application security requirements”¹¹².</p> <p>However, a VCN will use a Security List to define default rules. By default, the security list will only open port 22 for SSH access. You must modify the security list, or create NSGs, to allow access to all of the networking ports used by the Delphix Engine and the various source and target engines.</p> <p>This dual implementation of firewall, or security, rules may be different from other clouds. Please see OCI documentation for best practices.</p> <p>See Network Connectivity Requirements (see page 177) for information about specific port configurations.</p>

4.3.10.5 Storage configuration



You must always attach a minimum of 2 storage pools to the Delphix Engine; one for rpool and other for domain0 pool.

¹¹² https://docs.cloud.oracle.com/en-us/iaas/Content/Network/Concepts/securityrules.htm#Security_Rules

Requirements	Notes
<p>Allocate initial storage equal to the size of the physical source database storage.</p> <p>Attach a minimum of four (4), equally sized, storage devices to the Delphix Engine.</p> <p>Add storage when storage capacity approaches 30% free.</p> <p>Must use Block Volume for data storage.</p> <p>Block Volumes must be attached using Paravirtualized mode.</p>	<p>Currently supported Instance Types, or Shapes, only support Block Volumes; File Storage is not supported.</p> <p>Paravirtualized block devices are required; currently, iSCSI devices are not supported.</p> <p>Elastic Performance Configuration Options (aka Volume Performance Policy): use Higher Performance.</p> <p>For high redo rates and/or high DB change rates, allocate an additional 10-20 %.</p> <p>Add new storage by provisioning new volumes of the same size. This enables the Delphix Engine to achieve higher I/O rates by distributing load among devices and queueing more I/O operations to its storage.</p>

4.3.10.6 Additional OCI configuration notes

- When running low on storage space, Delphix recommends adding additional equivalently sized block storage volumes, or devices, instead of resizing existing volumes.
- If you must expand existing storage volumes, then this must be done using the “online” resizing strategy specified in OCI documentation; “offline” storage resizing is not supported and may lead to unexpected downtime. If an existing storage volume is expanded, then use the Setup, or sysadmin, interface to expand each storage “device,” or volume. The additional storage, as a result of a resize, will not be available for use until the storage devices are explicitly instructed to make use of the additional space.
- If expanding storage volumes, it is recommended that all volumes are expanded to the same size. When storage volumes, or devices, are the same size the Delphix product is able to balance I/O distribution among the disks for optimal performance.
- Hot removal of storage volumes is not supported.

4.3.10.7 Installing OCI

4.3.10.7.1 Download and verify the Delphix Engine image

1. Contact your account manager to request access to the OCI variant of the Delphix product.
2. Follow the link given by your Delphix solutions architect. Download the `Delphix_6.x.x.x_...Standard_OCI.qcow2` file and the `SHA256SUMS` file.

3. Once both files have finished downloading and assuming both files were downloaded to the same directory, you can run the following command to verify the download: `$ grep -i OCI.qcow2 ./SHA256SUMS | sed -E 's,Appliance_Images/(Controlled_Availability)?, ,g' | sha256sum --check`

4.3.10.7.2 Upload the Delphix Engine image as an object

1. Authenticate with OCI and navigate to the [Infrastructure Console](#)¹¹³.
2. Use the navigation menu to reach the **Object Storage Buckets, Core Infrastructure**, page (Hamburger Menu > Object Storage > Object Storage).
3. Remember to set your **List Scope Compartment**. This will depend on your organization's strategy for managing OCI resources.
4. [Create a storage bucket](#)¹¹⁴ or select an existing bucket.
5. Click the blue **Upload** button.
6. In the **Upload Objects** modal window, specify an optional prefix and choose the OCI specific QCOW2 file that was previously downloaded.
7. Click the blue **Upload** button.

4.3.10.7.3 Creating a custom compute image from an object

1. Authenticate with OCI and navigate to the **Infrastructure Console**.
2. Use the navigation menu to reach the **Compute Custom Images, Core Infrastructure**, page (Hamburger Menu > Compute > Custom Images).
3. Remember to set your **List Scope Compartment**. This will depend on your organization's strategy for managing OCI resources.
4. Click the blue **Image Import** button.
5. In the **Import Image** modal window, select a suitable compartment in the **Create In Compartment** field that conforms to your organization's strategy on managing OCI resources.
6. In the **Name** field enter a unique name to identify the Custom Compute Image. You may want to use the same, resulting name of the image object from the previous step, Upload the Delphix Engine Image as an Object.
7. For **Operating System** select **Linux**.
8. Next, identify an object by specifying its Compartment, Bucket, and Object Name. Or, specify an Object Storage URL. **Note:** The Object Details will identify this value as **URL Path (URI)**.

¹¹³ <https://console.us-phoenix-1.oraclecloud.com/>

¹¹⁴ https://docs.cloud.oracle.com/en-us/iaas/Content/GSG/Tasks/addingbuckets.htm#Putting_Data_into_Object_Storage

9. For **Image Type** select **QCOW2**.
10. For **Launch Mode** select **Paravirtualized Mode**.
11. For organizations that have a tagging policy for cloud-based resources, expand the **Tagging Options** section, and define tags.
12. Click the blue **Import Image** button.

4.3.10.7.4 Launching the Delphix Engine

1. Authenticate with OCI and navigate to the **Infrastructure Console**.
2. Use the navigation menu to reach the **Compute Instances, Core Infrastructure**, page (Hamburger Menu > Compute > Instances).
3. Remember to set your **List Scope Compartment**. This will depend on your organization's strategy for managing OCI resources.
4. Click the blue **Create Instance** button.
5. In the **Create Compute Instance** window pane, specify a unique name for the VM.
6. For the **Create In Compartment** field, select a suitable compartment that conforms to your organization's strategy on managing OCI resources.
7. In the **Image or operating system** section, click the Change Image button. Switch to the Custom Images tab. Find the Delphix image that corresponds to the instance you wish to deploy. Click the blue **Select Image** button. **Note:** If the Delphix Custom Image is not visible, look for the **Change Compartment** option near the top of the current window pane.
8. Each Availability Domain has its own quota, it is ok to use AD-1, AD-2, or AD-3 - but, be sure to make note of which Availability Domain you are using. **Note:** Compute Instances and attached Storage will need to be in the same Availability Domain.
9. In the **Shape** section click the **Change Shape** button. For **Instance type** specify **Virtual Machine** and for **Shape series** use **Intel Skylake**. Then select an OCI Shape that is supported by Delphix.
10. Continue on to the **Configure networking** section. This part is critical, if the network isn't specified correctly, you are likely to run into firewall issues; please consult your IT or DevOps teams. If your organization is using Network Security Groups (NSGs), mark the **Use Network Security Groups to Control Traffic** checkbox; again, please consult your IT or DevOps teams. Lastly, select the Do Not Assign a Public IP Address radio button; if you must deviate from this guidance then you are highly encouraged to engage your IT or DevOps teams.
11. You may skip the **Boot Volume** section.
12. In the **Add SSH Keys** select the **No SSH Keys** radio option. The Delphix product is a closed appliance and manages users independently.
13. In general, you can skip all of the Advanced Options. For organizations that have a tagging policy for cloud-based resources, expand into the Advanced Management section, and look for the Tagging sub-section to define tags.
14. Click the blue **Create button** - wait about 2-5 minutes for the Delphix Engine instance to boot.

4.3.10.7.5 Create block storage volumes

1. Authenticate with OCI and navigate to the **Infrastructure Console**.
2. Use the navigation menu to reach the **Block Volumes, Core Infrastructure**, page (Hamburger Menu > Block Storage > Block Volumes).
3. Remember to set your **List Scope Compartment**. This will depend on your organization's strategy for managing OCI resources.
4. Click the blue **Create Block Volume** button.
5. In the **Create Block Volume** modal window, specify a unique name for this Block Volume. It can be helpful if this name is descriptive or identifies the VM it is intended to be attached to and ends in a sequence number.
6. For the **Availability Domain**, this value **MUST** be the same Availability Domain used for the Delphix Engine instance, otherwise, this volume will not be available for use.
7. In the **Volume Size and Performance** section, select the **Custom** option. Set the size of the volume to be sufficiently large, with room for growth, to support the databases that will be virtualized, or masked, by this Delphix Engine. And, set the **Default Volume Performance** to the **Higher Performance** setting.
8. A **Backup Policy** is not required and can be left blank or **No Backup Policy Selected**. However, depending on your organization's needs, you may consider selecting a Backup Policy.
9. For **Encryption**, it is ok to use the default option, **Encrypt Using Oracle-Managed Keys**. Optionally, if you want, or need, to manage encryption keys independently then use the **Encrypt Using Customer-Managed Keys** option.
10. For organizations that have a tagging policy for cloud-based resources, expand the **Tagging Options** section, and define tags.
11. Uncheck the checkbox that says **View Detail Page After This Block Volume is Created**. This will prevent you from navigating away from the Block Volumes page, because, more often than not, you will need to create multiple Block Volumes at the same time.
12. Click the blue **Create Block Volume** button.
13. A Delphix Engine requires a minimum of four (4) equally sized Block Volumes. Repeat Steps 4-12 as many times as necessary.

4.3.10.7.6 Attach block storage volumes

1. Authenticate with OCI and navigate to the **Infrastructure Console**.
2. Use the navigation menu to reach the **Block Volumes, Core Infrastructure**, page (Hamburger Menu > Block Storage > Block Volumes).
3. Remember to set your **List Scope Compartment**. This will depend on your organization's strategy for managing OCI resources.

4. From the list of pre-existing Block Volumes, identify the resources you wish to attach to a Delphix Engine and wait until the volume's state becomes Available.
5. Select one of the **Block Volumes** to enter the **Block Volume Details** page.
6. On the left-hand side, locate the **Resources** menu and select **Attached Instances**.
7. If the Block Volume has not been previously attached to another VM, then you will be able to click the blue **Attach to Instance** button.
8. In the Attach to Instance modal window, specify the **Attachment Type** as **Paravirtualized**. Currently, iSCSI is not supported.
9. For **Access Type** use the **READ/WRITE** option.
10. Next, identify a Delphix Engine by selecting an instance, or by specifying an instance OCID. If you don't see the Delphix Engine instance in the Select an Instance drop-down menu, you may need to use the Change Compartment option. Block Volumes can only be attached to VM instances that were created in the same Availability Domain - if these values do not match, you will need to either re-provision Block Volumes or the Delphix Engine, in the correct Availability Domain.
11. Click the blue Attach button.
12. Repeat Steps 4-11 until all associated Block Volume resources have been attached to the Delphix Engine instance.

4.3.10.7.7 Configuring masking

Once deployed, go to [First Time Setup](#) (see page 180) section to learn how to activate the masking service now that you have the software installed.


4.3.11 VMware installation

The Delphix Engine is a virtual appliance that runs on a hypervisor. In this section, you'll find requirements to run Delphix on VMware including supported versions and instance configurations as well as recommended configuration parameters for optimal performance.

The Delphix Engine is intensive both from a network and a storage perspective. If the Delphix Engine competes with other virtual machines on the same host for resources it will result in increased latency for all operations. As such, it is crucial that your ESXi host is not over-subscribed, as this eliminates the possibility of a lack of resources for the Delphix Engine. This includes allowing a percentage of CPU resources for the hypervisor itself as it can de-schedule an entire VM if the hypervisor is needed for managing IO or compute resources.

4.3.11.1 Supported ESX versions

Requirements	Notes
VMware Cloud VMware ESX/ESXi 8.0 VMware ESX/ESXi 7.0, 7.0u1, 7.0u2, 7.0 U3c ESX/ESXi 6.7 U3 VMware ESX/ESXi 6.5 U1, 6.5 U3 VMware ESX/ESXi 6.0	More recent versions of VMware are preferred, such as ESX/ESXi 6.0 - 7.0 U3c


 If a minor release version is listed as supported, then all patch releases applicable to that minor release are certified.

4.3.11.2 Virtual CPUs

Requirements	Notes
8 vCPUs	<ul style="list-style-type: none"> - CPU resource shortfalls can occur both on an over-committed host as well as competition for host resources during high IO utilization. - CPU reservations are strongly recommended for the Delphix VM so that Delphix is guaranteed the full complement of vCPUs even when resources are overcommitted. - It is suggested to use a single core per socket unless there are specific requirements for other VMs on the same ESXi host.
Never allocate all available physical CPUs to virtual machines	<ul style="list-style-type: none"> - CPU for the ESXi Server to perform hypervisor activities must be set aside before assigning vCPUs to Delphix and other VMs. - We recommend that a minimum of 8-10% of the CPUs available are reserved for hypervisor operation. (e.g. 12 vCPUs on a 128 vCore system).

4.3.11.3 Memory

Requirements	Notes
128 or higher GB vRAM (recommended) 64GB vRAM (minimum)	<ul style="list-style-type: none"> - The masking service on the Delphix Engine uses its memory to process database and file blocks. - More memory can sometimes improve performance. Memory reservation is a requirement for the Delphix VM. - Overcommitting memory resources in the ESX server will significantly impact the performance of the Delphix Engine. - Reservation ensures that the Delphix Engine will not stall while waiting for the ESX server to page in the engine's memory.

 Do not allocate all memory to the Delphix VM.

Never allocate all available physical memory to the Delphix VM. You must set aside memory for the ESX Server to perform hypervisor activities before you assign memory to Delphix and other VMs. The default ESX minimum free memory requirement is 6% of the total RAM. When free memory falls below 6%, ESX starts swapping out the Delphix guest OS. We recommend leaving about 8-10% free to avoid swapping


For example, when running on an ESX Host with 512GB of physical memory, allocate no more than 470GB (92%) to the Delphix VM (and all other VMs on that host).

4.3.11.4 Network

Requirements	Notes
The ova is pre-configured to use one virtual ethernet adapter of type VMXNET 3.	<ul style="list-style-type: none"> - Jumbo frames are highly recommended to reduce CPU utilization, decrease latency, and increase network throughput. (typically 10-20% throughput improvement) - If additional virtual network adapters are desired, they should also be of type VMXNET 3.
A 10GbE NIC in the ESX Server is recommended.	For VMs having only gigabit networks, it is possible to aggregate several physical 1GbE NICs together to increase network bandwidth (but not necessarily to reduce latency). Refer to the VMware Knowledge Base article NIC Teaming in ESXi and ESX ¹¹⁵ . However, it is not recommended to aggregate NICs in the Delphix Engine VM.

¹¹⁵ <https://kb.vmware.com/s/article/1004088>

4.3.11.5 Storage

 Always attach a minimum of 2 storage pools to the Delphix Engine; one for rpool and the other for domain0 pool.

There are three types of data that Delphix stores on disk, which are:

1. **Delphix VM configuration storage:** stores data related to the configuration of the Delphix VM. VM Configuration Storage includes the VMware ESX configuration data as well as log files.
2. **Delphix Engine system disk storage:** stores data related to the Delphix Engine system data, such as the Delphix .ova settings.
3. **Metadata storage:** stores metadata used by the Masking service.

4.3.11.5.1 General requirements

Requirements	Notes
Storage used for Delphix must be provisioned from storage that provides data protection.	For example, using RAID levels with data protection features, or equivalent technology. The Delphix engine product does not protect against data loss originating at the hypervisor or SAN layers.

4.3.11.5.2 Delphix VM configuration storage

The Delphix VM configuration should be stored in a VMFS volume (often called a "datastore").

Requirements	Notes
The VMFS volume should have enough available space to hold all ESX configuration and log files associated with the Delphix Engine.	If a memory reservation is not enabled for the Delphix Engine (in violation of memory requirements stated above), then space for a paging area equal to the Delphix Engine's VM memory must be added to the VMFS volume containing the Delphix VM configuration data.

4.3.11.5.3 Delphix Engine system disk storage

The VMFS volume must be located on shared storage in order to use vMotion and HA features.

Requirements	Notes
The Delphix Engine system disk should be stored in a VMDK.	The VMDK for the Delphix Engine System Disk Storage is often created in the same VMFS volume as the Delphix VM definition. In that case, the datastore must have sufficient space to hold the Delphix VM Configuration, the VMDK for the system disk, and a paging area if a memory reservation was not enabled for the Delphix Engine.
The Delphix .ova file is configured for a 127GB system drive.	The VMFS volume where the .ova is deployed should, therefore, have at least 127GB of free space prior to deploying the .ova.

4.3.11.5.4 Delphix Engine metadata storage

Shared storage is required in order to use vMotion and HA features. In addition to making sure the latest VMware patches have been applied, check with your hardware vendor for updates specific to your hardware configuration. VMDKs (Virtual Machine Disks) or RDMs (Raw Device Mappings) operating in virtual compatibility mode can be used for data storage.

Requirements	Notes
The minimum recommended storage size is 50 GB.	

In addition to making sure the latest VMware patches have been applied, check with your hardware vendor for updates specific to your hardware configuration.

4.3.11.6 Additional VMware configuration notes

- Running Delphix inside of vSphere is supported.
- Using vMotion on a Delphix VM is supported.
- Device passthrough is not supported.

4.3.11.7 Installing OVA on VMware

1. Download the OVA file from Delphix's Download site. Note, you will need a support login from your sales team or a welcome letter. Navigate to "Virtual Appliance" and download the appropriate OVA. If unsure, use the HWv11 OVA type.
2. Login using the vSphere client to the vSphere server (or vCenter Server) where you want to install the Delphix Engine.
3. In the vSphere Client, click **File**.

4. Select **Deploy OVA Template** and then browse to the OVA file. Click **Next**.
5. Select a hostname for the Delphix Engine. This hostname will be used in configuring the Delphix Engine network.
6. Select the data center where the Delphix Engine will be located.
7. Select the cluster and the ESX host.
8. Select one (1) data store for the Delphix OS. This datastore can be thin-provisioned and must have 127GB of free space to accommodate the Delphix operating system.
9. The Delphix VM Configuration Storage requires a minimum of 50GB. The VMFS volume should have enough available space to hold all ESX configuration and log files associated with the Delphix Engine. The Delphix Engine system disk should be stored in a VMDK system drive. The VMFS volume must be located on shared storage in order to use vMotion and HA features.
10. Select the virtual network you want to use. If using multiple physical NICs for link aggregation, you must use vSphere NIC teaming. Do not add multiple virtual NICs to the Delphix Engine itself. The Delphix Engine should use a single virtual network.
11. Click **Finish**. The installation will begin and the Delphix Engine will be created in the location you specified.
12. Once the Delphix Engine has been created proceed to [Setting up the Delphix Engine \(see page 180\)](#) to configure the system.

4.3.12 Containerized masking installation

This page describes how to utilize Kubernetes images to deploy a containerized version of the Continuous Compliance Engine. Since Continuous Compliance is a tool used for data masking, the term “masking” should be taken as a reference to the primary function of a Continuous Compliance Engine.

With a few small exceptions, containerized masking provides the same functionality and user experience as one would expect with a Continuous Compliance Engine deployed in the typical fashion, on a VM.

Containerized masking is designed to run on any Certified Kubernetes platform, the list of supported platforms can be found at <https://www.cncf.io/certification/software-conformance>. Containerized masking is also OCI-compliant and may use any container runtime within a Certified Kubernetes platform that implements the OCI Runtime Specification, including CRI-O, Docker, and Podman.



Delphix regularly tests against a range of popular Kubernetes platforms with the goal of covering a representative sample of implementations. The following Kubernetes platforms have been explicitly tested by Delphix and are recommended for use:

- Microk8s
- AWS EKS

4.3.12.1 Obtaining the images

Containerized masking utilizes three integrated containers to deliver, in essence, the same masking experience as provided on a Continuous Compliance Engine deployed on a VM. The containerized form allows for rapid spin up/tear down of ephemeral engines to handle automated workflow deployments. These three containers are delivered in a compressed archive (`.tar.gz`) for convenience.

Licensed versions of these bundles are available for download from the download.delphix.com¹¹⁶ site. In the folder for each version are two files. One file is HTML instructions similar to this page that can be downloaded for an offline copy of the installation instructions. The second file is the `masking_docker_images.tar.gz` bundle with the container images.

Docker is employed to build the container images, which produces a set of [Open Source \(OCI\) images](#)¹¹⁷ for each container. The intention is to make the containers as vendor independent as possible.

4.3.12.2 Setup

Containerized masking is intended to run as a pod on Kubernetes with three containers:

1. `delphix-masking-app` – Serves the application UI and API, and executes masking jobs.
2. `delphix-masking-database` – Stores various application configurations.
3. `delphix-masking-proxy` – Serves as a reverse proxy, handling HTTP and HTTPS traffic for the UI and API.

The UI and API are served from internal ports 8080 and 8443. When deploying the application, the Kubernetes config must provide a Service that directs external HTTP traffic to port 8080 and HTTPS traffic to port 8443, as shown in the example `kubernetes-config.yaml` file.

The pod also requires a single volume per instance. This storage should be attached to both the app container and the database container.

- This volume should be attached to the `delphix-masking-database` container at location `/var/delphix/postgresql` with a subpath of `postgresql`.
- This volume should be attached to the `delphix-masking-app` container twice. Once at location `/var/delphix/masking/` with a subpath of `masking` and once at location `/var/delphix/postgresql` with a subpath of `postgresql`.

This volume should have at least 2GB of space for each container, though certain configurations may require significantly more space.

This storage volume should be created as a persistent volume. If it is not, masking job configurations will have to be recreated each time the pod is restarted. Also, certain diagnostic information captured in the logs will be lost when the pod is restarted unless the volume is persistent.

Because this volume is persistent, the pod should be deployed as a StatefulSet.

¹¹⁶ <https://download.delphix.com/folder/1926/Delphix%20Product%20Releases/Containerized%20Masking>

¹¹⁷ <https://opencontainers.org/about/overview/>

4.3.12.2.1 Network management

The proxy container has built-in configurations to act as a reverse proxy. It is recommended that the main `nginx.conf` file remains unmodified; instead, modify the individual component configuration files that get incorporated into the main `nginx.conf` file through include statements (such as `proxy.conf` for the reverse proxy-related configs and `ssl.conf` for HTTPS related configs).

To modify any nginx related files, such as config files or certificates and keys, an external volume should be bind mounted to the proxy container at `/etc/config`. During container startup, if the proxy container detects bind mounted files at the locations listed below, it will ignore the config files that are built into the proxy container's image and will instead use the mounted files.

4.3.12.2.1.1 HTTPS certificates

If the proxy container does not detect an external certificate in the expected location, it will generate and use a self-signed certificate.

The expected locations of each file are shown below:

File	Description
<code>/etc/config/nginx/nginx.conf</code>	main configs file
<code>/etc/config/nginx/proxy.conf</code>	reverse proxy configs
<code>/etc/config/nginx/ssl/ssl.conf</code>	ssl configs
<code>/etc/config/nginx/ssl/nginx.crt</code>	ssl certificate
<code>/etc/config/nginx/ssl/nginx.key</code>	ssl private key
<code>/etc/config/nginx/ssl/dhparam.pem</code>	DH parameters file

4.3.12.2.2 OWASP CSRFGuard

The [OWASP CSRFGuard](https://owasp.org/www-project-csrfguard/)¹¹⁸ product has been employed as part of the protections that are built-in to the masking product. The supplied NginX proxy container rewrites a packet's Host header with the contents of the X-Forwarded-Host header if it exists so that CSRFGuard will accept proxied packets.

¹¹⁸ <https://owasp.org/www-project-csrfguard/>

This results in a requirement. If the Pod is placed behind a proxy device that re-writes the Host header, that proxy must add an X-Forwarded-Host header containing the original host value.

4.3.12.2.3 Sample configuration

The following configuration file shows an example of how Containerized masking might be deployed. Details will vary based on the use case, environment, and product version.

```
apiVersion: v1
kind: Service
metadata:
  name: delphix-masking
spec:
  type: NodePort
  selector:
    app: masking
  ports:
    - name: http
      port: 8080
      nodePort: 30080
    - name: https
      port: 8443
      nodePort: 30443
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: delphix-masking
spec:
  selector:
    matchLabels:
      app: masking
  serviceName: delphix-masking
  template:
    metadata:
      labels:
        app: masking
    spec:
      securityContext:
        runAsUser: 65436 # masking user
        runAsGroup: 50 # staff group
        fsGroup: 50
        #
        # Some volume providers, such as hostProvider, do not support fsGroup.
        # If you are using such a volume provider, use an init container to
        # change the ownership of each volume to 65436:50 and the permissions
        # to 775.
        #
        runAsNonRoot: true
      containers:
```

```
- image: delphix-masking-database:6.0.16.0-c1
  name: mds
  ports:
    - containerPort: 5432
      name: mds
  volumeMounts:
    - name: masking-persistent-storage
      mountPath: /var/delphix/postgresql
      subPath: postgresql
- image: delphix-masking-app:6.0.16.0-c1
  name: app
  ports:
    - containerPort: 8284
      name: http
  volumeMounts:
    - name: masking-persistent-storage
      mountPath: /var/delphix/masking
      subPath: masking
    - name: masking-persistent-storage
      mountPath: /var/delphix/postgresql
      subPath: postgresql
  startupProbe:
    httpGet:
      scheme: HTTPS
      path: /masking/api/system-information
      port: 8443
    failureThreshold: 30
    periodSeconds: 10
    timeoutSeconds: 10
  livenessProbe:
    httpGet:
      scheme: HTTPS
      path: /masking/api/system-information
      port: 8443
    initialDelaySeconds: 300
    failureThreshold: 1
    periodSeconds: 10
    timeoutSeconds: 10
  readinessProbe:
    httpGet:
      scheme: HTTPS
      path: /masking/api/system-information
      port: 8443
    initialDelaySeconds: 30
    periodSeconds: 60
    timeoutSeconds: 10
- image: delphix-masking-proxy:6.0.16.0-c1
  name: proxy
  ports:
    - containerPort: 8080
      name: http
    - containerPort: 8443
```

```

        name: https
    volumeClaimTemplates:
    - metadata:
        name: masking-persistent-storage
      spec:
        accessModes:
        - ReadWriteOnce
        resources:
        requests:
        storage: 4Gi

```

4.3.12.2.4 Deployment

Load the container images obtained from the download site into some Kubernetes container registry, then deploy the masking Pod using a config file similar to the example provided above.

```
kubectl apply -f <path-to-config-file>
```

4.3.12.3 Debugging

In a support case, a Delphix Support engineer may ask for a support bundle containing diagnostic information. The preferred method of generating a support bundle is to use the API endpoints as shown in the [API Call for Generating a Support Bundle](#) (see page 974) document. The [API Client Documentation](#) (see page 878) may assist with more information on various uses of the API Client.

4.3.12.3.1 Generating and retrieving a support bundle from CLI

If the API endpoints are not functioning properly or there are difficulties accessing them, a support bundle can be gathered by running the following command-line commands from the Kubernetes layer of the node hosting the pod – Kubernetes admin permissions are required to perform these actions.

The exact name of the tarball file created by this command can then be found using `kubectl exec`. For example:

```
$ kubectl exec -it <pod name> -c app -- /bin/bash /opt/delphix/masking/bin/
generate_container_support_bundle.sh
```

```
$ kubectl exec delphix-masking-0 -c app -- find /var/delphix/masking/ -name 'dlpx-
support-*'
/var/delphix/masking/dlpx-support-4b3e2af2-1d00-43f5-b45b-c84dba62648a-20211201-18-21-5
3.tar.gz
```

The tarball can then be copied out of the pod using `kubectl cp`. For example:

```
$ kubectl cp delphix-masking-0:/var/delphix/masking/dlpx-support-4b3e2af2-1d00-43f5-
b45b-c84dba62648a-20211201-18-21-53.tar.gz -c app dlpx-support-4b3e2af2-1d00-43f5-
b45b-c84dba62648a-20211201-18-21-53.tar.gz
```

The tarball can then be provided to the Delphix Support engineer by uploading it to `upload.delphix.com` and adding the associated case number in the matching field.

4.3.12.4 Managing LDAP over TLS/SSL for containerized masking

4.3.12.4.1 Prerequisites

The below libraries should be installed on the containerization masking engine:

- `openssl`
- `keytool`

4.3.12.4.2 Get the LDAP certificate

Get the LDAP certificate from the LDAP server: for example, if the LDAP server is: `qa-ad.delphix.com`

```
openssl s_client -showcerts -connect qa-ad.delphix.com:636 -servername qa-
ad.delphix.com
```

4.3.12.4.2.1 Create a `user-ldap.cer` file from the above LDAP certificate:

Create a file with the name `user-ldap.cer` and copy the code from the above output file to this file (only copy from `-----BEGIN CERTIFICATE-----` till `-----END CERTIFICATE-----`).

Below is the sample output needs to be copied to the file: `user-ldap.cer`

```
-----BEGIN CERTIFICATE-----
MIIGfDCCBGSGAwIBAgIIEqvHrbNVb88wDQYJKoZIhvcNAQELBQAwcjELMAkGA1UE
BhMCVVMxCzAJBgNVBAGTAkNBMRUwEwYDVoQHEwxSZWR3b29kIENpdHkxCzAJBgNV
BAoTALFBMQwwCgYDVoQLDANSJkQxJDAiBgNVBAMTG3FhLWFkZGMwMS5xYS1hZC5k
ZWxwaGl4LmNvbTAeFw0yMDEwMTIyMjUwMDBaFw0zMDUwMTIyMjUwMDBaMHIxCzA
-----END CERTIFICATE-----
```

4.3.12.4.3 Import certificate to Keystore

Import certificate to keystore using `keytool` utility like below, assuming running the below command from the same location where we created `user-ldap.cer` file.

```
keytool -import -trustcacerts -alias .masking_certs -file user-ldap.cer
-keystore .masking_certs -storepass changeit -noprompt
```

You can verify the imported certificate as below:

```
keytool -list -keystore .masking_certs -v
```

4.3.12.4.4 Create configmap entry based on LDAP certificate

1. use the Kubernetes command to create a configmap, for example:

```
kubectl create configmap ldap-ssl-config --from-file=.masking_certs
```

Here `ldap-ssl-config` is the name of the created configmap entry, `.masking_certs` file contains the LDAP certificate. To verify that configmap entry is added to the pod instance run the following command:

```
kubectl get configmap
```

4.3.12.4.5 Mount the configured configmap as volume

Add configmap entry as a volume to the pod instance in its config `.yaml` file. If you already have other volumes defined that new entry can go under the existing volumes section. If not create a **volumes:** section as shown below:

```
volumes:
  - name: ldap-ssl-cert-volume
    configMap:
      name: ldap-ssl-config
```

Here `ldap-ssl-cert-volume` is a name for the provided volume, `ldap-ssl-config` is the name of the previously created configmap entry.

Now we are ready to mount that volume to **app** container. Under the **containers:** section of the pod's config `.yaml` file, find the **app** container and add another entry to its **volumeMounts:** as shown below:

```
- name: ldap-ssl-cert-volume
  mountPath: /var/delphix/ssl/.masking_certs
  subPath: .masking_certs
```

Here `ldap-ssl-cert-volume` is a pod level provided volume, `.masking_certs` is a name of the certificate file (originally provided by the configured configmap).

For reference, see the below-attached sample screenshot from `kubernetes-config.yaml`

```

volumes:
  - name: ldap-ssl-cert-volume
    configMap:
      name: ldap-ssl-config
  - name: nfs-pv-storage2
    persistentVolumeClaim:
      claimName: nfs-pvc2
containers:
  - image: delphix-masking-database:25.0.0.0
    imagePullPolicy: Never # This image has to be built locally
    name: mds
    ports:
      - containerPort: 5432
        name: mds
    volumeMounts:
      - name: masking-persistent-storage
        mountPath: /var/delphix/postgresql
        subPath: postgresql
  - image: delphix-masking-app:25.0.0.0
    imagePullPolicy: Never # This image has to be built locally
    name: app
    lifecycle:
      postStart:
        exec:
          command: ["/bin/bash", "-c", "ls -ltra /var/delphix/ssl/.masking_certs;"]
    ports:
      - containerPort: 8284
        name: http
    volumeMounts:
      - name: ldap-ssl-cert-volume
        mountPath: /var/delphix/ssl/.masking_certs
        subPath: .masking_certs
      - name: masking-persistent-storage
        mountPath: /var/delphix/masking
        subPath: masking
      - name: masking-persistent-storage
        mountPath: /var/delphix/postgresql
        subPath: postgresql
      - name: nfs-pv-storage2
        mountPath: /var/delphix/masking/remote-mounts/nfs_2
    env:
      - name: MASK_DEBUG
        value: "true"

```

Now deploy the pods as usual using the config file:

```

kubectl create -f kubernetes-config.yaml
kubectl get pods

```

4.3.12.4.6 Enable ApplicationSettings for LDAP over TLS/SSL

Once the Containerization Masking engine is deployed successfully, setup and enable the LDAP over TLS/SSL accordingly.

4.4 Naming requirements

This section describes the naming requirements for Masking Engine objects which are allowed to be created/renamed manually.

4.4.1 Affected configurable objects

configurable objects
algorithm
application
connector
domain
environment
file format
job
profiling group
record type
role
rule set
search expression

For all of the above:

- Leading/trailing white space is not allowed
- The following special characters are not allowed:

Symbol	Name
[open bracket
]	close bracket
(open parenthesis
)	close parenthesis
{	open brace
}	close brace
~	tilde
!	exclamation mark
@	at
#	pound
\$	dollar
%	percent
^	carat
*	asterisk
"	quote
?	question mark

Symbol	Name
:	colon
;	semi-colon
,	comma
/	forward slash
\	back slash
\\	double back slash
`	back quote
+	plus
=	equal
<	less than
>	greater than
'	single quote
	pipe

4.4.2 Upgrade

During an upgrade of a Masking Engine to a 6.0 or later release, a name with leading or trailing white space will be automatically trimmed, and a counter value might be appended to the end of the name to prevent a naming conflict. For example:

pre-upgrade name	post-upgrade name	upgrade change
"alg_SecureLookup"	"alg_SecureLookup"	no change

pre-upgrade name	post-upgrade name	upgrade change
"alg_SecureLookup"	"alg_SecureLookup1"	leading white space trimmed and counter value appended
"alg_SecureLookup"	"alg_SecureLookup2"	trailing white space trimmed and counter value appended

If any name from the above-mentioned "configurable entities" table has a restricted special character - an upgrade will fail with the corresponding error message.

4.4.3 Maximum name lengths

Access the [API client](#)¹¹⁹ to find out the maximum name length for a particular object by navigating to `http://YourMaskingEngine.YourDomain.com/masking/api-client` to access the API client for the engine.

201
Success No links

Media type

Controls Accept header.

Example Value | Schema

```

MaskingJob {
  jobId integer($int32)
  readOnly: true
  # The ID number of the masking job. This field is auto-generated by the Masking Engine.
  jobName* string
  # The name of the masking job. Once the masking job is created, this field cannot be changed.
  rulesetId* > [...]
  rulesetType > [...]
  createdBy > [...]
  createdTime > [...]
  email > [...]
  feedbackSize > [...]
  jobDescription > [...]
  maxMemory > [...]
  minMemory > [...]
  multiTenant > [...]
  numInputStreams > [...]
  onTheFlyMasking > [...]
  databaseMaskingOptions DatabaseMaskingOptions > {...}
  onTheFlyMaskingSource OnTheFlyMaskingSource > {...}
  failImmediately > [...]
  enabledTasks > [...]
  streamRowLimit > [...]
}
example: OrderedMap { "jobName": "some_masking_job", "rulesetId": 7, "jobDescription": "This example illustrates a MaskingJob with just a handful of the possible fields set. It is meant to exemplify a simple JSON body that can be passed to the endpoint to create a MaskingJob.", "feedbackSize": 100000, "onTheFlyMasking": false, "databaseMaskingOptions": OrderedMap { "batchUpdate": true, "commitSize": 20000, "dropConstraints": true, "prescript": OrderedMap { "name": "my_prescript.sql", "contents": "ALTER TABLE table_name DROP COLUMN column_name;" }, "postscript": OrderedMap { "name": "my_postscript.sql", "contents": "ALTER TABLE table_name ADD column_name VARCHAR(255);" } }
                    
```

400
Bad request No links

403
Forbidden access No links

¹¹⁹ <https://masking.delphix.com/docs/latest/masking-api-client>

4.4.4 Create/rename

If an attempt is made to create a new entity (or to modify the name of the existing one) with leading or trailing white space or any of the special characters listed above, the operation will fail on a 6.0 or later release with a corresponding error message.

4.4.5 Environment export/import

If any entity name exported from a pre-6.0 version contains leading or trailing white spaces or the special characters listed above, the import operation will fail on a 6.0 or later release with a corresponding error message.

4.4.6 Sync

If a sync bundle from a pre-6.0 version contains leading or trailing white space or any of the special characters listed above, then the Sync import operation will fail on a 6.0 or later release, with a corresponding error message.

4.5 Graphical user interface

4.5.1 Grid UI

This section gives detailed information about how to use the Grid displayed on the Engine UI. UI for Domains, Classifiers, Profiler Sets, Roles, Inventory, Rule Sets, etc displays information using a Grid structure. Each implemented Grid supports functionality like sorting, filtering, pinning, etc. Each subsection gives information about how to use Grid functionalities.

4.5.1.1 Sorting

To identify if the column supports sorting or not

- Hover on the column name. If the `:handpointer:` icon appears i.e. column header is clickable then the column is sortable.

Follow these steps to **sort** the grid based on any one of the grid column data.

- Click on the column header on the grid. The records in the grid will be sorted based on the data in that column.
- Click on the column again to change the sorting order. A vertical directional arrow symbol indicates the sorting order applied (↑ Ascended, ↓ Descending, and blank default sorting order).

CONTINUOUS COMPLIANCE 24.0.0.0

[Environments](#) | [Monitor](#) | [Settings](#) | [Admin](#) | [Audit](#)

Async Tasks

Job Executions

Job Executions

Reset ✕

Environment	Job Name	Status	Progress	Submit Time	Start Time	Type	Actions
env_W3466FEU	prof_1MVP92FA	✔ SUCCEEDED	✔ 5 of 5 - Profiling Complete	31 May 2024 11:48 IST	31 May 2024 11:48 IST	PROFILE	ⓘ
env_OHDTV69	prof_VM6CB1B6	✘ CANCELLED	✘ 5 of 5 - Profiling Complete	31 May 2024 11:47 IST	31 May 2024 11:47 IST	PROFILE	ⓘ
env_XH3M04HZ	prof_2WP7CBOH	✔ SUCCEEDED	✔ 5 of 5 - Profiling Complete	31 May 2024 11:46 IST	31 May 2024 11:46 IST	PROFILE	ⓘ
env_OA5E6DDP	prof_LEC54WA8	⚠ FAILED	⚠ 5 of 5 - Profiling Complete	31 May 2024 11:44 IST	31 May 2024 11:44 IST	PROFILE	ⓘ
env_QAXIIN8R	prof_47GTRFIF	✔ SUCCEEDED	✔ 5 of 5 - Profiling Complete	31 May 2024 11:43 IST	31 May 2024 11:43 IST	PROFILE	ⓘ
env_U1NIG02X	mask_TLYQSJLX	✘ CANCELLED	✘ 10 of 10 - Job Completed	31 May 2024 11:39 IST	31 May 2024 11:39 IST	MASK	ⓘ
env_0H2F7534	mask_XHB54T4U	⚠ FAILED	⚠ 10 of 10 - Job Completed	31 May 2024 11:38 IST	31 May 2024 11:38 IST	MASK	ⓘ
env_XL8L6VSR	mask_HSNW6TOB	⚠ WARNING	⚠ 10 of 10 - Job Completed	31 May 2024 11:36 IST	31 May 2024 11:36 IST	MASK	ⓘ
env_YH2M983L	reidentify_FIYYE5...	✔ SUCCEEDED	✔ 10 of 10 - Job Completed	31 May 2024 11:34 IST	31 May 2024 11:34 IST	RE-IDENTIFY	ⓘ
env_YH2M983L	tokenize_JH6NSM...	✔ SUCCEEDED	✔ 10 of 10 - Job Completed	31 May 2024 11:33 IST	31 May 2024 11:33 IST	TOKENIZE	ⓘ
env_NAM8F24V	tokenize_A4FF1C...	✔ SUCCEEDED	✔ 10 of 10 - Job Completed	31 May 2024 11:31 IST	31 May 2024 11:31 IST	TOKENIZE	ⓘ
env_HNEJJNEH	mask_2TR94XA1	✘ CANCELLED	✘ 10 of 10 - Job Completed	31 May 2024 00:06 IST	31 May 2024 00:06 IST	MASK	ⓘ
env_HLEKYFPS	mask_310XQVVY	⚠ FAILED	⚠ 10 of 10 - Job Completed	31 May 2024 00:04 IST	31 May 2024 00:04 IST	MASK	ⓘ

Displaying 1 to 15 of 56

In the case of grouping, sorting is applied within each group.

- Children of groups can be collapsed and expanded by clicking on the down arrow icon ▼ beside each group name.
- The count next to the group name in parentheses shows the total number of children under that group.

CONTINUOUS COMPLIANCE 26.0.0.0

[Environments](#) | [Monitor](#) | [Settings](#) | [Admin](#) | [Audit](#)

Environments > ENV > testRuleset

Back

testRuleset

All Logical Keys
All Filters
Actions ▼

Status: New Submit

Table	Column	Data Type	Domain	Algorithm	File Format	Actions
▼ TEST1 (3) <small>Filter Condition</small>						ⓘ ⚙ ⌂
	FIRST_NAME	VARCHAR2 (2000)	FIRST_NAME	dlpx-core:FirstName		✎
	ID <small>Primary Key, Index</small>	NUMBER (0)				✎
	LAST_NAME	VARCHAR2 (2000)			json_filter.json	📄 ✎
> TEST10 (3) <small>Logical Key</small>						ⓘ ⚙ ⌂
▼ TEST2 (3)						ⓘ ⚙ ⌂
	FIRST_NAME	VARCHAR2 (2000)				✎
	ID <small>Primary Key, Index</small>	NUMBER (0)				✎

Displaying 1 to 9 of 19

4.5.1.2 Filtering

To identify if the column supports filtering or not,

- Hover on the column header of the grid and click the ≡ icon that appears in the right corner.
- If there is a tab with the 🗑️ icon then filtering is supported for that column.

Follow these steps to add a **Filter** on the grid to show columns or fields only matching certain criteria.

- Hover on any column header of the grid and click the ≡ icon that appears in the right corner.
- When the dialogue appears, choose the the tab with filter icon, then select a required filter criteria and provide the required inputs to filter the grid.

The screenshot displays the 'Algorithms' configuration page in the Continuous Compliance interface. The main content area features a table with the following columns: Algorithm Name, Framework Name, Mask Type, and Owner. A dropdown menu is currently open over the 'Owner' column header, showing a list of filterable values: 'All', 'System', 'deleted-user', 'admin', and 'user_9HKGP8JC'. The 'All' option is selected, indicated by a checkmark. The page also includes a sidebar with navigation options like 'Classifiers', 'Data Formats', 'Domains', 'Expressions', 'JDBC Drivers', and 'Profile Sets'. The top navigation bar includes 'Environments', 'Monitor', 'Settings', 'Admin', and 'Audit'. The bottom right corner of the table area shows 'Displaying 1 to 13 of 68'.

- The applied filter criteria are displayed in the chips on the top of the grid.

Settings > Algorithms

Algorithms

Extension Support Policy + Algorithm

Owner: admin Reset

Algorithm Name	Framework Name	Mask Type	Owner	Actions
ACCOUNT_TK	Tokenization	STRING	admin	...
ALG_CCM7V5ZQ	Tokenization	STRING	admin	...
ALG_H1L7WQC	Tokenization	STRING	admin	...
ALG_HJJQCK5	Tokenization	STRING	admin	...
ALG_I4EVFGIE	Tokenization	STRING	admin	...
ALG_K7CD9EAZ	Tokenization	STRING	admin	...
ALG_QMM90EBQ	Tokenization	STRING	admin	...
ALG_R328IUG9	Tokenization	STRING	admin	...
ALG_SOBM1VNV	Tokenization	STRING	admin	...
ALG_V4YL4UR6	Tokenization	STRING	admin	...
NAME_TK	Tokenization	STRING	admin	...

Displaying 1 to 12 of 12

- To remove a filter criteria, click on the cross mark on the applied filter criteria value listed.
- To remove all filter criteria click on the **Reset** button.
- These filters will be persisted in the browser cache until the user resets it or clears the browser cache.
- On some screens, Grid doesn't display filter in chips, and filters are not persisted i.e. [Monitoring Single Job](#) (see page 513).

In the case of grouping, filtering is applied within each group.

Table	Column	Data Type	Domain	Algorithm	File Format	Actions
TEST1 (1) Filter Condition	ID Primary Key, Index	NUMBER (0)				🔑 ⏴ ⏵
TEST10 (1) Logical Key	ID Primary Key, Index	NUMBER (0)				🔑 ⏴ ⏵
TEST2 (1)	ID Primary Key, Index	NUMBER (0)				🔑 ⏴ ⏵
TEST3 (1)	ID Primary Key, Index	NUMBER (0)				🔑 ⏴ ⏵
TEST4 (1)	ID Primary Key, Index	NUMBER (0)				🔑 ⏴ ⏵

Column: ID Reset

Displaying 1 to 10 of 14

5 types of filters are displayed according to the data of the column.

1. Text Filter with option ["contains"]

Monitor > Job Executions

Job Executions

0 Jobs Queued 0/7 Jobs Running 0/6028 Memory Usage(MB)

Environment	Job Name	progress	Submit Time	Start Time	Type	Actions
env_9U1AURVV	prof_9KI3AMBU	5 of 5 - Profiling Complete	4 Jun 2024 15:18 IST	4 Jun 2024 15:18 IST	PROFILE	👁
env_ITDX1LUK	prof_PVBC9FDJ	5 of 5 - Profiling Complete	4 Jun 2024 15:18 IST	4 Jun 2024 15:18 IST	PROFILE	👁
env_J3DPGFWE	prof_KR7BZONJ	✔️ SUCCEEDED 5 of 5 - Profiling Complete	4 Jun 2024 15:17 IST	4 Jun 2024 15:17 IST	PROFILE	👁
env_8MWQ5HKE	prof_660H799W	❌ FAILED 5 of 5 - Profiling Complete	4 Jun 2024 15:16 IST	4 Jun 2024 15:16 IST	PROFILE	👁
env_B2H8YD2L	prof_95R44MCG	✔️ SUCCEEDED 5 of 5 - Profiling Complete	4 Jun 2024 15:16 IST	4 Jun 2024 15:16 IST	PROFILE	👁
env_GQGOSNE	mask_8W09Y5MY	❌ CANCELLED 10 of 10 - Job Completed	4 Jun 2024 15:16 IST	4 Jun 2024 15:16 IST	MASK	👁
env_F2J2RNOL	mask_MBIH644H	❌ FAILED 10 of 10 - Job Completed	4 Jun 2024 15:15 IST	4 Jun 2024 15:15 IST	MASK	👁
env_JKUCMCP8	mask_GITMTYUR	⚠️ WARNING 10 of 10 - Job Completed	4 Jun 2024 15:15 IST	4 Jun 2024 15:15 IST	MASK	👁

2. Number filter with options ["Equals", "Greater than", "Greater than or equals", "Less than", "Less than or equals", "Not Equals"]

Monitor > Job Executions

Job Executions

0 Jobs Queued 0/7 Jobs Running 0/6028 Memory Usage(MB)

Reset X

Environment	Job Name	Execution	Progress	Submitted	Start Time	Type	Actions
env_9U1AURVY	prof_9KI3AM...	91	5 of 5 - Profiling Complete	4 Jun 2024 1...	4 Jun 2024 1...	PROFILE	👁
env_ITDX1LUK	prof_PVBC9F...	90	5 of 5 - Profiling Complete	4 Jun 2024 1...	4 Jun 2024 1...	PROFILE	👁
env_J3DPGFWE	prof_KR7BZ...	89	5 of 5 - Profiling Complete	4 Jun 2024 1...	4 Jun 2024 1...	PROFILE	👁
env_8MW05HKE	prof_660H79...	88	5 of 5 - Profiling Complete	4 Jun 2024 1...	4 Jun 2024 1...	PROFILE	👁
env_B2H8YD2L	prof_95R44...	87	SUCCEEDED	5 of 5 - Profiling Complete	4 Jun 2024 1...	PROFILE	👁
env_GQGHOSNE	mask_8W09...	86	CANCELLED	10 of 10 - Job Completed	4 Jun 2024 1...	MASK	👁
env_F2J2RNOL	mask_MBIH6...	85	FAILED	10 of 10 - Job Completed	4 Jun 2024 1...	MASK	👁
env_JKUCMCPB	mask_GITMT...	84	WARNING	10 of 10 - Job Completed	4 Jun 2024 1...	MASK	👁

3. Single Select List filter - Select one from the available list of options

The screenshot shows the 'Algorithms' settings page. A table lists various algorithms with columns for Algorithm Name, Framework Name, Mask Type, and Owner. A dropdown filter is open over the Owner column, showing a list of users including 'System', 'deleted-user', 'admin', and 'user_9HKGP8JC'. The 'All' option is selected.

Algorithm Name	Framework Name	Mask Type	Owner
ACCOUNT_SL	Secure Lookup	STRING	System
ACCOUNT_TK	Tokenization	STRING	admin
ADDRESS LINE 2 SL	Secure Lookup	STRING	System
ADDRESS LINE SL	Secure Lookup	STRING	System
ALG_CCM7V5ZQ	Tokenization	STRING	admin
ALG_H1L7WQC	Tokenization	STRING	admin
ALG_HJQCCK5	Tokenization	STRING	admin
ALG_J4EVFGIE	Tokenization	STRING	admin
ALG_K7CD9EAZ	Tokenization	STRING	admin
ALG_QMM90EBQ	Tokenization	STRING	admin
ALG_R328IUG9	Tokenization	STRING	admin
ALG_SOBM1VNV	Tokenization	STRING	admin

4. Multi Select List Filter - Select one or more from the available list of options

Monitor > Job Executions

Job Executions

0 Jobs Queued 0/7 Jobs Running 0/6028 Memory Usage(MB)

Status: FAILED, CANCELLED Reset X

Environment	Job Name	Status	Submit Time	Start Time	Type	Actions
env_ITDX1LUK	prof_PVBC9FDJ	CANCELLED	4 Jun 2024 15:18 IST	4 Jun 2024 15:18 IST	PROFILE	👁
env_8MWOSHKE	prof_660H799W	FAILED	4 Jun 2024 15:16 IST	4 Jun 2024 15:16 IST	PROFILE	👁
env_GQGHOSNE	mask_8W09Y5MY	CANCELLED	4 Jun 2024 15:16 IST	4 Jun 2024 15:16 IST	MASK	👁
env_F2J2RN0L	mask_MBIH644H	FAILED	4 Jun 2024 15:15 IST	4 Jun 2024 15:15 IST	MASK	👁
env_694EG91G	prof_400EVHVDV	CANCELLED	3 Jun 2024 18:02 IST	3 Jun 2024 18:02 IST	PROFILE	👁
env_TLJCOLY4	prof_WIQYIOWD	FAILED	3 Jun 2024 18:00 IST	3 Jun 2024 18:00 IST	PROFILE	👁
env_T4UHI5QV	mask_OIOBBVB0	CANCELLED	3 Jun 2024 18:00 IST	3 Jun 2024 18:00 IST	MASK	👁
env_2NC6VHVX	mask_O6FURGUH	FAILED	3 Jun 2024 18:00 IST	3 Jun 2024 18:00 IST	MASK	👁

5. Date filter - Picks range between two dates

Monitor > Job Executions

Job Executions

0 Jobs Queued 0/7 Jobs Running 0/6028 Memory Usage(MB)

Reset X

Environment	Job N...	Execution ID	Status	Progress	Submit Time	Actions
Env1	sample_pr...	100	CANCELLED	5 of 5 - Profiling Complete	4 Apr 2024 14:19 IS	👁
Env1	sample_pr...	99	SUCCEEDED	5 of 5 - Profiling Complete	4 Apr 2024 14:16 IS	👁
Env1	sample_pr...	98	SUCCEEDED	5 of 5 - Profiling Complete	4 Apr 2024 14:13 IS	👁

4.5.1.3 Pinning columns

Follow these steps to pin a column on the grid.

- Hover on any column header of the grid and click the ≡ icon that appears in the right corner.
- When the dialogue appears, choose the tab with ≡ icon, then choose the **Pin Column** and choose the desired option.

Monitor > Job Executions

Job Executions

0 Jobs Queued 0/7 Jobs Running 0/6028 Memory Usage(MB)

Reset X

Type	Name	Status	Progress	Submit Time	Start Time	Actions
PROFILE	660H799W	SUCCEEDED	5 of 5 - Profiling Complete	4 Jun 2024 15:18 I...	4 Jun 2024 15:18 I...	
PROFILE	prof_95R44MCG	CANCELLED	5 of 5 - Profiling Complete	4 Jun 2024 15:18 I...	4 Jun 2024 15:18 I...	
PROFILE	660H799W	SUCCEEDED	5 of 5 - Profiling Complete	4 Jun 2024 15:17 I...	4 Jun 2024 15:17 IST	
PROFILE	660H799W	FAILED	5 of 5 - Profiling Complete	4 Jun 2024 15:16 I...	4 Jun 2024 15:16 I...	
PROFILE	env_B2H8YD2L prof_95R44MCG	SUCCEEDED	5 of 5 - Profiling Complete	4 Jun 2024 15:16 I...	4 Jun 2024 15:16 I...	
MASK	env_GQGHSNE mask_8W09Y5MY	CANCELLED	10 of 10 - Job Completed	4 Jun 2024 15:16 I...	4 Jun 2024 15:16 I...	
MASK	env_F2J2RN0L mask_MBIH644H	FAILED	10 of 10 - Job Completed	4 Jun 2024 15:15 I...	4 Jun 2024 15:15 IST	
MASK	env_JKUCMCP8 mask_GITMTYUR	WARNING	10 of 10 - Job Completed	4 Jun 2024 15:15 I...	4 Jun 2024 15:15 IST	
RE-IDENTIFY	env_K0ME9FK9 reidentify_DZG0AS...	SUCCEEDED	10 of 10 - Job Completed	4 Jun 2024 15:14 I...	4 Jun 2024 15:14 I...	

4.5.1.4 Auto-sizing columns

Follow these steps to adjust the column size.

- Hover on any column header of the grid and click the ≡ icon that appears in the right corner.
- When the dialogue appears, choose the tab with ≡ icon, then choose the desired option from the list.

Monitor > Job Executions

Job Executions

0 Jobs Queued 0/7 Jobs Running 0/6028 Memory Usage(MB)

Reset X

Environment	Job Name	Status	Submit Time	Start Time	Type	Actions
env_9U1AURVY	prof_9KI3AMBU	SUCCEEDED	4 Jun 2024 15:18 IST	4 Jun 2024 15:18 IST	PROFILE	
env_ITDX1LUK	prof_PVBC9FDJ	CANCELLED	4 Jun 2024 15:18 IST	4 Jun 2024 15:18 IST	PROFILE	
env_J3DPGFWF	prof_KR7BZONJ	SUCCEEDED	4 Jun 2024 15:17 IST	4 Jun 2024 15:17 IST	PROFILE	
env_8MWQ5HKE	prof_660H799W	FAILED	4 Jun 2024 15:16 IST	4 Jun 2024 15:16 IST	PROFILE	
env_B2H8YD2L	prof_95R44MCG	SUCCEEDED	5 of 5 - Profiling Completed	4 Jun 2024 15:16 IST	PROFILE	
env_GQGHSNE	mask_8W09Y5MY	CANCELLED	10 of 10 - Job Completed	4 Jun 2024 15:16 IST	MASK	
env_F2J2RN0L	mask_MBIH644H	FAILED	10 of 10 - Job Completed	4 Jun 2024 15:15 IST	MASK	
env_JKUCMCP8	mask_GITMTYUR	WARNING	10 of 10 - Job Completed	4 Jun 2024 15:15 IST	MASK	

4.5.1.5 View more columns

To see the other grid columns which are not displayed by default on the grid.

- Hover on any column header of the grid and click on the ≡ icon that appears in the right corner.
- When the dialogue appears, choose the last tab, then choose any desired columns from the list below to view it on the grid.

Monitor > Job Executions

Job Executions

0 Jobs Queued 0/7 Jobs Running 0/6028 Memory Usage(MB)

Reset X

Job Name	Execu...	Status	Submit Time	Start Time	Type	Actions
prof_9KI3AMBU	91	SUCCEEDED	4 Jun 2024 15:18 IST	4 Jun 2024 15:18 IST	PROFILE	
prof_PVBC9FDJ	90	CANCELLED	4 Jun 2024 15:18 IST	4 Jun 2024 15:18 IST	PROFILE	
prof_KR7BZONJ	89	SUCCEEDED	4 Jun 2024 15:17 IST	4 Jun 2024 15:17 IST	PROFILE	
prof_660H799W	88	FAILED	4 Jun 2024 15:16 IST	4 Jun 2024 15:16 IST	PROFILE	
prof_95R44MCG	87	SUCCEEDED	4 Jun 2024 15:16 IST	4 Jun 2024 15:16 IST	PROFILE	
mask_8W09Y5MY	86	CANCELLED	4 Jun 2024 15:16 IST	4 Jun 2024 15:16 IST	MASK	
mask_MBIH644H	85	FAILED	4 Jun 2024 15:15 IST	4 Jun 2024 15:15 IST	MASK	
mask_GITMTYUR	84	WARNING	4 Jun 2024 15:15 IST	4 Jun 2024 15:15 IST	MASK	
reidentify_DZG0...	83	SUCCEEDED	4 Jun 2024 15:14 IST	4 Jun 2024 15:14 IST	RE-IDENTIFY	

4.5.1.6 Actions

To perform an action on any of the rows, Click the (...) button to the right of the corresponding row under the **Actions** column, list of actions will be visible. If any action is not supported or not permitted for that row then it will be disabled.

C CONTINUOUS COMPLIANCE 24.0.0.0

[Environments](#)
[Monitor](#)
[Settings](#)
[Admin](#)
[Audit](#)

Admin > Roles

Roles

+ Role

Name ↑	Type	Actions
All Privileges	DEFAULT	...
DBA	DEFAULT	...
Environment Owner	DEFAULT	...
IT Security Analyst	DEFAULT	...
Operator	DEFAULT	...
role_I0MC90LY	CUSTOM	...
SME	DEFAULT	...

View

Edit

Duplicate

Delete

Displaying 1 to 7 of 7

4.5.1.7 Copy cell data

Data from the grid cell can be copied by first selecting a cell and then using the CTRL+C or Command+C. The cell will be highlighted once the content has been copied.

X CONTINUOUS COMPLIANCE 24.0.0.0
 Environments Monitor Settings Admin Audit
👤 ?

Algorithms
Settings > Algorithms
🔗 Extension Support Policy + Algorithm

- Algorithms
- Classifiers
- Data Formats
- Domains
- Expressions
- JDBC Drivers
- Profile Sets

Algorithms

Algorithm Name	Framework Name	Mask Type	Owner	Actions
ACCOUNT_SL	Secure Lookup	STRING	System	⋮
ACCOUNT_TK	Tokenization	STRING	deleted-user	⋮
ADDRESS LINE 2 SL	Secure Lookup	STRING	System	⋮
ADDRESS LINE SL	Secure Lookup	STRING	System	⋮
BUSINESS LEGAL ENTITY SL	Secure Lookup	STRING	System	⋮
COMMENT_SL	Secure Lookup	STRING	System	⋮
CREDIT_CARD	Payment Card	STRING	System	⋮
DATE_SHIFT(DISCRETE)	Date Shift Discrete <small>PLUGIN</small>	LOCAL_DATE_TIME	System	⋮
DATE_SHIFT(FIXED)	Date Shift	LOCAL_DATE_TIME	System	⋮

Displaying 1 to 10 of 59

4.5.2 Select/Deselect all the records

To select all the records while doing any selection, use the checkbox given in the header to select all the records. The same checkbox can be also used for deselecting all the records.

Add Database Table ✕

Add Tables

Check one or more table names to select them for inclusion in the Rule Set.

Table Name


- data_table
- data_table1
- data_table10
- data_table11
- data_table12
- data_table2
- data_table3

Total number of tables selected: 13

Cancel Save

4.5.3 List Editor UI

This section gives detailed information about how to use the List editor on the Engine UI. UI for Add and Edit algorithms uses this for updating details. We've taken an example of how to add constants for the Numeric expression algorithm using List Editor.

To add items to the list click on the edit icon  on the right side. If there are no items present in the list then it will display a message related to an empty list.

Add Algorithm



- Details
- Configuration
- Summary

Configuration

Configure the algorithm.

Expression
input * 0.5

Input Type
double

Replacement Value for Nonconforming Data
XXX

Constants

There are no constants.



Numeric Expression

Framework Description

Numeric Expression masks input by evaluating it within a one-line, user-defined expression, which must be a valid Java expression that returns a java.math.BigDecimal object or another object that can be converted into a BigDecimal. The expression can reference user-defined constant variables that remain fixed for the life of the algorithm, as well as a special long integer "seed" constant whose value is based on the algorithm key.

Cancel Back **Next** Save

On clicking the edit icon will open a new section for adding items.

Add Algorithm



- Details
- Configuration
- Summary

Configuration

Configure the algorithm.

Expression
input * 0.5

Input Type
double

Replacement Value for Nonconforming Data
XXX

Constants

There are no constants. +

Numeric Expression


Framework Description


Numeric Expression masks input by evaluating it within a one-line, user-defined expression, which must be a valid Java expression that returns a java.math.BigDecimal object or another object that can be converted into a BigDecimal. The expression can reference user-defined constant variables that remain fixed for the life of the algorithm, as well as a special long integer "seed" constant whose value is based on the algorithm key.



Cancel Back **Next** Save



Click **+** on the top right of the section to add an item. It will display input fields related to configuration. Here Name and Value inputs are displayed.




Constants



Learn about Constants 

To add another item on top use  on the top right, to add another item below any item use  beside the item. To delete an item click  beside an item.

Constants

+

Learn about Constants i

Name

var1

Value

4

+ 🗑

Name

var2

Value

6

+ 🗑

× ✓

To save the changes click on ✓ . To discard the changes click on × . Once changes are saved details will be displayed.

Add Algorithm



- Details
- Configuration
- Summary

Configuration

Configure the algorithm.

Expression
input * 0.5

Input Type
double

Replacement Value for Nonconforming Data
XXX

Constants 

Name	var1
Value	4
Name	var2
Value	6

Numeric Expression

Framework Description

Numeric Expression masks input by evaluating it within a one-line, user-defined expression, which must be a valid Java expression that returns a `java.math.BigDecimal` object or another object that can be converted into a `BigDecimal`. The expression can reference user-defined constant variables that remain fixed for the life of the algorithm, as well as a special long integer "seed" constant whose value is based on the algorithm key.

Cancel Back **Next** Save

4.6 Managing application settings

This section describes how continuous compliance application settings can be managed. Visit [API calls for masking administration](#)¹²⁰ to see the supported types of administrative APIs.

4.6.1 The Application Settings screen

Application Settings allows an administrator to change the Delphix Continuous Compliance Engine settings. Presently there are 12 categories of settings:

- LDAP settings
- General settings
- Mask settings
- ASDD settings
- JOB settings
- SMTP settings
- engineSync settings
- Async settings
- Algorithm settings
- FileUpload settings
- Database Settings
- Profile settings.

¹²⁰ <https://masking.delphix.com/docs/latest/api-calls-for-masking-administration>

Refer [here \(see page 0\)](#) for a detailed description of each application's settings before making changes. Over time, more settings will be added to give the admin direct control over the product's various settings.

Application Settings

Application settings are used to control the behavior of the application and are not intended to be changed frequently. Changing these settings can have a significant impact on the behavior of the application. Please consult the documentation before making changes.

Admin > Application Settings

Application Settings

Setting Group	Setting Name	Setting Value
general (19)		
	AllowPasswordResetRequest	true
	NumSimulJobsAllowed	7
	PasswordResetLinkDuration	5
	EnableMonitorRowCount	true
	PasswordTimeSpan	23
	PasswordCount	3
	JobOutputCleanerPeriodIntervalSeconds	86400
	MaximumQueuedJobs	0
	MountConsistencyCheckPeriod	5

4.6.2 Modify Application Settings values

Application settings are used to control the behavior of the application and are not intended to be changed frequently. Changing these settings can have a significant impact on the behavior of the application.

Use these instructions to modify the application settings value.

1. From the **Admin** section, select the **Application Settings** option.
2. A list of all application settings appears in a grid grouped by settings group name.
3. You can filter or sort data by the various informational fields, by clicking on those respective fields. More information on grid filtering and sorting can be found on the [Graphical user interface](#)¹²¹ page.
4. To find a particular setting, input the **Setting Name** into the search field located in the table header under "Setting Name."
5. Clicking on the pencil icon positioned in the rightmost column of each setting will reveal an input field. For a boolean setting value, a drop-down with values true & false will be visible and for other value types, an input text area will be displayed.
6. After entering the desired setting value, click outside the input field. A confirmation dialog will then appear, giving you the option to save or discard the changes.

¹²¹ <https://masking.delphix.com/docs/latest/graphical-user-interface>

The screenshot shows the 'Application Settings' page in the Continuous Compliance Admin interface. The top navigation bar includes 'Environments', 'Monitor', 'Settings', 'Admin' (selected), and 'Audit'. A warning banner at the top states: 'Application Settings: Application settings are used to control the behavior of the application and are not intended to be changed frequently. Changing these settings can have a significant impact on the behavior of the application. Please consult the documentation before making changes.'

The left sidebar contains navigation links: 'About', 'Application Settings' (selected), 'Logs', 'Roles', 'Users', and 'Utilization'. The main content area is titled 'Application Settings' and displays a table of settings under the 'general (19)' group.

Setting Group	Setting Name	Setting Value	
general (19)	AllowPasswordResetRequest	true	
	NumSimulJobsAllowed	7	
	PasswordResetLinkDuration	5	
	EnableMonitorRowCount	true	
	PasswordTimeSpan	<input type="text" value="true"/> false	
	PasswordCount	3	
	JobOutputCleanerPeriodIntervalSeconds	86400	
	MaximumQueuedJobs	0	
	MountConsistencyCheckPeriod	5	
	RulesetRefreshThreads	8	
	MaximumMemoryForJobs	0.0	



- This screen is accessible only to admin users.
- The application settings UI displays only settings which are enabled for users to view.

4.7 Users and roles

The Delphix Masking Service has a flexible and robust users and roles system that allows you to give users fine-grain privileges over what environments they have access to and what tasks they can and can not perform.

4.7.1 What are roles?

A defined role is what is used to give certain users privileges over certain environments and tasks. Roles can be defined by selecting a subset of actions that can be taken on certain objects.

Navigate to **Admin > Roles** to view all roles. From here, the User can view and modify roles.

Admin > Roles

Roles + Role

Name ↑	Type	Actions
All Privileges	DEFAULT	...
DBA	DEFAULT	...
Environment Owner	DEFAULT	...
IT Security Analyst	DEFAULT	...
Operator	DEFAULT	...
role_I0MC90LY	CUSTOM	...
SME	DEFAULT	...

Displaying 1 to 7 of 7

The roles on the screen can be filtered or sorted by fields **Name** and **Type** by clicking on the respective fields. More information on grid filtering and sorting can be found [here \(see page 232\)](#).

4.7.1.1 Role Type

With the 13.0.0.0 release, a New parameter Role type has been added to the product. There are 2 types of roles. **CUSTOM** and **DEFAULT**. All built-in system roles will have a type of **DEFAULT** and others will be the type of **CUSTOM**.

There have been 5 roles added to the product with the role type **DEFAULT**.

1. IT Security Analyst
2. DBA
3. SME
4. Operator
5. Environment Owner

If a role with the name 'All Privileges' exists on the product then it will be marked as a **DEFAULT** role type.

If a role with any of the above names already exists then the role being added will be renamed by appending sequence number. For example, If '**DBA**', and '**DBA 1**' are already present in the system then the role with the name '**DBA 2**' will be added to the product with **DEFAULT** role type.

4.7.1.2 Actions

When defining a role, you can select one or more of the following actions for the role to be able to perform:

- **View:** Be able to view the object and important information about the object.
- **Add:** Be able to add an instance of an object.
- **Update:** Be able to update/edit an instance of an object.
- **Delete:** Be able to delete an instance of an object.
- **Copy:** Be able to create a copy of an object.
- **Export:** Be able to export an object from a Delphix Engine.
- **Import:** Be able to import an exported object into a Delphix Engine

Please note that not all of these actions are available for all objects in the masking service.

4.7.1.3 Objects

While defining a role, permission to perform the above actions can be defined on a per-object basis. These objects include:

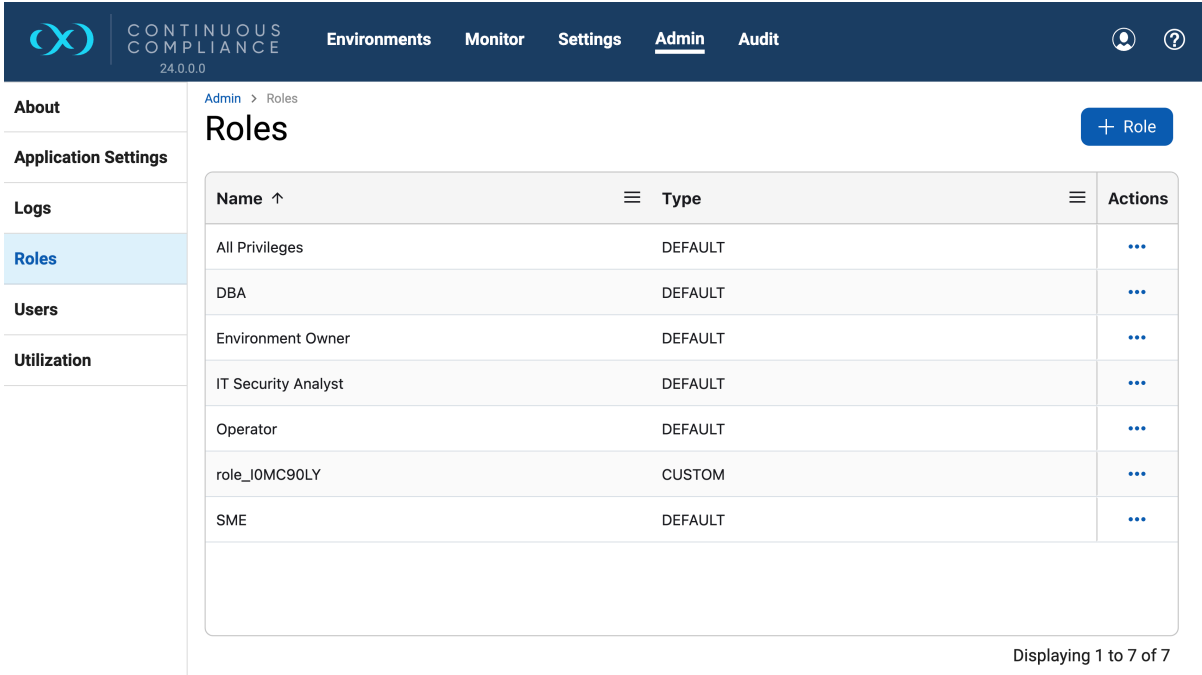
General	Jobs	Settings	Report	Approval Workflow
Environment	Profile Job	Domains	Inventory Report	Approve Inventories
Connection	Masking Job	Algorithms		
Ruleset	Tokenize Job	Plugins		
Inventory	Re-identify Job	Profile Expression		
		Profiler Set		
		File Format		
		JDBC Drivers		
		Password Vault		
		User		
		Diagnostic		

Refer to [Delphix Masking Terminology \(see page 303\)](#) for definitions of these objects. View privilege for Plugins and JDBC Drivers should be always true for any type of role. Environment Export privilege permission is no longer supported.

4.7.1.4 Adding a role

To add a role follow these steps:

- 1. Login into the **Masking Engine**, and select **Admin > Roles**.
- 2. Click the **+ Role** button from the top-right corner just above the roles grid.



- 3. A full-screen dialog will appear for adding a new role. Enter a **Role Name**. The far-left column lists the items for which you can set actions.

Add Role



Role Name

General	View	Add	Update	Delete	Copy	Import	Export
Environment	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
Connection	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			
Rule Set	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
Inventory	<input type="checkbox"/>		<input type="checkbox"/>			<input type="checkbox"/>	<input type="checkbox"/>
Jobs	View	Add	Update	Delete	Run		
Profile Job	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
Masking Job	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
Tokenize Job	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
Re-identify Job	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
Settings	View	Add	Update	Delete			
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			
Report	View						
	<input type="checkbox"/>						
Approval Workflow	View						
	<input type="checkbox"/>						

Cancel Save

- Select the checkboxes for the corresponding actions that you want to apply. If there is no checkbox, that action is not available. For example, if you want this role to have View, Add, Update, Delete, and Run actions for masking jobs, select the corresponding checkboxes in the **Masking Job** row.

Add Role



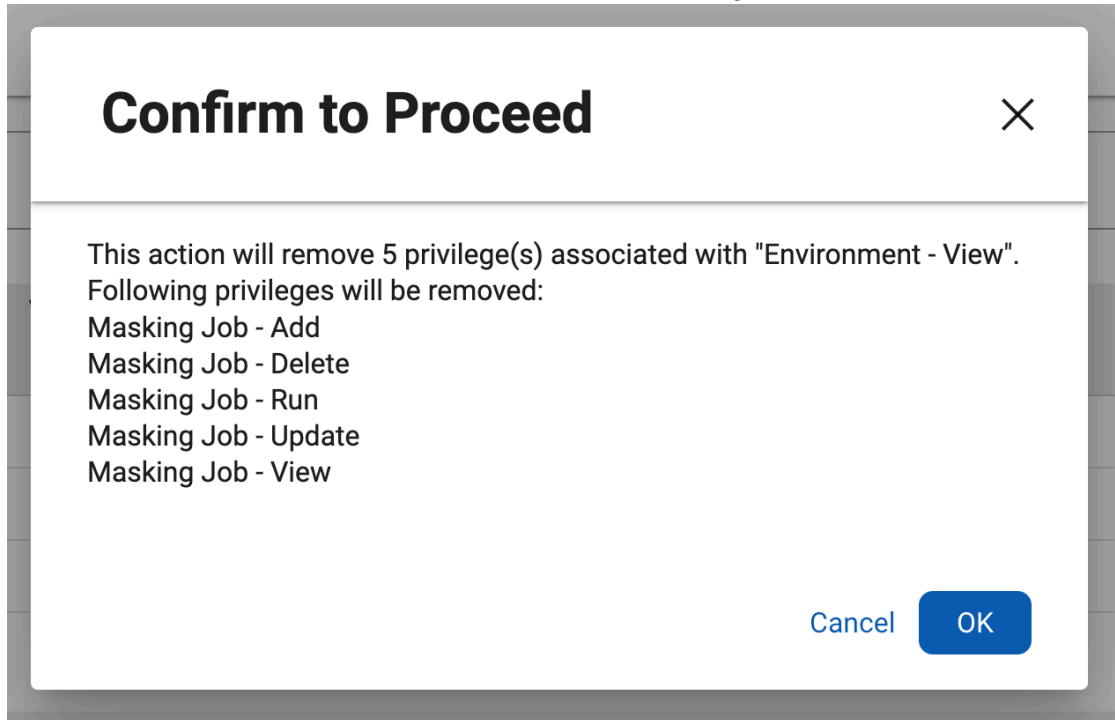
Role Name
Demo Role

General	View	Add	Update	Delete	Copy	Import	Export
Environment	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Connection	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			
Rule Set	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
Inventory	<input type="checkbox"/>		<input type="checkbox"/>			<input type="checkbox"/>	<input type="checkbox"/>
Jobs	View	Add	Update	Delete	Run		
Profile Job	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
Masking Job	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Tokenize Job	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
Re-identify Job	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
Settings	View	Add	Update	Delete			
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			
Report	View						
	<input type="checkbox"/>						
Approval Workflow	View						
	<input type="checkbox"/>						

Cancel Save

- If you are removing any actions with dependents selected then, a dialog box will appear asking for confirmation for removing actions with all its dependents.

- a. For example, Masking Job [View, Add, Update, Delete, Run] depends on Environment - View. Therefore, if the user tries to remove Environment - View, it will ask for confirmation to remove it with the list of dependents. If users do not wish to remove it then they can select the **Cancel** or **Close icon** else can select **OK** and move forward with changes.



6. When you are finished assigning privileges for this Role, click **Save**.

4.7.1.5 Recommended roles

While every organization will differ in what users and roles they define, Delphix uses these common/popular roles. Please note that each defined user can only have one role assigned to them.

- **Administrator:** This role is assigned by enabling a user's Administrator setting in either the UI or API. A user with this role has unrestricted access to all the engine functions. Specifically, the user has all privileges available through the roles system and the following additional, Administrator-only privileges:
 - [Sync](#) (see page 842)
 - A User's `apiAccess` and `userStatus` setting
 - Audit Page
 - Admin > Users Generate Key Button
 - Admin > Email Notification
 - Admin > Utilization
 - Deletion of any object: An Admin can delete any object, such as any Algorithm, Domain, Profile Expression, or Profile Set. In contrast, a user with the **All Privileges** role can only delete objects they created.

- Settings > Roles
- **IT Security analyst:** Unrestricted access for all settings functions; access to all application functions except environment and environment create, delete, update.
- **All Privileges:** Unrestricted access to an application environment; central admin or security analyst will determine if this role can modify settings.
- **DBA:** Manage connections for the application database, scripting, and scheduling (no settings).
- **SME/Analyst/Developer:** Manage inventories, create, and view jobs.
- **Operator:** All job privileges.
- **Environment Owner:** Approve workflow and inventories, privileges to view for settings and environment.

4.7.1.6 Modifying Roles

Users can perform 4 types of action on this screen by clicking the (...) button to the right of the corresponding row under the **Actions** column.

The screenshot shows the 'Roles' page in the Continuous Compliance Admin interface. The breadcrumb is 'Admin > Roles'. The table lists the following roles:

Name ↑	Type	Actions
All Privileges	DEFAULT	...
DBA	DEFAULT	...
Environment Owner	DEFAULT	...
IT Security Analyst	DEFAULT	...
Operator	DEFAULT	...
role_IOMC90LY	CUSTOM	...
SME	DEFAULT	...

The 'Actions' dropdown menu includes: View, Edit, Duplicate, and Delete. The status at the bottom right indicates 'Displaying 1 to 7 of 7'.

4.7.1.6.1 1. View Role

The user can view the role details. Everything on the dialog form will be disabled in case of view is selected.

View Role



Role Name
Demo Role

General	View	Add	Update	Delete	Copy	Import	Export
Environment	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
Connection	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			
Rule Set	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
Inventory	<input type="checkbox"/>		<input type="checkbox"/>			<input type="checkbox"/>	<input type="checkbox"/>

Jobs	View	Add	Update	Delete	Run
Profile Job	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Masking Job	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Tokenize Job	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Re-identify Job	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Settings	View	Add	Update	Delete
Domains	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Algorithms	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Plugins	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Close

4.7.1.6.2 2. Edit Role

The user can edit the actions of roles. The role name is not allowed to be edited. It will be disabled.

Edit Role



Role Name
Demo Role

General	View	Add	Update	Delete	Copy	Import	Export
Environment	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
Connection	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			
Rule Set	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
Inventory	<input type="checkbox"/>		<input type="checkbox"/>			<input type="checkbox"/>	<input type="checkbox"/>

Jobs	View	Add	Update	Delete	Run
Profile Job	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Masking Job	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Tokenize Job	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Re-identify Job	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Settings	View	Add	Update	Delete
Domains	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Algorithms	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Plugins	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Cancel Save

4.7.1.6.3 3. Duplicate Role

On selecting a duplicate option, actions will be pre-selected to the new role screen dialog, and the user can give a new name and duplicate the role.

Duplicate Role ✕

^ General	View	Add	Update	Delete	Copy	Import	Export
Environment	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Connection	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			
Rule Set	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
Inventory	<input type="checkbox"/>		<input type="checkbox"/>			<input type="checkbox"/>	<input type="checkbox"/>

^ Jobs	View	Add	Update	Delete	Run
Profile Job	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Masking Job	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Tokenize Job	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Re-identify Job	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

^ Settings	View	Add	Update	Delete
Domains	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Algorithms	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Plugins	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Cancel Save

4.7.1.6.4 4. Delete Role

Delete Role ✕

Are you sure you want to delete "Demo Role" role?

Cancel Confirm

- Roles with **CUSTOM** type can be added, edited, deleted, duplicated, and viewed.

- Roles with **DEFAULT** type cannot be added, edited, or deleted. It can only be viewed and duplicated.

To Modify or Add a role using Masking API, follow these steps:

1. Access the API client on your Masking Engine, from `http://myMaskingEngine.myDomain.com/masking/api-client`.
2. Login into the Masking Engine and select the Role endpoint.

role		^
GET	/roles Get all roles	⌵ 🔒
POST	/roles Create role	⌵ 🔒
GET	/roles/{roleId} Get role by ID	⌵ 🔒
PUT	/roles/{roleId} Update role by ID	⌵ 🔒
DELETE	/roles/{roleId} Delete role by ID	⌵ 🔒

3. Execute API requests by providing request parameters and body as mentioned in the example. ([Sample JSON for Add/Update role API](#) (see page 266))

4.7.2 What are users?

Once you have your roles defined, it is time to create users with those roles. We highly recommend creating independent users for each individual who will have access to the masking service.

Navigate to **Admin > Users** to view all users. The users on the screen can be filtered or sorted by the various informational fields by clicking on the respective fields. More information on grid filtering and sorting can be found [here](#)¹²².

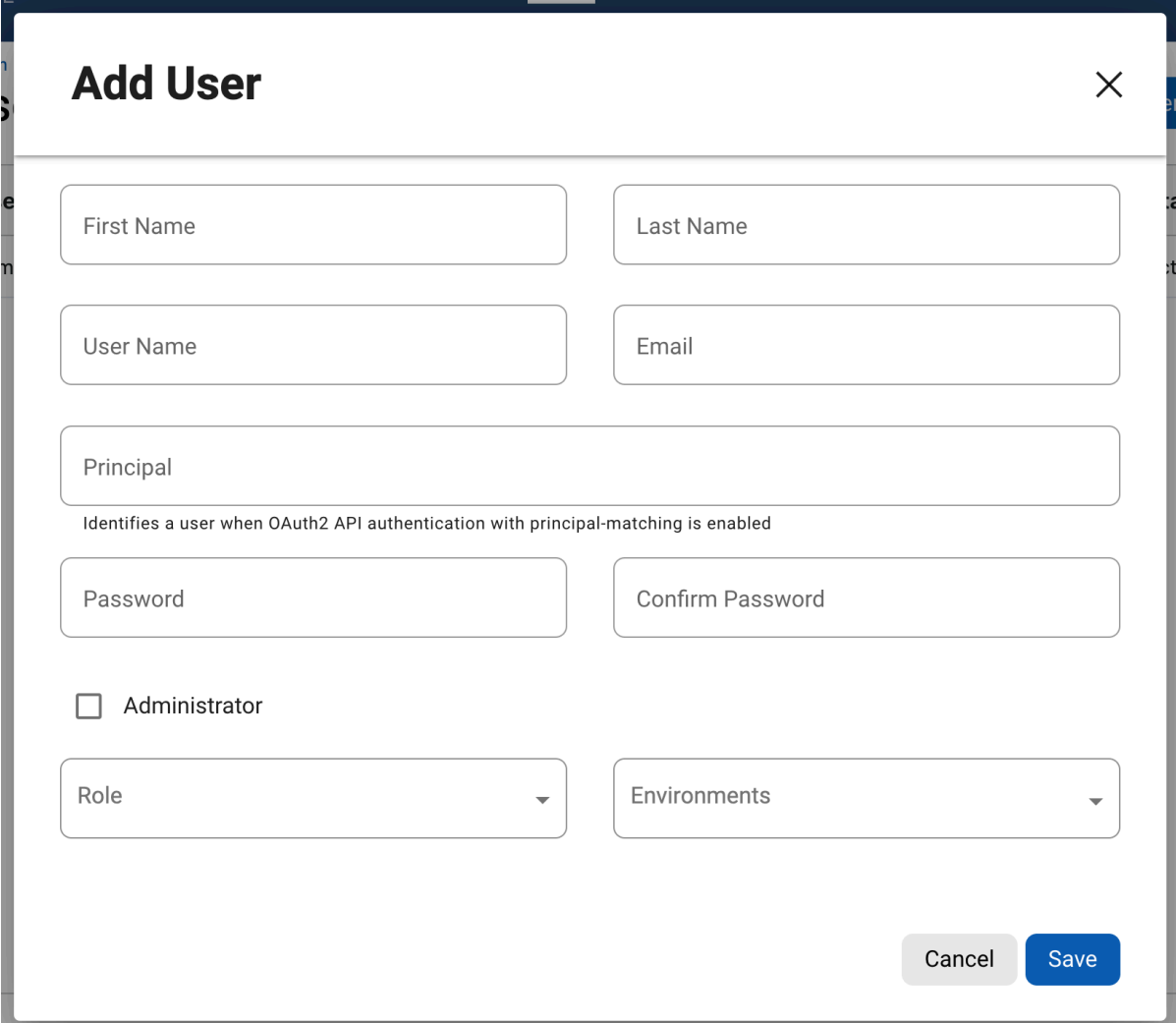
¹²² <https://masking.delphix.com/docs/latest/graphical-user-interface>

 Sortable and Filterable columns are User Name, Last Name, First Name, Email, and Admin.

4.7.2.1 Adding a user

To create a new user using the Masking UI follow these steps:

1. Login into the **Masking Engine**, and select **Admin > Users**.
2. Click the **+ User** button from the top-right corner just above the user grid.
3. A dialog will appear for adding a new user.



Add User ✕

First Name

Last Name

User Name

Email

Principal

Identifies a user when OAuth2 API authentication with principal-matching is enabled

Password

Confirm Password

Administrator

Role

Environments

Cancel Save

4. You will be prompted for the following information:
 - **First Name:** (Optional) The user's given name
 - **Last Name:** (Optional) The user's surname

- **User Name:** The login name for the user
- **Email:** The user's e-mail address (mailable from the Delphix Masking Engine server for purposes of job completion e-mail messages)
- **Principal:** Identifies this user on external identity services. Used for OAuth2 API authentication when the principal is selected as the field to match users with access tokens.
- **Password:** The password that the Delphix Masking Engine uses to authenticate the user on the login page. The password must be at least eight characters long, but no longer than 65 characters. It must also contain a minimum of one uppercase character, one unique character (!@#\$%^&*), and one number.
- **Confirm Password:** Confirm the password with double-entry to avoid data entry errors.
- **Administrator:** (Optional) Select the Administrator checkbox if you want to give this user Administrator privileges. (Administrator privileges allow the user to perform all Delphix Masking Engine tasks, including creating and editing users in the Delphix Masking Engine.) If you select the Administrator checkbox, the Roles and Environments fields disappear because Administrator privileges include all roles and environments.
- **Role:** Select the role to grant to this user. The choices here depend on the custom roles that you have created. You can assign one role per user name.
- **Environments:** Enter as many environments as this user will be able to access. Granting a user access to a given environment does not give them unlimited access to that environment. The user's access is still limited to their assigned role.

5. When you are finished, click **Save**.



When a user is created, it's Account Status is *Active* by default.

To create a new user using the Masking API follow these steps:

1. Access the API client on your Masking Engine, from `http://myMaskingEngine.myDomain.com/masking/api-client`.
2. Login into the Masking Engine and select the User endpoint.

user			^
GET	/users	Get all users	🔒 ↓
POST	/users	Create user	🔒 ↓
GET	/users/{userId}	Get user by ID	🔒 ↓
PUT	/users/{userId}	Update user by ID	🔒 ↓
DELETE	/users/{userId}	Delete user by ID	🔒 ↓
POST	/users/forgot-password	Send Reset password mail to the user	🔒 ↓
POST	/users/reset-password	Reset new password for the user	🔒 ↓

- Click Create users using the POST /users section and refer to the Example Value for parameters required for new users.

POST /users Create user 🔒 ^

Parameters

No parameters

Request body required application/json

The user to create

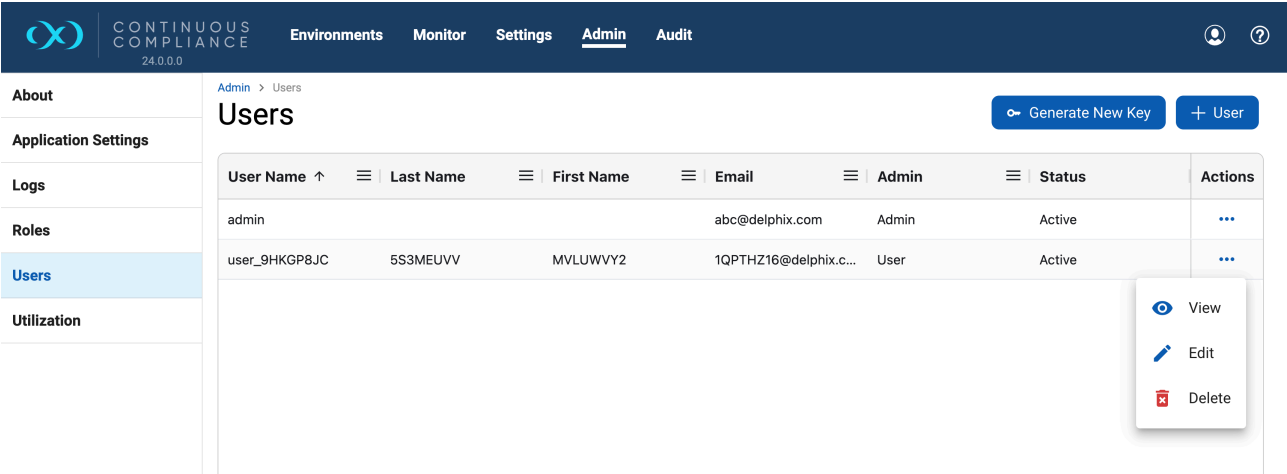
```
{
  "userName": "DelphixUser1",
  "password": "Password_123",
  "firstName": "First",
  "lastName": "Last",
  "email": "user@delphix.com",
  "isAdmin": false,
  "showWelcome": true,
  "userStatus": "ACTIVE",
  "nonAdminProperties": {
    "roleId": 1,
    "environmentIds": [
      1,
      3,
      7
    ]
  }
}
```

Execute

- Enter valid User creation JSON in the **body** section, refer to sample create users JSON ([Sample New User Create JSON](#) (see page 270))
- Click on **Execute**.

4.7.2.2 Modifying User

Users can perform 3 types of action on this screen by clicking the (...) button to the right of the corresponding row under the **Actions** column.



4.7.2.3 View User

For viewing user details. Everything on the dialog form will be disabled in case of view is selected.

View User

SIGU458K

I4NE8U1P

user_0600A2YB

ECC1PLNP@delphix.com

Identifies a user when OAuth2 API authentication with principal-matching is enabled

Administrator

Enable Welcome Page

role_3NETFZ97

Environments

ACTIVE

4.7.2.4 Edit User

The Edit user dialog will appear with existing user details.

The following user information can be modified through the **Edit User** screen:

1. First Name
2. Last Name
3. Email Address
4. Principal
5. Password
6. Administrator Status
7. Welcome Page Status

8. Account Status (cannot be changed to *Locked*)
9. User Roles (non-admin users only)
10. User Environments (non-admin users only)

Edit User ✕

First Name	SIGU458K	Last Name	I4NE8U1P
User Name	user_0600A2YB	Email	ECC1PLNP@delphix.com
Principal			
Identifies a user when OAuth2 API authentication with principal-matching is enabled			
<input type="checkbox"/> Change Password			
<input type="checkbox"/> Administrator			
<input checked="" type="checkbox"/> Enable Welcome Page			
Role	role_3NETFZ97	Environments	
Account Status	ACTIVE		

Cancel Save


To update user information using Masking API, follow these steps:

1. Access the API client on your Masking Engine, from `http://myMaskingEngine.myDomain.com/masking/api-client`.
2. Login into the Masking Engine and select the User endpoint.

user			^
GET	/users	Get all users	🔒 ↓
POST	/users	Create user	🔒 ↓
GET	/users/{userId}	Get user by ID	🔒 ↓
PUT	/users/{userId}	Update user by ID	🔒 ↓
DELETE	/users/{userId}	Delete user by ID	🔒 ↓
POST	/users/forgot-password	Send Reset password mail to the user	🔒 ↓
POST	/users/reset-password	Reset new password for the user	🔒 ↓

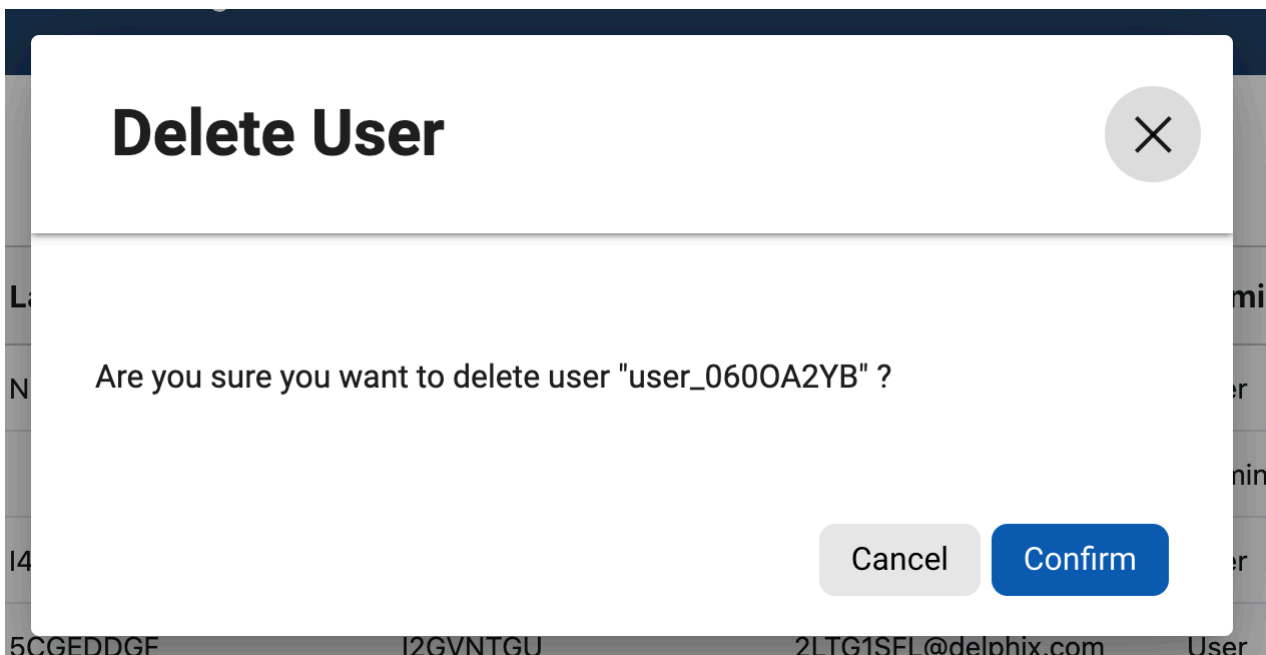
Click **Update user by ID** and refer to the **Example Value** for parameters required for updating users.

3. Enter valid User creation JSON in the **body** section, refer to sample create users JSON. ([Sample User JSON](#) (see page 270))
4. Click on **Execute**.

 User's Account Status will be automatically changed to *Locked* on 3 invalid login attempts.

4.7.2.5 Delete User

On clicking the Delete option, The confirmation dialog will appear for deleting the user. If do not wish to remove it then they can select the **Cancel** or **Close icon** else can select **Confirm** and move forward with changes.



To delete a user using Masking API follow these steps:

1. Access the API client on your Masking Engine, from the `http://myMaskingEngine.myDomain.com/masking/api-client`.
2. Login into the Masking Engine and select the User endpoint.

user			^
GET	/users	Get all users	🔒 ↓
POST	/users	Create user	🔒 ↓
GET	/users/{userId}	Get user by ID	🔒 ↓
PUT	/users/{userId}	Update user by ID	🔒 ↓
DELETE	/users/{userId}	Delete user by ID	🔒 ↓
POST	/users/forgot-password	Send Reset password mail to the user	🔒 ↓
POST	/users/reset-password	Reset new password for the user	🔒 ↓

3. Click **Delete user by ID** and enter the user ID for the user to be deleted.

DELETE /users/{userId} Delete user by ID 🔒 ^

Parameters

Name	Description
userId * required integer(\$int32) (path)	The ID of the user to delete

Execute

Responses

Code	Description	Links
200	Success	No links
400	Bad request	No links
401	Unauthorized access	No links
403	Forbidden access	No links
404	Not found	No links

4. Click on **Execute**.

4.7.3 Sample JSON

This section contains the following articles:

- [All privileges \(see page 266\)](#)
- [Create new user \(see page 269\)](#)

- [User update](#) (see page 270)

4.7.3.1 All privileges

```
{
  "roleName": "All Privileges",
  "environment": {
    "copy": true,
    "create": true,
    "delete": true,
    "export": true,
    "import": false,
    "run": false,
    "update": true,
    "view": true
  },
  "connector": {
    "copy": false,
    "create": true,
    "delete": true,
    "export": false,
    "import": false,
    "run": false,
    "update": true,
    "view": true
  },
  "ruleset": {
    "copy": true,
    "create": true,
    "delete": true,
    "export": false,
    "import": false,
    "run": false,
    "update": true,
    "view": true
  },
  "inventory": {
    "copy": false,
    "create": false,
    "delete": false,
    "export": true,
    "import": true,
    "run": false,
    "update": true,
    "view": true
  },
  "profileJob": {
    "copy": false,
    "create": true,
```



```
"delete": true,
"export": false,
"import": false,
"run": true,
"update": true,
"view": true
},
"maskingJob": {
  "copy": false,
  "create": true,
  "delete": true,
  "export": false,
  "import": false,
  "run": true,
  "update": true,
  "view": true
},
"tokenizeJob": {
  "copy": true,
  "create": true,
  "delete": true,
  "export": true,
  "import": true,
  "run": true,
  "update": true,
  "view": true
},
"reidentifyJob": {
  "copy": true,
  "create": true,
  "delete": true,
  "export": true,
  "import": true,
  "run": true,
  "update": true,
  "view": true
},
"scheduler": {
  "copy": false,
  "create": true,
  "delete": true,
  "export": false,
  "import": false,
  "run": false,
  "update": true,
  "view": true
},
"domain": {
  "copy": false,
  "create": true,
  "delete": true,
  "export": false,
```

```
"import": false,
"run": false,
"update": true,
"view": true
},
"algorithm": {
  "copy": false,
  "create": true,
  "delete": true,
  "export": false,
  "import": false,
  "run": false,
  "update": true,
  "view": true
},
"jdbcDriver": {
  "copy": false,
  "create": true,
  "delete": true,
  "export": false,
  "import": false,
  "run": false,
  "update": true,
  "view": true
},
"passwordVault": {
  "copy": false,
  "create": false,
  "delete": false,
  "export": false,
  "import": false,
  "run": false,
  "update": false,
  "view": false
},
"plugin": {
  "copy": false,
  "create": true,
  "delete": true,
  "export": false,
  "import": false,
  "run": false,
  "update": true,
  "view": true
},
"profileExpression": {
  "copy": false,
  "create": true,
  "delete": true,
  "export": false,
  "import": false,
  "run": false,
```

```

    "update": true,
    "view": true
  },
  "profileSet": {
    "copy": false,
    "create": true,
    "delete": true,
    "export": false,
    "import": false,
    "run": false,
    "update": true,
    "view": true
  },
  "fileFormat": {
    "copy": false,
    "create": true,
    "delete": true,
    "export": false,
    "import": false,
    "run": false,
    "update": false,
    "view": true
  },
  "user": {
    "copy": false,
    "create": true,
    "delete": true,
    "export": false,
    "import": false,
    "run": false,
    "update": true,
    "view": true
  }
}

```

4.7.3.2 Create new user

```

{
  "userName": "DelphixUser1",
  "password": "Password_123",
  "firstName": "First",
  "lastName": "Last",
  "email": "user@delphix.com",
  "isAdmin": false,
  "showWelcome": true,
  "userStatus": "ACTIVE",
  "nonAdminProperties": {
    "roleId": 1,
    "environmentIds": [

```

```

    1
  ]
}
}

```

4.7.3.3 User update

```

{
  "userName": "DelphixUser1",
  "password": "Password_123",
  "firstName": "First",
  "lastName": "Last",
  "email": "user@delphix.com",
  "isAdmin": false,
  "showWelcome": true,
  "userStatus": "ACTIVE",
  "nonAdminProperties": {
    "roleId": 1,
    "environmentIds": [
      1
    ]
  }
}

```

4.8 Best practices for defining masking roles

4.8.1 Introduction

The Delphix Masking Engine contains a role definition capability that enables admins to easily create roles for users. This section describes the typical roles and privileges that can be granted to users. It is recommended that the masking administrator implementing these roles consult IT Security and follow existing policies for data access. Roles are added by clicking the appropriate checkboxes within the add role function in the Settings tab. A sample RACI document and examples of roles / privileges are located below.

Roles for operating the Delphix Masking Engine are shared primarily between the masking administration team and the teams that support the applications that will be on-boarded to the Masking Engine. The admin will manage central functions of the engine including definition of custom domains, profiler expressions, algorithms, role and user definitions. The masking Engine is flexible enough to enable application teams with these functions as well, but it is recommended that these shared functions be managed by the admin team. The admin team should have an account registered with Delphix Support and be the main interface for issues and maintenance support from Delphix.

Masking processes can be developed for each application by the central admin team or the individual application teams, often determined by the volume of applications to be on-boarded. The RBAC model employed by Delphix Masking can support different implementation models. Your Delphix support team can assist in constructing roles to meet your needs.

Once roles are defined, they can be assigned to individual user IDs for the environments that those users have responsibility. Administrators will have access to all masking settings and environments by default.

- 1. Administrator access provides unlimited access to all functions and environments; this role should be granted to the central administration team.
- 2. All privileges is a default role (predefined) which will provide all functions for each environment a user is given access to.
- 3. Connector access should be controlled and administered by personnel responsible for database access.

4.8.2 Sample RACI

Teams: IT Security DM = Data masking admin team Application = App owner/SME DBA = Database admin QA = QA/Test environment owner PM = project management

Role	Description	Accountable	Responsible	Consulted	Informed
Security Policy	Determine data types that are sensitive for the enterprise.	IT Security	IT Security	DM, Application	DBA, QA
Program Management	Maintain program plan and implementation schedule, tracking and reporting.	PM	DM, Application	QA, IT Security	DBA
Inventory Management	Apply security policy to application schemas/ files.	Application	DM, Application	DBA, QA	IT Security
Data Masking	Build, maintain, schedule masking processes.	Application	DM, DBA	QA	IT Security
Masked Data Validation	Review and approve inventories and masked data.	Application	Application, DBA, QA	DM	IT Security

Role	Description	Accountable	Responsible	Consulted	Informed
Masked Data Deployment	Deploy masked data to required environments.	Application	Application, DBA, QA	DM, QA	IT Security
Environment Audit	Assure applications are compliant with masking.	IT Security	IT Security	DM, DBQ, QA	Application
Masking Administration	Manage masking tool central functions, create domains, profiler expressions, roles, users.	DM	DM	Application, IT Security, DBA	QA

4.8.3 Sample roles for Masking

Role	Description	<i>*Delphix Masking Functions</i>
Administrator	Manages masking server updates and upgrades; works with IT Security to update domains, algorithms and profiler expressions / sets.	Unrestricted access to all the engine functions. The Admin role is assigned via the checkbox in the add user page of the UI.
IT Security Analyst	Determines domains to be masked and high-level method for each domain and communicates them to administrator for inclusion in masking engine, responsible for masking audit functions.	Unrestricted access for all settings functions; access to all application functions except environment and environment create, delete, update.
Application Roles (per environment)		
All Privileges	Super user for an environment.	Unrestricted access for an application environment; central admin or security analyst will determine if this role can modify settings.

Role	Description	*Delphix Masking Functions
DBA	Manages user privileges, database performance and schema definition.	Manage connectors for application database, scripting and scheduling (no settings).
SME / Analyst / Developer	Application subject matter expert, application developer, data analyst, application architecture.	Manage inventories, create, view jobs.
Operations Roles (per environment)		
Operator	Schedule jobs, execute jobs, verify results, run automation scripts.	All job privileges.
Environment Owner	Determine workflow, monitor tool usage for environment.	Approve workflow and inventories, privileges to view for settings and environment.

4.9 Audit logs

Delphix helps you keep a record of user actions taken in the UI or directly through our REST APIs. You can access these audit logs directly from our UI or through our APIs.

4.9.1 Audit logs UI page

The Audit Logs can be found on the UI under the Audit tab. The user should be an admin to access this page.

Audit

 Export PDF


Timestamp ↓	User	Status	Activity	Audit Action	Audit Target
5 Jun 2024 10:49 IST	admin	SUCCEEDED	Viewed all User.	GET_ALL	USER
5 Jun 2024 10:49 IST	admin	SUCCEEDED	Viewed all Environment.	GET_ALL	ENVIRONMENT
5 Jun 2024 10:49 IST	admin	SUCCEEDED	Viewed all User.	GET_ALL	USER
5 Jun 2024 10:49 IST	admin	SUCCEEDED	Viewed all User.	GET_ALL	USER
5 Jun 2024 10:49 IST	admin	SUCCEEDED	Viewed all User.	GET_ALL	USER
5 Jun 2024 10:49 IST	admin	SUCCEEDED	Viewed all System Information.	GET_ALL	SYSTEM_INFORM...
5 Jun 2024 10:48 IST	admin	SUCCEEDED	Logged in user (id=7, userName=admin).	LOGIN	USER
4 Jun 2024 20:14 IST	admin	SUCCEEDED	Viewed all Mainframe Dataset Connector.	GET_ALL	MAINFRAME_DAT...
4 Jun 2024 20:14 IST	admin	SUCCEEDED	Viewed all File Connector.	GET_ALL	FILE_CONNECTOR

Displaying 1 to 9 of 776

This page contains information on what action occurred, the user that performed the action, and the time at which the action occurred.

The logs on the screen can be filtered or sorted by the various informational fields by clicking on the respective fields. More information on grid filtering and sorting can be found [here \(see page 232\)](#).

It also provides an option to download logs as PDFs. Users can select the **Export PDF** option from the top right corner and Logs will be downloaded as a PDF file.

 • All the columns are filterable and sortable.

• Filters and sort order applied to the grid will also be applied to Exported PDF file.

4.9.2 Audit log APIs

With 5.3.2.0, Delphix introduced an endpoint to get all Audit Logs. This endpoint contains the user name, action type, target, status, start time, and end time. For more information please refer to [API documentation. \(see page 878\)](#)

4.9.3 What gets logged?

User actions are categorized into the following:

Cancel	Create	Delete	Edit	Export	Get	Get All
--------	--------	--------	------	--------	-----	---------

Import	Lock	Login	Logout	Run	Test	Unlock
--------	------	-------	--------	-----	------	--------

The objects that user actions target are categorized into the following:

Algorithm	Analytics	Application	Application Log	Async Task	Audit Log	
Column Metadata	Database Connector	Ruleset Connector	Database Ruleset	Domain	Encryption Key	Environment
Execution	File Connector	File Download	File Field Metadata	File Format	File Metadata	File Ruleset
File Upload	LDAP	Mainframe Dataset Connector	Mainframe Dataset Field Metadata	Mainframe Dataset Format	Mainframe Dataset Metadata	Mainframe Dataset Ruleset
Masking Job	Profile Expression	Profile Job	Profile Set	Re Identification Job	Role	SSH Key
SSO	Syncable Object	System Information	Table Metadata	Tokenization	User	

4.9.4 Retention policy

The default policy stores the last one million Audit Log entries. Any entries older than the most recent million are removed daily. Additionally, there is a fail-safe mechanism that prevents an attacker from forcing an unbounded number of actions to be logged to overload the system's disk space. In the event that such an attack occurs, Delphix also logs it to the application logs.

4.9.5 Recommendation

If a full record of all Audit Log entries is desired, Delphix recommends using the new API to periodically retrieve new entries from the Audit Logs.

4.10 Monitor Async Task Status

This section describes how users can monitor the progress and Async Task status of an Import, Export or any other async actions performed.

4.10.1 Async task status

To check the async task status:

1. From the **Monitor** section, select the **Async Task Status** option.
2. A list of all Async operations performed appears in a grid, this grid is by default listed with the latest operation on the top.
3. You can filter or sort data by the various informational fields, by clicking on those respective fields. More information on grid filtering and sorting can be found in the [Graphical user interface](#)¹²³ page.
4. To find a particular Async Task Status for an Import/Export action performed, input the **Async Task Id** from the response of the corresponding Import/Export into the search field located in the table header under "Task ID."

The screenshot shows the 'Monitor > Async Tasks' page. The table below is a grid of Async Tasks. The 'Status' column uses green checkmarks for successful tasks and a red warning icon for failed tasks. The 'Actions' column contains a three-dot menu for each row. A tooltip is visible over the Actions menu for task 14, showing 'Download File' and 'Cancel Task' options.

Task ID	Type	Start Time ↓	End Time	Status	Actions
15	Algorithm Create	5 Jun 2024 11:22 IST	5 Jun 2024 11:22 IST	✓	⋮
14	Export	5 Jun 2024 11:22 IST	5 Jun 2024 11:22 IST	⚠	⋮
13	Export	5 Jun 2024 11:21 IST	5 Jun 2024 11:22 IST	✓	⋮
12	Algorithm Create	31 May 2024 11:30 IST	31 May 2024 11:30 IST	✓	⋮
11	Algorithm Create	30 May 2024 23:54 IST	30 May 2024 23:54 IST	✓	⋮
10	Algorithm Create	30 May 2024 20:38 IST	30 May 2024 20:38 IST	✓	⋮
9	Algorithm Create	28 May 2024 13:02 IST	28 May 2024 13:02 IST	✓	⋮
8	Algorithm Create	28 May 2024 12:52 IST	28 May 2024 12:52 IST	✓	⋮
7	Algorithm Create	28 May 2024 12:28 IST	28 May 2024 12:28 IST	✓	⋮
6	Algorithm Create	28 May 2024 12:04 IST	28 May 2024 12:04 IST	✓	⋮
5	Algorithm Create	28 May 2024 12:03 IST	28 May 2024 12:03 IST	✓	⋮
4	Algorithm Create	28 May 2024 12:03 IST	28 May 2024 12:03 IST	✓	⋮

Displaying 6 to 19 of 21

From the result grid, you can also download the exported file for the export operation by clicking the **Download File** option from the **Actions (...)** button on the corresponding row.

You can also download the log file for the failed import/export operations by clicking the **Download File** link on the corresponding row.

For the cancelable task to cancel the ongoing async task, click on the **Actions (...)** button on the corresponding row and choose **Cancel Task**.

¹²³ <https://masking.delphix.com/docs/latest/graphical-user-interface>

4.11 Kerberos configuration

4.11.1 Introduction

As of 5.3.0.0, the Continuous Compliance Engine supports Kerberos authentication for Oracle, MSSQL Server, and Sybase connections. Utilizing this service requires the presence of a Kerberos Key Distribution Center (KDC) server, as well as additional configuration actions, to be done on both the Continuous Compliance Engine and the database.

This page presents configuration instructions for enabling and using Kerberos on the Continuous Compliance Engine, as well as reference configurations for enabling Kerberos on the Databases. Although other configurations are possible, the configurations exemplified in this page have been validated by Delphix.



Kerberos is not supported for containerized masking deployments at this time.

4.11.2 Terminology

Throughout this page, the following example values are used. To recreate these reference environments, these values must be replaced with real values appropriate for your network environment:

- `.bar.com` – The DNS domain of the network
- `BAR.COM` – The Kerberos domain
- `me-host` – The hostname of the Continuous Compliance Engine
- `foo-kcd` – The hostname KDC server
- `krbuser` – The Kerberos principal to be granted access to the database for masking

4.11.3 Configuring Kerberos on the appliance

This section details the steps required to configure Kerberos on your appliance.

Launch the Delphix Server Setup UI and perform the following steps to enable Kerberos:

1. From the **Network Authorization** widget, click **Modify**.
2. Select the checkbox before **Use Kerberos authentication to communicate with remote hosts** field.

Network Authorization

[Modify](#)

Kerberos Configuration
Disabled

Host Connection Authentication
Disabled

3. Click the plus symbol to add record(s) for your KDCs, and populate other fields appropriately for your network environment. Upon pressing **Save**, your configuration will be tested. If the engine is able to authenticate to the KDC with the supplied configuration, the configuration is applied immediately.

Network Authorization ✕

KERBEROS CONFIGURATION

Use Kerberos authentication to communicate with remote hosts

Kerberos Key Distribution Center host(s)

+

Hostname [^]	Port
foo-kdc.bar.com	88

Realm

Principal

Keytab

_krbuser_keytab_base64_

HOST CONNECTION AUTHENTICATION

When connecting to hosts, you can provide username-password pairs when setting up the connection, or you can utilize one or more Enterprise Password Vault systems by adding them to your engine setup.

Click the + to add a vault

+

Vault name	Hostname	Port	Vault Type	Auth Method

Cancel
Save

4.11.4 Creating masking database connectors using Kerberos

Once the Continuous Compliance Engine is configured for Kerberos, creating Connectors using Kerberos authentication is simple:

- Select a source type that supports Kerberos.

Create Connection ✕

- Details
- Credentials
- Custom Properties
- Summary

Details

Specify details to identify this connector.

Connection Name

Select Source Type

Select Connection Method

Hostname/IP (Basic)
 JDBC URL (advanced)

Cancel Back Next Test Connection Save

1 Details step

- In the Credentials step, under the Select Authentication Type dropdown, select **Kerberos Authentication**.

Create Connection ✕

- Details
- Credentials
- Custom Properties
- Summary

Credentials

Specify the credentials for Oracle connector.

Schema Name

Host

Port

SID

Select Authentication Type

Principal Name

Password field can be left blank to use keytab

Password

Select Authentication Type

- Kerberos Authentication
- Username/Password
- Kerberos Authentication ✓
- Password Vault

Cancel Back Next Test Connection Save

The key tab is used if the same user principal configured in Server Setup is used, thus, it is not necessary to enter a password in the Connector definition.

For Sybase database Connectors, it is necessary to supply the service principal name as an additional configuration item. For Oracle DB, this value is determined automatically. For MSSQL Server it is determined based on the reverse DNS mapping of the Server Name (refer to the section on MSSQL Server below).

- If any changes are made to the underlying `krb5.conf` configuration file, these changes will not be recognized by the engine until after the next database connection attempt. Therefore, expect to have to click Test Connection twice after making any changes to the `krb5.conf` file. It does not matter if the first connection attempt succeeds or fails.

4.11.5 Reference database configurations

The following pages are reference Kerberos configuration procedures and troubleshooting notes for the supported databases. These are meant to serve as examples to be further customized according to the user's specific network environment and security needs.

4.11.6 Oracle and Kerberos

4.11.6.1 Overview

This page describes how to set up an Oracle database instance for Kerberized connections. The following steps are described:

- Creating a service principal and adding it to the database system
- Configuring the database to use Kerberos authentication
- Creating database users identified via Kerberos
- Troubleshooting tips

4.11.6.2 Prerequisites

This page assumes you already have a Kerberized network environment with an MIT Kerberos KDC. These procedures have been tested successfully with Oracle database versions 11.2.0.2, 11.2.0.4, and 12.2.1. Oracle database version 12.1.0.1 did not pass testing.

The following is needed from your Kerberos environment:

- The `krb5.conf` file
- A user principal and associated password or key tab to be used for logging into the Oracle database
- The ability to create a service principal for the Oracle database and retrieve the associated key tab

In addition, these example values are used:

- Oracle database host – `ora-db.bar.com`
- Oracle service name – `oracle`

4.11.6.3 Creating the Oracle service principal

A service principal is a unique identity for a service on a network, used primarily for Kerberos authentication. The format for naming the Oracle service principal combines the service name and the host, followed by the domain.

Given the example values provided where the service name is `oracle` and the host is `ora-db.bar.com`, the domain being `bar.com`, the service principal would aptly be named `oracle/ora-db.bar.com@bar.com`.

The hostname is whatever the database thinks its hostname is, so the output of `uname -n` on the database system rather than the actual DNS name. Typically, these values would be the same, but this is not always the case.

1. On the KDC, run the following:

```
# kadmin.local
kadmin.local: addprinc -randkey oracle/ora-db@bar.com
kadmin.local: ktadd -norandkey -k /var/tmp/ora-db.keytab oracle/ora-db@bar.com
```

2. Copy the resulting key tab file (`/var/tmp/ora-db.keytab`) to the Oracle database at this location: `/etc/v5srvtab`.
3. As a root user on the Oracle database, ensure the key tab has the correct permissions:

```
# chown root:oinstall /etc/v5srvtab
# chmod 440 /etc/v5srvtab
```

4. Finally, copy `/etc/krb5.conf` from the KDC to `/etc/krb5.conf` on the Oracle database. This file should be readable by all users.

4.11.6.4 Configuring the Oracle database for Kerberos

Log into the Oracle database system as the appropriate user for the database in question.

```
$ cd $ORACLE_HOME
$ vi network/admin/sqlnet.ora
```

- For **Oracle 11**, add the following:

```
SQLNET.KERBEROS5_CONF=/etc/krb5.conf SQLNET.AUTHENTICATION_SERVICES=(BEQ, KERBEROS5)
SQLNET.KERBEROS5_CONF_MIT=true SQLNET.AUTHENTICATION_KERBEROS5_SERVICE=oracle
```


- For **Oracle 12**, add the following:

```
NAMES.DIRECTORY_PATH=(TNSNAMES, EZCONNECT, HOSTNAME) SQLNET.KERBEROS5_CONF=/etc/
krb5.conf SQLNET.AUTHENTICATION_SERVICES=(BEQ,KERBEROS5PRE,KERBEROS5)
SQLNET.KERBEROS5_CONF_MIT=true SQLNET.AUTHENTICATION_KERBEROS5_SERVICE=oracle
```

i For **Oracle 11** databases (not needed on Oracle 12), use the `$ vi dbs/init.ora` command to open the `init.ora` file using `vi`.

Add this line at the end – `OS_AUTHENT_PREFIX=""`

- `OS_AUTHENT_PREFIX` is an Oracle initialization parameter that affects operating system authentication. Setting this parameter to an empty string (`""`) means that no prefix is required for Oracle to recognize operating system user names.

4.11.6.5 Creating a database User Identified via Kerberos

1. Log into the Oracle database system as the appropriate database user and open a database session as the DBA:

```
$ sqlplus / as sysdba
```

2. On **Oracle 12**, to alter your session to create the user in one of the PDBs:

```
SQL> alter session set container=MYPDB;
```

3. Create the user that will connect to the database using Kerberos:

```
SQL> create user krbdbuser identified externally as 'krbuser@BAR.COM';
```

4. Grant the user privileges necessary for masking.

The following example grants all privileges:

- **Oracle 11**

```
SQL> grant all privilege to krbdbuser;
```

- **Oracle 12** (Customize permissions as necessary for your environment)

```
SQL> grant connect,resource to krdbuser;  
SQL> grant create tablespace, drop tablespace to krdbuser;  
SQL> grant create table to krdbuser;  
SQL> grant create sequence to krdbuser;  
SQL> grant select_catalog_role to krdbuser;  
SQL> grant unlimited tablespace to krdbuser;  
SQL> grant select_catalog_role to krdbuser;  
SQL> grant alter system to krdbuser;  
SQL> grant sysoper to krdbuser;  
SQL> grant dba to krdbuser;
```

4.11.6.6 Troubleshooting

- Connecting via JDBC with Kerberos authentication from Continuous Compliance requires a Kerberos login, followed by JDBC connect. A failure stack with an error in the login function indicates a misconfiguration on either the engine or KDC – the engine has not attempted to communicate with the database at that point. Failure stacks are saved in the debugging log for masking.
- Login exceptions that mention a checksum error mean either the password or key tab supplied does not match the expected password/key on the KDC for the principal you are trying to use. Server Setup verifies that your key tab works at configuration time, but it could stop working if the key for your principal is updated on the KDC.
- Prior to Oracle 12, Oracle databases instances assume they can create/write a particular temporary file to store Kerberos credentials for the database. This means if you attempt to run multiple Kerberized instances of Oracle 11 on the same system or VM, and the databases run as different system users, the first Oracle instance that performs Kerberos authentication will create and own this file. Kerberos authentication will fail to function on all other instances.

4.11.7 MSSQL Server and Kerberos

4.11.7.1 Overview

This is an overview of the step necessary to get your Continuous Compliance Engine talking to an MS SQL Server database using Kerberos authentication. Since Active Directory already uses Kerberos for authentication, little or no additional configuration is need on the MSSQL Server database.

This page describes how to set up an MSSQL Server database instance for Kerberized connections. Since Active Directory already uses Kerberos for authentication, little or no additional configuration is needed on the MSSQL Server database. The following steps are described:

- Create the necessary SPNs (Service Principal Names) for your MSSQL database service in Active Directory
- Create the DB Connector on the Continuous Compliance Engine
- Creating a keytab for an Active Directory User
- Troubleshooting tips

4.11.7.2 Prerequisites

Configuring cross-realm trust between Active Directory and an MIT KDC Server is a complex topic, and will not be described here. In the absence of such a setup, it is possible to make the Delphix Appliance a Kerberos client of the Active Directory (AD) Server. In this configuration, no additional KDC is necessary. The example below assumes this kind of configuration.

This section of the document uses these example values in addition to or instead of those mentioned above:

- The MSSQL server database is named `mssql-db.bar.com`¹²⁴.
- The Active Directory user configured for masking access to the MSSQL database is `aduser` (rather than `krbuser` in other examples elsewhere in this document).
- The Active Directory user that starts the MS SQL Server service on the DB Server is `dbuser`.

4.11.7.3 Creating SPNs for the Database Service

MS SQL Server service will typically register several SPNs with Active Directory upon startup. However, there are several conditions which can cause these SPNs to not be registered successfully, or to be registered with service names other than those that are expected by the Microsoft JDBC Driver for SQL Server employed by Continuous Compliance.

The service principal name for an MS SQL Server expected by Continuous Compliance is: `MSSQLSvc/`

In addition, it is **required** that a reverse mapping exist in DNS from the IP address of the MS SQL Server system to the FQDN registered.

The following commands may be run in PowerShell on the MS SQL Server to assist in debugging SPN related issues:

List all SPNs for `dbuser`:

```
setspn -L -U dbuser
```

Deleting an old SPN associated with `dbuser`:

```
setspn -U -D MSSQLSvc/other-server.ad.bar.com:SQL2008R2 dbuser
```

Here's how to create the SPN describe above:

```
setspn -U -S MSSQLSvc/mssql-db.bar.com:1433 dbuser
```

4.11.7.4 Creating the Database Connector on the Continuous Compliance Engine

Once the above steps are complete, creating the database connector can be performed using the procedure above. Enter the username and optionally, password of the Active Directory user in the Connector definition. Be sure that the Active Directory user has sufficient access to the MS SQL Database for masking.

The password field can be left blank when creating the connector if the user is the same user configured in Server Setup for the appliance. Since keytabs are not typically used in an Active Directory environment, it may be useful to create one manually, to avoid having a password in the DB Connector.

¹²⁴ <http://mssql-db.bar.com>

4.11.7.5 Creating a keytab file for an Active Directory user

On a unix or MAC system with MIT Kerberos CLI utilities installed:

```
# ktutil

ktutil: addent -password -p krbuser -k 1 -e arcfour-hmac

<type password for krbuser>

ktutil: addent -password -p krbuser -k 1 -e aes128-cts-hmac-sha1-96

<type password for krbuser>

ktutil: addent -password -p krbuser -k 1 -e aes256-cts-hmac-sha1-96

<type password for krbuser>

ktutil: write_kt /var/tmp/krbuser.keytab

ktutil: exit

# base64 /var/tmp/krbuser.keytab ;# This is string to user for keytab in Server Setup
kerberos configuration
```



kvno doesn't matter when using Kerberos keytabs with Active Directory. The password must match the active password for the Active Directory user in question

4.11.7.6 Troubleshooting tips

The client uses the incorrect service name. This will typically manifest an exception mentioning cred, like:

```
Caused by: org.ietf.jgss.GSSEException: No valid credentials provided (Mechanism
level: Fail to create credential. (63) - No service creds)
at sun.security.jgss.krb5.Krb5Context.initSecContext(Krb5Context.java:770)
at sun.security.jgss.GSSContextImpl.initSecContext(GSSContextImpl.java:248)
at sun.security.jgss.GSSContextImpl.initSecContext(GSSContextImpl.java:179)
at
com.microsoft.sqlserver.jdbc.KerbAuthentication.intAuthHandShake(KerbAuthentication.j
ava:163) ... 101 common frames omitted
```

```
Caused by: sun.security.krb5.internal.KrbApErrException: Fail to create credential.
(63) - No service creds at
sun.security.krb5.internal.CredentialsUtil.acquireServiceCreds(CredentialsUtil.java:1
62)
at sun.security.krb5.Credentials.acquireServiceCreds(Credentials.java:458)
at sun.security.jgss.krb5.Krb5Context.initSecContext(Krb5Context.java:693) ... 104
common frames omitted
```

This could happen if using the JTDS JDBC driver and your MSSQL Server's IP address does not have a reverse mapping DNS. In that case, the driver could construct a service name like **MSSQLSvc/**, and try to use it.

Either correct the DNS to have a valid reverse mapping for the IP of your SQL server or manually add an SPN to the active directory for the name of the JDBC client being used. Determine the user that starts MSSQL Server on the database machine using the following command in PowerShell:

```
setspn -AU MSSQLSvc/ :1433
e.g. setspn -AU MSSQLSvc/10.43.100.101:1433 AD\dbuser
```

The database server has multiple DNS names (FQDNs). In this case, SPNs may be registered only for some of them. It may be necessary to add SPNs for the other FQDNs as above. The MS SQL Server did not automatically register an SPN. There is a limit (in the thousands) to the number of SPNs that may be registered for a given Active Directory user. It is quite possible to hit this limit in an environment where many MS SQL Server VMs are actively created and destroyed with the same configuration.



In Active Directory, setspn is not creating a service principal with distinct key as is typical for services on MIT KDCs - rather it is mapping the service principal to the key for the Active Directory user in question.

4.11.7.7 The SPN for the SQL Server is registered to the incorrect Active Directory account

Manifests as an exception with this text: GSS failure: Defective token detected (Mechanism level: AP_REP token id does not match!)

Resolution: From PowerShell on the MS SQL Server:

```
PS> setspn -Q <SPN>
```

This will show what the user has the SPN registered.

```
PS> setspn -U -D <SPN> <WRONG_ACCT>
```

This will unregister the SPN from that user

```
PS> setspn -AU <SPN> <CORRECT_ACCT>
```

4.11.8 Sybase and Kerberos

Creating a principal and corresponding keytab on the KDC

1. SSH into the KDC as the user with sufficient privileges to run `kadmin.local`
2. Run the Kerberos configuration CLI with `kadmin.local`
3. Add a new principal you want to authenticate as later with: `add_principal <>` We're going to continue to use **krbuser** as our example Kerberos principal.
4. Once you've created the principal and provided it a password, we need to generate a keytab for it. Do so via the following command:

```
ktadd -norandkey -k v5srvtab krbuser
```

In this case, `v5srvtab` is the keytab filename, and it will be placed into whatever directory you've invoked `kadmin.local` from. Presumably, this will be the home directory of the machine.

1. You now have everything you need done on the KDC, but you will need your keytab file later as well as the **krb5.conf** file that is located in the home directory of the KDC, so consider moving them somewhere (probably your local machine) that will be convenient for you to access later.

Configuring the Sybase image for Kerberos

1. Startup a Sybase database.
2. **Note:** Each Sybase database machine may have multiple Sybase instances running on it at a given point in time. In this case, I am configuring the `ASE_1550_S5` instance, but these steps can be done on any instance so long as you change the `$SYBASE_HOME` directories accordingly.
3. Connect to the particular Sybase instance you are working on and invoke the following sql statement: `sp_configure 'use security services', 1`
4. Continue to create a user with the same name as the principal name you created previously on the KDC, in this case **krbuser**: `sp_addlogin krbuser, <password>`
5. Change your `$SYBASE` environment variable to point to the Sybase directory for whichever instance you are configuring. In this case, we want to do: `export SYBASE=/opt/sybase/15-5`
6. Open the `$SYBASE/interfaces` file, and find the header for whichever Sybase instance you are configuring. In our case, it is **ASE_1550_S5**. You should see something that looks like this:


```
ASE1550_S5
```

```
`master tcp ether 10.43.89.241 5500`
`master tcp ether localhost 5500`
```

```
`query tcp ether 10.43.89.241 5500`
`query tcp ether localhost 5500`
```

You want to add the following line to this:

```
secmech 1.3.6.1.4.1.897.4.6.6
```

This line is **static**, while the other lines in **this** section are dynamically generated **for** your instance. So, your **final** result should look something like **this**:

```
ASE1550_S5
```

```
master tcp ether 10.43.89.241 5500 < your numbers will vary
master tcp ether localhost 5500 < your numbers will vary
query tcp ether 10.43.89.241 5500 < your numbers will vary
query tcp ether localhost 5500 < your numbers will vary
```

1. Navigate to **\$SYBASE/OCS-15_0/config**. You should see **libtcl64.cfg** and **libtcl.cfg**
2. Change the contents of **libtcl64.cfg** to be this:

```
`[DIRECTORY]`
`;ldap=libsybdldap.so ldap://ldaphost/dc=sybase,dc=com`
`[SECURITY]`
`csfkrb5=libsybskrb64.so secbase=@bar.com libgss=/lib64/libgssapi_krb5.so.2.2`
```

```
[FILTERS];ssl=libsybfssl.so`
```

2. Change the contents of **libtcl.cfg** to be this:

```
`[DIRECTORY]`
`;ldap=libsybdldap.so ldap://ldaphost/dc=sybase,dc=com`
`[SECURITY]`
`csfkrb5=libsybskrb64.so secbase=@bar.com libgss=/lib64/libgssapi_krb5.so.2.2`
`[FILTERS]`
`;ssl=libsybfssl.so`
```

3. **Note** that the **@bar.com** value is our realm name that is determined by the KDC. Realistically, you should never have to deal with this, and it should never change, but if for some reason it does, that value needs to be updated.
4. Create a directory for those Kerberos config files you created on the KDC in the previous set of steps:

```
sudo mkdir /krb
```

Copy into **/krb** your keytab file **v5srvtab** and config file **krb5.conf** that you took off of the KDC earlier.

1. Head to **\$SYBASE/ASE-15_0/install** and open the **RUN_ASE1550_S5** file. We're going to add information so that Sybase knows where to find our keytab and our krb5.conf file, so change the content to look like this:

```
#!/bin/sh

#

# ASE page size (KB) : 4096

# Master device path: /opt/sybase/devices/data5/S5_master.dat

# Error log path: /opt/sybase/errorlogs/ASE1550_S5.log

# Configuration file path: /opt/sybase/15-5/ASE-15_0/ASE1550_S5.cfg

# Directory for shared memory files: /opt/sybase/15-5/ASE-15_0

# Adaptive Server name: ASE1550_S5

#

export **KRB5_KTNAME**=/krb/v5srvtab

export **KRB5_CONFIG**=/krb/krb5.conf

/opt/sybase/15-5/ASE-15_0/bin/dataserver \

-kASE1550_S5@bar.com \

-d/opt/sybase/devices/data5/S5_master.dat \

-e/opt/sybase/errorlogs/ASE1550_S5.log \

-c/opt/sybase/15-5/ASE-15_0/ASE1550_S5.cfg \

-M/opt/sybase/15-5/ASE-15_0 \

-sASE1550_S5 \
```

1. Reboot the Sybase instance you're working so that it reads in all of these configuration changes.
2. Connect to the Sybase instance as the **dbo** user so that you may give dbo privileges to your Kerberos authentication login on a particular database within the instance. Below is an example of doing so with the database **potatoes**:

```
>> sql5

1> use potatoes
```



```

2> go
1> sp_addalias instructions, dbo
2> go
Alias user added.
(return status = 0)

```

1. Now, to access the Sybase instance via Kerberos and confirm success, you can do the following set of commands (I put these three lines into a script called **connect.sh** for future convenience):

```

#!/bin/sh
kinit -k -t /krb/v5srvtab <>
export SYBASE='/opt/sybase/15-5'
/opt/sybase/15-5/OCS-15_0/bin/isql64 -V -SASE1550_S5

```

Testing by creating a Kerberos connector on the Delphix Engine

1. Start by configuring your engine for Kerberos. SSH into the engine as the Delphix user and run the following command: `/opt/delphix/server/bin/jmxtool tunable set enabled_features KERBEROS true`
2. Log into the Delphix Engine and proceed through the first-time setup.
3. Once the first-time setup is complete, log into the Delphix Setup page, proceed to Preferences > Kerberos Configuration. Add the information for your KDC to configure it with the principal name you created earlier, **krbuser**. You can get the keytab by running the following command on your keytab file: `base64 v5srvtab`

Copy the output as plaintext into the keytab field of the Kerberos configuration box.

Finally, create a Sybase connector with parameters that look like this, and if your “test connection” attempt succeeds you’re all set!

Create Connection



- Details
- Credentials
- Custom Properties
- Summary

Details

Specify details to identify this connector.

Connection Name
Connector_test

Select Source Type
Database - Sybase

Select Connection Method
 Hostname/IP (Basic) JDBC URL (advanced)

Cancel Back Next Test Connection Save

Create Connection



- Details
- Credentials
- Custom Properties
- Summary

Credentials

Specify the credentials for Sybase connector.

Schema Name
dbos

Database Name
potatoes

Host
sybaseHostname.bar.com

Port
4000

Select Authentication Type
Kerberos Authentication

Principal Name
krbuser


Service Principal
ASE1550_S5

Cancel Back Next Test Connection Save

4.12 Password vault configuration

4.12.1 Introduction

The Continuous Compliance Engine supports the use of HashiCorp, CyberArk and Google Cloud Platform (GCP) password vaults for connections to Db2, MariaDB, MSSQL, MySQL, Oracle, PostgreSQL, YugabyteDB, CockroachDB, and Sybase databases. Utilizing this feature requires the presence of either a HashiCorp, CyberArk, or GCP vault/secret manager, as well as additional configuration actions on the Continuous Compliance Engine.

 Password vault authentication is not supported for containerized masking deployments at this time.

4.12.2 Configuring a password vault on the appliance

Before attempting to access a password vault, the CA certificate for the vault must first be added to the Compliance Engine's trust store. Certificates can be managed through the Delphix Server Setup UI and the steps for doing so can be found on the [TrustStore settings](#)¹²⁵ page.

Currently, password vaults and the associated credential paths can only be configured on the appliance using the API. The Continuous Compliance Engine's web API includes two endpoints, `password-vaults` and `credential-paths` for managing the setup of vaults and credentials.

4.12.2.1 Setting up a password vault

The POST action on the `password-vaults` endpoint is used to provide information on the type of vault to be accessed and the location of the server hosting the vault.

4.12.2.1.1 HashiCorp

For a HashiCorp vault, the body of the request will be similar to:

```
{
  "name": "HashiVault",
  "vaultType": "HASHICORP",
  "configJson": {
    "host": "123.45.67.89",
    "port": 8200,
  }
}
```

¹²⁵ <https://cd.delphix.com/docs/latest/truststore-settings>

```

    "namespace": "sample/child",
    "authType": "TOKEN",
    "token": "hvs.kvITvwsI4gs"
  },
  "description": "Vault description is optional"
}

```

i Namespaces are only relevant when using the Enterprise version of the HashiCorp product. If this field is specified, it should match the namespace being used on the HashiCorp server.

To use either AppRole or Certificate based authentication, the following substitutions can be made to the above example:

```

"authType": "APPROLE",
  "roleId": "your-role-id",
  "secretId": "your-secret"

```

or

```

"authType": "CERTIFICATE",
  "certificate": "-----BEGIN CERTIFICATE-----\nMIa1ZqA=\n-----END CERTIFICATE-----",
  "privateKey": "-----BEGIN RSA PRIVATE KEY-----\nUw9aPq\n-----END RSA PRIVATE KEY-----",
  "roleName": "sampleRole"

```

4.12.2.1.2 CyberArk

For CyberArk, the request body will be similar to:

```

{
  "name": "CyberVault",
  "vaultType": "CYBERARK",
  "configJson": {
    "host": "cyberark01.myserver.com",
    "port": 443,
    "appId": "MyApp",
    "authType": "CERTIFICATE",
    "certificate": "-----BEGIN CERTIFICATE-----\nMIa1ZqA=\n-----END CERTIFICATE-----"
    "privateKey": "-----BEGIN PRIVATE KEY-----\nMIa1ZqA=\n-----END PRIVATE KEY-----"
  },
  "description": "Vault description is optional"
}

```

4.12.2.1.3 Google Cloud Platform

For a GCP password vault, referred to as a Secret Manager the request will be similar to:

```
{
  "name": "GCPVault",
  "vaultType": "GCP",
  "configJson": {
    "authType": "SERVICE_ACCOUNT",
    "serviceAccountIdJson": {...}
  },
  "description": "Vault description is optional"
}
```

The only authType for GCP is `SERVICE_ACCOUNT`.

The `serviceAccountIdJson` value in its entirety is retrieved from the **Service Account** section of **IAM & Admin** in the google account. This is done by creating and downloading a JSON key for the service account. Permissions to create this key must be enabled. The key contains multiple fields and handles all properties of the connection to the secret manager. Google documents the process [here](#)¹²⁶. An example of the JSON looks like this:

```
{
  "type": "service_account",
  "project_id": "test-project",
  "private_key_id": "test-key-id",
  "private_key": "test-key",
  "client_email": "test-email",
  "client_id": "test-client-id",
  "auth_uri": "test-auth-uri",
  "token_uri": "test-token-uri",
  "auth_provider_x509_cert_url": "test-cert-url",
  "client_x509_cert_url": "test-client-cert-url"
}
```

i The Google documentation recommends setting up Workload Identity Federation instead of service account ID for some applications. This approach is not supported for masking engine password vaults.

¹²⁶ <https://cloud.google.com/iam/docs/keys-create-delete>

4.12.2.2 Setting up a credential path

Credential paths are used to specify the location of the credentials within a password vault.

4.12.2.2.1 HashiCorp

The Continuous Compliance Engine currently supports two types of HashiCorp secrets engines: `database` and `key-value-v2`.

The request body for a HashiCorp credential path will be similar to:

```
{
  "credentialPathName": "HashiCredentialPath",
  "description": "Credential path description is optional",
  "passwordVaultId": 1,
  "credentialParameters": {
    "engineType": "KEY_VALUE_V2",
    "engine": "secret-engine-name",
    "path": "secret-path",
    "usernameKey": "username",
    "passwordKey": "password"
  }
}
```

Database secrets engines support dynamic secrets by generating database credentials based on configured roles. When using a database secrets engine, set `engineType` to **DATABASE** and use `role` to specify the name of the role to create credentials against.

```
"credentialParameters": {
  "engineType": "DATABASE",
  "engine": "database-engine-name",
  "role": "my-role",
  "usernameKey": "username",
  "passwordKey": "password"
}
```

4.12.2.2.2 CyberArk

The request body for a CyberArk credential path will be similar to:

```
{
  "credentialPathName": "CyberCredentialPath",
  "description": "Credential path description is optional",
  "passwordVaultId": 1,
```

```
"credentialParameters": {
  "queryString": "Safe=DevTest;Folder=Root;Object=postgres01"
}
```

4.12.2.2.3 Google Cloud Platform

The request body for a GCP credential path will be similar to:

```
{
  "credentialPathName": "GCPCredentialPath",
  "description": "Credential path description is optional",
  "passwordVaultId": 1,
  "credentialParameters": {
    "versionId": "latest",
    "projectId": "example-project",
    "usernameKey": "username",
    "passwordKey": "password"
  }
}
```

Every time a GCP credential value is changed in Secret Manager it will be versioned, in most cases "latest" will be desired and is specified as a string like above. This is the most recent version of the secret. An integer for a different version can be substituted.

The value for parameters "usernameKey" and "passwordKey" specify the names of the secrets saved in Google Secret Manager used to provide a username and password for database connection. A secret for each is required. The names of the secrets are unimportant as long as they correspond to the required login information. As an example, if a secret called "my_database_username" is created in GCP with the appropriate value, then "my_database_username" would be the value of the "usernameKey" parameter in the credential path.

The project ID is a numeric value, not the name of the GCP project. The numeric value can be found in the path for each secret in the project. If one of the secrets is selected, the path appears in the UI, and will be formatted like this:

```
projects/999999999999/secrets/my_database_username
```

The numeric value following /projects is the project ID.

4.12.3 Configuring the database connector

Database connectors can be configured to use a password vault through either the Continuous Compliance Engine UI or the APIs.

4.12.4 UI configuration

When creating or editing a Db2, MariaDB, MSSQL, MySQL, Oracle, PostgreSQL, YugabyteDB, CockroachDB, or Sybase database connector, Select 'Password Vault' option from the **Authentication Type** dropdown and then select the required credential path from the **Credential Path** dropdown. If the "Test Connection" run succeeds then it is complete.

4.12.5 API configuration

`CredentialPathId` is an optional field when creating a DB2, MariaDB, MSSQL, MySQL, Oracle, PostgreSQL, YugabyteDB, CockroachDB, or Sybase database connector via the API. Setting this value to the id of an existing credential path object will result in the connector using password vaults to retrieve the credential. As an example:

```
{
  "connectorName": "psql-connector",
  "databaseType": "POSTGRES",
  "environmentId": 1,
  "host": "mpv-psql.mydb.co",
  "port": 5432,
  "databaseName": "postgres",
  "schemaName": "public",
  "credentialPathId": 1
}
```

4.13 Db2 connector license installation





At this time, Containerized Masking only supports the Db2 LUW connector. The Db2 Mainframe and Db2 iSeries connectors cannot be enabled for Containerized Masking.

If you have been licensed to use the Continuous Compliance Db2 Connector for Mainframe or Db2 Connector for iSeries, you will need to obtain the respective Db2 Connector package (tar file) and apply it to your Masking Engine(s). Each package is intended to be installed and run from a workstation or laptop, not from the Delphix Appliance. These packages contain a script that must be used in a bash shell and depends on the availability of the `curl` and `ssh` commands to install the respective license on your remote Delphix Appliance.

4.13.1 Applying Db2 connector for mainframe

1. Go to the [Home > Delphix Product Releases > Continuous Compliance Connectors > IBM Db2 for z/OS¹²⁷](#) folder on the Delphix Download site.
2. Select the folder with the most recent release number that is less than or equal to your engine version.
3. Download the `DB2MaskingMainframe.tar` file,
4. Extract its contents using `tar -xvf DB2MaskingMainframe.tar`.
5. Run `cd db2-license`.
6. Run `./installdb2license.sh -h your-engine-hostname`. You will be prompted for a username and password.

 To view a help message, run `./installdb2license.sh` with no arguments.

 To use TLS/SSL, specify the `-s` flag, your secure port (`-P` option) and trusted server certificate (`-C` option).

The script will enable the Db2 Mainframe connector, shutdown the Masking Engine, and then start the Masking Engine. To stop and start the Masking Engine, the script will prompt the user for the Delphix **sysadmin** password. After this process completes, "Database - MAINFRAME DB2" will appear in the Connector drop-down of the Masking Engine UI and can be used in the same way as other database connectors to profile and mask rule sets.

4.13.2 Applying Db2 connector for iSeries


1. Go to the [Home > Delphix Product Releases > Continuous Compliance Connectors > IBM Db2 for iSeries¹²⁸](#) folder on the Delphix Download site.
2. Select the folder with the most recent release number that is less than or equal to your engine version.
3. Download the `DB2MaskingiSeries.tar` file.

¹²⁷ <https://download.delphix.com/folder/5225/Delphix%20Product%20Releases/Continuous%20Compliance%20Connectors/IBM%20Db2%20for%20z/OS>

¹²⁸ <https://download.delphix.com/folder/5219/Delphix%20Product%20Releases/Continuous%20Compliance%20Connectors/IBM%20Db2%20for%20iSeries>

4. Extract its contents using `tar -xvf DB2MaskingISeries.tar`.
5. Run `cd db2-license`.
6. Run `./installdb2license.sh -h your-engine-hostname`. You will be prompted for a username and password.

 To view a help message, run `./installdb2license.sh` with no arguments.
















 To use TLS/SSL, specify the `-s` flag, your secure port (`-P` option) and trusted server certificate (`-C` option).















The script will enable the Db2 iSeries connector, shutdown the Masking Engine, and then start the Masking Engine. To stop and start the Masking Engine, the script will prompt the user for the Delphix **sysadmin** password. After this process completes, "Database - iSeries DB2" will appear in the Connector drop-down of the Masking Engine UI and can be used in the same way as other database connectors to profile and mask rule sets.




4.14 Continuous Compliance Engine icon reference

This topic illustrates the icons that appear on the Continuous Compliance Engine graphic user interface and describes the meaning of each.

Icon	Description
	Add or Create
	Export
	Import
	Delete
	Run

Icon	Description
	Cancel
	Job Success
	Job Created
	Error or Failed
	Queued
	Running
	Skipped
	Warning
	Close
	Download
	Test Connection
	Database
	File
	Refresh
	Ruleset Refresh Success

Icon	Description
N/A	Ruleset refresh is not applicable for file rulesets
	Ruleset Refresh in Progress
	Ruleset Refresh Failed
	Ruleset Refresh Waiting
	Ruleset Refresh Cancel
	View
	Edit
	Edit
	Information
	Duplicate
	Edit File Format
	View Redefine Condition
	Edit Redefine Condition
	Download File
	Manage Mappings

Icon	Description
	Update Custom SQL for table in Database Inventory
	Update Filter Condition for table in Database Inventory
	Update Logical Key for table in Database Inventory

4.15 Delphix masking terminology

Before getting started with the Continuous Compliance Engine, an overview of universal terms and concepts will build and unify how different masking components come together. The following provides a brief overview of the key concepts within the masking service.

4.15.1 Products

Term	Definition
Delphix Continuous Data	Delphix Continuous Data is a Delphix product to deliver data on-demand to application developers and testers. Running as a virtual appliance, it is sometimes referred to as a Data Engine.
Delphix Continuous Data with Elastic Data	Elastic Data is a storage feature of Delphix Continuous Data that allows optimal data management to minimize costs through the use of block and elastic storage.
Delphix Continuous Compliance	Delphix Continuous Compliance is a Delphix product for discovering sensitive data and replacing it with realistic but fictitious data. Running as a virtual appliance, it is sometimes referred to as a Compliance Engine.
Data Control Tower	Data Control Tower (DCT) is a Delphix product that provides a data mesh to unify data governance, automation, and compliance across all applications and cloud platforms.

4.15.2 Interfaces

Term	Definition
Delphix Setup	Delphix Setup is the Delphix Continuous Data's user interface for system administrators to configure their engine settings, such as storage, support bundle, authentication, and network configurations.
Delphix Management	Delphix Management is Delphix Continuous Data's user interface for product administrators to manage their virtualized or masked datasets.
Delphix Self-Service	Delphix Self-Service is the Delphix Continuous Data user interface designed specifically for project teams, application developers, and testers to manage their virtualized datasets.
Command Line Interface (CLI)	Command Line Interface (CLI) is the engine's terminal interface which allows users to perform various administrative commands. This is not to be confused with the <code>dxtoolkit</code> .
Delphix Download Portal	Delphix Download Portal ¹²⁹ is the location where users download Delphix's products.
Delphix Support Portal	Delphix Support Portal ¹³⁰ is the location where users receive support for Delphix's products.

4.15.3 Core concepts

Term	Definition
Virtualization	Virtualization describes the capability of producing a functioning database or filesystem copy that is lightweight and ephemeral.
Masking	Masking describes the capability of iterating through a dataset to identify all sensitive fields and replace them with desensitized values to eliminate risk in lower environments.

¹²⁹ <https://download.delphix.com/>

¹³⁰ <https://support.delphix.com/>

Term	Definition
Data Source	Data Source is a database or unstructured files located in a user's environment. It generally describes ingestion sources and is typically located in a Source or Staging Environment.
Dataset	Dataset is an instance of any collection of data, such as VDB, dSource, vFiles, data source, or database. The dataset may or may not be managed by Delphix Continuous Data.

4.15.4 Data Source Connectors

Term	Definitions
Connector	Connector refers to the Delphix Continuous Data's data source connection mechanism. A connector enables Delphix Continuous Data functionality with a specific data source system or DBMS. See other connector types for specific details.
Standard Connector	Standard Connector are connectors that are built and supported by Delphix. They are included for free with a Delphix Continuous Data License Agreement.
Select Connector	Select Connector are connectors that are built and supported by Delphix but require a separate License Agreement.
Premium Connector	Premium Connector are connectors that are built and supported by a third party. They often require a separate License Agreement.
Plugin	Plugin is the software delivery framework for many Connectors. The user must upload the plugin into Delphix Continuous Data to install the associated Connector.

4.15.5 Continuous Compliance

Term	Definition
Application	An Application is a tag that is assigned to one or more environments. We recommend using an application name that is the same as the application associated with the environments.
Connector	Connector refers to the Delphix Continuous Data's data source connection mechanism. A connector enables Delphix Continuous Data functionality with a specific data source system or DBMS. See other connector types for specific details.
Domain	A domain represents a correlation between various sensitive data categories (social security numbers) and the way it should be secured.
Environment	An environment is a construct that can be used to describe a collection of masking jobs associated with a group of data sources.
dSource	dSource is the copy of a source database's persistent data layer that Delphix Continuous Data uses to create and update virtual databases (VDBs). Based on the ingestion model and data source type, the dSource could be exposed through a mount point and interacts with a Staging Database Instance, Database, or Files. Consult the data source connector documentation for specific details.
Virtual Database (VDB)	Virtual Database (VDB) is a full read-and-write copy of the source data that is provisioned from either a dSource or another VDB. A VDB is provisioned and managed by Delphix Continuous Data.
Timeflow	Timeflow describes the timeline of data of a virtual database or dSource.
Snapshot	Snapshot represents the state of a dataset at a specific moment in time. They are used to create or refresh the same or another timeline.
Hooks	Hooks are mechanisms that allow the execution of custom operations at specific points in various processes like linking, provisioning, and managing virtual datasets.
Delphix Connector	Delphix Connector is a service that runs on the Windows Staging and Target Environments to enable communication with Delphix Continuous Data. The Delphix Connector should not be confused with Data Source Connectors.

Term	Definition
Ruleset	A rule set is group of tables or flat files within a particular data source that a user may choose to run profile, masking, or tokenization jobs on.
Source Environment	Source Environment is the "production" or "golden" environment that contains the ideal database the user would like to virtualize in Delphix Continuous Data.
Target Environment	Target Environment is the configured infrastructure, with available database binaries, in which virtual database copies will be hosted.
Host	Host is a single server within the environment collection. An environment is considered to be one or more hosts. For example, a RAC environment contains multiple hosts.

4.15.6 Masking algorithms

The following terminology is around the different Algorithms that users may use to secure their data.

Term	Definition
Algorithm Framework	A type of masking algorithm. One or more usable instances of an <i>algorithm framework</i> may be created. For example, "FIRST NAME SL" is an instance of the Secure Lookup <i>algorithm framework</i> .
Algorithm Instance	A named combination of algorithm framework and configuration values. <i>Algorithm instances</i> are applied to data fields and columns in the inventory in order to mask data.
Built-in Algorithm	An algorithm instance or framework included with the Masking Engine software. This includes several built-in algorithm instances that provide masking behavior that doesn't correspond to any built-in algorithm framework.
Non-conformant Data	Some masking algorithms require data to be in a particular format. The required format may vary by the configuration of the algorithm instance. For example, a particular Segment Mapping algorithm might be configured to expect a 10 digit number. Data which doesn't fit the pattern expected by an algorithm is called <i>nonconforming data</i> or <i>non-conformant data</i> . By default, <i>non-conformant data</i> is not masked, and warnings are recorded for the masking job. Warnings are indicated by a yellow triangle warning marker next to the job execution in Environment and Job Monitor pages. Whether <i>non-conformant data</i> results in a warning or failure is configurable for each algorithm instance.

Term	Definition
Collision	<p>The term <i>collision</i> describes the case where a masking algorithm masks two or more unique input values to the same output value. For example, a first name Secure Lookup algorithm might mask both "Amy" and "Jane" to the same masked value "Beth". This may be desirable, in the sense that it further obfuscates the original data, however <i>collisions</i> are problematic for data columns with uniqueness constraints.</p>
Secure Lookup	<p>The most commonly used algorithm framework. Secure lookup works by replacing each data value with a new value chosen from an input file. Replacement values are chosen based on a cryptographic hash of the original value, so masking output is consistent for each input. Secure lookup algorithms are easy to configure and work with different languages.</p> <p>When this algorithm replaces real data with fictional data, <i>collisions</i>, described above, are possible. Because many types of data, such as first or last name, address, etc, are not unique in real data, this is often acceptable. However, if unique masking output for each unique input is required, consider using a mapping or segment mapping algorithm, described below.</p>
Segment Mapping	<p>This algorithm permutes short numeric or alpha-numeric values to other values of the same format. This algorithm is guaranteed to not produce collisions, so long as the set of permissible mask values is at least as large as input or "real" set. The maximum number of digits or characters in the masked value is 36. You might use this method if you need columns with unique values, such as Social Security Numbers, primary key columns, or foreign key columns.</p>
Mapping	<p>Similar to secure lookup, a mapping algorithm allows you to provide a set of values that will replace the original data. There will be no collisions in the masked data, because each input is always matched to the same output, and each output value is only assigned to one input value. In order to accomplish this, the algorithm records, in an encrypted format, all known input to output mappings.</p> <p>You can use a mapping algorithm on any set of values, of any length, but you must know how many values you plan to mask, and provide a set of unique replacement values sufficient to replace each unique input value.</p> <p>NOTE: When you use a mapping algorithm, you cannot mask more than one table at a time. You must mask tables serially.</p>
Binary Lookup	<p>Replaces objects that appear in object columns. For example, if a bank has an object column that stores images of checks, you can use binary lookup algorithm to mask those images. The Delphix Engine cannot change data within images themselves, such as the name on X-rays or driver's licenses. However, you can replace all such images with a new, fictional image. This fictional image is provided by the owner of the original data.</p>

Term	Definition
Tokenization	<p>The only type of algorithm that allows you to reverse its masking. For example, you can use a tokenization algorithm to mask data before you send it to an external vendor for analysis. The vendor can then identify accounts that need attention without having any access to the original, sensitive data. Once you have the vendor's feedback, you can reverse the masking and take action on the appropriate accounts.</p> <p>Like mapping, a tokenization algorithm creates a unique token for each input such as "David" or "Melissa." The Delphix Engine stores both the token and original so that you can reverse masking later.</p>
Min Max	<p>Values that are extremely high or low in certain categories allow viewers to infer someone's identity, even if their name has been masked. For example, a salary of \$1 suggests a company's CEO, and some age ranges suggest higher insurance risk. You can use a min max algorithm to move all values of this kind into the midrange.</p>
Data Cleaning	<p>Does not perform any masking. Instead, it standardizes varied spellings, misspellings, and abbreviation for the same name. For example, "Ariz," "Az," and "Arizona" can all be cleaned to "AZ."</p>
Free Text Redaction	<p>Helps you remove sensitive data that appears in free-text columns such as "Notes." This type of algorithm requires some expertise to use, because you must set it to recognize sensitive data within a block of text.</p> <p>One challenge is that individual words might not be sensitive on their own, but together they may be. This algorithm uses profiler sets to determine which information it needs to mask. You can decide which expressions the algorithm uses to search for material such as addresses. For example, you can set the algorithm to look for "St," "Cir," "Blvd," and other words that suggest an address. You can also use pattern matching to identify potential sensitive information. For example, a number that takes the form 123-45-6789 is likely to be a Social Security Number.</p> <p>You can use free text redaction algorithm to show or hide information by displaying either a "deny list" or an "allow list."</p>

4.15.7 Profile job concepts

The following set of concepts are options available to the user for configuring a profiling job.

Term	Definition
Job Name	A free-form name for the job you are creating. Must be unique.

Term	Definition
Multi-Tenant	Check the box if the job is for a multi-tenant database. This option allows existing rulesets to be reused to mask identical schemas via different connectors. The connector can be selected at job execution time.
Rule Set	Select a ruleset that this job will execute against.
No. of Streams	The number of parallel streams to use when running the jobs. For example, you can select two streams to run two tables in the ruleset concurrently in the job instead of one table at a time.
Min Memory (MB) <i>optional</i>	Minimum amount of memory to allocate for the job, in megabytes.
Max Memory (MB) <i>optional</i>	Maximum amount of memory to allocate for the job, in megabytes.
Feedback Size <i>optional</i>	The number of rows to process before writing a message to the log. Set this parameter to the appropriate level of detail required for monitoring your job. For example, if you set this number significantly higher than the actual number of rows in a job, the progress for that job will only show 0 or 100%
Profile Sets <i>optional</i>	The name of a profile set, which is a subset of expressions (for example, a subset of financial expressions).
Comments <i>optional</i>	Add comments related to this job.
Email <i>optional</i>	Add email address(es) to which to send status messages. Separate addresses with a comma (,).

4.15.8 Masking job concepts


These concepts are options available to the user for configuring a masking job.

Term	Definition
Job Name	A free-form name for the job you are creating. Must be unique across the entire application.
Masking Method	Select either In-Place or On-The-Fly.
Multi-Tenant	Check the box if the job is for a multi-tenant database. This option allows existing rulesets to be reused to mask identical schemas via different connectors. The connector can be selected at job execution time.
Rule Set	Select a ruleset for this job to execute against.
Masking Method	Select either In-place or On-the-fly.
Min Memory (MB) optional	Minimum amount of memory to allocate for the job, in megabytes.
Max Memory (MB) optional	Maximum amount of memory to allocate for the job, in megabytes.
Update Threads	The number of update threads to run in parallel to update the target database. For database using T-SQL, multiple update/insert threads can cause deadlock. If you see this type of error, reduce the number of threads that you specify in this box.
Commit Size	The number of rows to process before issuing a commit to the database.
Feedback Size	The number of rows to process before writing a message to the logs. Set this parameter to the appropriate level of detail required for monitoring your job. For example, if you set this number significantly higher than the actual number of rows in a job, the progress that job will show 0% or 100%.
Disable Trigger <i>optional</i>	Whether to automatically disable database triggers. The default is for this check box to be clear and therefore not perform automatic disabling of triggers.
Drop Index <i>optional</i>	Whether to automatically drop indexes on columns which are being masked and automatically re-create the index when the masking job is completed. The default is for this check box to be clear and therefore not perform automatic dropping of indexes.

Term	Definition
Prescript <i>optional</i>	Specify the full pathname of a file that contains SQL statements to run before the job starts, or click Browse to specify a file. If you are editing the job and a pre script file is already specified, you can click the Delete button to remove the file. (The Delete button only appears if a prescript file was already specified.)
Postscript <i>optional</i>	Specify the full pathname of a file that contains SQL statements to be run after the job finishes, or click Browse to specify a file. If you are editing the job and a postscript file is already specified, you can click the Delete button to remove the file. (The Delete button only appears if a postscript file was already specified.)
Comments <i>optional</i>	Add comments related to this masking job.
Email <i>optional</i>	Add email address(es) to which to send status messages.

4.16 Changing the IP address of the Delphix Engine

You can change the IP address of the Delphix Engine either from the User Interface or using the Command-Line Interface.

-  For Containerized Masking, networking is handled via the underlying kubernetes infrastructure. There is no interface through the application to change the IP address. Changing the IP address of a containerized instance requires those changes to happen in kubernetes and its underlying nameservice. Frequently it is managed by the network proxy that directs traffic to the containerized instance.

4.16.1 Pre-requisites

- Ensure that no masking jobs are running.

4.16.2 Changing the IP address from the user interface

Perform the following procedure to change the IP address of the Delphix Engine from the UI.

1. Launch the Delphix Setup application.

2. Go to **System > Server Setup** in the Delphix Management interface, or click **Server Setup** in the Delphix Engine login screen.
3. In the **Network** panel, click **Modify**.
4. Under **DNS Services**, enter the new IP address.
5. Click **Ok**.
6. Refresh all environments by clicking the **Refresh** option on the Environments screen.

4.16.3 Changing the IP address using CLI

Perform the following procedure to change the IP address of the Delphix Engine using CLI.

1. Log into the Delphix CLI using your sysadmin account.

```
delphix> network
delphix network> setup
delphix network interface> list
NAME
vmxnet3s0
delphix network interface> select vmxnet3s0
delphix network interface 'vmxnet3s0'> get
  type: NetworkInterface
  name: vmxnet3s0
  addresses:
    0:
      type: InterfaceAddress
      address: 10.1.2.3/24
      addressType: STATIC
      enableSSH: true
      state: OK
  dataNode: DATA_NODE-34
  device: vmxnet3s0
  macAddress: 0:c:29:32:96:a3
  mtu: 1500
  mtuRange: 60-9000
  reference: NETWORK_INTERFACE-vmxnet3s0-DATA_NODE-34
  state: OK
```

2. Run the update command and update the address to the new IP address for the Delphix Engine.

```
delphix network interface 'vmxnet3s0'> update
delphix network interface 'vmxnet3s0' update *> edit addresses.0
delphix network interface 'vmxnet3s0' update addresses.0 *> get
Properties
  type: InterfaceAddress
  address: 172.16.151.154/24
  addressType: STATIC
  enableSSH: true
```

```
delphix network interface 'vmxnet3s0' update addresses.0 *> set address=10.1.2.4/24
delphix network interface 'vmxnet3s0' update addresses.0 *> get
  type: InterfaceAddress (*)
  address: 10.1.2.4/24 (*)
  addressType: STATIC (*)
  enableSSH: true (*)
```

3. Commit the operation.

```
delphix network interface 'vmxnet3s0' update addresses.0 *> commit
delphix network interface 'vmxnet3s0'> get
  type: NetworkInterface
  name: vmxnet3s0
  addresses:
    0:
      type: InterfaceAddress
      address: 10.1.2.4/24
      addressType: STATIC
      enableSSH: true
      state: OK
  dataNode: DATA_NODE-34
  device: vmxnet3s0
  macAddress: 0:c:29:32:96:a3
  mtu: 1500
  mtuRange: 60-9000
  reference: NETWORK_INTERFACE-vmxnet3s0-DATA_NODE-34
  state: OK
```

4.17 Stopping and starting the containerized Continuous Compliance Engine

4.17.1 Overview

This article describes how to stop and start the containerized Delphix Continuous Compliance engine. For information on performing the tasks for the Virtual Machine Masking Engine, please see the documentation located in the document [Starting, Stopping, and Restarting the Masking Engine](#) (see page 317).

Containerized deployments are dependent on a customer-created configuration file which can be named anything. For the purposes of this document, the default name of `kubernetes-config.yaml` will be used. Also, any command-line examples will assume that this file is in the current directory to simplify the example.

4.17.2 Starting the containerized Masking Engine

Starting the engine is a simple matter of asking Kubernetes to create the Pod described by the Pod configuration file. This is done with a single `kubectl` command.

```
$ kubectl create -f ./kubernetes-config.yaml
```

The Pod will take some time to start. The status of the Pod can be verified with another simple `kubectl` command.

```
$ kubectl get pods
NAME                READY   STATUS    RESTARTS   AGE
delphix-masking-0  3/3     Running   2 (5d14h ago)  13d
```

A Containerized Masking Pod consists of 3 containers. The above output demonstrates a Pod where 3/3 containers are `READY`. This is a Pod that is up and running and ready to accept connections.

ⓘ It is common for the first 2 containers of the Pod to enter a `READY` state very quickly and for the 3rd container to take some time to become ready. How long is dependent on a number of factors including the underlying infrastructure. (how powerful, how busy)

If the Pod `STATUS` indicates an error or the number of restarts is consistently climbing, that indicates that there is a problem with the Pod and debugging will need to be done to determine the problem and the appropriate resolution.

4.17.3 Stopping the containerized Masking Engine

Stopping a running Pod is simple despite some confusing terminology. The Kubernetes terminology for stopping a Pod is `delete`, but this command does not delete any of the containers or persistent volumes. It only stops the running Pod. The command to stop a running pod is of the same form as starting the Pod.

```
$ kubectl delete -f ./kubernetes-config.yaml
```

This tells Kubernetes to shut down whatever it previously started. Because the Containerized Masking Engine is a Stateful application, it has persistent storage. This persistent storage is not deleted when the Pod is shut down.

If a Pod that was shut down is then restarted, it will attempt to re-attach any persistent storage defined in the `kubernetes-config.yaml` file.

4.17.4 Removing persistent volumes / persistent volume claims

If it is necessary to delete any persistent volumes (PVs) and persistent volume claims (PVCs) associated with the Pod, that will have to be done manually. It is possible to locate any PVs and PVCs that exist with some simple `kubectl` commands.

```
$ kubectl get pv
NAME                                CAPACITY  ACCESS MODES  RECLAIM POLICY
STATUS  CLAIM                                STORAGECLASS
REASON  AGE
pvc-7fe1d352-de17-4132-b2a1-152f4e9cfefc  20Gi      RWX           Delete
Bound  container-registry/registry-claim    microk8s-hostpath
183d
pvc-a7275ce3-b630-4d4c-9712-b5124358cb7f  4Gi       RWO           Delete
Bound  default/masking-persistent-storage-delphix-masking-0  microk8s-hostpath
15d
nfs-pv                                500Mi     RWO           Retain
Bound  default/nfs-pvc                                nfs-storage
13d

$ kubectl get pvc
NAME                                STATUS  VOLUME
CAPACITY  ACCESS MODES  STORAGECLASS  AGE
masking-persistent-storage-delphix-masking-0  Bound  pvc-a7275ce3-b630-4d4c-9712-
b5124358cb7f  4Gi          RWO           microk8s-hostpath  15d
nfs-pvc                                Bound  nfs-pv
500Mi     RWO          nfs-storage    13d
```

To completely remove a Pod (a clean slate) would require the removal of any PVs and PVCs associated with the Pod. The first step is to shut down the Pod. Once the Pod is no longer running, removing the PVC will frequently also remove the associated PV.

Removing either the PV or PVC is a simple matter of using the appropriate `kubectl` command. To illustrate removing a PVC, simply take note of the name of the object. From the example output above, there is a PVC named `masking-persistent-storage-delphix-masking-0`. To remove it, use the following command.

```
$ kubectl delete pvc masking-persistent-storage-delphix-masking-0
```

4.18 Stopping, starting, and restarting the continuous compliance engine

4.18.1 Overview

This article describes how to stop, start, and restart the Virtual Machine-based Delphix Continuous Compliance engine. Use cases, troubleshooting tips before a restart, and steps in the CLI are outlined in the following sections. For instructions on stopping and starting the Containerized Masking Engine, please see the document [Stopping and Starting the Containerized Masking Engine \(see page 317\)](#).


4.18.2 Use cases examples

Stopping and starting the Masking Engine may be required when performing:

- Masking Engine maintenance work.
- Backup and Restore.

Restarting the Masking Engine may be required if:


- The Masking Engine is unreachable or unresponsive.
- A Masking Job is in an incorrect state.

 Stopping and Starting the Masking Engine will terminate all running jobs; this includes Imports, Inventory Scans, Profiling and Masking Jobs, etc.

4.18.3 Troubleshooting before a restart

If the Masking Engine is unreachable, the following should always be checked before a restart:

- Verify that the Engine is reachable over the network using ping.

 Verify that no jobs are running (unless the job should be terminated). If a root cause investigation is needed, please open a case with Delphix Support and upload a support bundle.

- Containerized Masking functions vary differently from the Virtual Machine deployment. For information on performing these same functions for Containerized Masking, please see the documentation page for [Stopping and Starting the Containerized Masking Engine](#) (see page 317).

Using the shell or putty, access the Masking Engine and login using the sysadmin user.

- The sysadmin password is the password set when the Masking Engine was configured.

```
# Access CLI using SSH.
ssh sysadmin@<yourEngine>
```

4.18.4 Using the Command-Line Interface (CLI)

The CLI provides means to access information and execute commands on the Engine without a GUI; one of which is to stop and start the Continuous Compliance Engine. This is done using the system menu.

- At the CLI prompt, type **system**.
- At the system prompt, do one of the following, depending on the desired action:
 - To enable the engine: type **startMasking** and then commit.
 - To disable the engine: type **stopMasking** and then commit.
 - To restart the engine: type **stopMasking** and commit, then **startMasking** and commit.
- To exit the CLI, type **exit**.

- If the Masking Engine fails to start, it could be worth waiting a few minutes (2 minutes or so) and then try `stopMasking`, followed by `startMasking` again. Startup failure could be the masking service entering Maintenance Mode. You cannot clear Maintenance Mode by entering `startMasking`; you must use `stopMasking`, followed by `startMasking`. If this fails, Delphix Support needs to investigate why the service failed.

4.18.4.1 Restarting the Masking Engine example

Below is an example of how to restart the Continuous Compliance Engine using the CLI.

```


$ ssh sysadmin@yourEngine
Password:
yourEngine> system
yourEngine system> ls
startMasking stopMasking
yourEngine system> stopMasking
yourEngine system stopMasking *> commit
yourEngine system> startMasking
yourEngine system startMasking *> commit
yourEngine system> exit
Connection to yourEngine closed

```

4.19 Upgrading the Continuous Compliance Engine

4.19.1 Upgrades for virtual Compliance Engines

Upgrading Delphix appliances is a multi-step process. This process will affect the availability of the Compliance Engine administrative interface and virtual datasets during the operation, based on the type of upgrade chosen.

-  Customers running version 5.3.9 and earlier that are requesting an upgrade to 6.0.0.0 and above, please contact Delphix Support to help coordinate this upgrade.

Upgrading from 6.0.x to 6.0.x includes pre-checks packaged in the upgrade image, thus, contacting Delphix Support is **not required** for this upgrade (e.g. 6.0.0.0 -> 6.0.9.0).

For more information on upgrades and the process, please visit the [Upgrading the Delphix Engine](#)¹³¹ documentation section. The page is located in the Continuous Data documentation suite, but is relevant to upgrading Continuous Compliance.

4.19.2 Upgrades for containerized Compliance Engines

Containerized Masking is generally expected to be used in an ephemeral fashion. The general process for utilizing newer versions is to upload the new set of containers and deploy new engines from them.

There is not currently a certified process by which to upgrade a Containerized Masking Engine in-place. If you have a need this, please contact your Delphix Representative and inquire about opening an enhancement request.

¹³¹ <https://cd.delphix.com/docs/latest/upgrade>

4.20 Utilization

4.20.1 Overview

Delphix helps with keeping track of Compliance engine utilization. You can access the utilization reports from the **Admin > Utilization** page. This article discusses the UI.

4.20.2 Utilization UI page

The **Utilization** page can be found on the UI under the **Admin** tab. This page provides the user the capability to generate two types of utilization reports in PDF format:

- **Jobs:** The report lists the number of job executions for a specified environment and date range.
- **Database Size:** This report is designed to assist customers with usage-based pricing contracts. The report contains the databases masked and their respective size.
 - Currently, this information is only gathered for Oracle databases (Oracle 12c or greater). Support for Oracle 11g and other database platforms will be added in future releases.
 - There is also an API endpoint to get database usage information. For more information, please refer to [API documentation](#). (see page 878)

4.20.3 The Jobs Utilization Report

To generate the jobs utilization report:

1. Select the **Utilization Type** as Jobs.
2. Select the **Job Environment** from the list.
3. Provide **Start Date** and **End Date**.

The screenshot shows the 'Utilization' page in the Continuous Compliance application. The top navigation bar includes 'Environments', 'Monitor', 'Settings', 'Admin', and 'Audit'. The left sidebar has 'About', 'Application Settings', 'Logs', 'Roles', 'Users', and 'Utilization' (highlighted). The main content area shows the breadcrumb 'Admin > Utilization' and the title 'Utilization'. Below the title are three input fields: 'Utilization Type' (set to 'Jobs'), 'Select Job Environment' (set to 'env_RP35XPSP'), and 'Start Date - End Date (UTC)' (set to '04/01/2024 - 04/03/2024'). A 'Generate PDF' button is located at the bottom of the form.

4. Click **Generate PDF**.

4.20.4 The Database Size Report

To generate the database size report:

1. Select the **Utilization Type** as Database Size.
2. Select a Date Range from the predefined list of date ranges. Select Custom to supply a custom Start Date and End Date.

3. Click **Generate PDF**.

4.20.4.1 Support matrix

The Delphix engine supports database size calculation for the following databases:

4.20.4.1.1 Oracle

Data Source	Version	Availability	Setup Instructions
Oracle	Oracle 11g	Unavailable	
	Oracle 12c and later	Available	Preparing Oracle Database for Profiling/ Masking (see page 323)
All other data sources		Unavailable	

Multiple options can be selected.

5 Preparing data

This section includes the following topics:

- [Database user permissions for executing masking and profiling job \(see page 322\)](#)
- [Preparing Oracle database for profiling/masking \(see page 323\)](#)
- [Preparing SQL server database for profiling and masking \(see page 326\)](#)
- [Preparing Sybase database for profiling and masking \(see page 328\)](#)

5.1 Database user permissions for executing masking and profiling job

5.1.1 Introduction

This section provides the recommended list of permissions required for executing Masking and Profiling jobs on the Continuous Compliance Engine. This page provides general permission recommendations. The subsequent pages in this section provide detailed recommendations for specific databases.

Delphix recommends that a separate Database user (i.e. named *Masking User*) be created across all the databases with the appropriate permissions on the schemas to be masked. If needed create multiple users. The appropriate permissions for the database *Masking User* are listed below.

The benefits of having a separate DB *Masking User*:

- Replicating the new user (and privileges) are easier
- Access Audits are much easier
- Can be created as a central AD user and used at many places simultaneously

5.1.2 List of database entitlements required to run masking jobs

- Read data from Tables
- Write data to Tables
- Update data in tables
- Create indexes
- Drop indexes
- Create triggers
- Drop triggers
- Disable triggers
- Enable triggers
- Alter tables add column
- Alter table delete column
- Create constraints
- Delete constraints

- Disable constraints
- Enable constraints

5.1.3 List of database entitlements required to run profiling jobs

- View Definition (Schema)
- Read Data from Tables

5.2 Preparing Oracle database for profiling/masking

5.2.1 Overview

Before masking your data, it is important to prepare your database. This article explains the required changes, reasons for the changes, and instructions on how to make the changes.

5.2.2 Archive logging

What is Archive Logging?

Oracle Database lets users save filled groups of redo log files to one or more offline destinations, known collectively as the archived redo log, or more simply the archive log. The process of turning redo log files into archived redo log files is called **archiving**. This process is only possible if the database is running in ARCHIVELOG mode. Users can choose automatic or manual archiving.

Why is it important to make this change?

Archive logging will slow down masking processes and absorb CPU resources that could be used by the masking process. Furthermore, since masking will change every row in every table being masked logs are only needed for short term recovery and transaction backout.

The choice of whether to enable the archiving of filled groups of redo log files depends on the availability and reliability requirements of the application running on the database. If you cannot afford to lose any data in your database in the event of a disk failure, use ARCHIVELOG mode. The archiving of filled redo log files can require you to perform extra administrative operations.

How exactly do I make this change? (exact commands, etc).

```
ALTER DATABASE NOARCHIVELOG;
```

5.2.3 DB/VDB memory allocation

What is SGA? A system global area (SGA) is a group of shared memory structures that contain data and control information for one Oracle database instance. If multiple users are concurrently connected to the same instance, then the data in the instance's SGA is shared among the users. Consequently, the SGA is sometimes called the shared global area.

An SGA and Oracle processes constitute an Oracle instance. Oracle automatically allocates memory for an SGA when you start an instance, and the operating system reclaims the memory when you shut down the instance. Each instance has its own SGA.

The SGA is read/write. All users connected to a multiple-process database instance can read the information contained within the instance's SGA, and several processes write to the SGA during the execution of Oracle. When automatic SGA memory management is enabled, the sizes of the different SGA components are flexible and can adapt to the needs of a workload without requiring any additional configuration. The database automatically distributes the available memory among the various components as required, allowing the system to maximize the use of all available SGA memory. Make sure the DB/VDB memory allocation is sufficient for the workload. Delphix's best practices for sizing a VDB will handle most masking requirements. If you plan to run many concurrent masking jobs a small memory allocation will negatively impact the performance of the masking jobs.

Why is it important to make this change?

To assure that masking jobs will perform at an optimum level.

How exactly do I make this change? (exact commands, etc). Set automatic SGA memory management to enabled. If not allowed set the SGA based on the diagnosis from the AWR report generated during a masking job. The DBA is best suited to make the appropriate tuning changes to the SGA parameters for the version of Oracle being masked.

5.2.4 Undo tablespace size and undo retention time:

What is tablespace? Every Oracle Database must have a method of maintaining information that is used to roll back or undo, changes to the database. Such information consists of records of the actions of transactions, primarily before they are committed. These records are collectively referred to as undo.

Undo records are used to: - Roll back transactions when a ROLLBACK statement is issued - Recover the database - Provide read consistency - Analyze data as of an earlier point in time by using Oracle Flashback Query - Recover from logical corruptions using Oracle Flashback features

When a ROLLBACK statement is issued, undo records are used to undo changes that were made to the database by the uncommitted transaction. During database recovery, undo records are used to undo any uncommitted changes applied from the redo log to the datafiles. Undo records provide read consistency by maintaining the before image of the data for users who are accessing the data at the same time that another user is changing it.

Why is it important to make this change?

The masking Engine updates or inserts masked data in batches. In the case of an insert, it only requires the current transaction size for the commit of each table being masked. The default per table stream is 10k rows. However, with an update, the transaction is not complete until the entire table is masked. So, the more tables and more rows and the wider (size) each row is in each table, the more undo space is needed to complete the transaction. Large tables, such as DW tables or history and Audit tables, most often need an increase to the Undo space and undo Retention time for updates. If space or time is exceeded then the masking job may fail with an ORA-01555, Snapshot too old error.

How exactly do I make this change? (exact commands, etc).

It is highly recommended to increase the Undo space and undo Retention time when running in-place jobs on large tables. A general rule of thumb is 2 or 3 times the size of the largest table(s), or if there are multiple tables running at the same time, then all tables combined. A DBA is best suited to make the necessary UDNO Space and UNDO Retention changes.

5.2.5 Redo logs are optimally sized

What is Redo Logs?

The most crucial structure for recovery operations is the redo log, which consists of two or more preallocated files that store all changes made to the database as they occur. Every instance of an Oracle Database has an associated redo log to protect the database in case of an instance failure.

Why is it important to make this change?

The most important reason to make this change is to keep performance optimal. If redo logs are too small, then the log switching will occur too often, using up valuable Oracle resources.

How exactly do I make this change? (exact commands, etc).

A DBA is best suited to make these changes appropriately.

5.2.6 Change PCTFREE to 40-50:

What is PCTFREE?

PCTFREE and PCTUSED are used together, but PCTFREE is critical for updates. The larger the PCTFREE value the more updates can be done.

Why is it important to make this change?

PCTFREE aids in performance increases for updating Oracle during masking. The Masking Engine does many updates at the same time in batch mode. The more that can be done without DB overhead the faster the masking jobs run.

How exactly do I make this change? (exact commands, etc).

A DBA is best suited to make these changes.

5.2.7 Change primary Key To ROWID:

What is ROWID?

For each row in the database, the ROWID pseudocolumn returns the address of the row. Oracle Database rowid values contain information necessary to locate a row.

Why is it important to make this change?

This is especially important in masking for performance. IF ROWID is used then Oracle will manage the updates for the rows it tracks using ROWID. This makes updates much faster. On occasion, there may be a key (PK/FK/UK) or ID column with an index that is faster, but generally, ROWID is the fastest.

How exactly do I make this change? (exact commands, etc).

Add ROWID as the logical key on each table in the ruleset using the Masking Engine GUI. Also, in a script you should drop foreign keys, and if possible indices and disable triggers and recreate them after the masking job has been run for any of these types of columns being masked.

5.2.8 Masking user privileges:

The following privileges are required for the database user provided in the source connector configuration

- **SELECT** on V\$CONTROLFILE
- **SELECT** on V\$DATAFILE_HEADER
- **SELECT** on V\$INSTANCE

Why is it important to make this change?

These privileges are needed to calculate the size of the source database.

How exactly do I make this change? (exact commands, etc).

SQL commands to grant required privileges to the user

```
GRANT SELECT on v_$controlfile to [mask_user];  
GRANT SELECT on v_$datafile_header to [mask_user];  
GRANT SELECT on v_$instance to [mask_user];
```

5.3 Preparing SQL server database for profiling and masking

Before masking your data, it is important to prepare your database. This section explains the required changes, reasons for the change, and the instructions to make the change.

5.3.1 Logging

What is Simple Recovery Model?

SQL Database Simple Recovery model - Automatically reclaims log space to keep space requirements small, essentially eliminating the need to manage the transaction log space. Operations that require transaction log backups are not supported by the simple recovery model.

Why is it important to make this change?

Reducing the overhead of the transaction logging and the size of the files before checkpoints increases the masking speed significantly.

How exactly do I make this change?

Either (a) use SQL Server Management Studio to open the DB properties dialog box and select the “simple recovery model” or (b) issue the `SET RECOVERY SIMPLE` statement from a SQL query tool. Please see [this reference](#)¹³² for more details.

5.3.2 DB/VDB memory allocation

What is min/max memory in SQL Server?

Memory is allocated at the SQL Server level, so all the DBs will share the entire load. The max memory should be close to the maximum available on the server.

Why is it important to make this change?

To assure that masking jobs will perform at an optimum level.

How exactly do I make this change?

Use SQL Server Management Studio and change the max memory allocation for the server.

¹³² <https://docs.microsoft.com/en-us/sql/relational-databases/backup-restore/view-or-change-the-recovery-model-of-a-database-sql-server>

5.3.3 Primary/Foreign/DMS_ROW_ID Keys

What is a key?

A key is a unique, non-null value that identifies a row in the database.

Why is it important to make this change?

Using a PK or Foreign key is critical for fast updates. When a table does not have an identity column with an index or a PK/FK then the masking engine will alter the table to have an Identity column, DMS_ROW_ID to optimize performance.

How exactly do I make this change?

A logical key can be added to a table in the Masking Engine Ruleset for each table, if there is a specific column that would find the row to update faster than the current PK/FK.

5.3.4 Creating a masking user and privileges

It is highly recommended to create a database user, and possibly a role, for use by the Masking Engine. This user should be created in a non-Production environment and not in your production environment. The following permissions are needed:

- db_datareader
- db_datawriter
- db_ddladmin

SQL commands to add a user with the required privileges:

```
USE [mask_db]
GO
CREATE LOGIN [mask_user] WITH PASSWORD=N'delphix123'
GO
CREATE USER [mask_user] FOR LOGIN [mask_user]
GO
USE [mask_db]
GO
ALTER ROLE [db_datareader] ADD MEMBER [mask_user]
GO
USE [mask_db]
GO
ALTER ROLE [db_datawriter] ADD MEMBER [mask_user]
GO
USE [mask_db]
GO
ALTER ROLE [db_ddladmin] ADD MEMBER [mask_user]
GO
```

5.4 Preparing Sybase database for profiling and masking

Before masking data, the database(s) must be prepared. This page provides configuration resources, information on how memory requirements are determined, table keys, creating roles, and creating user privileges.

5.4.1 Determining the amount of memory SAP ASE needs

As referenced in SAP's [Determining the amount of memory SAP ASE needs](#)¹³³ page, the total memory SAP ASE requires to start is the *sum of all memory configuration parameters*, plus the *size of the procedure cache*, plus the *size of the buffer cache*, where the size of the procedure cache and the size of the buffer cache are expressed in round numbers rather than in percentages. Visit the page mentioned above for more elaborated detail.

5.4.2 Determining SAP ASE memory configuration

As referenced in SAP's [Determining SAP ASE memory configuration](#)¹³⁴ page, the total memory allocated during system start-up is the sum of memory required for all the configuration needs of SAP ASE. You can obtain this value from the read-only configuration parameter `total logical memory`. Visit the page mentioned above for more elaborated detail.

5.4.3 In-place masking and table keys


A key is a unique, non-null value that identifies a row in the database table. When masking using In-Place, this key looks up the row to update the masked record. For performance, this key needs to be indexed. The best index is a Clustered Index.

- **Primary Key (PK):** If there is a Primary Key, the Continuous Compliance Engine will use this key.
- **IDENTITY column:** If there is no PK, but there is an IDENTITY column on the table, the Continuous Compliance Engine will use this. If this column is not indexed, the Continuous Compliance Engine will add a non-clustered index called `i_MASKING_GENERATED_IDENTITY_TMP`.
- **Logical Key (LK):** You can add a Logical Key to override the key used. The Logical Key can be one or multiple columns. The Logical Key has to be unique and can't have NULLs.
- **No Key:** If there is no Key, the Continuous Compliance Engine has an automated feature that will add an IDENTITY column called `MASKING_GENERATED_IDENTITY_TMP`. This column is removed at the end of the masking job execution. This column will be indexed with a non-clustered index called `i_MASKING_GENERATED_IDENTITY_TMP`.
 - An alternative would be using On-the-Fly masking.

¹³³ https://help.sap.com/docs/SAP_ASE/3bdda6b0ffad441aab4fe51e4e876a19/a86647e6bc2b10148590975389f4734e.html?version=16.0.2.9

¹³⁴ https://help.sap.com/docs/SAP_ASE/3bdda6b0ffad441aab4fe51e4e876a19/a866c44fbc2b101481b18b39e4cb726d.html

5.4.4 Transaction logs

 When a table is altered, Sybase ASE will create log entries (needed for Rollback and other DB features so these are unavoidable). The transaction log size required for masking is larger than the combined size of all tables masked.


The management of Transaction Logs on Sybase is a manual DBA task.

5.4.4.1 On-the-Fly

Sybase Transaction Logs can be turned off on Insert. To do this, you need to enable [select into/bulkcopy/pllsort](#)¹³⁵ on the database. Visit SAP's [The transaction log](#)¹³⁶ page for more information.

5.4.4.2 Sizing Transaction Logs

In general, the amount of tables and records masked can result in large transactions. Each database's transaction log should be managed appropriately to allow the masking jobs to run; failure in doing so can result in the suspension of the transaction and the masking job appears to hang.

 Review the ASE documentation [Managing Free Space with Thresholds](#)¹³⁷ on how to manage the transaction log threshold. Resizing the database can be necessary to have a larger transaction log. When resizing a Delphix VDB, make sure any new log devices are created in the VDB's underlying `datafile` directory, provided by the Delphix Engine. For more information, visit the [Resizing an SAP ASE VDB](#)¹³⁸ page.

5.4.5 Creating a masking user and privileges

It is highly recommended to create a database user and possibly a role to mask. This user should not be created in production, but should be created in non-production. The following permissions are needed:

- Syntax to add a user and give privileges:

¹³⁵ https://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.dc31654_1251/html/sag/X80055.htm

¹³⁶ <https://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.help.sqlanywhere.12.0.1/dbadmin/da-dbfiles-s-4160005.html>

¹³⁷ <https://help.sap.com/viewer/3bdda6b0ffad441aab4fe51e4e876a19/16.0.3.7/en-US/a8c58629bc2b10148a2c8f38befbcac8.html>

¹³⁸ <https://delphixdocs.atlassian.net/wiki/spaces/CD/pages/5341432/Resizing+an+SAP+ASE+VDB>

```
sp_adduser mask_user;  
CREATE user NEWUSER;  
CREATE LOGIN mask_user WITH PASSWORD Delphix_123; --THIS MUST BE DONE IN MASTER  
CREATE USER mask_user IDENTIFIED BY Delphix_123;  
GRANT SELECT ON PII_V2 TO mask_user; GRANT INSERT ON PII_V2 TO mask_user; GRANT  
DELETE ON PII_V2 TO mask_user; GRANT ALTER ON PII_V2 TO mask_user; GRANT UPDATE ON  
PII_V2 TO mask_user;  
GRANT ALTER ANY TABLE TO mask_user;
```

- Adaptive Server requires a two-step process to add a user: `sp_addlogin`, followed by `sp_adduser`.

```
CREATE LOGIN MASK_SUPER_USER WITH PASSWORD Delphix_123;  
sp_addlogin MASK_SUPER_USER, Delphix_123;  
GRANT ROLE sa_role TO MASK_SUPER_USER;
```


6 Connecting data

This section contains the following topics:

- [Managing environments](#) (see page 331)
- [Managing remote mounts for VM continuous compliance engines](#) (see page 340)
- [Managing remote mounts for containerized masking](#) (see page 348)
- [Managing SSL/TLS over JDBC for containerized masking](#) (see page 352)
- [Managing connectors](#) (see page 355)
- [Managing extended connectors](#) (see page 389)
- [Managing rule sets](#) (see page 400)
- [Managing file formats](#) (see page 427)
- [Managing inventories](#) (see page 451)
- [Managing record types and header/footer records](#) (see page 475)
- [Managing jobs](#) (see page 482)
- [Whole file masking](#) (see page 522)

6.1 Managing environments

This section describes how you can create and manage your environments in the masking service.

As a reminder, environments are used to group certain sets of objects within the Masking Engine. They can be thought of as folders/containers where a specified user can create manage connectors, rule sets, and jobs.

The Main Environment screen lists all the environments the logged in user has access to. It is the first screen that appears when a user logs into Delphix.

The screenshot shows the 'Environments' page in the Continuous Compliance interface. The top navigation bar includes the logo and tabs for 'Environments', 'Monitor', 'Settings', 'Admin', and 'Audit'. Below the navigation, the page title 'Environments' is displayed along with buttons for '+ Application', '+ Environment', and an 'Actions' dropdown. A table lists the environments, with the following data visible:

ID ↑	Application	Environment	Purpose	Approval	Actions
1	App	Env1	Masking	Disabled	...

At the bottom right of the table area, it says 'Displaying 1 to 1 of 1'.

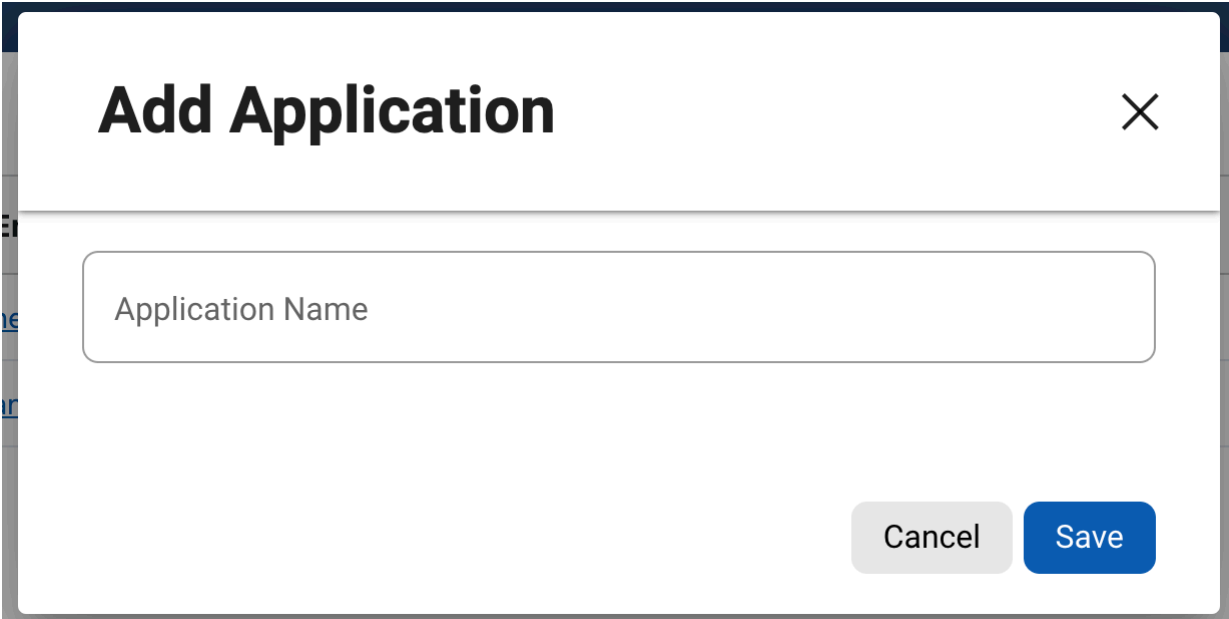
The main **environments** screen contains the following information and actions:

- **ID** – The numeric ID of the environment used to refer to the environment from the Masking API.
- **Application** – A way to indicate the name of the application whose data will be managed within this environment.
- **Environment** – The name of the environment.
- **Purpose** – The purpose of the environment.
- **Actions** – List of actions
 - **Edit** – Edit the environment. See more details below.
 - **Export** – Export the environment. See more details below.
 - **Duplicate** – Copy the environment. See more details below.
 - **Delete** – Delete the environment. See more details below.

The environments on the screen can be filtered or sorted by the various informational fields by clicking on the respective field. More information on grid filtering and sorting can be found in their respective sections, in the [Graphical user interface](#)¹³⁹ page..

6.1.1 Adding an application

For an environment to be created, an application needs to be specified. Here are the steps to add an application:

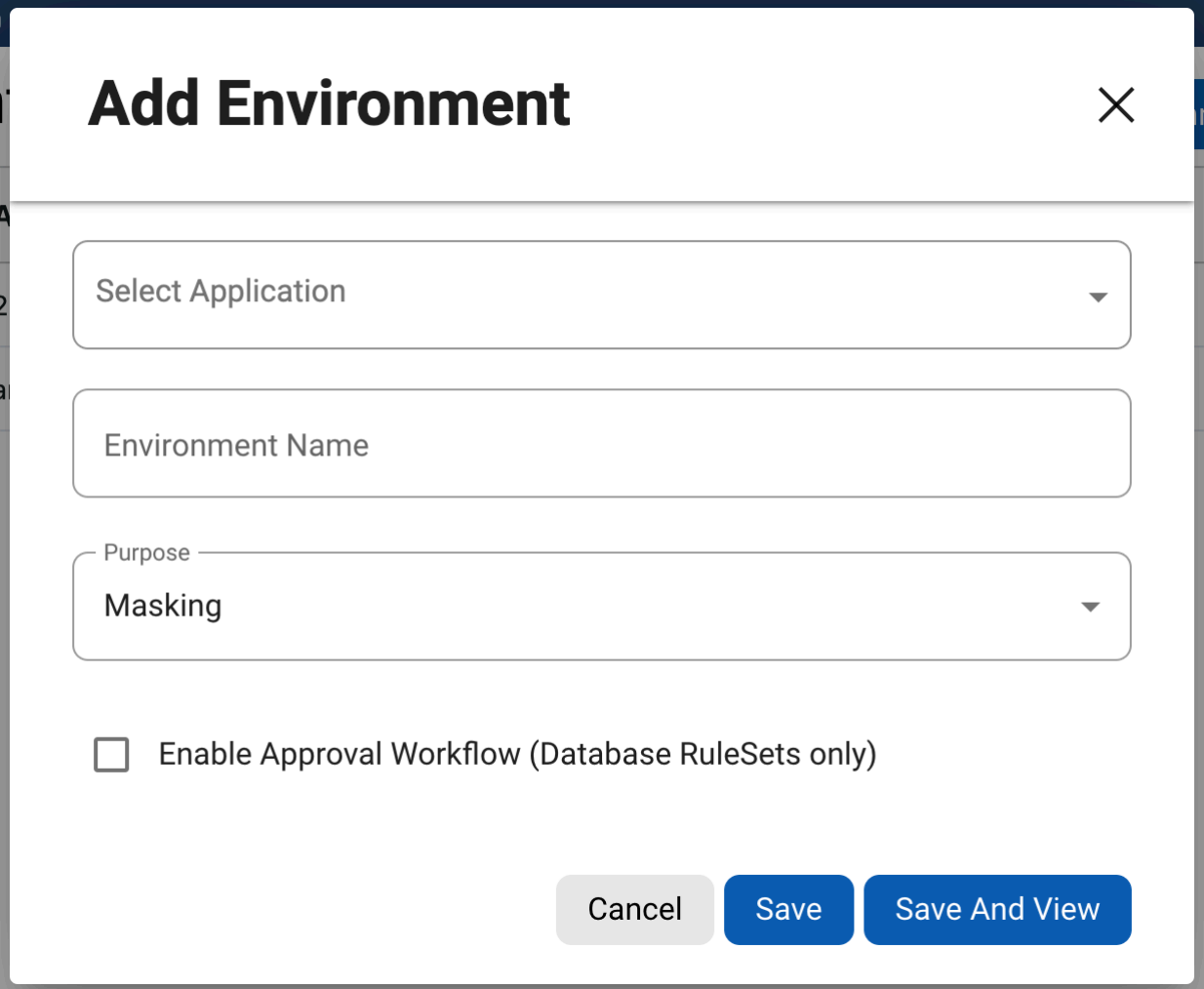


1. On the main environments page, near the upper right-hand corner of the screen, click on the **+** **Application** button.
2. The screen prompts you for the following items:
 - a. Application Name
3. Click **Save** to return to the **Environments List** screen.

¹³⁹ <https://masking.delphix.com/docs/latest/graphical-user-interface>

6.1.2 Creating an environment

Here are the steps you need to take to create an environment:



Add Environment ✕

Select Application ▼

Environment Name

Purpose
Masking ▼

Enable Approval Workflow (Database RuleSets only)

Cancel Save Save And View

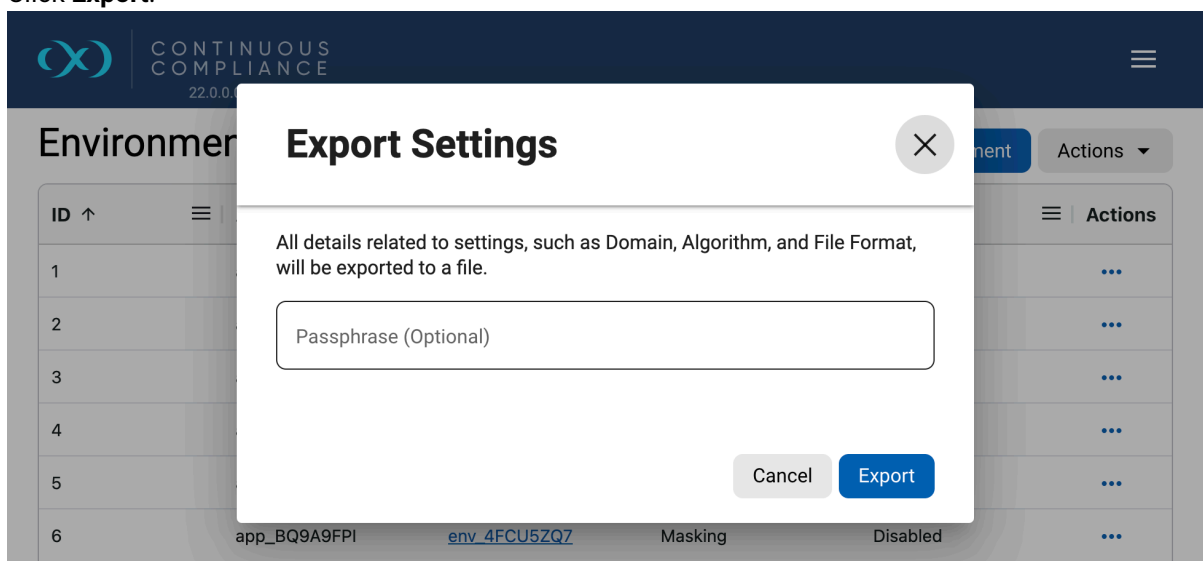
1. On the main environments page, in the upper right-hand corner of the screen, click on the **+** **Environment** option.
2. The screen prompts you for the following items:
 - **Application Name** – Dropdown menu displaying the names of applications for associating with the environment, for informative purposes.
 - **Environment Name** – The display name of the new environment.
 - **Purpose** – The type of masking workflow for the environment: Mask or Tokenize/Re-Identify.
 - **Enable Approval Workflow (Database rulesets only)** – Whether or not to require approvals of inventories before masking jobs can be run in the environment. Applicable for Database rulesets only.

3. Either click **Save** to return to the **Environments List** screen, or click **Save And View** to display the environments **Jobs** screen.

6.1.3 Exporting settings

To export the Settings:

1. On the main environments page, in the upper right-hand corner of the screen, click on the **Action** drop-down list and select the **Export Settings** option.
2. The screen prompts you to take the input for the optional **Passphrase**.
3. Click **Export**.

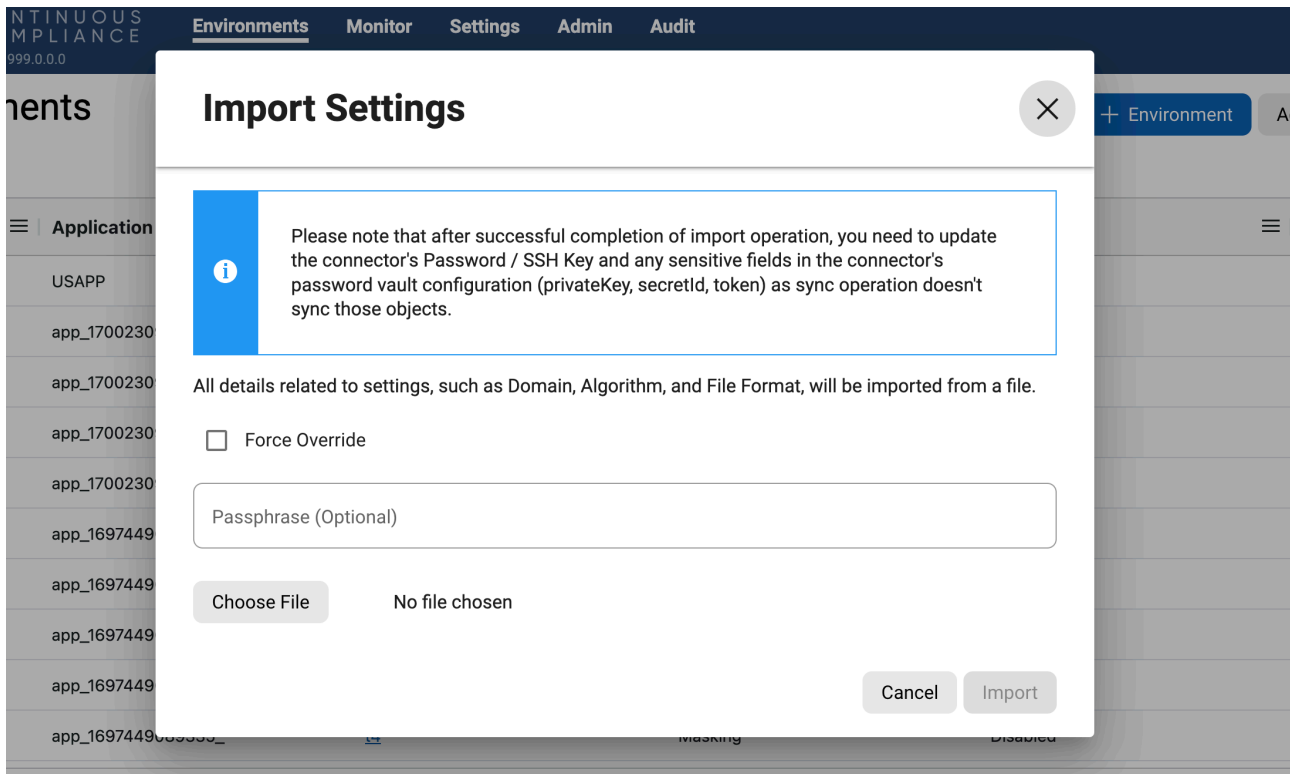


All the information related to Settings (Domain, Algorithm, File Format and so on) is exported to a file.

A status message will display in the bottom sheet containing the **Async Task ID** generated for the export. You can utilize this ID to monitor the export status from the **Monitor** → **Async Tasks** page. Once the export process finishes, it will automatically update the status to the corresponding Async Task entry. On successful export, to download the exported file manually, click on the "**Download file**" option in the Actions (...) dropdown corresponding to the respective Async Task ID entry.

6.1.4 Importing settings

Once you have exported your settings, you can easily import it into another Masking Engine. To import settings:



1. On the main environments page, in the upper right-hand corner of the screen, click on the **Action** drop-down list and select the **Import Settings** option.
2. The screen prompts you for the following items:
3. **Passphrase** – You can input the **Passphrase** (optional). If the settings were exported using a passphrase then you must use the same passphrase for the import settings as well otherwise the import operation will fail.
4. **Force Overwrite** – Specify whether the import should fail if an object already exists with the same ID or the existing object should be overwritten. Click on the force overwrite checkbox if you want to overwrite the existing object.
5. **Settings File** – Click on **Choose File** button to browse for the exported settings file that contains the information you want to import. (This file must be a previously exported masking environment.)
6. Click **Import** button to start the import operation.

A status message appears in the bottom sheet with the Async Task Id generated for the import, using which you can check the import status from [Async Task Status](#)¹⁴⁰ page. When the import operation is complete, it will automatically update the current status of the import operation on the [Async Task Status](#)¹⁴¹ page.

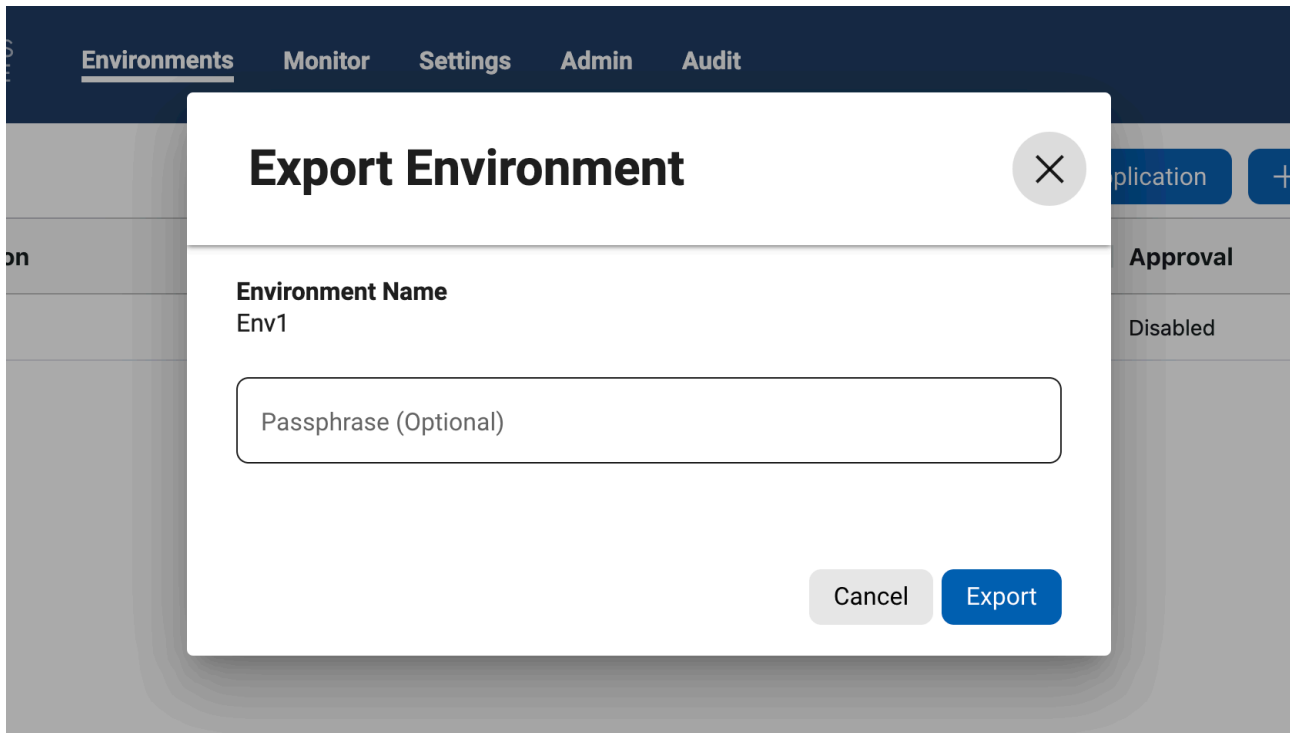
6.1.5 Exporting an environment

For a variety of different reasons (the main one being moving environments between masking engines), you may want to export all the objects within an environment (connectors, rule sets, masking jobs, etc).

¹⁴⁰ <https://delphixdocs.atlassian.net/wiki/x/0wGNBw>

¹⁴¹ <https://delphixdocs.atlassian.net/wiki/x/0wGNBw>

To export an environment use the Export Environment option available in the Masking UI. To export an individual environment:



1. Click on the Actions (...) drop-down icon beside each environment entry on the Environment Listing screen, and choose the **Export** option. Alternatively, you can click the **Export Environment** button on the **Environment Jobs** screen.
2. A pop-up is displayed with the following items pre filled :
 - **Environment Name** : The display name of the new environment.
3. You can input the optional **Passphrase**.
4. Click **Export**.

All the information for the specified environment (connectors, rule sets, inventory, jobs, and so on) is exported to a file.

A status message will display in the bottom sheet containing the **Async Task ID** generated for the export. You can utilize this ID to monitor the export status from the **Monitor** → **Async Tasks** page. Once the export process finishes, it will automatically update the status to the corresponding Async Task entry. On successful export, to download the exported file manually, click on the "**Download file**" option in the Actions (...) dropdown corresponding to the respective Async Task ID entry.

6.1.6 Importing an environment

Once you have exported your environment, you can easily import it into another Masking Engine. To import an environment

Import Environment

Please note that after successful completion of import operation, you need to update the connector's Password / SSH Key and any sensitive fields in the connector's password vault configuration (privateKey, secretId, token) as sync operation doesn't sync those objects.

Import Settings Force Overwrite

Select or Enter Application

Select or Enter Environment

OTF Environment

Passphrase (Optional)

Settings File

Choose File No file chosen

Environment File

Choose File No file chosen

Cancel Import

1. On the main environments page, in the upper right-hand corner of the screen, click on the **Select Action** drop-down list and select the **Import Environment** option.
2. The screen prompts you for the following items:
 - **Import Settings** – Click the checkbox if you want to import settings as well.
 - **Force Overwrite** – Specify whether the import should fail if an object already exists with the same ID or the existing object should be overwritten. Click on force overwrite checkbox if you want to overwrite the existing object.
 - **Application** – You can select the existing application from the application drop-down or you can enter the application name to create a new application.
 - **Environment** – You can select the existing environment from the environment drop-down or you can enter the environment name to create a new environment.
 - **OTF Environment** – Click on **OTF Environment** checkbox to import the on-the-fly connectors into that environment.

- **Source Application** – You can select the existing application from the source application drop-down or you can enter the application name to create a new application.
- **Source Environment** – You can select the existing environment from the environment drop-down or you can enter the environment name to create a new environment.
- **Passphrase** – You can input the **Passphrase** by clicking the **Use Passphrase** checkbox. If the exported file is used the passphrase then you should use the same passphrase for the import as well.
- **Settings File** – Click on **Select...** button to browse for the exported settings file that contains the information you want to import. (This file must be a previously exported masking environment.)
- **Environment File** – Click on **Select...** button to browse for the exported environment file that contains the information you want to import. (This file must be a previously exported Masking environment.)

3. Click **Import** button to start the import operation.

A status message appears in the bottom sheet with the Async Task Id generated for the import, using which you can check the import status from [Async Task Status](#)¹⁴² page. When the import operation is complete, it will automatically update the current status of the import operation on the [Async Task Status](#)¹⁴³ page.

6.1.7 Editing an environment

To change the properties of an environment, do the following:

The screenshot shows the 'Edit Environment' dialog box in the Continuous Compliance interface. The dialog has a title bar with a close button (X). Below the title bar, there are three input fields: 'Select Application' with a dropdown arrow, 'Environment Name', and 'Purpose' with a dropdown arrow. Below these fields is a checkbox labeled 'Enable Approval Workflow (Database RuleSets only)'. At the bottom of the dialog are three buttons: 'Cancel', 'Save', and 'Save And View'. The background shows a table of environments with columns for 'Application' and 'Environment Name'.

¹⁴² <https://delphixdocs.atlassian.net/wiki/x/0wGNBw>

¹⁴³ <https://delphixdocs.atlassian.net/wiki/x/0wGNBw>

1. Click on the Actions (...) drop-down icon beside each environment entry on the Environment Listing screen, and choose the **Edit** option.
2. The pop-up prompts you for the following information:
 - **Application Name** – Dropdown menu displaying the names of applications for associating with the environment, for informative purposes.
 - **Environment Name** – The display name of the new environment.
 - **Purpose** – Read only, cannot be modified.
 - **Enable Approval Workflow (Database rulesets only)** – Whether or not to require approvals of inventories before masking jobs can be run in the environment. Applicable for Database rulesets only.
3. Click **Save** or **Save And View**.

6.1.8 Copying an environment

A user can also easily create an exact copy of a certain environment. This is a very powerful feature when wanting to have several similar but not exact environments but don't want to start from scratch. To copy an environment do the following:

1. Click on the Actions (...) drop-down icon beside each environment entry on the Environment Listing screen, and choose the **Duplicate** option.

2. The pop-up prompts you for the following information:
 - **Environment Name** – The display name of the new environment.
 - **Application Name** – Dropdown menu displaying the names of applications for associating with the environment, for informative purposes.
 - **Purpose** – Read only, cannot be modified.
 - **Enable Approval Workflow (Database rulesets only)** – Whether or not to require approvals of inventories before masking jobs can be run in the environment. Applicable for Database rulesets only.
3. Click **Duplicate** or **Duplicate And View**.

6.1.9 Deleting an environment

To delete an environment:

- Click on the Actions (...) drop-down icon beside each environment entry on the Environment Listing screen, and choose the **Delete** option.
- A confirmation dialog appears, on confirming the particular environment will be deleted.



Clicking the **Delete** icon deletes EVERYTHING for that environment: connections, inventory, rule sets, and so on. It does not delete universal settings like algorithms, domains, etc.

6.2 Managing remote mounts for VM continuous compliance engines

This section describes how you can mount an NFS/CIFS location inside the Continuous Compliance engine and use it in a masking job for engines deployed on virtual machines. For information on file mounts for containerized masking, please refer to the [Managing remote mounts for containerized masking](#)¹⁴⁴ page.

In order to access the files shared over NFS/CIFS server from the Masking Engine, complete the following two steps:

1. Create and connect a mount using Mount Filesystem API endpoint.
2. Create a file connector with Filesystem Mount Point mode or upload a XML/Copybook file format using Filesystem Mount Point mode.

¹⁴⁴ <https://masking.delphix.com/docs/latest/managing-remote-mounts-for-containerized-masking>

6.2.1 Mount filesystem API

The **Mount Filesystem** APIs are used to perform normal CRUD operations(Create, Read, Update, and Delete) along with three mount operations connect(mount), disconnect(unmount), and remount on a mounted object.

6.2.1.1 Mount information

To create a mount entry, information about the mount is passed. Some of them are required and some are optional.

- Required Information:
 - **mountName**: The name of the mount. This name is used to refer to this mount in the connector creation and file format upload UIs.
 - **hostAddress**: The NFS/CIFS server address.
 - **mountPath**: The remote path shared by the NFS/CIFS server. For a CIFS mount, this should be the path after the hostname/IP address, with any backslashes (\) replaced with a slash (/). For example, \\10.0.0.1\Share would be entered as /Share.
 - **type**: The type of the server. CIFS, NFS3, or NFS4.
- Optional Information:
 - **options**: The mount options.
 - **connectOnStartup**: Whether this mount should be connected or not when the server starts.



When a server shuts down, all the mounts are disconnected.

6.2.2 Mount options

The API supports passing many mount options. Not all of them are supported by a server. After a mount is connected, you might see the options field has many options that were not passed by you or some options that have been eliminated that were passed by you. The options field shows effective options only. The applied options are gathered after a mount is connected.

The API also restricts the usage of some mount options.

6.2.2.1 Enforced options

The following mount options are enforced and added to the list of options for all mounts:

- **nosuid**: The filesystem cannot contain set userid files.
- **noexec**: No executable script can be run from the mount.
- **nodev**: The filesystem cannot contain special devices.

6.2.2.2 Minimal options

Although `options` is an optional field, it is required for CIFS mounts to pass credentials. The following options are required for CIFS mounts:

- **username:** The username to connect to the CIFS server.
- **password:** The password of the user.
- **domain:** The domain of the user.

For example, `"options": "username=abc,password=pass,domain=DOMAIN"`

For NFSv3 mounts, `options` are not required, therefore can be `null`.

For NFSv4 mounts, the following option is required:

- **nfsvers:** The NFS protocol version number. For example, `"options": "nfsvers=4.0"`

6.2.2.3 Version options

The version information is passed using `vers` option. The supported versions based on mount types are

Mount Type	Supported Versions
CIFS	2.0, 2.1, 3.0
NFS3	3, 3.0
NFS4	4, 4.0, 4.1, 4.2

6.2.2.4 Generic options

Some mount options are generic which can be applied to all the mount types while some are mount specific options. In the case of *remount* operation, only generic options can be modified. The list of allowed generic options are:

`async`, `atime`, `auto`, `context`, `defaults`, `defcontext`, `diratime`, `dirsync`, `fscontext`, `group`, `iversion`, `lazytime`, `loud`, `mand`, `_netdev`, `noatime`, `noauto`, `nodev`, `nodiratime`, `noexec`, `nofail`, `noiversion`, `nolazytime`, `nomand`, `norelatime`, `nostrictatime`, `nosuid`, `nouser`, `owner`, `relatime`, `_rnetdev`, `ro`, `rootcontext`, `rw`, `silent`, `strictatime`, `sync`, and `user`.

6.2.3 CRUD operations

6.2.3.1 Create

The **create** endpoint is used to create a mount entry. It takes all the information about a mount as its input and creates a mount entry. It doesn't do any kind of validation about the mount's accessibility. The validation is done during the *connect* operation.

6.2.3.2 Read

The **read** endpoints are used to retrieve information about a mount. There are two *read* endpoints.

1. *get all*: To get information about all mounts.
2. *get*: To get information about any particular mount identified by its id.

6.2.3.3 Update

The **update** endpoint is used to modify any information of a mount. Update operation can be performed only on a disconnected mount.

6.2.3.4 Delete

The **delete** endpoint is used to delete a mount entry. A mount can be deleted only if it is not being used in any of the connectors.

6.2.4 Mount operations

Apart from normal CRUD operations, there are three special mount related operations exposed through the API.

6.2.4.1 Connect

The **connect** endpoint is used to mount a remote mount inside the masking engine. If the connect operation succeeds then, the options field is updated with the applied mount options.

6.2.4.2 Disconnect

The **disconnect** endpoint is used to unmount a remote mount from the Masking Engine.

6.2.4.3 Remount

The API supports the **remount** operation. This can be used to remount an active or to connect a disconnected mount and also to update some mount information. This can update *mountName*, *connectOnStartup* and generic *options* only. For other updates, use the normal update API.

6.2.4.4 Resolve mount consistency

A script runs in the background to keep the data in the *mount_information* table and mounts in sync. If for some reason, the data for a mount mounted inside the mount engine and data corresponding to that mount in *mount_information* table becomes inconsistent, the mount is unmounted. For example, if a mount is in a disconnected state in DB but it is mounted in the engine, then it will be unmounted.

6.2.5 Using mounts

A mount can be used while creating File connectors.

6.2.5.1 File connector

1. **Details:** While creating a connector, when any file source type is selected, the details step provides the connection mode dropdown. There are six options:
 - a. AWS S3
 - b. Other S3 Compatible Storage
 - c. Filesystem Mount Point
 - d. SFTP
 - e. FTP
 - f. FTPS (Mainframe file type only)

Create Connector



- Details
- Credentials
- Summary

Details

Specify details to identify this connector.

Connector Name
Fixed-width-connector

Select Source Type
File - Fixed

Select Connection Mode
AWS S3

- AWS S3 ✓
- Other S3 Compatible Storage
- File System Mount Point
- SFTP
- FTP

Cancel Back Next Test Connection Save

2 Details Step

- 2. **Credentials:** On selecting the Filesystem Mount Point option, the mount name and a path inside the mount need to be specified in the Credentials step.

Create Connection



- Details
- **Credentials**
- Summary

Credentials

Specify the credentials for Fixed Width connector.

Select Mount
mount_5LU0ZG6D

Path Under Mount
test/subdir/sub2

Cancel Back **Next** Test Connection Save

3 Credentials Step

3. **Summary:** Here you can view the following information for File System Mount Point:

- Connection Name: This is the name of the connection being created/edited.
- Mount Name: This is a list of mount names created in the engine.
- Remote Path: The complete remote path. On selecting a mount name and typing a path in the above input box, this gets updated.
- Path Under Mount: A path relative to the path mounted. By default, it is at the root of the remote Mount path.

Create Connection



- Details
- Credentials
- **Summary**

Summary

View the connector details below. Click the Back button to make changes or click the Save button to save Fixed Width connector.

Connection Summary

Connection Name
connector_test

Mount
mount_5LU0ZG6D

Remote Path
qa-sftp-7add-qar-130664-27e4593a.dlpxdc.co./var/tmp/masking-mount/test/subdir/sub2

Path Under Mount
test/subdir/sub2

Cancel Back Next Test Connection **Save**

4 Summary Step

- =
 A connector can be created even if a mount is in a disconnected state but it should be in an active state when a ruleset is being created or when a job is run.

6.2.5.2 Sync mounts

A mount can be synced from a source engine to a target engine using [Sync APIs](#)¹⁴⁵. Syncing a file connector using a mount also syncs the related mounts. The following mount information fields are synced:

- mountName
- hostAddress
- mountPath
- options
- connectOnStartup
- type

In case of CIFS mounts, the password is not synced. In order to set the password in the target engine, update the mount's options and ensure to include the password in the options.

6.2.6 Recommended mount server configuration

The NFS and CIFS servers should be configured in such a way that the files are readable and writeable by the Masking Engine.

6.2.6.1 CIFS server

The user-provided to connect to the mount should have read and write permission on the mount.

6.2.6.2 NFS server

1. The Masking Engine's server IP should have read and write permission on the mount.
2. For NFS, the access to a file is controlled based on the UID and GID. In order to give read & write permission to the Masking Engine on the share path, the path should be shared with the following options:

```
<mount path> <masking engine ip>(rw,all_squash,anonuid=<uid>,anongid=<gid>)
# uid and gid is of the owner of the shared path on the server
```

¹⁴⁵ <https://delphixdocs.atlassian.net/wiki/spaces/CC/pages/114560050>

6.3 Managing remote mounts for containerized masking

This section describes how to mount an NFS mountpoint inside the Containerized Masking Engine. For information on file mountpoints for Virtual Machine Masking, please refer to [Managing File Mounts](#) (see page 340).

In Containerized Masking, much more control is available to the admin at the Kubernetes layer. That advantage is used to simplify file systems mounts for Containerized Masking. This document will describe the process using NFS as an example mountpoint type.



Restriction

Filesystem mount points must be mounted as a subdirectory of `/var/delphix/masking/remote-mounts/`.



Restriction

In order for Kubernetes to utilize some particular network filesystem, the underlying host will typically need to be able to support that filesystem. In this example, to support mounting NFS filesystems, the underlying OS needs to be able to perform an nfs mount. This is typically enabled by installing the `nfs-client` package. For example, if the kubernetes cluster runs on top of a debian-type linux distro, the package would need to be installed using `apt install nfs-client` on each node to ensure all nodes have the necessary utilities to handle mounting NFS filesystems.

6.3.1 Creating the mountpoint connection in Kubernetes

To establish a remote mount using NFS, the first step is creating the NFS connection to the remote NFS host. This is accomplished utilizing a special NFS persistent volume. This can be added to the beginning of the `kubernetes-config.yaml` file or created as separate config files just for this purpose. If separate config files are created, they will have to be applied before the main Pod config is applied.

Both a Persistent Volume (PV) and Persistent Volume Claim (PVC) are necessary and the YAML for each of these looks like the following snippets.

6.3.1.1 NFS Persistent Volume YAML

```
apiVersion: v1
  kind: PersistentVolume
  metadata:
    name: nfs-pv
```

```

spec:
  capacity:
    storage: 500Mi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  storageClassName: nfs-storage
  mountOptions:
    - hard
    - nfsvers=4.1
  nfs:
    server: <your NFS server host>
    path: <the exported directory on the NFS server, for example /var/tmp/
masking-mount>

```

6.3.1.2 NFS persistent volume claim YAML

```

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: nfs-pvc
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  storageClassName: nfs-storage
  resources:
    requests:
      storage: 500Mi # change corresponding to actual requirements

```

6.3.2 Using the mountpoint in the pod configuration

Next, the recently created NFS PVC must get mounted into the application container. This is achieved by editing the existing Pod config YAML and adding 2 objects. First, attaching the PVC to the pod as a volume. Second, linking that volume into the application container. This is demonstrated in the excerpt below.

6.3.2.1 Excerpt of kubernetes-config.yaml to show support for NFS volumes

```

#
# Example of volume definition per Persistent Volume Claim
#
volumes:
  - name: nfs-pv-storage
    persistentVolumeClaim:
      claimName: nfs-pvc

```

```


containers:
  - image: delphix-masking-app:6.0.16.0-c1
    name: app
    ports:
      - containerPort: 8284
        name: http
    volumeMounts:
      - name: masking-persistent-storage
        mountPath: /var/delphix/masking
        subPath: masking
      - name: masking-persistent-storage
        mountPath: /var/delphix/postgresql
        subPath: postgresql
      #
      # Example of mounting an external volume
      #
      # Mount path is the directory on the `app` container to be mounted to the
      # remote provided Persistent Volume.
      # It should always start with the `/var/delphix/masking/remote-mounts`
      # and to be followed with customer named sub-directory per mount.
      # That sub-directory will automatically be created on the Masking Engine
`app` container.
      #
      - name: nfs-pv-storage
        mountPath: /var/delphix/masking/remote-mounts/nfs_example

```

6.3.3 Using the mountpoint in the UI

Once a properly configured Pod is started, the configured NFS filesystem can be accessed in the UI using the same process that was previously used for non-containerized instances documented in [Managing Remote Mounts for VM Masking Engines \(see page 340\)](#). The one sticking point is that these mount points (in the dropdown list) by default are named "mountpoint_1", "mountpoint_2", etc.

It is possible to rename the default mount point names to something more friendly. This is done via the `PUT /mount-filesystem/{mountID}` API endpoint.

-  The `/mount-filesystem` API has a large set of functionality that is used to manage filesystem mounts in the Virtual Machine deployment of the Masking Engine. For Containerized Masking, most of that functionality is handled by Kubernetes itself rendering the API tasks useless and therefore disabled. The only functionality available in Containerized is the endpoint that allows you to update an existing mount and only to update its name.

6.3.4 Other types of filesystem mountpoint

The above example has used NFS, but it is possible to mount any filesystem that Kubernetes will support. To mount CIFS or some other supported remote filesystem is possible so long as the same general procedure is followed including:

- creating the various Kubernetes objects (such as the PV and PVC)
- mounting it under the `/var/delphix/masking/remote-mounts/` required path

6.3.5 Known limitations

- You can't configure mount point manually (i.e. using API endpoints). Only mount points provided by Kubernetes will be detected.
- Customized mount points can't be synced from Appliance Masking Engine. If the sync bundle contains any mount point created via API - importing that bundle to containerized Masking Engine will fail.
- Masking Sync is incapable of altering your various Kubernetes config YAML files which is the only way to mount a filesystem in Containerized Masking.
- You can't edit existing mount points at containerized Masking Engine.
- Mount points are named automatically by Masking Engine
- You can delete (via API `mountFilesystem`) only those mount points which are not provided by Kubernetes (for example were synced in), and not associated with any existing connector.

6.3.6 Local file masking troubleshooting

If Masking Engine is not responsive at `<your-masking-engine-URL>:30080/masking` - there might be need to troubleshoot. If you are not sure what's the name of the masking pod you can find all pods in the given Kubernetes's cluster by running **`kubectl get pod`** command. The one with the word **masking** will be the desired pod. If multiple masking pods are run on the same instance - look for

`delphix-masking-*` names, Pod status could be seen by running **`kubectl get pod <your-masking-pod-name>`**. If not all 3 containers are in the running status - let's get the description of the pod: **`kubectl describe pod delphix-masking-0`**. In the output of the above command there is the health information for each container, their status and the latest errors that prevented the pod from a successful startup. Most probably those errors will give a hint on what went wrong. If Masking Engine was working fine prior to adding the Local File Masking configuration, the error reasons could be (but not limited to):

- the configured remote Persistent Volume is not accessible
- the directory configured for remote Persistent Volume doesn't exist
- the yaml files entries you've added are not correctly indented (yaml files are indentation sensitive).
After fixing the found problem and tearing down all created Kubernetes instances (in the opposite order) - start applying those again.

If Masking Engine application is up and running, but the configured masking job fails - verify the write permissions are granted to the masking target directory (on the corresponding mounted Persistent Volume).

6.4 Managing SSL/TLS over JDBC for containerized masking

On the VM instance, we use the Virtualization Engine's Setup App to manage certificates and trust stores for SSL/TLS needs. Since Containerized Masking Engine runs alone - we need to provide another way of creating the truststore and storing the SSL certificate. There are multiple options of establishing truststore on linux container. Below is an example of using Kubernetes for this purpose.

- uploading the saved certificate to configmap
- mounting that configmap as volume
- creating a truststore and uploading there the configured certificates

6.4.1 Prerequisites

Database is configured with SSL listener. To establish the SSL/TLS connection over JDBC we should know:

- database URL,
- SID,
- SSL listener port,
- SERVICE_NAME (for database service where SSL listener is enabled)
- SSL_SERVER_CERT_DN (SSL server certificate distinguished name) - could be found from the generated certificate, for example by using the openssl utility:

```
openssl x509 -in ssl_cert.crt -text
```

Here **ssl_cert.crt** is a name of the file containing the desired certificate (the one that was copied from the Database).

6.4.2 Create configmap entry based on database provided SSL/TLS certificate

1. save SSL/TLS certificate as .crt file.
2. use Kubernetes command to create a configmap, for example:

```
kubectl configmap ora-18 --from-file=ssl_cert.crt
```

Here **ora-18** is the name of the created configmap entry, **ssl_cert.crt** file contains the SSL/TLS certificate. To verify that configmap entry is added to the pod instance run the following command:

```
kubectl get configmap
```

6.4.3 Mount the configured configmap as volume

Add configmap entry as a volume to the pod instance in its config .yaml file. If you already have other volumes defined that new entry can go under the existing volumes section. If not create a **volumes:** section as shown below:

```
volumes:
  - name: ora-ssl-cert-volume
    configMap:
      name: ora-18
```

Here *ora-ssl-cert-volume* is a name for the provided volume, *ora-18* is the name of the previously created configmap entry.

Now we are ready to mount that volume to **app** container. Under the **containers:** section of the pod's config .yaml file, find the **app** container and add another entry to its **volumeMounts:** as shown below:

```
- name: ora-ssl-cert-volume
  mountPath: /var/delphix/ssl/ssl_cert.crt
  subPath: ssl_cert.crt
```

Here **ora-ssl-cert-volume** is a pod level provided volume, **ssl_cert.crt** is a name of the certificate file (originally provided by the configured configmap).

If using multiple SSL/TLS certificates - the above steps to be repeated for each certificate.



Attention!

The used mountPath `/var/delphix/ssl/` is a preconfigured location on the app container where certificates should be stored! That's where the truststore will look for customer provided certificates.

6.4.4 Create trust store and upload all mounted SSL/TLS certificates

We suggest using Kubernetes's lifecycle postStart hook to create the truststore and load the certificates:

In the pod's config .yaml file in the **containers:** section, find the **app** container and add to a lifecycle section to contain a **postStart:** hook as shown below

```
name: app
  lifecycle:
    postStart:
      exec:
```

```
command: ["/bin/bash", "-c", "for filename in /var/delphix/ssl/*.crt;
do keytool -import -trustcacerts -keystore /var/delphix/ssl/.masking_certs -storepass
changeit -noprompt -alias $(basename \"$filename\" .crt) -file \"$filename\"; done"]
```

Here we use the keytool utility to create the truststore `/var/delphix/ssl/.masking_certs` and to load all the mounted certificates found in the `/var/delphix/ssl/` directory.

6.4.5 Configure SSL/TLS over JDBC connector

Now any required SSL/TLS certificates are uploaded to the truststore on Containerized Masking Engine. We can use them to establish the JDBC connection. In the connector settings for the advanced Oracle database connector the URL to be configured as following:

```
jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=<your oracle DB URL>)
(PORT=<port where SSL listener is configured>))(CONNECT_DATA=(SERVICE_NAME=<service
name>))(SECURITY=(SSL_SERVER_CERT_DN="<distinguished name of the SSL certificate>")))
```

6.4.6 SSL/TLS over JDBC troubleshooting

1. verify the file contains the exact SSL/TLS certificate (copied from the DB). It should look like:

```
-----BEGIN CERTIFICATE-----
MIIBkDCB+gIBADANBgkqhkiG9w0BAQQFADARMQ8wDQYDVQQDEwZiYmRoY3AwHhcNMjIwOTAxMDA0
...
uVWk84o=
-----END CERTIFICATE-----
```

1. verify the certificate is mounted under the correct `/var/delphix/ssl/` directory.
2. verify the certificate is uploaded to the truststore by logging into the bash on the app container and checking truststore exists and how many certificates are loaded:

```
keytool -list -keystore /var/delphix/ssl/.masking_certs -v
```

1. if **app** container didn't start - most probably the mount was not configured correctly. Check the pod description for errors:

```
kubectl describe pod delphix-masking-0
```

Particularly check for indentation issues in the YAML entries because Kubernetes is very sensitive to indentation.

6.5 Managing connectors

Connectors are used to define data sources that the Continuous Compliance Engine can connect to and are grouped within environments. To navigate to the **connectors** screen, click on an environment and then click the **Connectors** tab.

The screenshot shows the Continuous Compliance interface for environment 'Env1'. The 'Connectors' tab is selected, displaying a table of connectors. The table is organized into three groups based on the Meta Data Source: DATABASE (3 connectors), FILE (5 connectors), and MAINFRAME (1 connector). Each connector row includes its Connector ID, Name, Type, and an Actions menu (three dots).

Meta Data Source	Connector Id ↑	Name	Type	Actions
▼ DATABASE (3)				
	16	MySQLDBConnector ⚠	Mysql	⋮
	17	LocalhostPostgresDB	Postgres	⋮
	18	PostgresDBConnector	Postgres	⋮
▼ FILE (5)				
	18	FixedFileConnector ⚠	Fixed Width	⋮
	19	DelimitedFileConnector ⚠	Delimited	⋮
	20	XMLFileConnector ⚠	XML	⋮
	21	WholeFileConnector ⚠	Fixed Width	⋮
	22	JSONFileConnector	Json	⋮
▼ MAINFRAME (1)				
	1	MainFrameFileConnector	Dataset	⋮

The **Connectors** screen contains the following information and actions:

- **Meta Data Source:** The source of the connector. E.g. Database, File, or Mainframe. The rows are grouped by Meta Data Source.
- **Connector ID:** The numeric ID of the connector used to refer to the connector from the masking API.
- **Name:** The name of the connector.
- **Type:** The specific type of connector.
- **Edit:** Edit the connector. See more details below.
- **Test Connection:** Test the connector. See more details below.
- **Delete:** Delete the connector. See more details below.

The rows on the screen can be filtered or sorted by **Connector ID**, **Name** and **Type**(filtering only) within each Meta Data Source group by clicking on the respective field. More information on grid filtering and sorting can be found [here](#)¹⁴⁶.



Missing Authentication

If an authentication is missing, an aptly named warning icon will show next to **Name**. Hover over the icon to identify if it's missing the password, SSH key, credential path, or access keys.



For more information about connectors including features, versions and TLS setup, please refer to [Data Source Support](#) (see page 154).

6.5.1 Database connectors

The fields that appear are specific to the **DBMS Type** you select. If you need assistance determining these values, please contact your database administrator.


- **Schema Name:** The schema that contains the tables that this connector will access.
- **Database Name:** The name of the database to which you are connecting. **Note:** the database name field is case-sensitive. It must match exactly with the name of the current database as known to the instance.
- **Host Name/IP:** The network hostname or IP address of the database server.
- **Port:** The TCP port of the server.
- **SID:** (Oracle only) Oracle System ID (SID).
- **Instance Name:** (MSSQL Server only) The name of the instance. This is optional. If the instance name is specified, the connector ignores the specified "Port" and attempts to connect to the "SQL Server Browser Service" on port 1434 to retrieve the connection information for the SQL Server instance. If the instance name is provided, be sure to make exceptions in the firewall for port 1434 as well as the particular port that the SQL Server instance listens to.
- **Custom Driver Name:** (Generic only) The name of the JDBC driver class, including Java package name.
- **JDBC URL:** (Generic and Advanced connector mode for Oracle, MS SQL Server, and Sybase only) The custom JDBC URL, typically including hostname/IP and port number.


¹⁴⁶ <https://masking.delphix.com/docs/latest/graphical-user-interface>

6.5.2 Database connector properties

6.5.2.1 Getting properties

To retrieve all properties set on the connector, make a request to the `GET database-connector/{id}/properties` endpoint. This endpoint will respond with all default properties set by the driver, superimposed by any properties specified by an uploaded connection properties file. If a properties file is uploaded for a connector, this list can also be viewed through the UI on the database connector form, where you can sort by `Property`, `Value`, or `Modified`. The `Modified` field signifies whether the property value is the default or modified by the uploaded properties file.

-  The database name field is case-sensitive. It must match exactly with the name of the current database as known to the instance.

-  Only a valid JDBC URL is required to retrieve properties of a connector; a valid connection to the database server is not necessarily required.

Edit Connection



- Details
- Credentials
- Custom Properties
- Summary

Custom Properties

Specify connection properties(Optional) for Postgres connector.

Connection Properties ?

Choose File

Property	Value	Modified
localSocketAddress		False
adaptiveFetch	false	False
useSpnego	false	False
quoteReturningIdentifiers	true	False
logUnclosedConnections	false	False
binaryTransfer	true	False
sslpassword	REDACTED	False

Displaying 16 to 23 of 78

6.5.2.2 Setting properties

Properties can sometimes be set through the JDBC URL or through a connection properties file. Customizing the JDBC URL is limited to **Advanced**, **Generic**, and **Extended** Connectors, while uploading a properties file is supported by all database connectors. All properties files must have the extension `.properties` and must adhere to Java properties file syntax. Even if a property specified in the properties file is not technically supported by the JDBC driver, it will still be passed along to the driver when building the JDBC Connection. All provided and unsupported properties will be logged whenever the properties file is loaded.



The properties file is assumed to be written using ISO 8859-1 character encoding.

6.5.2.3 Security considerations

The property key or value provided in a database connector's properties file will not be regulated and is subject to any user with `CREATE` or `UPDATE` connector privileges. This means that even supported sensitive properties such as `user`, `password`, `hostname`, etc. will be available in plain text to anyone with the `VIEW` connector privilege.

If possible, specify sensitive properties through relevant form fields which will be obfuscated in all places or through the JDBC URL which will still be visible in plain text to any user with the `VIEW` connector privilege but will be redacted in support bundles.

6.5.3 File connectors

The following values appear when any of the file connector types are selected:

- **Connector Name:** The name of the file connector (specific to your Delphix application and unrelated to the file itself).
- **Connection Mode:** Filesystem Mount Point, SFTP, FTP, FTPS (only for mainframe datasets), AWS S3 and Other S3 Compatible Storage.



Due to networking complications in containerized masking, FTP and FTPS is currently disabled in containerized deployments. Delphix is researching options to re-enable FTP (for containerized masking) at a future date.

Create Connector



- **Details**
- Credentials
- Summary

Details

Specify details to identify this connector.

Connector Name

Select Source Type

Select Connection Mode

- AWS S3 ✓
- Other S3 Compatible Storage
- File System Mount Point
- SFTP
- FTP

5 Details Step

Create Connection



- Details
- Credentials
- Summary

Credentials

Specify the credentials for Fixed Width connector.

Port	22
Server Name	some-host.example.com
Path	/data/files
<input type="checkbox"/> User Directory as root	
Select Authentication Type	Username/Password ▼
Username	username
Password

6 Credentials Step

The rest of the values appear based on the selected **Connection Mode** value.

For **Filesystem Mount Point** connection mode, refer to the corresponding section on the [Managing Remote Mounts \(see page 340\)](#) page.

For **AWS S3 & Other S3 Compatible Storage** connection modes, refer to the corresponding section on the [Amazon Simple Storage Service \(S3\) connector for files¹⁴⁷](#) page.

For **SFTP, FTP & FTPS** connection modes, the following values appear under the credential step:

- **Path:** The path to the directory where the file(s) are located.
- **Server Name:** The name of the server used to connect to the file.
- **Port:** The port used to connect to the server.
- **User Name:** The user name to connect to the server.
- **Password:** (non-Public Key Authentication only) The associated password for the server.

¹⁴⁷ <https://delphixdocs.atlassian.net/wiki/x/WIBZBw>

- **Public Key Authentication:** (Optional) (Only appears for SFTP) Select this from 'Select Authentication Type' dropdown to specify a public key. Use the following instructions in this [Knowledge base](#)¹⁴⁸ article to complete.

Create Connection ✕

- Details
- **Credentials**
- Summary

Credentials

Specify the credentials for Fixed Width connector.

Port
22

Server Name
some-host.example.com

Path
/data/files

User Directory as root

Select Authentication Type
Username/Password

Username
username

Password
.....

Cancel Back **Next** Test Connection Save

7 User Directory as root



If you plan to do on-the-fly masking, you will need to create a separate environment and connector as the source for the files to be masked. The masked files will be put into the directory being pointed to by the connector created previously (the target). However, the file

¹⁴⁸[https://support.delphix.com/Continuous_Compliance_Engine_\(formerly_Masking_Engine\)/Managing_Public_Key_Infrastructure_for_use_with_Masking_SFTP_Connectors_\(KBA1850\)?mt-deflection=true&mt-query=Unable%20to%20Create%20Valid%20SSH%20Key](https://support.delphix.com/Continuous_Compliance_Engine_(formerly_Masking_Engine)/Managing_Public_Key_Infrastructure_for_use_with_Masking_SFTP_Connectors_(KBA1850)?mt-deflection=true&mt-query=Unable%20to%20Create%20Valid%20SSH%20Key)

path specified in the connector of the target rule set must point to an existing file the target directory. It does not have to be a copy of the file, just an entry in the directory with the same name. It will be replaced by the masked file.

Starting with version 6.0.9.0 the SFTP mode is extended with the **User Directory as root** flag. If the Path defined is relative to the User-home-dir as configured on the SFTP Server, select the flag below.

If the connector is configured via the API then that flag is accessible as `userDirIsRoot`, for example:

```
{
  "connectorName": "Test SFTP Connector",
  "environmentId": 2,
  "fileType": "DELIMITED",
  "connectionInfo": {
    "connectionMode": "SFTP",
    "path": "/delimited",
    "host": "yourSFTPServer",
    "loginName": "xxxxx",
    "password": "xxxxx",
    "port": 22,
    "userDirIsRoot": true
  }
}
```

Limitations

- S3 Role-based connectivity is restricted to Continuous Compliance instances hosted on AWS EC2. Connectivity attempts from outside the AWS environment using this method will result in connection failures.

6.5.4 Create, test, edit, and delete a connector


6.5.4.1 Create a connector

On the top-right corner of the **Connectors** page, click on the **+ Connection** button. The **Create Connection** wizard appears.

1. **Details:** input the following information:

- **Connector Name:** The name of the connector.
- **Source Type:** The type of data source (Oracle, Sybase, etc.). Each source type in this dropdown is preceded by the Meta Data Source (Database, File, or Mainframe).
- **Connection Mode** (Oracle, MSSQL Server, and Sybase only) Choose a connection type:

- **Basic:** Basic connection information.
- **Advanced:** The full JDBC connect string including any database parameters.

 Basic & Advanced mode is not available for all source types.

Create Connection



- Details**
- Credentials
- Custom Properties
- Summary

Details

Specify details to identify this connector.

Connection Name
Sybase Test

Select Source Type
Database - Sybase

Select Connection Method

- Hostname/IP (Basic) JDBC URL (advanced)

Cancel Back **Next** Test Connection Save

8 Details Step

2. **Credentials:** Provide the connection information pertaining to the source type selected.
 - a. The fields that appear are specific to the **Source Type** you selected. If you need assistance determining these values, please contact your database administrator.

Create Connection



- Details
- **Credentials**
- Custom Properties
- Summary

Credentials

Specify the credentials for Sybase connector.

Schema Name

Database Name

Host

Port

Select Authentication Type

Username

Password

9 Credentials Step

- Custom Properties:** Upload the properties file (typically a Java properties file) to specify configurations for the JDBC connection.

Create Connection



- Details
- Credentials
- Custom Properties
- Summary

Custom Properties

Specify connection properties(Optional) for Sybase connector.

Connection Properties ⓘ

Choose File

Property	Value	Modified
ⓘ No items to display		

Cancel Back Next Test Connection Save

10 Custom Properties Step

- Summary:** This step provides comprehensive details of all the configurations provided. After verification of the present information, click **Save**.

Create Connection



- Details
- Credentials
- Custom Properties
- **Summary**

Summary

View the connector details below. Click the Back button to make changes or click the Save button to save Sybase connector.

Connection Summary	
Connection Name	Sybase Test
Schema Name	dbos
Database Name	potateos
Host	sybase-host.example.com
Port	4000
Username	username

[Cancel](#) [Back](#) [Next](#) [Test Connection](#) [Save](#)

11 Summary Step

6.5.4.2 Editing a connector

Perform the following steps to edit a connector:

1. In the **Connectors** tab, click on either **Edit** under the action dropdown or directly click on the connector name link of the connector you want to edit on the connector grid.

The screenshot shows the 'demo-env' environment page with the 'Connectors' tab selected. The table below lists the connectors:

Meta Data Source	Connector Id	Name	Type	Actions
DATABASE (1)	1	local-postgres	Postgres	...
FILE (1)	1	NewDel	Delimited	...
MAINFRAME (1)	1	Mainframe	Dataset	...

The 'local-postgres' connector is highlighted with a red arrow. A dropdown menu is open for this connector, showing the following actions:

- Edit
- Test Connection
- Delete

- 2. Change any information necessary across its steps. For more information on the fields refer [Database Connector Fields](#) (see page 0).

Edit Connection



- Details
- **Credentials**
- Custom Properties
- Summary

Credentials

Specify the credentials for Postgres connector.

Schema Name
public

Database Name
potateos

Host
some-host.example.com

Port
5432

Select Authentication Type
Username/Password

Username
username

Change Password

Cancel

Back

Next

Test Connection

Save

3. To change the password, select the checkbox next to **Change Password**
4. To change connection properties or upload a new one, select **Choose File** and upload the properties file in the Custom Properties Step.

Edit Connection



- Details
- Credentials
- Custom Properties
- Summary

Custom Properties

Specify connection properties(Optional) for Postgres connector.

Connection Properties ⓘ

Choose File

Property	Value	Modified
localSocketAddress		False
adaptiveFetch	false	False
useSpnego	false	False
quoteReturningIdentifiers	true	False
logUnclosedConnections	false	False
binaryTransfer	true	False
sslpassword	REDACTED	False

Displaying 16 to 23 of 78

- Cancel
- Back
- Next
- Test Connection
- Save

12 Custom Properties Step

5. Click **Save** or verify all details in the summary step then click **Save**.

6.5.4.3 Testing a connector

Testing a connector can be done while creating/editing the connector or from the grid.

Information
The Inventory Tab has been removed from this screen. To access the inventory of a rule set, click on the hyperlink associated with its name on the Rule Set listing screen.

Environments > demo-env
demo-env
Application: app1 | Purpose: MASK | Approval: Disabled

+ Connector

Jobs Rule Sets **Connectors**

Meta Data Source	Connector Id ↑	Name	Type	Actions
▼ DATABASE (1)	1	local-postgres	Postgres	...
▼ FILE (1)	1	NewDel	Delimited	...
▼ MAINFRAME (1)	1	Mainframe	Dataset	...

- Edit
- Test Connection
- Delete

Edit Connection



- Details
- **Credentials**
- Custom Properties
- Summary

Credentials

Specify the credentials for Postgres connector.

Schema Name
public

Database Name
potatoes

Host
some-host.example.com

Port
5432

Select Authentication Type
Username/Password

Username
username

Change Password

Cancel Back **Next** Test Connection Save



13 Test Connection on Wizard

Test Connection in the wizard allows you to test the connection anytime, to check the validity of the connector information that was added.

When you click **Test Connection**, Delphix uses the information from the wizard to attempt a database connection. When finished, a status message appears indicating success or failure.

6.5.4.4 Deleting a connector

To delete a connector, click **Delete** under the action icon. When you delete a connector, you also delete its rule sets and inventory data.

The screenshot shows the 'demo-env' environment page with the 'Connectors' tab selected. A table lists connectors grouped by 'Meta Data Source'. A context menu is open over the 'local-postgres' connector, with the 'Delete' option highlighted by a red arrow.

Meta Data Source	Connector Id	Name	Type	Actions
DATABASE (1)	1	local-postgres	Postgres	...
FILE (1)	1	NewDel	Delimited	
MAINFRAME (1)	1	Mainframe	Dataset	

The dialog box is titled 'Delete Connector' and contains the text: 'Are you sure you want to delete "LocalhostPostgresDB" Connector?'. At the bottom, there are two buttons: 'Cancel' and 'Confirm'.

14 Delete Connector

On clicking **Delete**, it will prompt for confirmation. Click on **Confirm** to delete the connector.

6.5.5 Authentication types

Authentication type dropdown allows you to select the type of authentication you want to provide for the source type selected, the options available here is also determined by the source type selected.

Select Authentication Type

Username/Password

Username/Password ✓

Kerberos Authentication

Password Vault

15 Authentication Type Dropdown

6.5.5.1 Username/Password

- **Username:** The user login this connector will use to connect to the database (not applicable for Kerberos Authentication).
- **Password:** The password associated with the Login ID or Username (this password is stored encrypted).

6.5.5.2 Kerberos authentication

Before Kerberos may be used to authenticate the database, the appliance must be properly configured (refer to the [Kerberos configuration instructions](#) (see page 277)). If this authentication is selected, the application authenticates with the Kerberos KDC before connecting to the database, then uses its Kerberos credentials to authenticate to the database instead of a login/password. When Kerberos is enabled, the **Login ID** field is treated as the Kerberos user principal name.

- **Password:** If supplied, is used to authenticate the user principal with the KDC. The password field may be left blank if the keytab set during appliance configuration contains keys for the user principal.
- **Principal Name:** The name of the Kerberos user principal to use when authenticating with the KDC. The realm portion of the principal may be omitted if it matches the configured default realm.
- **Service Principal:** (Sybase only) The name of the Sybase service instance.

 Kerberos functionality has been disabled in containerized masking.

6.5.5.3 Password Vault

- **Password Vault:** Whether to use a password vault to authenticate to the database instead of a login ID and password. Before a password vault may be used, it must be properly configured. If this authentication type is selected, the selected Credential Path is used to obtain database credentials from the password vault it references.
- **Credential Path:** The path to credentials in a password vault to use for database authentication in lieu of a login ID and password.

6.5.6 Mainframe MVS storage access

The Continuous Compliance Engine can connect to a wide range of host systems, including Mainframes, Linux, and Windows, as long as they support SFTP, FTP, and FTPS protocols. Furthermore, mainframe MVS storage can be accessed using the FTP and FTPS protocol. In order to access the mainframe data set and effectively mask its contents, you must adhere to the z/OS path name guidelines described below. Failing to meet these criteria will result in the user being directed towards the USS storage, rather than the mainframe MVS storage.

- A data set name consists of one or more parts connected by periods. Each part is called a qualifier.
- Each qualifier must begin with an alphabetic character (A to Z) or the special character @, #, or \$.
- The remaining characters in each qualifier can be alphabetic, special, or numeric (0 to 9) characters.
- Each qualifier must be 1 to 8 characters in length.
- The provided path or qualifier should end with a period dot (.).



Masking large datasets in-place on mainframe MVS storage

For in-place jobs involving large datasets that exceed the configured system default size of the MVS storage, users should manually create a dataset with the same name as the input dataset, appending a **.msk** extension. This newly created dataset should have a larger size than the input dataset and must be allocated before executing the job.

6.5.6.1 Valid connection with Mainframe

Edit Connection



- Details
- **Credentials**
- Summary

Credentials

Specify the credentials for Mainframe connector.

Port
22

Server Name
some-host.example.com

Path
DATA.FILES

Username
username

Change Password

Success
Connection Succeeded


- Close
- Cancel
- Back
- Next
- Test Connection
- Save

6.5.6.2 Invalid connection with Mainframe

Edit Connection



- Details
- Credentials
- Summary

 Connection Failed: sample-host.example.com: Name or service not known

Credentials

Specify the credentials for Mainframe connector.

Port
21

Server Name
sample-host.example.com

Path
/VALBMCN3

Username
delphix

Change Password

Cancel Back **Next** Test Connection Save

16 As described in the banner at the top, the request is failing due to an error with the Server Name address. Verify that the server name is correct and that the server is accessible from your network.

Edit Connection



- Details
- Credentials
- Summary

Connection Failed: Expecting / to follow the hostname in URI "ftp://sample-host.example.com:21VALBMCN3".

Credentials

Specify the credentials for Mainframe connector.

Port
21

Server Name
sample-host.example.com

Path
VALBMCN3

Username
delphix

Change Password

Cancel Back Next Test Connection Save

17 As described in the banner at the top, the request is failing due to an error with the Path format, indicating that a slash must follow the hostname URI.

6.5.6.3 FTPS Connector for Mainframe storage

FTPS is an encryption protocol like FTP with support for TLS/SSL to establish a secure and encrypted channel for data transfer between the Continuous Compliance Engine and the mainframe storage. This added layer of security ensures that all data exchanged remains confidentially and tamper-proof.

Create Connection



- Details**
- Credentials
- Summary

Details

Specify details to identify this connector.

Connection Name
test-mainframe

Select Source Type
Mainframe - Dataset

Select Connection Mode
FTPS

Cancel Back **Next** Test Connection Save

Create Connection



- Details
- Credentials**
- Summary

Credentials

Specify the credentials for Mainframe connector.

Port
21

Server Name
mainframe-native.example.com

Path
MASKING.

Username
username

Password
.....

Cancel Back **Next** Test Connection Save

If the connector is configured via the API, then in connection mode "FTPS" needs to be passed, for example:

```
{
  "connectorName": "MAINFRAME_FTPS",
  "environmentId": 1,
  "connectionInfo": {
    "connectionMode": "FTPS",
    "path": "MASKING.",
    "host": "zos.example.com",
    "loginName": "maskinguser",
    "password": "maskingpwd",
    "port": 21,
    "userDirIsRoot": false
  }
}
```

The FTPS connection mode is specifically designed and available exclusively for mainframe storage systems. This connection mode is not available with any other connectors.

To establish a successful connection to the mainframe using FTPS, it is important that users upload the Server SSL Certificate to the TrustStore of the Data Engine. Detailed instructions for adding the certificate to the Data Engine TrustStore can be found in the [TrustStore settings](#)¹⁴⁹ page.

6.5.6.4 Invalid/no certificate


The Continuous Compliance Engine experiences connection errors while trying to establish a connection with the mainframe USS storage. These errors come when either incorrect or invalid certificates are provided or when no certificates have been uploaded to the TrustStore of the Data Engine.

¹⁴⁹ <https://cd.delphix.com/docs/latest/truststore-settings>

Edit Connection



- Details
- Credentials
- **Summary**

 Connection Failed: Unable to find valid certificate in engine Truststore. Please upload the valid mainframe server certificate.

Summary

View the connector details below. Click the Back button to make changes or click the Save button to save Mainframe connector.

Connection Summary	
Connection Name	MainFrameFileConnector
Server Name	zos.example.com
Port	21
Path	files
Username	username

Cancel Back Next Test Connection Save

18 Connection failed when an Invalid or No SSL Certificate was provided

6.5.7 Amazon Simple Storage Service (S3) connector for files

The Continuous Compliance Engine supports connecting to files and mainframe datasets stored in S3 with either of the following connectors:

- **AWS S3 Connector:** Use this connector for native AWS S3 storage.
- **Other S3 Compatible Storage Connector:** Use this for object storage from vendors that support the S3 protocol (e.g., GCP).

Create Connector



- Details
- Credentials
- Summary

Details

Specify details to identify this connector.

Connector Name

Fixed-connector

Select Source Type

File - Fixed

Select Connection Mode

AWS S3

AWS S3

Other S3 Compatible Storage

File System Mount Point

SFTP

FTP

Cancel

Back

Next

Test Connection

Save

6.5.7.1 Configuring an AWS S3 Connector

1. **connectorName**: Specifies the name of the connector.
2. **environmentId**: Indicates the identifier of the environment where the connector will be configured.
3. **fileType**: Denotes the type of files to be managed by the connector.
4. **connectionInfo**: This section contains details necessary for establishing a connection to the S3 service.
 - **connectionMode**: Specifies the mode of connection, which is set to "AWS_S3" indicating that it connects to an S3 bucket.
 - **prefix**: Indicates the prefix to be used for identifying files within the S3 bucket.
 - **delimiter**: Specifies the delimiter used in the file paths within the S3 bucket.
 - **awsRegion**: Specifies the AWS region where the S3 bucket is located.
 - **awsBucketName**: Specifies the name of the S3 bucket to connect to.

- **awsAuthType:** The Continuous Compliance Engine offers support for connecting to S3 through two authentication methods: AWS secret-based authentication(**AWS_SECRET**) and AWS Roles based authentication(**AWS_ROLE**).

Secret-based authentication requires:

- **awsAccessKey:** The access key is a Alphanumeric string that uniquely identifies your AWS account.
- **awsSecretKey:** The secret key is associated with your access key and is used for signing requests to AWS services.

For more information related to prefix and delimiter, please refer to the [Amazon Simple Storage Service documentation](#)¹⁵⁰.

Create Connection



- Details
- Credentials**
- Summary

Credentials

Specify the credentials for Fixed Width connector.

Select Authentication Type

Cancel
Back
Next
Test Connection
Save

19 AWS S3 Connection Mode

¹⁵⁰ <https://docs.aws.amazon.com/AmazonS3/latest/userguide/using-prefixes.html>

6.5.7.1.1 Sample payloads

- **AWS Secret**

To connect to S3, Continuous Compliance requires a user's AWS Access Key and Secret Key for authentication.

```
{
  "connectorName": "JSON S3",
  "environmentId": 37,
  "fileType": "JSON",
  "connectionInfo": {
    "connectionMode": "AWS_S3",
    "prefix": "json/",
    "delimiter": "/",
    "awsRegion": "us-west-2",
    "awsBucketName": "masking-bucket-name",
    "awsAuthType": "AWS_SECRET",
    "awsAccessKey": "<Your AWS Access Key>",
    "awsSecretKey": "<Your AWS Secret Key>"
  }
}
```

- **AWS Role**

To establish secure communication between a masking engine hosted on EC2 instance and S3, we leverage instance profiles. This approach eliminates the need for static access keys and enhances security by dynamically providing temporary credentials.

```
{
  "connectorName": "JSON S3",
  "environmentId": 37,
  "fileType": "JSON",
  "connectionInfo": {
    "connectionMode": "AWS_S3",
    "prefix": "json/",
    "delimiter": "/",
    "awsRegion": "us-west-2",
    "awsBucketName": "masking-bucket-name",
    "awsAuthType": "AWS_ROLE"
  }
}
```

6.5.7.2 Configuring Other S3 Compatible storage Connector

1. **connectorName**: Specifies the name of the connector.
2. **environmentId**: Indicates the identifier of the environment where the connector will be configured.

3. **fileType**: Denotes the type of files to be managed by the connector.
4. **connectionInfo**: This section contains details necessary for establishing a connection to the S3 compatible storage service.
 - a. **connectionMode**: Specifies the mode of connection, which is set to "**S3_COMPATIBLE**" indicating that it connects to an S3 Compatible storage.
 - b. **s3CompatibleDetails**:
 - i. **bucketName**: Specifies the name of the S3 compatible storages bucket to connect to.
 - ii. **region**: Specifies the region where the S3 compatible storage bucket is located.
 - iii. **prefix**: Indicates the prefix to be used for identifying files within the S3 compatible bucket.
 - iv. **delimiter**: Specifies the delimiter used in the file paths within the S3 compatible bucket.
 - v. **accessKey**: The access key is a Alphanumeric string that uniquely identifies your account.
 - vi. **secretKey**: The secret key is associated with your access key and is used for signing requests to AWS services.
 - vii. **serviceEndpoint**: Specify the service endpoint parameter to indicate the URL of the storage service you want to connect to

Create Connector



- Details
- **Credentials**
- Summary

Credentials

Specify the credentials for Fixed Width connector.

20 Other S3 Compatible Connection Mode

i At present, this connection mode is only supported for storages that are compatible with GCP-hosted buckets. S3 buckets hosted on other clouds may not be compatible.

S3 Compatible Cloud Storage should support the following API.

- Bucket's APIs
 - ListObjects
- Object APIs
 - DeleteObject
 - GetObject
 - PutObject
- Multipart Upload APIs
 - InitiateMultipartUpload

- CompleteMultipartUpload
- UploadPart
- ListParts

6.5.7.2.1 Sample payloads

To connect to Other S3 Compatible storage, Continuous Compliance requires a user's Access Key and Secret Key for authentication.

```
{
  "connectorName":"S3 Compatible storage",
  "environmentId":37,
  "fileType":"JSON",
  "connectionInfo":{
    "connectionMode":"S3_COMPATIBLE",
    "s3CompatibleDetails" : {
      "prefix":"json/",
      "delimiter":"/",
      "region":"us-west-2",
      "bucketName":"s3-compatible-bucket-name",
      "accessKey":"<Your Access Key>",
      "secretKey":"<Your Secret Key>"
    }
  }
}
```

6.5.7.3 S3 upload sizing

S3 supports object sizes up to 5 TB. When uploading objects larger than 100 MB, Amazon recommends using a multipart upload. As the name implies, a multipart upload breaks the object into smaller parts where each is assigned a part number. S3 supports breaking an object into at most 10,000 parts.

When uploading a masked object, Continuous Compliance uses a multipart upload. The part size is calculated by multiplying 20% times the masking job's maximum memory and then dividing by the masking job's number of streams. For example if maximum memory is 2 GB and the number of streams is 1, then the part size would be $(20\% * 2 \text{ GB}) / 1 = 400 \text{ MB}$.

When working with a large object, you must configure the masking job's maximum memory and streams so that the object can be uploaded in at most 10,000 parts. For example if you need to mask a 5 TB object (the maximum size for an S3 object), then the part size must be greater than or equal to $5 \text{ TB} / 10,000 \text{ parts} \cong 525 \text{ MB}$. If you plan to use 1 stream, then the maximum memory should be $(1 * 525 \text{ MB}) / .2 = 2,625 \text{ MB}$ or more.

For more information related to multipart uploads, please refer to:

- [Amazon S3 Multipart Upload](#)¹⁵¹

¹⁵¹ <https://docs.aws.amazon.com/AmazonS3/latest/userguide/mpuoverview.html>

- [Amazon S3 Multipart Upload Limits](#)¹⁵²

6.6 Managing extended connectors

Extended Connectors allow you to upload additional JDBC Drivers to the Continuous Compliance Engine to enable masking of data sources not natively supported by Continuous Compliance.

6.6.1 Limitations

Delphix supports type 4 JDBC Drivers. These must be a pure-java .jar file that can be used simply by uploading it (or it's zip file) to the engine. Anything that requires compilation on the engine, or execution of any kind of install or licensing script, is not supported.

Extended Connectors don't support all of the features available for built-in connectors like Oracle. As of 6.0.9.0, the "Disable Constraint", "Disable Trigger" and "Drop Indexes" options can be implemented and enabled by driver support plugins, which are detailed [here \(see page 1050\)](#). Delphix provides support for Extended Connectors in accordance with our [Support Policy](#)¹⁵³.

Drivers that require a Java version higher than 8 are not supported.

6.6.2 Installing a new driver

To use a new JDBC driver, first you need to upload it to your Masking Engine. Since some drivers require multiple files, the driver and any additional files it needs to function should be put together in a single zip file. Even if a driver doesn't require additional files, it still needs to be zipped.

For example, to package the Informix JDBC driver for use with Continuous Compliance take all three files provided for Informix and zip them together:

```
$ ls
LICENSE.txt ifxjdbc.jar ifxlang.jar
$ zip informix.zip *
  adding: LICENSE.txt (deflated 70%)
  adding: ifxjdbc.jar (deflated 4%)
  adding: ifxlang.jar (deflated 4%)
$ ls
LICENSE.txt ifxjdbc.jar ifxlang.jar informix.zip
$
```

To upload the driver package to the engine, navigate to the **JDBC Drivers** under **Settings**.

¹⁵² <https://docs.aws.amazon.com/AmazonS3/latest/userguide/qfacts.html>

¹⁵³ <https://www.delphix.com/masking-help/jdbc-drivers-support>

The screenshot shows the 'JDBC Drivers' configuration page in the Continuous Compliance interface. The page includes a navigation menu on the left with options like Algorithms, Classifiers, Data Formats, Domains, Expressions, JDBC Drivers (selected), and Profile Sets. The main content area displays a table of installed drivers. The table has columns for Name, Class Name, Version, Date, Description, Checksum, and Actions. The table lists several drivers, including MySQL, Sybase, Oracle, MSSQL, Postgres, and three instances of DB2. A '+ JDBC Driver' button is located in the top right corner of the table area. Below the table, it says 'Displaying 1 to 8 of 8'.

Name	Class Name	Version	Date	Description	Checksum	Actions
MySQL	org.mariadb.jdbc.Driver	2.7.2	2 Apr 2024 14:00 ...	The default MySQL driver used for MySQL and MariaDB connectors.		...
Sybase	com.sybase.jdbc42.jdbc....	16.0	2 Apr 2024 14:00 ...	The default Sybase driver used for Sybase connectors.		...
Oracle	oracle.jdbc.OracleDriver	19.3	2 Apr 2024 14:00 ...	The default Oracle driver used for Oracle connectors.		...
MSSQL	com.microsoft.sqlserver.j...	12.4.1	2 Apr 2024 14:00 ...	The default MSSQL driver used for MSSQL connectors.		...
Postgres	org.postgresql.Driver	42.5.4	2 Apr 2024 14:00 ...	The default Postgres driver used for Postgres connectors.		...
DB2	com.ibm.db2.jcc.DB2Driver	4.32.28	2 Apr 2024 14:00 ...	The default DB2 driver used for DB2 connectors.		...
DB2	com.ibm.db2.jcc.DB2Driver	4.32.28	2 Apr 2024 14:00 ...	The default DB2 driver used for DB2 connectors.		...
DB2	com.ibm.db2.jcc.DB2Driver	4.32.28	2 Apr 2024 14:00 ...	The default DB2 driver used for DB2 connectors.		...

Clicking on the **+ JDBC Driver** button will bring up a dialog box to upload the driver zip file and enter the driver's configuration details.

Add JDBC Driver

Name

Description (Optional)

Class Name


(This is the class that implements java.sql.Driver)

Driver ⓘ
Select the JDBC driver for upload

Extensions Support Policy is defined [here](#)

The **Add JDBC Driver** screen lets you set the following information.

- **Name:** A human-readable name for the driver. Name it whatever is convenient for you. **Note:** Special Characters are not allowed in the Name field.
- **Description:** A human-readable description of the driver.
- **Class Name:** The Fully Qualified Class Name of the class in the JDBC driver that implements the java.sql.Driver interface. The class name will be in the documentation for the driver itself.
- **Select JDBC driver for upload:** Lets you select the zip file containing the driver and upload it.

 Users cannot update the driver support that a JDBC driver references or uses via the UI; as of now, that can only be done via the web API.

To remove an uploaded driver, click the Actions button to the right side corner of the **JDBC Drivers** list and select the option **Delete**. Note that the delete will fail if any Connectors exist that use the driver you're trying to delete.

If you find you need to edit a driver's configuration options later, click the Actions button to the right side corner of the **JDBC Drivers** list and select the option **Edit**.

6.6.3 Driver permissions

The Continuous Compliance Engine uses the Java Security Manager to prevent uploaded JDBC drivers from performing certain actions without your permission.

Uploaded drivers are granted all permissions *except* for the following non- `FilePermission` :

Class	Target	Action
<code>java.net.SocketPermission</code>	<code>localhost:-</code>	accept, connect, listen, resolve
<code>java.lang.RuntimePermission</code>	<code>exitVM</code>	
<code>java.lang.RuntimePermission</code>	<code>createClassLoader</code>	
<code>java.lang.RuntimePermission</code>	<code>accessClassInPackage.sun</code>	
<code>java.lang.RuntimePermission</code>	<code>setSecurityManager</code>	
<code>java.security.SecurityPermission</code>	<code>setPolicy</code>	
<code>java.security.SecurityPermission</code>	<code>setProperty.package.access</code>	

With regards to `FilePermissions`, `read` access is granted to all, though `write` is only allowed for the following directories:

- the masking user's home directory (`System.getProperty("user.home")`)
- the JVM's default temp directory (`System.getProperty("java.io.tmpdir")`)

Please note that both of these locations are shared, so care will need to be taken to avoid collisions.

The set of permissions granted to uploaded drivers is static and cannot be modified.

6.6.4 Extended logging

The Continuous Compliance Engine provides enhanced logging for extended connectors to assist in debugging connection problems. Enhanced logging can be enabled when the connector is created by checking the 'Enable Logger' box. Enhanced logging may have an impact on performance so you should enable it only when debugging connection problems.

Note that extended logging will not work with signed drivers such as MSSQL.

Enhanced Logging requires some additional permissions to be granted.

Class Name	Target Name	Action Name	Purpose
java.io.RuntimePermissi on	getClassLoader		Allows the driver to load the classes implementing the logging feature

6.6.5 Creating an extended connector

Creating a connector using an Extended Driver is very similar to creating a connector with built-in support.

It can be done by the following steps:

Details Step

Choose **Database - Extended** as the Source Type. Specify **Connection Name** which is a name for this connection.

Create Connection



- Details
- Credentials
- Custom Properties
- Summary

Details

Specify details to identify this connector.

Connection Name
Extended-Test

Select Source Type
Database - Extended

Cancel

Back

Next

Test Connection

Save

21 Details Step

Credentials Step

The following fields will be available:

- **JDBC Driver** Select the JDBC Driver you want to use for this connection
- **Login ID** The username the Masking Engine should connect to the target database with.
- **Password** The password to use to connect to the database
- **JDBC URL** You must provide the JDBC URL for the database to connect to. The exact format and available parameters are specific to the database you're connecting to. Consult your database vendors documentation for details.



Some databases allow you to specify usernames and passwords in the JDBC URL. It's best not to do this. The Continuous Compliance Engine is careful not to log the Login ID and Password in the Masking Engine's logs, but JDBC URLs may be logged unmodified.

Create Connection



- Details
- **Credentials**
- Custom Properties
- Summary

Credentials

Specify the credentials for Extended connector.

Schema Name

JDBC URL

Enable Logger

Select JDBC Driver

Extensions Support Policy is defined [here](#)

Username

Password

Cancel Back Next Test Connection Save

22 Credentials Step

Custom Properties Step

In this step, you can upload the connection properties (optional).

Create Connection



- Details
- Credentials
- Custom Properties
- Summary

Custom Properties

Specify connection properties(Optional) for Extended connector.

Connection Properties ⓘ

Choose File

Property	Value	Modified
No items to display		

Cancel Back Next Test Connection Save

23 Custom Properties Step

Summary Step

Here you can view all the details of the information entered for a quick review before saving. Information related to Selected JDBC Driver will also be available here.

Create Connection



- Details
- Credentials
- Custom Properties
- Summary

Summary

View the connector details below. Click the Back button to make changes or click the Save button to save Extended connector.

Connection Summary

Connection Name
Extended-Test

Schema Name
test

JDBC URL
jdbc:informix-sqli://sample-
host.example.com:9088/sysuser

JDBC Driver
Driver_1

Username
dbuser

JDBC Driver Detail

Type
com.informix.jdbc.IfxDriver

Version
3.70

Date Uploaded
2024-04-03T06:47:32.090Z

Uploaded By
admin

Default Driver
false

24 Summary Step

Once the connector is created, you can create rulesets, inventories, and jobs to profile and mask your data as with other types of connectors.

Extended Connectors can be edited and deleted in the same way as [Built In Connectors](#) (see page 355).

6.6.6 Synchronization

Connectors using extended JDBC Drivers can be synchronized similarly to other connectors. See [Working with Multiple Masking Engines](#) (see page 842) for details. When a job or connector requires an uploaded JDBC Driver, the driver will be exported along with the connector or job. JDBC Drivers are part of the **Global Object** and so will be synchronized whenever the Global Object is synchronized. They can also be synchronized individually.

6.6.7 License entitlement for commercial JDBC drivers

Continuous Compliance requires the installation of a Delphix license to use certain commercial JDBC drivers. Users who have purchased this entitlement will have access to their given license files on the [Delphix download site](#)¹⁵⁴.

¹⁵⁴ <https://download.delphix.com/>

6.6.7.1 Installing a license



- Commercial JDBC drivers from the following vendors must be licensed using this mechanism:
 - CData
- To synchronize License files attached to a JDBC driver while synchronizing a connector, Global Objects should be Synchronized first.

1. Download the license from the Delphix download site.
2. Upload the license to Continuous Compliance using either
 - a. the API using the `POST /license` API endpoint or
 - b. the UI using the **License** file upload button in the **Add or Edit JDBC Driver** dialogue box. The License button appears on entering a valid CData JDBC driver class name as shown in this screenshot:

Add JDBC Driver

Name

Description (Optional)

Class Name

(This is the class that implements java.sql.Driver)

Driver ⓘ
Select the JDBC driver for upload

License(s) ⓘ
If you have received license(s) from Delphix for this driver, select it for upload

6.6.7.2 Managing licenses

To view the licenses installed on an engine, use the `GET /license` API or `GET /ALL` API endpoint.

6.7 Managing rule sets

This section describes how rule sets can be created, edited, and removed.

6.7.1 The rule set screen

To access the rule sets linked to a specific environment, simply click on the “**Rule Sets**” tab from any location within that environment. This action will display the rule sets listing screen. If you haven't created any rule sets yet, the list will be empty.

The screenshot shows the 'Rule Sets' screen for a 'Demo_Environment'. The interface includes a top navigation bar with 'Environments', 'Monitor', 'Settings', 'Admin', and 'Audit' tabs. Below the navigation bar, there is an 'Information' banner stating: 'The Inventory Tab has been removed from this screen. To access the inventory of a rule set, click on the hyperlink associated with its name on the Rule Set listing screen.' The main content area features a breadcrumb 'Environments > Demo_Environment', a 'Back' link, and a '+ Rule Set' button. Below this, there are tabs for 'Jobs', 'Rule Sets' (selected), and 'Connectors'. A table lists 13 rule sets with columns for Rule Set ID, Name, Connector Name, Source Type, Metadata, Refresh/Save, and Actions.

Rule Set ID	Name	Connector Name	Source Type	Metadata	Refresh/Save	Actions
23	MysqlRuleset	MysqlDBConnector	Database	mysql	✓	...
24	HanaRuleset	HanaConnector	Database	extended	✓	...
25	PostgresRuleset	PostgresDBConnector	Database	POSTGRESQL	✓	...
26	LocalPostgresRuleset	LocalhostPostgresDB	Database	POSTGRESQL	✓	...
27	OracleRuleset	OracleConnector	Database	oracle	✓	...
28	WholeFileRs	WholeFileConnector	Fixed Width	File	N/A	...
29	JSONFileRs	JSONFileConnector	JSON	File	N/A	...
30	FixedFileRs	FixedFileConnector	Fixed Width	File	N/A	...
31	XMLFileRs	XMLFileConnector	XML	File	N/A	...
32	DelimitedFileRs	DelimitedFileConnector	Delimited	File	N/A	...
33	MainFrameFileRuleSet	MainFrameFileConnector	DataSet	Mainframe	N/A	...

The **rule sets** screen contains the following information and actions:

- **Rule Set ID** – The numeric ID of the rule set used to refer to the rule set from the Masking API.
- **Name** – The name of the rule set.
- **Connector Name** – The name of the connector.
- **Source Type** – The specific type of ruleset. One of Database, Fixed Width, Delimited, JSON, XML, or DataSet
- **Metadata** – The type of rule set (Database type, File, or Mainframe).
- **Refresh/Save** – Refresh status of the rule set. Only applies to Database rule sets. See [here](https://delphixdocs.atlassian.net/wiki/spaces/CC/pages/8324329/Managing+rule+sets#Refreshing-a-rule-set)¹⁵⁵ for more details.

¹⁵⁵ <https://delphixdocs.atlassian.net/wiki/spaces/CC/pages/8324329/Managing+rule+sets#Refreshing-a-rule-set>

- **Actions** – List of actions
 - **Edit** – Edit the rule set. See more details below.
 - **Duplicate** – Copy the rule set. See more details below.
 - **Delete** – Delete the rule set. See more details below.
 - **Refresh** – Refresh the rule set. This only applies to Database rule sets. See [here](#)¹⁵⁶ for more details.

The rule sets on the screen can be filtered or sorted by the various informational fields by clicking on the respective field. More information on grid filtering and sorting can be found [here](#) (see page 232).



- View permission of Inventory and File-Format is required to select a rule set and view the inventory.

6.7.2 Creating a new rule set

To create a new rule set, first select an Environment by clicking on its name. Then, navigate to the **Rule Set** tab and click on the "Add Rule Set" button located in the upper right-hand corner.

A dialog box prompts you to select the type of rule set you wish to create. You'll find four options to choose from: **Database**, **Semi-Structured**, **Flat Files**, and **Copybook**. Simply click on the option that corresponds to your choice, and then proceed by clicking on "Continue".

¹⁵⁶ <https://delphixdocs.atlassian.net/wiki/spaces/CC/pages/8324329/Managing+rule+sets#Refreshing-a-rule-set>

What type of data does this Rule Set contain? ×

Database

Select if source data for this Rule Set is in a **database**.

Semi-Structured Files

Select if source data for this Rule Set is in a **JSON** or **XML** file.

Flat Files

Select if source data for this Rule Set is in a **Delimited** or **Fixed-Width** format.

Copybook

Select if source data for this Rule Set is in a **mainframe**.

Close

Continue

An **Add Rule set wizard** appears comprising three steps, depending on the rule set type selection made earlier to start the creation of a new rule set. See "[The Add/Edit rule set wizard](#)"¹⁵⁷ for a complete description of this screen and additional options, which will guide you through the process of creating a new rule set.

6.7.3 Editing/Modifying a rule set

To edit a rule set, navigate to the **Rule Set tab** of an environment, Click on the (...) in the Actions column for the rule set, and select the **Edit** option from the menu.

An **Edit Rule set wizard** similar to the Add ruleset wizard appears, with data pre-filled depending on the corresponding rule set being edited. See "[The Add/Edit rule set wizard](#)"¹⁵⁸ for a complete description of this screen and additional options, modify the rule set as desired using the steps provided here.

¹⁵⁷ <https://delphixdocs.atlassian.net/wiki/spaces/CC/pages/8324329/Managing+rule+sets#The-Add/Edit-rule-set-wizard>

¹⁵⁸ <https://delphixdocs.atlassian.net/wiki/spaces/CC/pages/8324329/Managing+rule+sets#The-Add/Edit-rule-set-wizard>

6.7.4 The Add/Edit rule set wizard

6.7.4.1 Step 1: Details

1. Start by providing a name for the rule set you wish to create.
2. Select an appropriate connector from the dropdown menu.
 - The dropdown displays connectors that have already been created. If you haven't created any connectors yet, the dropdown will be empty, and you will not be able to proceed with rule set creation.
 - The connectors shown in this dropdown are filtered based on the rule set type chosen in the previous dialog.
 - Additionally, you can quickly filter the connector list to select the desired connector by typing into the input box.
3. Click on "Next" to proceed.



During the editing of a rule set, all input fields in this step will be non-editable and displayed in read-only mode. Click "Next" to proceed to adding, removing, or modify database tables, files, or file patterns within the rule set.

Add a Rule Set for a Database



- Details
- Data Tables
- Summary

Details

Name the Rule Set and select a connector.

Rule Set Name
DemoPostgresRuleset

Select Connector
PostgresDBConnector

Connector details

Connector Name
PostgresDBConnector

Database Type
POSTGRES

Database Name
hercules

Host
anp-spa.dlpxdc.co

Port
5432

Schema Name
public

User Name
metadataservice

Cancel Back Next Save

6.7.4.2 Step 2: Data Tables / Files

When initially creating a ruleset, this list will be empty. You can select data tables or files that are accessible by the chosen connector to add to this ruleset. When editing the file ruleset, this grid will display the file names already included in the ruleset being modified. In the case of a Database ruleset, this grid will display all the tables (Added, Removed, and Modified) in the ruleset.

Database Rule Set:

Add a Rule Set for a Database



- Details
- Data Tables
- Summary

Data Tables

Specify which data tables for the selected connectors to be included in this rule set.

<input type="checkbox"/> Table Name ↑	<input type="checkbox"/> Logical Key	<input type="checkbox"/> Filter	<input type="checkbox"/> Custom SQL
<input type="checkbox"/> TEST1		ID > 10	
<input type="checkbox"/> TEST10	ID		
<input checked="" type="checkbox"/> TEST2			
<input type="checkbox"/> TEST3			
<input type="checkbox"/> TEST4			
<input type="checkbox"/> TEST5			

Displaying 1 to 7 of 7

The “Data Tables” grid contains the following information,

- **Table Name** – The name of the table in the rule set.
- **Logical Key** – A logical key is a unique, non-null value that identifies a row in the database.
- **Filter** – A WHERE clause to filter or subset the data.
- **Custom SQL** – Customized SQL SELECT query for the table.

The tables on the screen can be filtered or sorted by clicking the respective fields—all the fields on the above screen support sorting and filtering. For more details on grid filtering and sorting, refer to the information [here](#) (see page 232).



While editing the ruleset, a read-only column named “**State**” is shown to indicate whether the table is newly added, modified, or removed. The state column will be blank for unmodified items.

Edit Database Rule Set



- Details
- **Data Tables**
- Summary

Data Tables

Specify which data tables for the selected connectors to be included in this rule set.

<input type="checkbox"/> Table Name ↑	Logical Key	Filter	Custom SQL	State
<input type="checkbox"/> TEST2				
<input type="checkbox"/> TEST3				
<input type="checkbox"/> TEST4				Removed
<input type="checkbox"/> TEST5	ID	ID > 100		Modified
<input type="checkbox"/> TEST6			Select * from TEST6 where ID ...	Modified
<input type="checkbox"/> TEST8				Added
<input type="checkbox"/> TEST9				Added

Displaying 2 to 9 of 9



- When editing a rule set with a large number of tables, the tables will load in batches. Although the full load may take some time, the UI will become editable as soon as a smaller set of tables is ready. A loader will be visible above the grid to indicate progress, and the number of tables loaded can be seen at the bottom.
- Using select all, before all the tables loaded to edit the rule set, will only affect the tables loaded, not all the tables present in the rule set.

Edit Database Rule Set



- Details
- **Data Tables**
- Summary

Data Tables

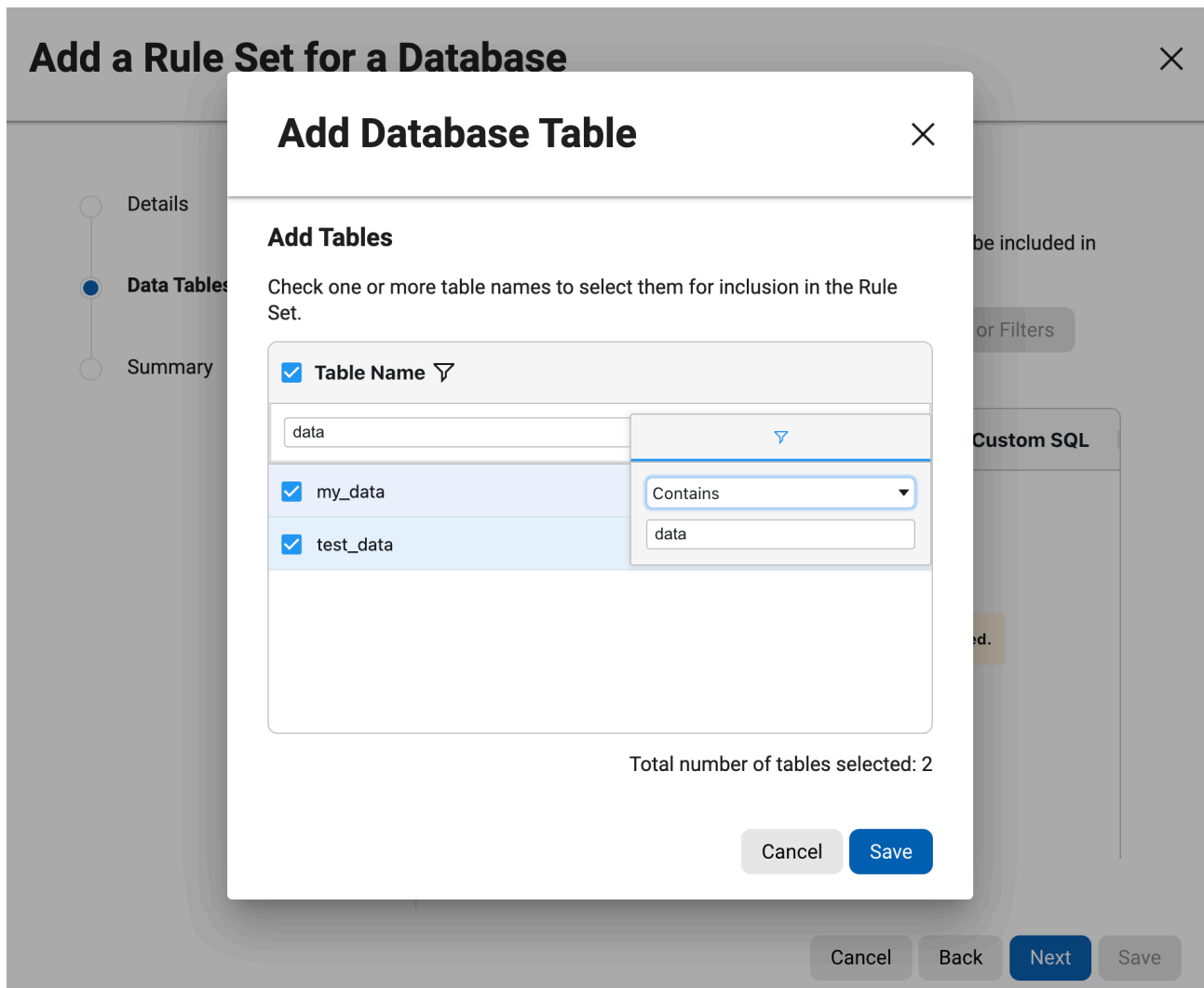
Specify which data tables for the selected connectors to be included in this rule set.

<input type="checkbox"/> Table Name ↑	Logical Key	Filter	Custom SQL	State
<input type="checkbox"/> TEST1				
<input type="checkbox"/> TEST10				
<input type="checkbox"/> TEST11				
<input type="checkbox"/> TEST12				
<input type="checkbox"/> TEST13				
<input type="checkbox"/> TEST14				
<input type="checkbox"/> TEST15				
<input type="checkbox"/> TEST16				
<input type="checkbox"/> TEST17				
<input type="checkbox"/> TEST18				

Displaying 1 to 10 of 30

Adding tables to a database rule set.

- Click on the **"Add Tables"** button to include new tables in the rule set being created or modified.
- A dialog box titled "Add Database Table" will appear, listing all the table names for the selected connector. From this list, you can select multiple tables and click **"Save"** to include them in the rule set being created or modified.
- The table names can be filtered either by using a substring or a starts with condition. For more details on grid filtering and sorting, refer to the information [here \(see page 232\)](#).
- All the tables can also be selected/deselected by using the checkbox given before the header **Table Name**, refer to the information [here. \(see page 0\)](#)



Flat Files, Semi-Structured, or Copybook Rule Sets:

Edit Flat files Rule Set



- Details
- Data Files
- Summary

Data Files

Select and configure which imported Data Format Files to include in this Rule Set.

<input type="checkbox"/>	File Name or Pattern	Format ...	End of R...	Delimiter	Text Enc...	Enclosur...	Escape ...	Is Pattern	State
Existing Files									
<input type="checkbox"/>	environment	default_de...	LF termina...	,		Double En...	No	No	
<input type="checkbox"/>	export.csv	default_de...	LF termina...	,		Double En...	No	No	
<input type="checkbox"/>	export1.csv	default_de...	LF termina...	,		Double En...	No	No	
<input type="checkbox"/>	huge2.csv	default_de...	LF termina...	,		Double En...	No	No	
<input type="checkbox"/>	newembedded.csv	default_de...	LF termina...	,		Double En...	No	No	
<input type="checkbox"/>	sample.csv_org	default_de...	LF termina...	,		Double En...	No	No	
<input type="checkbox"/>	sap.hana.xs.formLogi...	default_de...	LF termina...	,		Double En...	No	No	
Files Added									
<input type="checkbox"/>	TEST_1	default_de...	LF termina...	,		Double En...	No	No	Added
<input type="checkbox"/>	sample.csv	default_de...	LF termina...	,		Double En...	No	No	Added

Displaying 4 to 15 of 15

The "Data Files" step contains the following information and actions, depending on the connector type selected.

Fixed-Width: File Name or Pattern, Format File, End of Record, Mask Whole File, Is Pattern.

Delimited: File Name or Pattern, Format File, End of Record, Delimiter, Text Enclosure, Enclosure Escape Strategy, Escape Enclosure Escape Character, Is Pattern.

XML & JSON: File Name or Pattern, Format File, Is Pattern.

Copybook: File Name or Pattern, Format File, Variable Length, Is Pattern.

Properties of a Data File :

- **File Name or Pattern** – The name of the file or pattern in the rule set.
- **Format File** – Name of the file format created specific to the file type.
- **End of Record** – The string of characters that delineates the end-of-record for a file. Note that, for linux this is '\n', and for windows it is '\r\n'.
- **Mask Whole File** – This flag indicates whether the file is to be read as whole or line-by-line. Applicable only for Fixed-Width file type.
- **Is Pattern** – This flag indicates whether or not this file name represents a regular expression.
- **Delimiter** – The delimiter for a delimited file. This field should be left blank for other file types.
- **Text Enclosure** – The text enclosure for the file.
- **Enclosure Escape Strategy** – The character used to escape a literal enclosure character within an enclosed value. By default, this is equal to the enclosure value itself, so doubling the enclosure character escapes it.

- **Escape Enclosure Escape Character** – This flag indicates whether the enclosure escape character also escapes itself. For example, if the enclosure escape character is *, then the sequence ** would be treated as a single * character, rather than an escape.
- **Variable Length** – The record format type for the mainframe data set.



While editing the ruleset, a read-only column named "**State**" is shown to indicate whether the file is newly added, modified, or removed. The status column will be blank for unmodified items.

You can filter and sort the "**File Name or Pattern**" column. For more details on grid filtering and sorting, refer to the information [here \(see page 232\)](#).

Adding files or file patterns to a Flat File, Semi-Structured, or Copybook Rule Set.

- Click on "**Add File Name**" to include new file names in the rule set being created or modified.
 - A dialog box titled "Add File Name" will appear, the "**File Name**" table lists all the files accessible by the selected connector. You can choose multiple files to include in the rule set. File names can be filtered by using a substring condition. For more details on grid filtering and sorting, refer to the information [here \(see page 232\)](#).



File Name list

The files listed in the File Name table are only those that haven't been added to the ruleset yet.

Add File Name



Select one or more files to include them in the rule set. Configure the format and attributes of the files according to the specified file names.

File Name

File Name ⌵

⌵

<input checked="" type="checkbox"/>	TEST_1
<input checked="" type="checkbox"/>	test
<input checked="" type="checkbox"/>	test_1
<input checked="" type="checkbox"/>	testdata_delimited

Displaying 1 to 4 of 4

Properties

Select File Format ⌵

⌵ End of Record

Cancel

Save

- Click on "**Add File Pattern**" to input one or more regular expression patterns for matching the data files accessible by the selected connector.
 - A dialog box titled "Add File Pattern" will appear, in "**Regular Expression Pattern**" field you can specify multiple regular expression patterns. After typing a regular expression click on the option Add item shown in the dropdown or press **ENTER** key to include the regular expression as an input. To remove a pattern you've added, simply click on the :cancel: icon next to each corresponding regular expression entry.



File pattern syntax

Expressions are case sensitive. A file pattern uses the regular expression syntax defined by the Java Pattern class. The syntax is documented [here](#)¹⁵⁹. For example, the pattern `.*\txt` will match any file with a `.txt` extension, such as *example.txt*.

¹⁵⁹ <https://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html>

Add File Name Pattern



Specify one or more regular expression matching the file names to include them in the rule set. Configure the format and attributes of the files according to the specified file name patterns. A file pattern uses the regular expression syntax defined by the [Java Pattern class](#).

File Name Pattern

Regular Expression Pattern

Properties

Select File Format

End of Record

Select End of Record

Delimiter* CTRL

Do not use '^' or '\t' - use CTRL

Text Enclosure

Enclosure Escaping Strategy

Double Enclosure

Escape "Enclosure Escape Character"

This flag indicates whether the enclosure escape character also escapes itself. For example, if the enclosure escape character is *, then the sequence ** would be treated as a single * character, rather than an escape.

Cancel

Save

- Provide all required properties for the files or patterns as necessary. Please note that the fields displayed here may vary depending on the selected connector types. These properties will be applied to all selected files or the multiple regular expressions provided.
 - Selecting a file format from the drop-down menu is essential. Be sure to choose the correct format that aligns with the data in the selected files or regular expression patterns.

- For Mainframe you will see a checkbox for a variable length for dataset where each record within the dataset can have a different length.
- For all other file types, select the end-of-record to let the application know whether the file is in Windows/DOS format (CR+LF) or Linux format (LF).
- If the file is delimited, you will find a field available to input the delimiter.
- if the file is fixed-width type you'll find a checkbox labeled "Mask Whole File" available. Further information about this feature is available [here](#)¹⁶⁰.
- Click on the **Save** button to add all the files names or file patterns selected above to the Data Files grid.



The records are not added to the rule set until the Save button is clicked.

Reset All records

"Reset All" button will restore the rule set to its original state. When editing a rule set, this action reverts any changes since the last save. When creating a new rule set, the rule set is reset to an empty state.

Once you've completed adding, modifying or removing files/tables from the ruleset, click on "Next" to continue.

¹⁶⁰ <https://delphixdocs.atlassian.net/wiki/x/IIW>

6.7.4.3 Step 3: Summary

Edit Flat files Rule Set ×

Details

Data Files

Summary

Summary

Review the rule set configuration. Click Back to correct errors or Save to continue.

Details

Rule Set Group
Flat

Rule Set Name
DelimitedFileRs

Connector Name
DelimitedFileConnector

Connection Type
DELIMITED

Connection Mode
SFTP

Update All Files
FALSE

Remove All Files
FALSE

Total Data Files Added
1

Total Data Files Modified
1

Total Data Files Removed
1

Cancel Back Next Save

Review the rule set configuration. Click Back to correct errors or **Save** to continue.



- The "Save" button will remain inactive when no changes have been made to the edited rule set or when creating a new rule set without any tables or files included.
- Changing the connector selection will reset all modifications made to the rule set.

6.7.5 Removing a table or file from the ruleset

To remove a table or file from a rule set:

1. Click the Actions (...) drop-down to the right of a rule set on the **rule set** screen, and then select the **Edit** option.

The screenshot shows the Continuous Compliance user interface. At the top, there's a navigation bar with 'Environments', 'Monitor', 'Settings', 'Admin', and 'Audit'. Below it, an information banner states: 'The Inventory Tab has been removed from this screen. To access the inventory of a rule set, click on the hyperlink associated with its name on the Rule Set listing screen.' The main content area is titled 'env_143RWOK1' with a 'Back' link and a '+ Rule Set' button. Below the title, there are tabs for 'Jobs', 'Rule Sets', and 'Connectors'. The 'Rule Sets' tab is active, displaying a table with columns: Rule Set ID, Name, Connector Name, Source Type, Metadata, Refresh/Save, and Actions. The first row shows a rule set with ID '1', Name 'rs_5Q5ADQNA', Connector Name 'con_OB4SNLPV', Source Type 'Database', and Metadata 'oracle'. A dropdown menu is open over the 'Actions' column of this row, showing options: Refresh, Edit (highlighted with a red arrow), Duplicate, and Delete.

Rule Set ID	Name	Connector Name	Source Type	Metadata	Refresh/Save	Actions
1	rs_5Q5ADQNA	con_OB4SNLPV	Database	oracle	✓	⋮

2. In the rule set wizard's second step (either "Data Tables" or "Data Files"), you can select one or more tables or files either by clicking on the checkboxes or by clicking anywhere on the rows.
3. Click on the **"Remove Selected"** button, located on top of the grid.
4. A confirmation dialog will pop up, displaying the count of tables or files being removed from the ruleset.
5. Click on **Confirm**.
6. Any newly added tables or files will be removed from the grid. However, tables or files already saved to the rule set, their corresponding rows will be updated with the status "Removed".
7. Click on Next to verify the summary, and then click on **Save** button.

- i**
- Clicking the Confirm button in the dialog will not immediately delete the tables or files from the rule set. The changes will only take effect after clicking the Save button in the rule set wizard.

6.7.6 Removing or Modifying All records in a rule set

To remove or modify properties of all the tables or files included in a ruleset, click on checkbox beside the "File Name or Pattern" or "Table Name" column header in the grid. Then apply any action using the action buttons above the grid, like Edit Selected or Remove Selected.

i After checking the select all checkbox and then applying any actions like edit selected or remove selected, the grid listing table or file names become inactive and won't allow further actions until the last action is saved. You can use "Reset All" button to reset the grid to its initial state and discard all changes, enabling you to start modifying again.

Edit Flat files Rule Set



- Details
- Data Files
- Summary

Data Files

Select and configure which imported Data Format Files to include in this Rule Set.

<input type="checkbox"/>	File Name or Pattern	Format ...	End of R...	Delimiter	Text Enc...	Enclosur...	Escape ...	Is Pattern	State
Existing Files									
<input type="checkbox"/>	USER_PREFERENCES...	default_de...	LF termina...	,		Double En...	No	No	Removed
<input type="checkbox"/>	embeddedXML.csv	default_de...	LF termina...	,		Double En...	No	No	Removed
<input type="checkbox"/>	embeddedXML.csv_bkp	default_de...	LF termina...	,		Double En...	No	No	Removed
<input type="checkbox"/>	enclosure.csv_bkp	default_de...	LF termina...	,		Double En...	No	No	Removed
<input type="checkbox"/>	environment	default_de...	LF termina...	,		Double En...	No	No	Removed
<input type="checkbox"/>	export.csv	default_de...	LF termina...	,		Double En...	No	No	Removed
<input type="checkbox"/>	export1.csv	default_de...	LF termina...	,		Double En...	No	No	Removed
<input type="checkbox"/>	huge2.csv	default_de...	LF termina...	,		Double En...	No	No	Removed
<input type="checkbox"/>	newembedded.csv	default_de...	LF termina...	,		Double En...	No	No	Removed
<input type="checkbox"/>	sample.csv_org	default_de...	LF termina...	,		Double En...	No	No	Removed

Displaying 1 to 12 of 12

6.7.7 Modifying table properties in a rule set

6.7.7.1 Logical key

A logical key is a unique, non-null value that identifies a row in the database.

If your table has no primary keys defined in the database, and you are using an In-Place masking strategy, you must specify an existing column or columns to be a logical key. This logical key does not change the target database; it only provides information to Delphix. For multiple columns, separate each column using a comma.

i If no primary key is defined and a logical key is not defined an identity column will be created. This requires the user be authorized to modify the schema, or else the job may fail.

Edit Logical Key



Specify a logical key for the table(s) you've selected. Check that the column(s) specified exist in the table(s). Masking jobs will fail if the key column does not exist in the table(s).

To enter a logical key:

1. In the rule set wizard's "Data Tables" step, click on one or more checkboxes or click anywhere on the rows to select the desired tables.
2. Click on the **"Edit Logical Key"** button, located on top of the grid.
3. Enter the logical key value for the tables in the Logical Key field.
4. Click on **Save**.



- The logical key cannot be more than 1024 characters in length.
- Clicking the Save button in the dialog will not immediately update the table properties. The changes will only take effect after clicking the Save button in the rule set wizard.

6.7.7.2 Edit filter

Use this function to specify a filter to run on the data before loading it to the target database.

Edit Custom SQL or Filters



Table can have either Custom SQL or Filter

Adding a filter to the table will remove the table's existing custom SQL, and vice versa.

Specify a custom SQL or filter for the selected tables.

Custom SQL

Filter

Filter

my_data.id < 1000

The column name should be preceded by a prefix for the table name, like
customer.cust_id < 1000

Cancel

Save

To add a filter to a database rule set table or edit a filter:


1. In the rule set wizard's "Data Tables" step, click on one or more checkboxes or click anywhere on the rows to select the desired tables.
2. Click on the "Edit Custom SQL or Filters" button, located on top of the grid.
3. Ensure the **Filter** radio button is selected
4. Enter the where condition for this table in the text input area.
5. Click on **Save**.



- Be sure to specify column name with table name prefix, for example, customer.cust_id < 1000.
- Clicking the Save button in the dialog will not immediately update the table properties. The changes will only take effect after clicking the Save button in the rule set wizard.

6.7.7.3 Custom SQL

Use this function to supply a customized SQL SELECT Query for the table. Typically, this query will include a **WHERE** clause to filter or subset the data.

 The custom SQL must contain the primary key column (or columns if the table uses a composite primary key) and all columns that will be masked.

Edit Custom SQL or Filters



 **Table can have either Custom SQL or Filter**

Adding a filter to the table will remove the table's existing custom SQL, and vice versa.

Specify a custom SQL or filter for the selected tables.

Custom SQL

Filter

[Generate Custom SQL](#)

Custom SQL

//

Cancel

Save

To add or edit SQL code:

1. In the rule set wizard's "Data Tables" step, select a checkbox or click anywhere on the row to select the desired table.
2. Click on the "**Edit Custom SQL or Filters**" button, located on top of the grid.
3. Ensure the **Custom SQL** radio button is selected
4. Enter the custom SQL code for this table in the text input area or click on "**Generate Custom SQL**" to automatically suggest a custom SQL for you.
5. Click on **Save**.



- You cannot update custom SQL for multiple tables simultaneously. When multiple tables are selected, the Custom SQL radio button will be disabled.
- The "Generate Custom SQL" option will be available only for the tables that have already been added to the rule set.
- Clicking the Save button in the dialog will not immediately update the table properties. The changes will only take effect after clicking the Save button in the rule set wizard.

Delphix will run the query to subset the table based on the SQL you specify.



A table can have either Custom SQL or Filter at a given time. Adding a filter to the table will remove the table's existing custom SQL, and vice versa.

6.7.8 Control character support for delimited files

The user can specify control character as a delimiter/end of record from UI/API.



Control Character

There are two ways to specify control characters,

1. From UI click on the **CTRL** button to open a virtual keyboard, where you can choose the desired control character.
2. From UI or API manually input the control character in the input fields. Ensure it's in the format **\$(hex value of the control character)**, such as `$(01)` for ^A.

The control character value supports the UTF-8 character set. Avoid using '^t' or '\t'. Instead, utilize the CTRL button and select the appropriate control character.

6.7.8.1 Control character as a delimiter

In order to use control character as a delimiter, the user needs to click on **CTRL** button inside delimiter input text or manually enter the control character value.

Add File Name



Select File Format
default_delimited.fmt

End of Record
LF terminated (Unix)

Delimiter*
\${02} CTRL

Do not use '^t' or '\t' - u
Text Enclosure
*

Enclosure Escaping Strat
Double Enclosure

Escape "Enclos
This flag indicates wheth
if the enclosure escape ch
character, rather than an e

^@ [NUL]	^A [SOH]	^B [STX]	^C [ETX]
^D [EOT]	^E [ENQ]	^F [ACK]	^G [BEL]
^H [BS]	^I [HT]	^J [LF]	^K [VT]
^L [FF]	^M [CR]	^N [SO]	^O [SI]
^P [DLE]	^Q [DCL]	^R [DC2]	^S [DC3]
^T [DC4]	^U [NAK]	^V [SYN]	^W [ETB]
^X [CAN]	^Y [EM]	^Z [SUB]	^[[ESC]
^_ [FC]	^] [GS]	^^ [RS]	^_ [US]

Cancel Save

6.7.8.2 Control character as an end of record

In order to use control character as an end of record, the user needs to click on **CTRL** button inside custom end of record input text or manually enter the control character value.

Add File Name



Select File Format
default_delimited.fmt

End of Record
Custom

Custom End of Record*
\$[02] CTRL

Do not use '\r\n' or '\n' as Delimiter*	^@ [NUL]	^A [SOH]	^B [STX]	^C [ETX]
Do not use '^t' or '\t' - use Text Enclosure *	^D [EOT]	^E [ENQ]	^F [ACK]	^G [BEL]
	^H [BS]	^I [HT]	^J [LF]	^K [VT]
Enclosure Escaping Strategy Double Enclosure	^L [FF]	^M [CR]	^N [SO]	^O [SI]
	^P [DLE]	^Q [DCL]	^R [DC2]	^S [DC3]
	^T [DC4]	^U [NAK]	^V [SYN]	^W [ETB]
	^X [CAN]	^Y [EM]	^Z [SUB]	^[[ESC]
	^ [FC]	^] [GS]	^^ [RS]	^_ [US]

Escape "Enclosure Escape Character"

Cancel Save

6.7.8.3 Control character as a value

1. Control characters are supported as values in a delimited file. No special configuration is necessary. Simply configure the delimited file format as usual.
2. The user doesn't need to configure anything extra if the control character is only part of the value and not being used as a delimiter or end of record. However, the user needs to define delimiter/end of record as per the requirement.

6.7.9 Define enclosure escaping strategy for delimited files

The user can configure the enclosure escape character from the UI/API to escape the enclosure. To configure the enclosure escape character from the UI, user needs to select the "Enclosure Escaping Strategy" dropdown value as per below options on the edit rule set popup window,

6.7.9.1 Double enclosure

Double enclosure option will set the escape character value same as enclosure value. For example, if the enclosure escape character is " then escape character value will be " as well.

Add File Name ✕

Select File Format

default_delimited.fmt✕ ▼

End of Record

LF terminated (Unix)▼

Delimiter*

|CTRL

Do not use '^t' or '\t' - use CTRL

Text Enclosure

*

Enclosure Escaping Strategy

Double Enclosure▼

Escape "Enclosure Escape Character"

This flag indicates whether the enclosure escape character also escapes itself. For example, if the enclosure escape character is *, then the sequence ** would be treated as a single * character, rather than an escape.

CancelSave

6.7.9.2 Custom

By selecting custom option user can specify any single character as an enclosure escape character except the "escape sequences" and "control characters".

Add File Name ✕

LF terminated (Unix)

Delimiter* CTRL

Do not use '^t' or '\t' - use CTRL

Text Enclosure

Enclosure Escaping Strategy Custom

Custom Enclosure Escaping Strategy

Escape "Enclosure Escape Character"

This flag indicates whether the enclosure escape character also escapes itself. For example, if the enclosure escape character is *, then the sequence ** would be treated as a single * character, rather than an escape.

Cancel Save

Default Enclosure Escape Character

The default value for "Enclosure Escaping Strategy" is "Double Enclosure".

6.7.9.3 Escape "enclosure escape character"

Selecting this checkbox indicates whether the enclosure escape character also escapes itself. For example, if the enclosure escape character is " then the sequence "" would be treated as a single " character, rather than an escape.

6.7.9.4 Configure enclosure escape character for a large rule set

To configure the enclosure escape character for a large rule set, you can use this [API Script](#). (see page 983)

6.7.10 Refreshing a rule set

Refreshing a rule set will result in the columns in the tables in the rule set being rescanned. As a result, the inventory associated with the rule set will also be refreshed, but any pre-existing algorithm assignments will be retained.

To refresh a rule set:

1. Click the Actions (...) drop-down to the right of a rule set on the **rule set** screen, and then select the **Refresh** option.
2. The **Refresh/Save** icon on the grid will turn to an hourglass as the associated tables are rescanned.
3. After the refresh is complete, the **Refresh/Save** icon will return to the circular arrow.

The screenshot shows the Continuous Compliance user interface. At the top, there's a navigation bar with 'Environments', 'Monitor', 'Settings', 'Admin', and 'Audit'. Below it, an information banner states: 'The Inventory Tab has been removed from this screen. To access the inventory of a rule set, click on the hyperlink associated with its name on the Rule Set listing screen.' The main content area is titled 'env_143RWOK1' with a 'Back' link and a '+ Rule Set' button. Underneath, there are tabs for 'Jobs', 'Rule Sets', and 'Connectors'. The 'Rule Sets' tab is active, displaying a table with the following columns: Rule Set ID, Name, Connector Name, Source Type, Metadata, Refresh/Save, and Actions. The table contains one row for rule set 'rs_5Q5ADONA' with connector 'con_OB4SNLPV' and source type 'Database'. A dropdown menu is open over the 'Refresh/Save' column, showing options: Refresh (highlighted with a red arrow), Edit, Duplicate, and Delete.

Rule Set ID	Name	Connector Name	Source Type	Metadata	Refresh/Save	Actions
1	rs_5Q5ADONA	con_OB4SNLPV	Database	oracle	✓	...

6.7.11 Copying a rule set

If you copy a rule set, the inventory associated with that rule set will also be copied. Also, any filter conditions defined for that rule set will be copied.

To copy a rule set:

1. Click the Actions (...) drop-down to the right of a rule set on the **rule set** screen, and then select the **Duplicate** option.
2. The **Duplicate rule set** window appears.
3. Enter a **Name** for the new rule set.
4. Click **Save**.
5. Modify the rule set as you want, using the procedures described [here](#)¹⁶¹.

The screenshot shows the Continuous Compliance interface. At the top, there's a navigation bar with 'Environments', 'Monitor', 'Settings', 'Admin', and 'Audit'. Below it, an information banner states: 'The Inventory Tab has been removed from this screen. To access the inventory of a rule set, click on the hyperlink associated with its name on the Rule Set listing screen.' The main content area is titled 'env_143RWOK1' with a 'Back' link and a '+ Rule Set' button. Below the title, there are tabs for 'Jobs', 'Rule Sets', and 'Connectors'. The 'Rule Sets' tab is active, showing a table with columns: Rule Set ID, Name, Connector Name, Source Type, Metadata, Refresh/Save, and Actions. The first row has ID '1', Name 'rs_5Q5ADONA', Connector Name 'con_OB4SNLPV', Source Type 'Database', Metadata 'oracle', and a green checkmark in the Refresh/Save column. A context menu is open over the Actions column of this row, with options: Refresh, Edit, Duplicate (highlighted with a red arrow), and Delete.

Rule Set ID	Name	Connector Name	Source Type	Metadata	Refresh/Save	Actions
1	rs_5Q5ADONA	con_OB4SNLPV	Database	oracle	✓	...

6.7.12 Deleting a rule set

If you delete a rule set, the inventory associated with that rule set will also be deleted. Also, any filter conditions defined for that rule set will be deleted.

To delete a rule set, Click the Actions (...) drop-down to the right of a rule set on the **rule set** screen, and then select the **Delete** option.

¹⁶¹ <https://delphixdocs.atlassian.net/wiki/spaces/CC/pages/8324329/Managing+rule+sets#The-Add/Edit-rule-set-wizard>

The screenshot shows the 'Rule Sets' section for environment 'env_143RWOK1'. The table below lists the rule sets:

Rule Set ID	Name	Connector Name	Source Type	Metadata	Refresh/Save	Actions
1	rs_5Q5AD0NA	con_OB4SNLPV	Database	oracle	✓	⋮

The context menu for the first row includes: Refresh, Edit, Duplicate, and Delete. A red arrow points to the Delete option.

6.8 Managing file formats

Unlike databases, files typically do not have built-in metadata to describe the format of the fields in the file, so this information must be provided to Delphix. This is done through the **Settings** tab, where a menu item is available on the left **Data Formats**. Click on Data Formats to see tabs for **File** and **Mainframe** and a button to add a format.

The screenshot shows the 'Data Formats' section under the 'Settings' tab. The table below lists the data formats:

ID	Name	Type	Actions
1	996QGfZO.fmt	Delimited	✖
2	VI03HE18.fmt	Fixed Width	✖
4	D21RLWF6.xml	XML	✖
5	8MFLR5WL.fmt	Delimited	✖
6	HKH1PQ0B.json	JSON	✖
7	GOL7IZ7F.json	JSON	✖
9	00YRAG2C.fmt	Delimited	✖
10	53W8WINM.xml	XML	✖

Displaying 1 to 8 of 22

The formats on the screen can be filtered or sorted by the various informational fields, by clicking on those respective fields. More information on grid filtering and sorting can be found on the [Graphical user interface](#)¹⁶² page.

- The file inventory relies on the file format it represents. As a result, any changes to the file inventory affect all data files using that file format. Thus, altering the file inventory impacts all files conforming to that format across the application. This is applicable to all the file types Fixed-Width, Delimited, Mainframe, JSON & XML.

6.8.1 Assigning file formats to files

When creating a rule set, you have the option to assign a file format to a file or set of files that you are incorporating into the rule set. It is essential to assign an appropriate file format to the data files.

You can change the file format for a file or set of files already in the rule set by editing the rule set. In the Edit Rule set wizards second step **Data Files**, select required file(s), clicking on the **Edit Selected** button, an **Edit File Properties** pop-up window will appear. There is a drop-down to select the proper format for the file(s) selected.

¹⁶² <https://masking.delphix.com/docs/latest/graphical-user-interface>

Edit File Properties ✕

Select File Format ▾

End of Record

date_csv_format.fmt

db_to_delimited.fmt

default_delimited.fmt

test_1_delimited.fmt

Text Enclosure

Enclosure Escaping Strategy ▾

Double Enclosure

Escape "Enclosure Escape Character"

This flag indicates whether the enclosure escape character also escapes itself. For example, if the enclosure escape character is *, then the sequence ** would be treated as a single * character, rather than an escape.

Cancel
Save

If the file is a **Mainframe** data set with a copybook, you will see a checkbox to signify if the file is variable length.

For all **other file types**, select the end-of-record to let the Continuous Compliance Engine know whether the file is in Windows/DOS format (CR+LF) or Linux format (LF). If the file is a delimited file, there will be a space to put in the delimiter. If there are multiple files in the rule set, you will have to edit each one individually and assign it to the appropriate file format.

6.8.2 Record types

Record types can be used to perform conditional masking of the file records. If a file has a different set of records spread across multiple rows, the Continuous Compliance Engine should be able to understand all the unique records.

Take an example where a file has the following records in the first three columns of each row; first name, last name, and age – but the last column of each row has a unique record like IP address, ethernet address, etc. In this case, you would need to create a new record type for every unique record present in the file and assign a specific file format to all of the record types. For more information on adding a record type, see the [Managing record types and header/footer records](#)¹⁶³ page.

[-] Record types can only be managed via the **Formats** page. The **Inventory** page does not allow adding, updating, and deleting record types.

6.8.3 Redefine conditions

For **Mainframe** data sets, the File Format screen allows for the entry of redefine conditions, which are used to handle any occurrences of COBOL's REDEFINES construct that might appear in the copybook. In COBOL, the REDEFINES keyword allows an area of a record to be interpreted in multiple different ways. In the example below, each record can hold either the details of a person (PERSON-DET) or the details of a company (COMP-DET).

```

01 CS-CUSTOMER-RECORD.
  05 CUST-TYPE                PIC X(1) .
  05 PERSON-DET.
    10 PERSON-FIRSTNAME      PIC X(20) .
    10 PERSON-LASTNAME       PIC X(40) .
    10 PERSON-ADDRESS1       PIC X(50) .
    10 PERSON-CITY            PIC X(20) .
    10 PERSON-STATE           PIC X(5) .
    10 PERSON-ZIP             PIC X(10) .
    10 PERSON-SSN             PIC S9(9) COMP-3.
  05 COMP-DET                  REDEFINES PERSON-DET.
    10 COMP-ENTITYNM          PIC X(53) .
    10 COMP-ADDRESS1          PIC X(50) .
    10 COMP-CITY               PIC X(20) .
    10 COMP-STATE              PIC X(5) .
    10 COMP-ZIP                PIC X(10) .
    10 COMP-PHONE              PIC X(12) .

```

In order to do any masking, however, the Continuous Compliance Engine must be able to determine (for each record) which fields should be read for the correct algorithms to be applied. In order to do this, the Continuous Compliance Engine uses redefined conditions, which are specified in the format. Redefine conditions are boolean expressions that can reference any fields in the record when they are evaluated.

In the example copybook above, the field CUST-TYPE is used to indicate which group is present. If CUST-TYPE holds a 'P', a PERSON-DET group is present, and if it holds a 'C', COMP-DET is present.


¹⁶³ <https://masking.delphix.com/docs/latest/managing-record-types-and-header-footer-records>

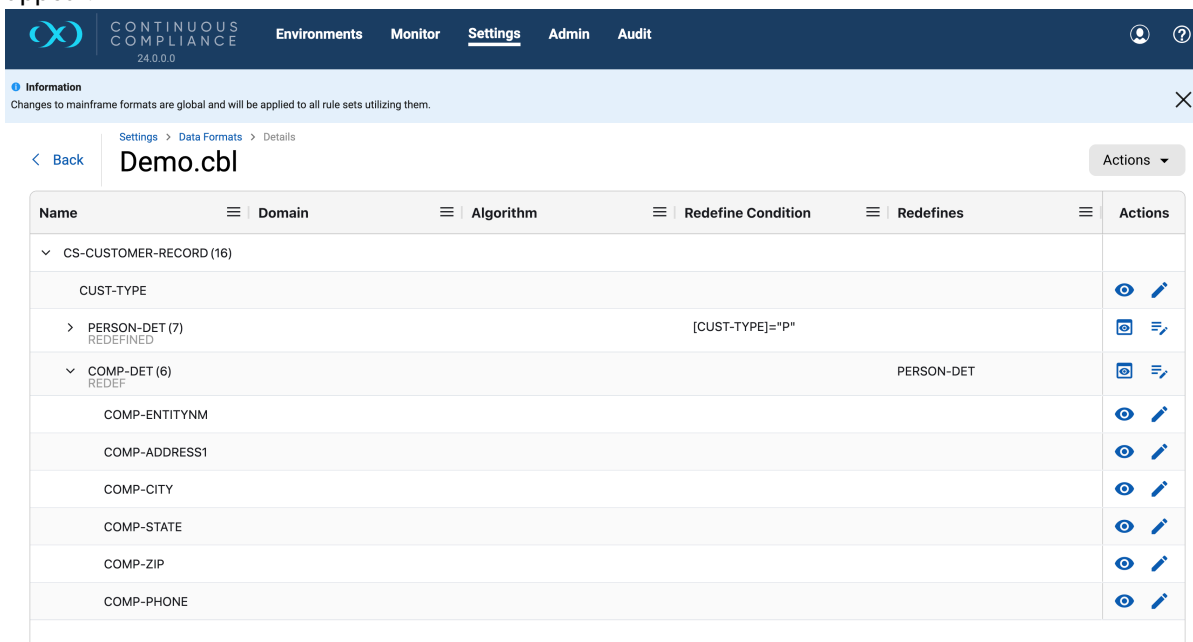
This can be expressed in the inventory by specifying a redefine condition with the value `[CUST-TYPE]='P'`. This expression indicates that, for each record read from the source file during the masking job, the value of the field CUST-TYPE should be read and compared against the string 'P'.

If it is equal, the Continuous Compliance Engine will read from the record the fields subordinate to PERSON-DET and will apply any masking algorithms specified on those fields. Similarly, a redefine condition with the value `[CUST-TYPE]='C'` should be applied to the COMP-DET field. Exactly one of the conditions should be evaluated to `true` for each group of redefined fields.

For example, a copybook might have fields A, B REDEFINES A, and C REDEFINES A. Of the Redefine Conditions attached to A, B, and C, one and only one should be evaluated to be true for each record.

6.8.3.1 Entering a Redefine condition

1. In the Format page for the file, click the Actions (...) button to the right of the corresponding field with a **REDEFINED** or **REDEF** type, then select the **Edit Redefine Condition** option  – an edit window will appear.



The screenshot shows the Continuous Compliance interface for the file 'Demo.cbl'. The top navigation bar includes 'Environments', 'Monitor', 'Settings', 'Admin', and 'Audit'. Below the navigation bar, there is an information message: 'Changes to mainframe formats are global and will be applied to all rule sets utilizing them.' The main content area shows a table with columns: Name, Domain, Algorithm, Redefine Condition, Redefines, and Actions. The table is organized into three main sections:

- CS-CUSTOMER-RECORD (16)**: Contains the field 'CUST-TYPE' with an eye icon and an edit icon.
- PERSON-DET (7) REDEFINED**: Contains the field 'PERSON-DET' with a camera icon, an edit icon, and a 'Redefine Condition' of '[CUST-TYPE]='P''.
- COMP-DET (6) REDEF**: Contains the field 'COMP-DET' with a camera icon and an edit icon, and a 'Redefines' value of 'PERSON-DET'. Below this are several other fields: 'COMP-ENTITYNM', 'COMP-ADDRESS1', 'COMP-CITY', 'COMP-STATE', 'COMP-ZIP', and 'COMP-PHONE', each with an eye icon and an edit icon.

2. Enter a condition in the text box that appears. This is the expression that, when evaluated to true, causes the subordinate fields to be read and (if they have algorithms assigned) masked.
3. Click **Save**.

Edit Redefine Condition ✕


Field Name
PERSON-DET

Redefine Condition

[CUST-TYPE]='P'

Cancel Save

4. For viewing Redefine Condition select  icon under the Actions column.

 Redefine conditions can be managed only via the **Formats** page, the **Inventory** page for Mainframe does not allow adding and updating redefine conditions.

6.8.3.2 Format for redefine conditions

Redefine conditions allow fields to be compared against either number or string literals. Square brackets [] enclosing a field name indicate a variable, which takes on the value of the named field:

```
[Field1] = 'An example String'
```

String literals can be enclosed in either single or double quotes. For fields that are numeric (e.g. PIC S99V9), the operators <, <=, >, and >= can be used in addition to the =operator:

```
[Field2] <= -10.5
```

Also, conditions can be joined using AND, OR, and NOT to form more complex conditions:


```
([Field3] > 2.5 AND [Field3] < 10) OR NOT [FIELD4] = 'Z'
```

6.8.4 Constructing file formats for upload

6.8.4.1 Delimited files

To import the file format for a **delimited file**, create a text document with the field names listed one per line describing the structure of the file to Delphix. The screenshot below shows the contents of an example delimited file format.

```
First_Name
Last_Name
DOB
SSN
Address
City
State
Zip_Code
```

 Notice there is no header and only a list of values.

Delimited file mismatch between format and data

Suppose the following delimited file format is being used, where the delimiting character on input is `,`.

```
FieldOne
FiledTwo
FiledThree
```

If the input data does not match the format:

- **Input Field Count < Format Field Count**
After masking, delimiters will be appended to match the total fields with file format. For example:
 - **Input data:** Data1, Data2
 - **Result after masking:** Masked1, Masked2,


- Note, one extra delimiter is added to match with the file format.
- **Data Field Count > Format Field Count**
After masking, the extra fields in the delimited file will be lost. For example:
 - **Input data:** Data1, Data2, Data3, Data4
 - **Result after masking:** Masked1, Masked2, Masked3

6.8.4.2 Fixed-width files

For fixed-width files, import a text file that describes the structure of the file to Delphix Continuous Compliance.

To input the file format for **fixed-width files**, create a text document with the field names and the length of each field, one combination per line. The screenshot below shows an example of content for fixed-width formats. In this format, the field name is followed by the length of the field, separated by a comma.

```
First_Name,20
Last_Name,30
DOB,10
SSN,11
Address,30
City,20
State,2
Zip_Code,10
```

 Note, there is no header – only a list of values.

Fixed-width file mismatch between format and data

For fixed-width files, caution should be taken to ensure that the field length is accurate. An incorrect field length will result in masking a field with the incorrect offset, which would have the consequence of not masking what was intended.

Multi-byte characters

For fixed-width files, field length is determined by the number of characters, rather than the number of bytes.

6.8.4.3 XML files

For XML file formats, you can use the data file itself or a subset of the file you want to mask as the format.

```
<?xml version="1.0" ?>
<catalog>
  <book id="1">
    <author>Soon</author>
    <title>A Comprehensive Guide</title>
    <genre>Computer</genre>
    <price>49.95</price>
    <publish_date>2001-04-16</publish_date>
    <description>DS explored in depth, are
    integrated into a comprehensive development
    environment.</description>
  </book>
</catalog>
```

6.8.4.4 JSON files

For JSON file formats, you can use the data file itself or a subset of the file you want to mask as the format.

```
[
  {
    "id":1,
    "first_name":"abc",
    "last_name":"xyz",
    "email":"abc.xyz@example.com",
    "gender":"Female",
    "ip_address":"10.0.0.1",
    "dob":"2002-07-19"
  }
]
```

6.8.4.5 Mainframe files

For Mainframe files, you can use the copybook file as the format.

```
01 CUSTOMER.
05 EMAIL PIC X(30).
05 DETAILS.
10 NAME.
15 LAST-NAME PIC X(15).
15 FIRST-NAME PIC X(8).
10 ADDRESS.
15 STREET PIC X(15).
15 CITY PIC X(15).
15 STATE PIC XX.
15 ZIP PIC 9(5).
```

6.8.5 Import, edit, and delete formats

6.8.5.1 Import file formats

For all file types other than Mainframe, you can import the file format via the **+ File Format** button, located in the **Files** tab. This will import the file directly into the Continuous Compliance Engine.

1. From **Home**, navigate to the **Settings** page and click into the **Data Formats** section on the left, then click the **Files** tab at the top.
2. Click the **+ File Format** button on the top-right, a window with the same name will appear.
3. Select a **Format Type** and click **Next**.

Add File Format



- Format Type
- Import Format
- Header & Footer
- Summary

Format Type

Specify the format type.

Format Type
Delimited

Delimited Formatting Example

The following is a sample file content for Delimited file format. With these formats just the field name is provided. Notice there is no headers and only a list of values.

```
First_Name
Last_Name
DOB
SSN
Address
City
State
Zipcode
```

Cancel
Back
Next
Save



Formatting Examples are shown in the wizard, based on each format type.

4. Import a format by clicking **Choose File**. The name of the file will be the name of the file format.
5. Browse for the file from which the fields will be imported, then click **Next**.

- a. The contents of the imported file vary for delimited, fixed-width, copybook (Mainframe), XML, and JSON file types.

Add File Format




- Format Type
- **Import Format**
- Header & Footer
- Summary

Import Format

Import a format as suggested in formatting examples in the previous step. This will apply to the default record type of this file format.

Choose File

default_delimited.fmt 

Cancel Back **Next** Save



Removing a selected file

If you accidentally selected an incorrect file, simply click the **x** button to the right of the file name and repeat the selection steps above.

6. Optionally, for only the delimited or fixed-width formats, users will see a step to configure the number of header or footer records for the file. Click **Next** with or without setting the header or footer, as needed.

Add File Format



- Format Type
- Import Format
- Header & Footer**
- Summary

Header & Footer

Specify the number of header and footer records for file format.

Cancel Back **Next** Save

7. View the summary on the last step to confirm the changes.
8. Click **Save** at any point after importing the format.

6.8.5.2 Import **Mainframe** formats

For Mainframe data sets, you can import the file format via the **+ Mainframe Format** button, located in the **Mainframe** tab. This will import the copybook directly into the Continuous Compliance Engine.

1. From **Home**, navigate to the **Settings** page and click into the **Data Formats** section on the left, then click the **Mainframe** tab at the top.
2. Click the **+ Mainframe Format** button on the top-right, a window with the same name will appear.

Settings > Data Formats

Data Formats

+ Mainframe Format

File Mainframe

ID ↑	Name	Type	Actions
1	UD1M7X0I.cbl	Copybook	
2	DY0RG3BU.cbl	Copybook	
3	U30DRU8D.cbl	Copybook	
4	KUBQUDH1.cbl	Copybook	
5	V2XR5B7S.cbl	Copybook	
6	HV8WQ5TM.cbl	Copybook	
7	OYDO0K0.cbl	Copybook	
8	2I6DNUXC.cbl	Copybook	
9	Demo.cbl	Copybook	

Displaying 1 to 9 of 9

3. The **Format Type** will be preselected with **Copybook**, click **Next** to continue.

Add Mainframe Format

Format Type

Specify the format type.


Format Type
Copybook

Copybook Formatting Example

The following is sample file content for a mainframe copybook format.

```
01 CUSTOMER.  
05 EMAIL PIC X(30).  
05 DETAILS.  
10 NAME.  
15 LAST-NAME PIC X(15).  
15 FIRST-NAME PIC X(8).  
10 ADDRESS.  
15 STREET PIC X(15).  
15 CITY PIC X(15).  
15 STATE PIC XX.  
15 ZIP PIC 9(5).
```

Cancel Back Next Save

 **Formatting Examples** is shown in the wizard for copybook.

4. Import a format by clicking **Choose File**.
5. Browse for the format file and click **Next**.

Add Mainframe Format ✕


○ Format Type

● **Import Format**

○ Summary

Import Format

Import a format as suggested in formatting examples in the Format Type step.

Entertainment.cbl 

Removing a selected file

If you accidentally selected an incorrect file, simply click the `:cancel:` button to the right of the file name and repeat the selection steps above.

6. View the summary on the last step to confirm the changes.
7. Click **Save** at any point after importing the format.

6.8.5.3 Edit a **file** format

Selecting the hyperlinked **Name** of a format in the corresponding column will initiate the file formats edit screen.

CONTINUOUS COMPLIANCE
24.0.0.0
Environments Monitor Settings Admin Audit
👤 ?

Algorithms

Classifiers

Data Formats

Domains

Expressions

JDBC Drivers

Profile Sets

Settings > Data Formats

Data Formats

+ File Format

File
Mainframe

ID ↑	Name	Type	Actions
1	996QQGFZO.fmt	Delimited	✖
2	VI03HE18.fmt	Fixed Width	✖
4	D21RLWF6.xml	XML	✖
5	8MFLR5WL.fmt	Delimited	✖
6	HKH1PQ0B.json	JSON	✖
7	GOL7IZ7F.json	JSON	✖
9	00YRAG2C.fmt	Delimited	✖
10	53W8WINM.xml	XML	✖

Displaying 1 to 8 of 22



The file inventory relies on the file format it represents. As a result, any changes to the file inventory affect all data files using that file format. Thus, altering the file inventory impacts all files conforming to that format across the application. This is applicable to all the file types Fixed-Width, Delimited, Mainframe, JSON & XML.

6.8.5.4 Delete a **file** format

Click the **Delete** icon under the **Actions** columns to the right of the corresponding format name in the list. The user will be prompted for confirmation.

Settings > Data Formats

Data Formats

+ File Format

File Mainframe

ID ↑	Name	Type	Actions
1	996QGFZO.fmt	Delimited	
2	VI03HE18.fmt	Fixed Width	
4	D21RLWF6.xml	XML	
5	8MFLR5WL.fmt	Delimited	
6	HKH1PQ0B.json	JSON	
7	GOL7IZ7F.json	JSON	
9	00YRAG2C.fmt	Delimited	
10	53W8WINM.xml	XML	

Displaying 1 to 8 of 22

6.8.6 Add, view, edit, and delete fields for file formats

6.8.6.1 Add file fields

New file fields can only be defined in delimited, fixed-width, and JSON file formats. Use the following steps to create a new field.

1. Navigate to **Settings** → **Data Format**, click the hyperlink of a format **Name** from the corresponding column. From there, an inventory-like screen will be available with several listed fields.



You can also navigate to the **Data Format** page from any **Rulesets** Inventory screen by clicking on the **Edit File Format** button. An information banner is added on the Environments Inventory page to help user with navigation.

CONTINUOUS COMPLIANCE 24.0.0.0

Environments Monitor Settings Admin Audit

Information
Changes to format files are global and will be applied to all rule sets utilizing them.

Settings > Data Formats > Details

00YRAG2C.fmt

+ Field + Record Type + Qualifiers + Header & Footer Actions

Reset

Record Type	Name	Position ↑	Domain	Algorithm	File Format	Automatic Up...	Actions
▼ All Records (6)							
	ID	1				Enabled	
	UNMASKED_00	2				Enabled	
	FIRSTNAME_00	3	FIRST_NAME	dlp-core:FirstName		Disabled	
	LASTNAME_00	4	LAST_NAME	dlpx-core:LastName		Disabled	
	DATA_00	5			CTJIOQ0W,json	Disabled	
	DATA_01	6				Enabled	
▼ new_record (2)							
	Name	1				Enabled	
	Email	2	EMAIL	dlpx-core:Email Uni...		Disabled	

2. Click on **+ Field** button appearing on the left corner just above the grid, to open an **Add Field** dialogue.

Add Field ✕


Formatting

Masking

Uncheck 'Enable Automatic Updates' to make Masking assignments editable.

Enable Automatic Updates

i

-  If you select a DATESHIFT algorithm or multi-column algorithms, more fields will appear in the dialogue. A DATESHIFT algorithm allows you to pick a date format from the dropdown list or specify your own date format.

3. Fill out the form and click **Save**.
4. The **Field Name** or **JSON Path** and all inputs in the **Formatting** section are mandatory.
5. The **Masking** section is optional and can be edited later as well.
6. Newly added fields will be reflected under the selected record type group on the page.



- **Enable Automatic Updates** will always be checked by default. Generally, If the user is making masking assignments then they don't want assignments to be overridden by the Profiler. To make masking assignments uncheck 'Enable Automatic Updates'.
- Users can uncheck 'Enable Automatic Updates', update masking assignments, and then check if they want to set the domain and algorithm but still allow the Profiler to change the assignment.

6.8.6.2 View, edit, or delete a file field

1. The fields can be **viewed**, **edited**, or **deleted** using the **Actions** to the right of the corresponding field.

Record Type	Name	Position ↑	Domain	Algorithm	File Format	Automatic Up...	Actions
All Records (6)							
	ID	1				Enabled	
	UNMASKED_00	2				Enabled	
	FIRSTNAME_00	3	FIRST_NAME	dplx-core:FirstName		Disabled	
	LASTNAME_00	4	LAST_NAME	dplx-core:LastName		Disabled	
	DATA_00	5			CTJIOQ0Wjson	Disabled	
	DATA_01	6				Enabled	
new_record (2)							
	Name	1				Enabled	
	Email	2	EMAIL	dplx-core:Email Uni...		Disabled	

2. **View** - is a read-only pre-filled dialogue (similar to Add Fields) and the user can not make edits.
3. **Edit** - prompts a pre-filled dialogue (similar to Add Fields) and the user can make edits as needed.
4. **Delete** - action, the user will be prompted for confirmation to Delete a field. Fields from XML and Mainframe formats cannot be deleted.
5. **Edit File Format** - action is used to edit the file format assigned to the field. It is applicable only in the case of Delimited file formats.



A field's masking settings (automatic updates setting, domain assignment, and algorithm assignment) can also be edited from the Inventory screen.

6.8.7 File structure for masking

This section describes the document structures that can be used for masking data files, including XML, JSON, and Mainframe. Use the pages below to get an understanding of how these files are structured for upload to the Continuous Compliance Engine.

- [XML structure \(see page 446\)](#)
- [JSON structure \(see page 448\)](#)

6.8.7.1 XML structure

This page aims to provide general tips for handling XML documents for masking, including editing XML structures and leveraging XPath to target data with precision. With the creation and manipulation of XML file formats, masked data should be handled per requirements.

6.8.7.1.1 Understanding XML structure

An XML document is both human-readable and machine-readable, which allows it to serve as a common medium for information exchange across diverse systems.

6.8.7.1.1.1 Definitions

- **Prolog** (optional): The prolog appears at the beginning of the XML document and contains metadata about the document itself, such as the XML version and the character encoding (e.g., `<?xml version="1.0" encoding="UTF-8"?>`).
- **Elements**: Elements are the building blocks of XML documents, denoted by tags. An element can contain text, other elements, or a mix of both. Elements are used to encase data points in a document, and typically consist of a start tag, content, and an end tag (e.g., `<name>John Doe</name>`).
- **Attributes**: Attributes provide additional information about elements. They are included within the start tag of an element and usually come in name/value pairs (e.g., `<postcode id="12345"/>`).
- **Root Element**: Every XML document must contain a single root element that encases all other elements. The root element provides a container for all data in the document to enforce a hierarchical structure.

6.8.7.1.1.2 Hierarchical structure

XML documents are inherently hierarchical, a feature that allows them to represent complex data structures effectively.

- **Parent and child elements:** Elements nested within other elements create parent-child relationships. This structure allows XML to represent complex data relationships naturally (e.g., a `Person` element might contain `FirstName`, `LastName`, and `ContactDetails` as child elements).
- **Sibling elements:** Elements that are at the same level of the hierarchy and share the same parent are called siblings. Sibling elements often represent similar types of data or repeated elements in a list (e.g., multiple `Person` elements within a `People` root element).

6.8.7.1.1.3 Use of XML in data masking

Masking operations on XML files typically involve modifying the content of elements or attributes to obfuscate sensitive data while maintaining the structural integrity of the document. Using XML's hierarchical nature, you can selectively apply masking rules to specific parts of the document without disrupting its overall format, to keep the masked data useful for testing or development purposes.

This structured approach not only helps in maintaining the logical grouping of data but also ensures that data masking can be done efficiently and effectively, targeting only those elements that contain sensitive information.

6.8.7.1.2 XML example

```
<Person>
  <First_Name>John</First_Name>
  <Last_Name>Doe</Last_Name>
  <DOB>1968-11-24</DOB>
  <State></State>
  <Postcode id=""/>
</Person>
```

6.8.7.1.3 Understanding XPath

XPath stands for XML Path Language, designed to use queries for selecting nodes from an XML document.

- **Expressions:** XPath uses path expressions to navigate through elements and attributes in an XML document.
- **Nodes:** In XPath, everything is treated as nodes, including elements, attributes, and even text.
- **@:** In XPath, this symbol is used to select attributes. For example, `@id` selects the `id` attribute of the context node.
 - To select the `name` attribute of an `employee` element, you would use the XPath expression `/employee/@name`.

6.8.7.1.3.1 XPath example

```
/Person
/Person/First_Name
/Person/Last_Name
/Person/DOB
/Person/State
/Person/Postcode
/Person/Postcode@id
```

6.8.7.2 JSON structure

In the context of data masking, a JSON file can be uploaded and used to specify which data fields (e.g. PII) should be masked to protect sensitive information. Accurately constructing a JSON file for data masking involves understanding the structure of JSON, knowing what fields need masking, and correctly using JSON Path syntax to specify those fields.

6.8.7.2.1 Understanding JSON structure

JSON data is presented in name/value pairs, with fields and values enclosed within curly braces `{}` for objects or square brackets `[]` for lists. Each data element within an object is separated by a comma. The correct format must be used; otherwise, it can result in errors during the masking process.

6.8.7.2.1.1 Definitions

- **Groups/fields:** Only specify fields that require masking. Fields not mentioned will not be masked. The presence of the field is vital; its value can be `null` since it is not relevant to the masking process—this means the value is merely a placeholder.
- **Objects:** In JSON, an object is a collection of key-value pairs where each key (or field) is a string, and the value can be a string, number, boolean, array, object, or null. Objects are delineated by curly braces `{}`. Each key-value pair in an object is separated by a comma, and the key and value in each pair are separated by a colon.
- **Lists (Arrays):** In JSON, an array is an ordered list of values. An array begins with a left square bracket `[` and ends with a right square bracket `]`. The values in the array are separated by commas and can include any data type—strings, numbers, objects, other arrays, etc. Arrays do not associate values with keys.
- **Fields:** In the context of JSON, fields refer to the names in key-value pairs within an object. Fields act as identifiers and are always strings. Each field in an object should be unique within that object. Fields provide a way to access the values they are associated with.
- **Values:** Values in JSON can be of any data type: strings, numbers, objects, arrays, booleans, or null. Values are the data assigned to fields in objects or the elements within an array. They represent the actual data contained within the JSON structure.

6.8.7.2.1.2 Example 1: List format with null values

```
[
  {
    "First_Name": "John",
    "Last_Name": "Doe",
    "DOB": "1968-11-24",
    "SSN": "123-45-6789",
    "Address": "123 4th Ave",
    "City": "Allentown",
    "State": null,
    "Zipcode": null
  }
]
```

6.8.7.2.1.3 Example 2: Object format with null values

```
{
  "First_Name": "John",
  "Last_Name": "Doe",
  "DOB": "1968-11-24",
  "SSN": "123-45-6789",
  "Address": "123 4th Ave",
  "City": "Allentown",
  "State": null,
  "Zipcode": null
}
```

6.8.7.2.2 Understanding JSON path

A JSON Path allows you to specify and locate specific parts of a JSON document. The syntax includes a variety of operators and expressions, but at its most basic, it starts with `$` to represent the root object or element.

- The `$` symbol denotes the root element of the JSON document.
- `['FieldName']` targets a specific field by name.

6.8.7.2.2.1 File format XPath examples

For **Example 1** (List format):

- `[$*]['First_Name']` targets the "First_Name" field of each object in the list.

For **Example 2** (Object format):

- `['First_Name']` directly targets the "First_Name" field in the root object.

6.8.7.2.3 JSON file formats

When profiling and masking JSON files, file formats must be provided to determine the JSON structure of the data file. To prevent overloading the masking engine, uploading a file format that only contains the necessary JSON structure is suggested. Only one record per JSON structure is needed. An example of a JSON file format and data file pair is below.

6.8.7.2.3.1 Example JSON file format

```
[
  {
    "id": 1,
    "name": {
      "first": "Alex",
      "last": "Johnson"
    },
    "address": {
      "street": "Main Street",
      "city": "Albany",
      "state": "MA",
      "zip": 04739
    }
  }
]
```

6.8.7.2.3.2 Example JSON data file

```
[
  {
    "id": 1,
    "name": {
      "first": "Jay",
      "last": "Green"
    },
    "address": {
      "street": "Jaeg Highway",
      "city": "Nacihju",
      "state": "NV",
      "zip": 24077
    }
  },
  {
    "id": 2,
```



```
[
  {
    "name": {
      "first": "Lottie",
      "last": "Bush"
    },
    "address": {
      "street": "Cacdol Key",
      "city": "Buwiju",
      "state": "OK",
      "zip": 86168
    }
  },
  {
    "id": 3,
    "name": {
      "first": "Henry",
      "last": "Rice"
    },
    "address": {
      "street": "Suvek Turnpike",
      "city": "Dennelema",
      "state": "IN",
      "zip": 48587
    }
  }
]
```


The JSON file format contains the JSON structure and fields to profile all of the records in the data file.

6.9 Managing inventories



An **inventory** describes all of the data present in a particular rule set and defines the methods that will be used to secure it. Inventories typically include a table or file name, column/field name, data classification, and the chosen algorithm.

6.9.1 The inventory screen

From anywhere within an environment, click on the **Rule Sets** tab to see the rule set screen. This displays the list of rule sets for the environment. Click on the **Rule Set name** to access the given rule set's inventory screen.


CONTINUOUS COMPLIANCE 22.0.0.0

[Environments](#)
[Monitor](#)
[Settings](#)
[Admin](#)
[Audit](#)

Information
The Inventory Tab has been removed from this screen. To access the inventory of a rule set, click on the hyperlink associated with its name on the Rule Set listing screen.

[Environments](#) > Demo_Environment
 + Rule Set

Demo_Environment
Application: Demo_Application | Purpose: MASK | Approval: Enabled

Rule Set ID ↑	Name	Connector Name	Source Type	Metadata	Refresh/Save	Actions
23	MySQLRuleSet	MySQLDBConnector	Database	mysql	✓	⋮
24	HanaRuleSet	HanaConnector	Database	extended	✓	⋮
25	PostgresRuleSet	PostgresDBConnector	Database	POSTGRESQL	✓	⋮
26	LocalPostgresRuleSet	LocalhostPostgresDB	Database	POSTGRESQL	✓	⋮
27	OracleRuleSet	OracleConnector	Database	oracle	✓	⋮
28	WholeFileRs	WholeFileConnector	Fixed Width	File	N/A	⋮
29	JSONFileRs	JSONFileConnector	JSON	File	N/A	⋮
30	FixedFileRs	FixedFileConnector	Fixed Width	File	N/A	⋮
31	XMLFileRs	XMLFileConnector	XML	File	N/A	⋮
32	DelimitedFileRs	DelimitedFileConnector	Delimited	File	N/A	⋮


CONTINUOUS COMPLIANCE 24.0.0.0

[Environments](#)
[Monitor](#)
[Settings](#)
[Admin](#)
[Audit](#)




[Back](#)

[env_W3466FEU](#) > **con_7Z1OKJH0**
File Format
Actions ▾

File Format

Reset ✕

Record Type	Name	Position ↑	Domain	Algorithm	File Format	Automatic Upd...	Actions
▼ All Records (6)							
	ID	1				Enabled	👁️ ✎️ 🗑️
	UNMASKED_00	2				Enabled	👁️ ✎️ 🗑️
	FIRSTNAME_00	3	FIRST_NAME	dlpx-core:FirstName		Enabled	👁️ ✎️ 🗑️
	LASTNAME_00	4	LAST_NAME	dlpx-core:LastName		Enabled	👁️ ✎️ 🗑️
	DATA_00	5	TELEPHONE_NO	dlpx-core:Phone Uni...		Enabled	👁️ ✎️ 🗑️
	DATA_01	6	EMAIL	dlpx-core:Email Uniq...		Enabled	👁️ ✎️ 🗑️

Displaying 1 to 7 of 7


The rows on the screen can be filtered or sorted by the various informational fields by clicking on the respective field. More information on grid filtering and sorting can be found [here](#) (see page 232).



- View permission of Inventory and File-Format is required to select a rule set and view the inventory.
- Sorting is applied within each group when rows are grouped by **Record types** or **Table name**.
- The Mainframe grid does not have sorting enabled because the memory is shared across record types and dataset masking is sensitive to the order of the field in the Copybook format.
- To show **Masked** column on the grid which shows information if the particular column/field is assigned with an algorithm, hover on the **Algorithm** column, click on the ≡ icon appearing on the right side corner of the dialogue, and choose the **Not blank** option from the drop down.

6.9.2 Assigning algorithms

To set criteria for sensitive columns or fields:

1. Click the Edit icon  under the **Actions** column to the right of the corresponding column or field for the **Edit Field/Column** dialogue to appear.

Edit Field ✕

Field Name
FIRSTNAME_00

Notes (Optional) //

Formatting

Record Type
All Records ▼

Position
3

Masking

Uncheck 'Enable Automatic Updates' to make Masking assignments editable.

Enable Automatic Updates

Data Model
Plain ▼ i

Select Domain
FIRST_NAME ▼

Select Algorithm
dlpx-core:FirstName ▼

Cancel
Save



- From the Inventory screen, the Edit Field/Column dialogue allows you to edit properties only under the **Masking** section.
- To edit the **Formatting** properties of a file field, go to the **Settings** → **Data Formats** and edit the corresponding format. You can also navigate to **Settings** → **Data Formats** from Inventory screen using the **Edit File Format** button present at the right corner, just above the grid.

2. You can add/remove notes in the **Notes** text field.
3. Choose **Enable Automatic Updates**:
 - a. **Check** (Enabled)
The default setting. A profiling job can determine or update whether to mask a column.

b. **Uncheck** (Disabled)

The user decides whether to mask/unmask a column. The user's choice overrides the profiling job.



- **Enable Automatic Updates** will always be checked by default. Generally, If the user is making masking assignments then they don't want assignments to be overridden by the Profiler. To make masking assignments uncheck 'Enable Automatic Updates'.
- Users can uncheck 'Enable Automatic Updates', update masking assignments, and then check it if they want to set the domain and algorithm but still allow the Profiler to change the assignment.

Edit Field ✕

Notes (Optional) //

Formatting

Record Type
All Records ▼

Position
3

Masking

Uncheck 'Enable Automatic Updates' to make Masking assignments editable.

Enable Automatic Updates

Data Model
Plain ▼

i

Select Domain
FIRST_NAME × ▼

Select Algorithm
dlpx-core:FirstName × ▼

Cancel
Save

3. From the **Domain** drop-down list, select the appropriate sensitive data element type.
4. The Continuous Compliance Engine defaults to a **Masking Algorithm**, as specified in the Settings screen. If necessary, you can override the default algorithm.
 - a. To select a different masking algorithm, choose one from the **Algorithm** drop-down list. For detailed descriptions of these algorithms, see the [Out-of-the-box algorithm instances](https://masking.delphix.com/docs/latest/out-of-the-box-algorithm-instances)¹⁶⁴ article.

¹⁶⁴ <https://masking.delphix.com/docs/latest/out-of-the-box-algorithm-instances>

[-] If you select a **DATESHIFT** algorithm and are not masking a datetime or timestamp column, you must specify a **Date Format** (this field only appears if a **DATESHIFT** algorithm is selected from the Masking Algorithm dropdown). The default format is **YYYY-MM-DD** in the legacy UI. A dropdown provides the capability to add a new date format or select from the existing list in the dropdown. Click on the **i** or **?** icon next to the dropdown for more suggestions on valid formats.


4. Once complete, click **Save**, which must be done for any edits to take effect.

6.9.3 Managing database inventory settings

- Database inventory screen lists **Columns** from all the tables in the rule set, the number beside each table name in parentheses is the total number of columns in that particular table.
- If a database column is a **Primary Key, Foreign Key, or Index**, it will be indicated below the column name.
- Metadata for the database column appears under the **Data Type** column including its **Length** mentioned in parentheses. This information is read-only.
- By default, only **Table Name, Column Name, Data Type, Domain, Algorithm, and File Format** columns will be displayed in the database inventory screen.
- If the [Inventory Approval Workflow](#) (see page 0) is enabled for the environment, a **Status** button appears just above the grid if any changes like **Add/Edit/Delete/Import** inventory are done to any column properties.

Table	Column	Data Type	Domain	Algorithm	File Format	Actions
TEST1 (3) Filter Condition						Filter, Sort, Expand
	FIRST_NAME	VARCHAR2 (2000)	FIRST_NAME	dlpx-core:FirstName		Edit
	ID Primary Key, Index	NUMBER (0)				Edit
	LAST_NAME	VARCHAR2 (2000)			json_filter.json	Edit, Import
TEST10 (3) Logical Key						Filter, Sort, Expand
TEST2 (3)						Filter, Sort, Expand
	FIRST_NAME	VARCHAR2 (2000)				Edit
	ID Primary Key, Index	NUMBER (0)				Edit

Displaying 1 to 9 of 19

- Logical Key, Filter, and Custom SQL can also be updated for individual tables on the Inventory screen. More information on these fields can be found [here \(see page 0\)](#).
 - Click  for updating the **Logical Key** under the **Actions** column on the row with the table name.


Add/Edit Key Column ✕

⚠ Caution
Make sure that the column(s) specified exist in the table. The masking jobs will fail if the key column does not exist in the table.

Table: TEST1

Key Column(s)

Cancel Save

- Click  for updating **Filter** Condition under the **Actions** column on the row with the table name.

Add/Edit Filter ✕

⚠ Caution
Make sure that the filter provided is valid. There will be adverse effects if the filter is not valid.

Table: TEST1

Filter

ID > 10

Cancel Save

- Click [<>](#) for updating **Custom SQL** under the **Actions** column on the row with the table name. It can show system-generated custom SQL.

Add/Edit Custom SQL ✕

⚠ Caution
Custom SQL cannot be applied as Filter is already applied to the table. To apply custom SQL, remove the Filter first. Below Statement is the System generated SQL.

Table: TEST1

Custom SQL

```
SELECT ROWID, "ID", "LAST_NAME", "FIRST_NAME" FROM  
"DLPXDBORA"."TEST1" where ID > 10
```

Cancel Save

6.9.4 Managing a fixed-width or delimited file inventory settings

- Search/select a **file** or **file format** under the **File Format** dropdown to create or edit the inventory of sensitive data. The **Record Types** and **Fields** for that specific file will appear in the grid below.
- Fields are listed in groups of **Record Types**, which can be collapsed and expanded by clicking on the down arrow icon \downarrow beside each Record Type Name. Users can also **Expand All** and **Collapse All** using the **Actions** button in the right corner, just above the grid.
- The count next to the Record Type in parentheses shows the total number of fields in that record.

Record Type	Name	Position	Domain	Algorithm	File Format	Automatic Upd...	Actions
All Records (6)							
	ID	1				Enabled	
	UNMASKED_00	2				Enabled	
	FIRSTNAME_00	3	FIRST_NAME	dlpX-core:FirstName		Enabled	
	LASTNAME_00	4	LAST_NAME	dlpX-core:LastName		Enabled	
	DATA_00	5	TELEPHONE_NO	dlpX-core:Phone Uni...		Enabled	
	DATA_01	6	EMAIL	dlpX-core:Email Uniq...		Enabled	

• To add a new Field, Record Type to the inventory, or to manage qualifiers or headers & footers of the selected file format, click on the **Edit File Format** button in the right corner. Navigate to the selected Settings Format screen and use **Add Field**.

6.9.5 Managing a JSON file inventory settings

- Search/select a **file** or **file format** under the **File Format** dropdown to create or edit the inventory of sensitive data.

Environments > env_YH2M983L > con_RRJ60IZJ

con_RRJ60IZJ

File Format: 2YG8EQRK.json

Reset

JSON Path ↑	Domain	Algorithm	Group Number	Logical Field Name	Automatic Updates	Actions
[\$*]					Enabled	
[\$*]['address']					Enabled	
[\$*]['address']['street_address']	NULL_SL	dipx-core:CM Alpha-Nu...			Enabled	
[\$*]['dob']	NULL_SL	ALG_Y18RJV6A			Enabled	
[\$*]['email']	NULL_SL	ALG_Y18RJV6A			Enabled	
[\$*]['first_name']					Enabled	

Displaying 1 to 6 of 6

• To add a new JSON path to the inventory, click on **Edit File Format** button in the right corner that will navigate to the respective selected Edit Formats screen, where you can use the **Add Field** button.

• Profiling is not supported on the JSON file rule sets.

6.9.6 Managing an XML file inventory settings

- Search/select a **file** or **file format** under the **File Format** dropdown to create or edit the inventory of sensitive data.
- The fields are displayed with XPath in a flat-grid structure and are, by default, sorted by XPath.

Environments > env_60QEKHNO > con_YN1U40F2

con_YN1U40F2

File Format: D21RLWF6.xml

XPath ↑	Domain	Algorithm	Group Number	Logical Field Name	Automatic Updates	Actions
/root					Enabled	
/root/data					Enabled	
/root/data/DATA_00	TELEPHONE_NO	dlpx-core:Phone Unique			Enabled	
/root/data/DATA_01	EMAIL	dlpx-core:Email Unique			Enabled	
/root/data/FIRSTNAME_00	FIRST_NAME	dlpx-core:FirstName			Enabled	
/root/data/LASTNAME_00	LAST_NAME	dlpx-core:LastName			Enabled	
/root/data/UNMASKED_00					Enabled	
/root/data@ID					Enabled	

Displaying 1 to 8 of 8

• XML attributes can be identified by “@” in the XPath.

6.9.7 Managing Mainframe inventory settings

- Search/select a **file** or **file format** under the **File Format** dropdown to create or edit the inventory of sensitive data.
- Fields are listed in groups by a parent field which can be collapsed and expanded by clicking on the down arrow icon beside the field name. Users can also **Expand All** and **Collapse All** using the **Actions** button in the right corner, just above the grid.
- The count next to the Field name in parentheses shows the total number of children fields.
- If a field is type of **REDEFINED** or **REDEF** then it will be indicated below the field name.

Information
Manage redefine conditions for the fields at the format level by selecting 'Edit Mainframe Format'. You can handle global masking assignments directly on this page.

Environments > env_QAXINBR > rs_MD0S420A

< Back **rs_MD0S420A** Mainframe Format Actions

File Format: Demo.cbl

Name	Domain	Algorithm	Redefine Condition	Redefines	Actions
CS-CUSTOMER-RECORD (16)					
CUST-TYPE					👁️ ✎
PERSON-DET (7) REDEFINED			[CUST-TYPE]="P"		
COMP-DET (6) REDEF				PERSON-DET	
COMP-ENTITYNM					👁️ ✎
COMP-ADDRESS1	ADDRESS	AddrLookup			👁️ ✎
COMP-CITY					👁️ ✎
COMP-STATE					👁️ ✎
COMP-ZIP					👁️ ✎
COMP-PHONE					👁️ ✎

• To edit Redefine conditions, click on the **Edit Mainframe Format** button at the right corner above the grid to navigate to the respective selected Edit Formats screen, where you can edit Redefine conditions. For more information on Redefine condition see the [Managing File Formats](#)¹⁶⁵ article.

• Masking a node with Level 88 children is not supported. A Level 88 node refers to the condition name in the Copybook format. If we allow masking a node with Level 88 children, the dataset file generated after masking will no longer be compatible with its Copybook format.

6.9.8 Importing and exporting an inventory

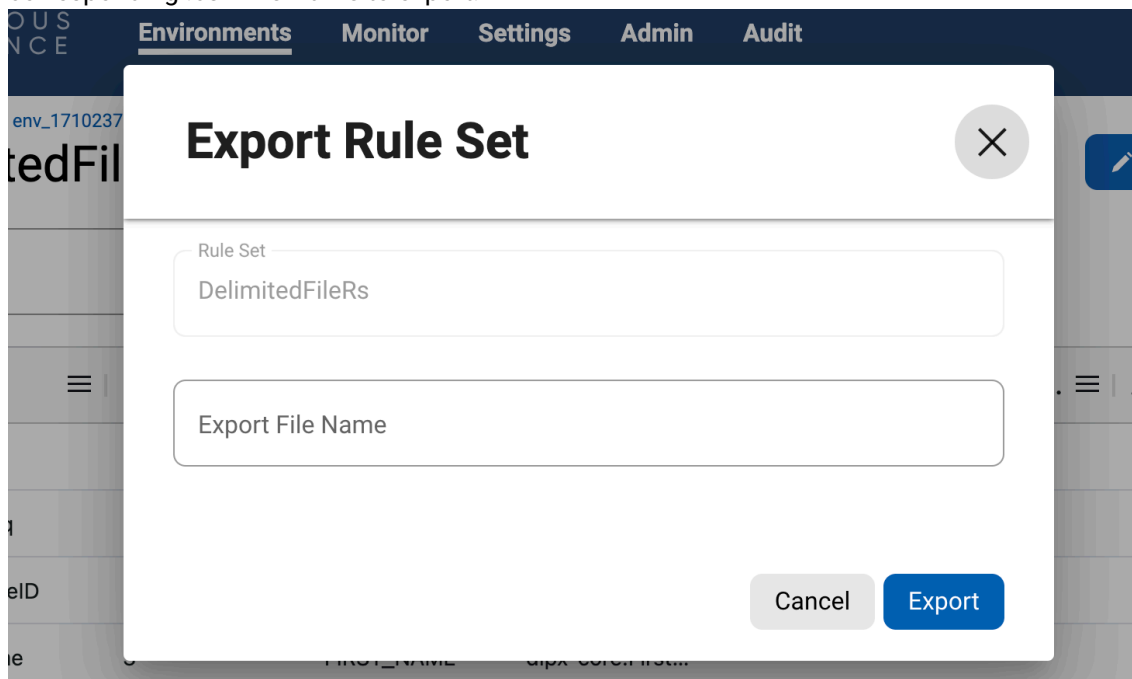
• Importing and exporting an inventory in CSV format is only supported for database, delimited, fixed-width, and XML inventories.

¹⁶⁵ <https://masking.delphix.com/docs/latest/managing-file-formats>

- Database rule-set exported CSV file will contain data sorted by Table Name and Column Name.

6.9.8.1 To export an inventory

1. Click on the **Actions** button in the right corner just above the grid, then choose **Export Rule Set** from the options.
2. The **Export Ruleset** window appears with the name of the currently selected **Rule Set** and provide a corresponding **.csv File Name** to export.

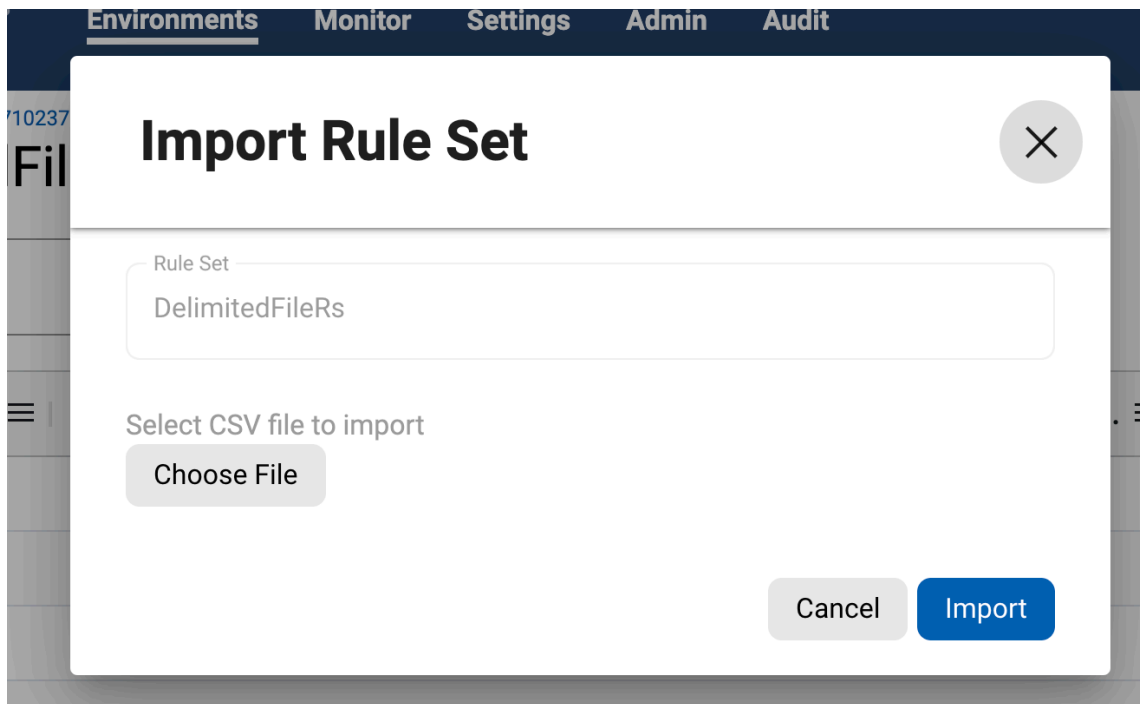


3. Click **Export**.

When the export operation is complete, a .csv file with the name provided earlier will start to be **Downloaded** on the browser.

6.9.8.2 To import an inventory

1. Click on the **Actions** button in the right corner just above the grid, then choose **Import Rule Set** from the options.
2. The **Import Rule Set** window appears with the name of the currently selected **Rule Set**.



3. Click **Choose File** to browse for the exported comma-separated (.csv) file.
4. Click **Import**.

When the import operation is complete, the inventory you imported appears in the **Rule Set** list for this environment.



- Only one rule set can be imported at a time.
- The format of an imported .csv file must exactly match the format of the exported inventory. If you plan to import an inventory, you should export it first and then update the exported file as needed before importing it.
- After importing the inventory to a 10.0.0.0 version or above Compliance Engine from older versions, rule set refresh is mandatory when the inventory has any document store type assignments, or the user needs to perform document store type masking on the columns from the imported inventory.

6.9.9 Document Store Type masking

This feature provides the ability to mask structured documents that are stored in database columns and delimited files. This is done by marking a column/delimited field as **Structured** and assigning a respective **Document Store Type** and **File Format** to it.

With the release of version 10.0.0.0 of the Continuous Compliance Engine, the document store type masking will support automatic datatype identification. This will be done by using the [JDBC SQL Type](#)¹⁶⁶ associated with columns. String and BLOB types will be supported for document store type masking.



With version 10.0.0.0 release

- In the case of existing rule sets, a rule set refresh is mandatory before using Document Store Type masking.
- Masking jobs having rule sets with Document Store Type assignments will need mandatory rule set refresh. Without rule set refresh job will not be allowed to run.
- Masking jobs having rule sets without document store type assignments will not need rule set refresh.
- Rule set refresh is not required for newly created rule sets.



- **For Database columns**

- The database column type should be from one of the following JDBC SQL Types: CHAR , NCHAR , VARCHAR , NVARCHAR , CLOB , NCLOB , LONGVARCHAR , LONGNVARCHAR , BLOB , SQLXML .
- BLOB type will not be supported for MySQL databases.
- SQLXML type will be only supported for Oracle databases.
- The file format must be either XML or JSON.

- **For Delimited file-fields**

- Document store type masking for delimited field is supported when JSON or XML data is enclosed by Enclosure and has enclosure escaping strategy as Double Enclosure .
- Only the double enclosure escaping strategy is supported. Custom enclosure escaping strategy and "enclosure escape character" functionality are not supported for delimited fields with structured data.
- More details on how to assign enclosure to delimited file-rule set can be found [here](#) (see page 400).

Database columns with a supported data type or a Delimited file field provide a setting called **Data Model**, which can be configured as either **Plain** or **Structured**.

As shown in the image below, columns with Plain selected as the Data Model can be masked as a single value by assigning a **Domain** and **Algorithm**.

¹⁶⁶ <https://docs.oracle.com/javase/8/docs/api/java/sql/JDBCType.html>

When the Structured value is selected for the Data Model, a **Document Store Type** and **File Format** can be assigned as shown in the image below.

Edit Properties ✕

Table Name: info_table

Column Name: xml_data

Notes (Optional)

XML CLOB column having XML document data stored in it

Masking

Uncheck 'Enable Automatic Updates' to make Masking assignments editable.

Enable Automatic Updates

Data Model

Structured ▼ i


Document Store Type

XML ▼

Select File Format

books.xml ✕ ▼

Cancel Save

The image below shows the **Inventory** screen for the database rule set with a structured column. To quickly access an assigned File Format from this screen (books.xml in this example), click  icon under the **Actions** column.

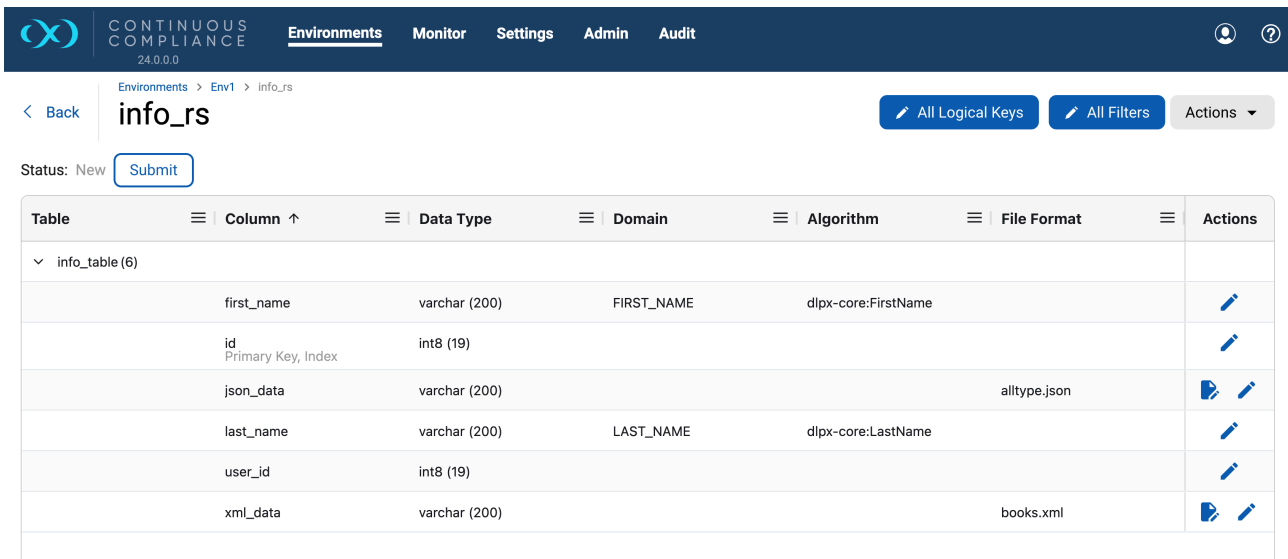


Table	Column ↑	Data Type	Domain	Algorithm	File Format	Actions
info_table (6)						
	first_name	varchar (200)	FIRST_NAME	dlpx-core:FirstName		
	id Primary Key, Index	int8 (19)				
	json_data	varchar (200)			alltype.json	
	last_name	varchar (200)	LAST_NAME	dlpx-core:LastName		
	user_id	int8 (19)				
	xml_data	varchar (200)			books.xml	

Users can assign algorithms to the fields inside the document store selected file format by either clicking on under **Actions** column in the grid or by going to **Settings > Data Format** and clicking on the assigned File Format name.

6.9.9.1 Multi-column algorithm support for document store type masking

With the release of version 10.0.0.0, [Multi-column algorithms](#)¹⁶⁷ will be supported for JSON and XML document store type masking with limited buffer-data size.

Buffer size (in bytes) will be using calculated using the below formula:

$$((\text{Max_memory_of_Job}/\text{No_of_streams_for_job}) * \text{CharStreamingBufferLimitRate}) / 100$$


- The default values will be used when the maximum memory and number of the stream for the job are not defined.
- Buffer-data size is configurable via the application setting `CharStreamingBufferLimitRate` under Mask group settings. For adjusting `CharStreamingBufferLimitRate`, refer to the [Masking API client](#)¹⁶⁸.

The fields having multi-column assignments should not exceed the limit of buffer data size. In case of exceeding the limit of buffer data size, the job will fail. Users can configure buffer size by adjusting `CharStreamingBufferLimitRate` to avoid exceeding the buffer data size issue.

¹⁶⁷ <https://masking.delphix.com/docs/latest/using-multi-column-algorithms>

¹⁶⁸ <https://masking.delphix.com/docs/latest/masking-client>


6.9.9.1.1 Multi-column algorithm with JSON file format

-  Multi-column algorithm is supported for JSON files and JSON in Document Store Type masking.
- Multi-column algorithm is not supported for JSON fields where,
 - JSON field is an array.
 - JSON fields are part of different arrays.
 - JSON fields are on different levels having one or more fields from JSON arrays.

Multi-column algorithm assignment for JSON fields will be validated at the time of assignment. If any of the above combinations are found while assigning a multi-column algorithm, that assignment will not be allowed.

Below is a sample JSON file format with valid and invalid multi-column assignment examples.

JSON Content	JSON Fields having Multi-column assignment	Invalid/Valid
<pre> { "data": { "founders": { "name": ["Founder_Name1", "Founder_Name2"] }, "hospital": { "name": "Hospital_Name", "registered_on": "2001-01-01", "inaugurated_on": "2002-01-01", "found_date": "1905-12-10", "patient": [{ "doa": "2001-07-31", "dob": "1907-08-01", "dol": "2005-04-12", "name": "Patient_Name1", "test_report": [{ "test_id": 1, "test_name": "TSH1", "dot": "2001-08-02", "result": "positive" }] }] }, "registration_id": 932884, "doctor_details": [{ "doj": "2001-07-31", "dol": "2004-01-02", "doctor_name": "Doctor_Name1" }] } } </pre>	$\$[\"data\"][\"founders\"][\"name\"][*]$ $\$[\"data\"][\"registration_id\"]$	Invalid. Multi-column algorithm assignment is not allowed as JSON field 'name' is an array
	$\$[\"data\"][\"hospital\"][\"patient\"][*][\"name\"]$ $\$[\"data\"][\"doctor_details\"][*][\"doctor_name\"]$	Invalid. Multi-column algorithm assignment is not allowed for JSON fields that are part of different JSON arrays. Here 'name' belongs to an array 'patient' and 'doctor_name' belongs to an array 'doctor_details'
	$\$[\"data\"][\"hospital\"][\"found_date\"]$ $\$[\"data\"][\"hospital\"][\"patient\"][*][\"doa\"]$	Invalid. Multi-column algorithm assignment is not allowed for these fields as one of the fields i.e. 'doa' belongs to an array and fields are on different levels.
	$\$[\"data\"][\"hospital\"][\"patient\"][*][\"doa\"]$ $\$[\"data\"][\"hospital\"][\"patient\"][*][\"dol\"]$	Valid
	$\$[\"data\"][\"hospital\"][\"registered_on\"]$ $\$[\"data\"][\"hospital\"][\"inaugurated_on\"]$	Valid
	$\$[\"data\"][\"doctor_details\"][*][\"doj\"]$ $\$[\"data\"][\"doctor_details\"][*][\"dol\"]$	Valid
	$\$[\"data\"][\"hospital\"][\"name\"]$ $\$[\"data\"][\"registration_id\"]$	Valid
	$\$[\"data\"][\"hospital\"][\"patient\"][*][\"dol\"]$ & $\$[\"data\"][\"hospital\"][\"patient\"][*][\"test_report\"][*][\"dot\"]$	Invalid. Multi-column algorithm assignment is not allowed for these fields as they are different levels having fields belonging to different arrays. 'dol' belongs to 'patient' while 'dot' belongs to 'test_report'

 Assigning a multi-column algorithm to an invalid combination of JSON fields will produce an error that shows JSON paths.

Edit Field



Invalid Multi-column assignment for following paths \$['books']['*']['name'],\$['cars']['*']['name']

6.9.9.1.2 Multi-column algorithm with XML file format


In the case of XML document store type masking, multi-column algorithm assignment to XML elements will not be validated at the time of assignment. XML can be difficult to find out if an element is a type of an array or a single element until the whole data is read. Here, the masking job will fail immediately when any of the invalid multi-column assignments are found while running the job. Make sure the algorithm assignment should follow the below rules.

- Multi-column algorithm for XML file masking is not supported.
- Multi-column algorithm assignment to XML attributes is not supported.
- Multi-column algorithm is not supported for XML elements where,
 - The element is a type of array.
 - Elements are part of different arrays.
 - Elements are on different levels having one or more elements of type array.

Below is a sample XML file format with valid and invalid multi-column assignment examples.

XML Content	XML elements having Multi-column assignment	Invalid/Valid
<pre> <data> <founders> <name>Whooley</name> <name>Boddy</name> </founders> <hospital id="101"> <name>Hospital_name</name> <registered_on>2001-01-01</registered_on> <inaugurated_on>2002-01-01</inaugurated_on> <found_date>1905-12-10</found_date> <patient> <doa>2001-07-31</doa> <dob>1907-08-01</dob> <dol>2005-04-12</dol> <name>Patient_Name1</name> </patient> <patient> <doa>2002-09-31</doa> <dob>1908-01-01</dob> <dol>2007-04-11</dol> <name>Patient_Name2</name> </patient> </hospital> <registration_id>932884</registration_id> <doctor_details> <doj>2001-07-31</doj> <dol>2004-01-02</dol> <doctor_name>Doctor_Name1</doctor_name> </doctor_details> <doctor_details> <doj>2005-07-07</doj> <dol>2007-01-03</dol> <doctor_name>Doctor_Name2</doctor_name> </doctor_details> </data> </pre>	/data/founders/name /data/registraion_id	Invalid. Multi-column algorithm assignment is not allowed as element 'name' is an array
	/data/hospital/patient/name /data/doctor_details/doctor_name	Invalid. Multi-column algorithm assignment is not allowed for elements that are part of different arrays. Here 'name' belongs to an array 'patient' and 'doctor_name' belongs to an array 'doctor_details'
	/data/hospital/found_date /data/hospital/patient/doa	Invalid. Multi-column algorithm assignment is not allowed for these as one of the elements i.e. 'doa' belongs to an array and elements are on different levels.
	/data/hospital/patient/doa /data/hospital/patient/dol	Valid
	/data/hospital/registered_on /data/hospital/inaugured_on	Valid
	/data/doctor_details/doj /data/doctor_details/dol	Valid
	/data/hospital/name /data/registration_id	Valid
	/data/hospital/@id /data/hospital/name	Invalid. Multi-column algorithm assignment is not allowed for XML attributes.

6.9.10 Inventory Approval Workflow (database rule sets only)

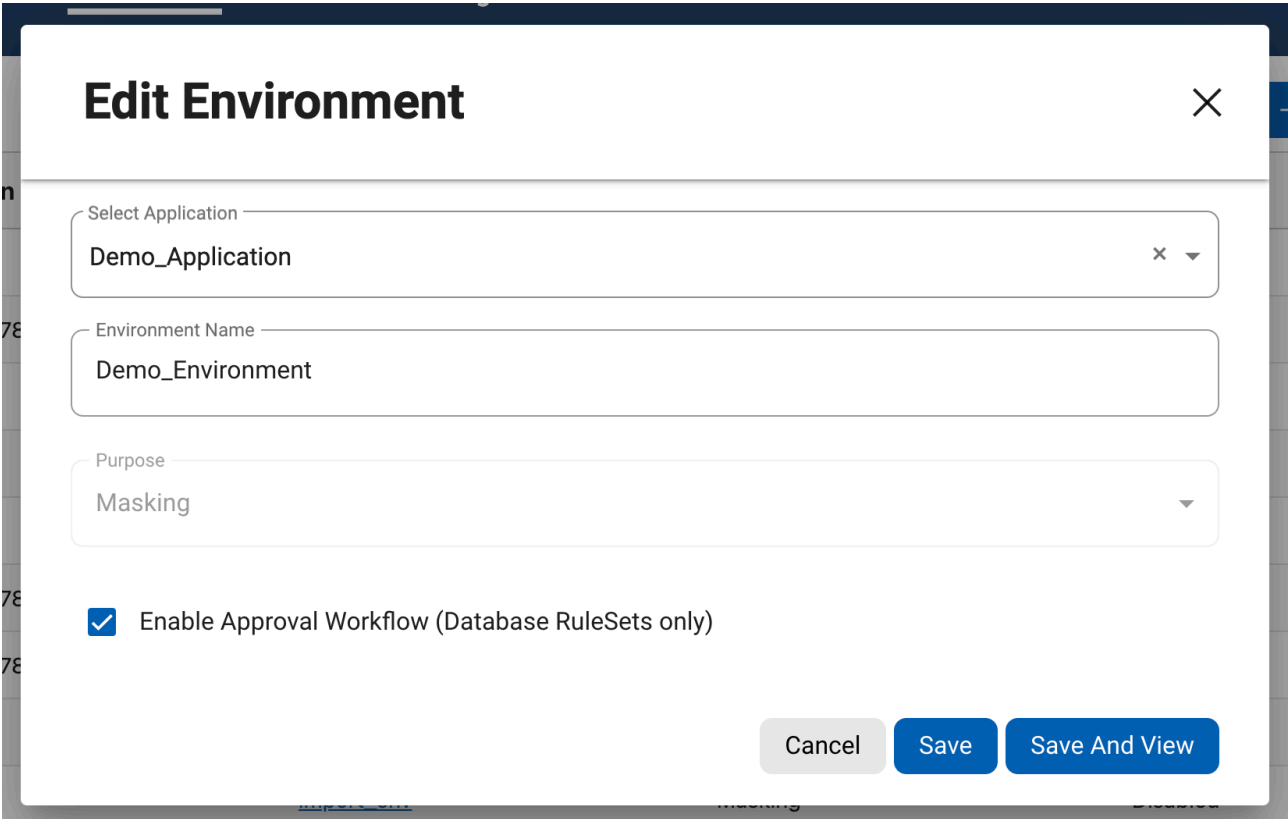
 This feature is only available for database rule sets.

When enabled, this feature requires a user (the “approver”) to approve a rule set’s inventory settings before a masking job for that rule set can be executed.

A database masking job will only be allowed to run if its rule set is in the **Approved** state. If the database rule sets state is either **New**, **Submitted**, or **Rejected**, and the user tries to run the masking job, then the user will receive the following error message: “Attempt to execute job while approval workflow is enabled and the rule set is not approved.”

6.9.10.1 Enabling Inventory Approval Workflow for an environment

Users can enable the inventory approval workflow for any environment by selecting the checkbox “Enable Approval Workflow (database rule sets only)” at the bottom of the **Add**, **Edit**, or **Copy** environment dialogue boxes.



Edit Environment [Close]

Select Application: Demo_Application [x] [v]

Environment Name: Demo_Environment

Purpose: Masking [v]

Enable Approval Workflow (Database RuleSets only)

Cancel Save Save And View

6.9.10.2 Workflow stages

- NEW**

When a user updates any column properties for a database rule set, the approval workflow status will be reset to NEW for that particular rule set. A masking job will not be able to run on any rule set in this status. Users will have to submit these inventories for approval by clicking on the **Submit** button which appears just above the grid.

The screenshot shows the Continuous Compliance user interface. At the top, there is a navigation bar with the logo and version '26.0.0.0', and menu items: Environments, Monitor, Settings, Admin, Audit. Below the navigation bar, the breadcrumb is 'Environments > ENV > testRuleset'. The page title is 'testRuleset'. There are buttons for 'All Logical Keys', 'All Filters', and 'Actions'. The status is 'New' with a 'Submit' button. Below the status is a table with columns: Table, Column, Data Type, Domain, Algorithm, File Format, and Actions. The table contains three sections: TEST1 (Filter Condition), TEST10 (Logical Key), and TEST2 (3). Each section lists columns with their data types and domains. For example, TEST1 has columns FIRST_NAME (VARCHAR2 (2000)), ID (NUMBER (0)), and LAST_NAME (VARCHAR2 (2000)).

Table	Column	Data Type	Domain	Algorithm	File Format	Actions
TEST1 (3) Filter Condition						🔑 🔍 <>
	FIRST_NAME	VARCHAR2 (2000)	FIRST_NAME	dlpx-core:FirstName		✎
	ID Primary Key, Index	NUMBER (0)				✎
	LAST_NAME	VARCHAR2 (2000)			json_filter.json	📄 ✎
TEST10 (3) Logical Key						🔑 🔍 <>
TEST2 (3)						🔑 🔍 <>
	FIRST_NAME	VARCHAR2 (2000)				✎
	ID Primary Key, Index	NUMBER (0)				✎

Displaying 1 to 9 of 19

- SUBMITTED**

Once the user modifies any properties in the database inventory and submits it for approval, an Admin user or any user whose role has the **Inventory Approval** privilege enabled will be able to approve/reject these changes by clicking on the **Approve/Reject** buttons appearing on the top of the grid. The Approve and Reject buttons will be hidden for the users without this privilege.

Environments > ENV > testRuleset

Back testRuleset All Logical Keys All Filters Actions

Status: Submitted Approve Reject

Table	Column	Data Type	Domain	Algorithm	File Format	Actions
TEST1 (3) Filter Condition						
	FIRST_NAME	VARCHAR2 (2000)	FIRST_NAME	dplx-core:FirstName		
	ID Primary Key, Index	NUMBER (0)				
	LAST_NAME	VARCHAR2 (2000)			json_filter.json	
TEST10 (3) Logical Key						
TEST2 (3)						
	FIRST_NAME	VARCHAR2 (2000)				
	ID Primary Key, Index	NUMBER (0)				
	LAST_NAME	VARCHAR2 (2000)				
TEST3 (3)						
	FIRST_NAME	VARCHAR2 (2000)				

Displaying 1 to 12 of 19

• **APPROVED**

If the database rule set status is Approved, masking jobs using this rule set may be executed.

Environments > ENV > testRuleset

Back testRuleset All Logical Keys All Filters Actions

Status: Approved

Table	Column	Data Type	Domain	Algorithm	File Format	Actions
TEST1 (3) Filter Condition						
	FIRST_NAME	VARCHAR2 (2000)	FIRST_NAME	dplx-core:FirstName		
	ID Primary Key, Index	NUMBER (0)				
	LAST_NAME	VARCHAR2 (2000)			json_filter.json	
TEST10 (3) Logical Key						
TEST2 (3)						
	FIRST_NAME	VARCHAR2 (2000)				
	ID Primary Key, Index	NUMBER (0)				
	LAST_NAME	VARCHAR2 (2000)				
TEST3 (3)						
	FIRST_NAME	VARCHAR2 (2000)				

Displaying 1 to 12 of 19

• **REJECTED**

If the database rule set status is Rejected, the user will have to re-modify the inventory properties set to the database columns and submit the inventory again for approval.

CONTINUOUS COMPLIANCE 26.0.0.0

Environments Monitor Settings Admin Audit

Environments > ENV > testRuleset

testRuleset

All Logical Keys All Filters Actions

Status: Rejected

Table	Column ↑	Data Type	Domain	Algorithm	File Format	Actions
TEST1 (3) Filter Condition	FIRST_NAME	VARCHAR2 (2000)	FIRST_NAME	dlpx-core:FirstName		
	ID Primary Key, Index	NUMBER (0)				
	LAST_NAME	VARCHAR2 (2000)			json_filter.json	
TEST10 (3) Logical Key						
TEST2 (3)	FIRST_NAME	VARCHAR2 (2000)	FIRST_NAME	dlpx-core:FirstName		
	ID Primary Key, Index	NUMBER (0)				
	LAST_NAME	VARCHAR2 (2000)				
TEST3 (3)	FIRST_NAME	VARCHAR2 (2000)				

Displaying 1 to 12 of 19

6.10 Managing record types and header/footer records

6.10.1 Records types

Record types and header/footer records are only applicable to delimited and fixed-width file formats.

Sometimes a delimited or fixed-width file will contain records with different formats. For example, this delimited file contains records with two different formats. Some records have an account number and postal address fields and other records have an account number and email address:

```
addresss, account1234, 9A Lexington Street, Bridgeport, Michigan, 48009
email, account1234 bar@example.com
address, account5678, 9B Agawam Village, Rochester, Colorado, 80015
address, account4321, 99 Mill River Drive, Davenport, Washington, 99336
email, account4321, foo@example.com
```

In this example, the first field differentiates the records: `address` for records with the format:

```
type, account_number, address, city, state, zip
```

and `email` for records with the format:

```
type, account_number, email_address
```

A field that differentiates records from one another is called a record type qualifier. The Compliance Engine supports up to three record type qualifiers.

In this case, define qualifiers that allow the Compliance Engine to differentiate between the different record types.

6.10.2 Header and footer records

Another common situation is for a delimited or fixed-width file to have one or more header or footer records. In this case, it may be desirable for the Compliance Engine to place unaltered copies of these records into the masked output file. For example, this delimited file's first record is a header that contains the names of each field:

```
hostname, IP  
foo.example.com, 10.11.12.13  
bar.exmaple.com, 10.14.15.16
```

In this situation, the Compliance Engine can be configured to treat the first record as a header so that its unmodified contents will be written into the masked file.

Navigation Options:

To access the Edit File Format screen, you can choose one of the following methods:

- i. Go to **Settings > Data Format**. In the **File** tab click on the required fixed-width or delimited file format name hyperlink to **Edit** the file format.
- ii. Navigate to **Environments > Ruleset**. Click on the required fixed-width or delimited ruleset name hyperlink to access the corresponding inventory-like screen. You can then select a file format or data file name from the dropdown and click on **Edit File Format** button.

Information
Changes to format files are global and will be applied to all rule sets utilizing them.

Settings > Data Formats > Details

00YRAG2C.fmt

+ Field + Record Type + Qualifiers + Header & Footer Actions

Reset

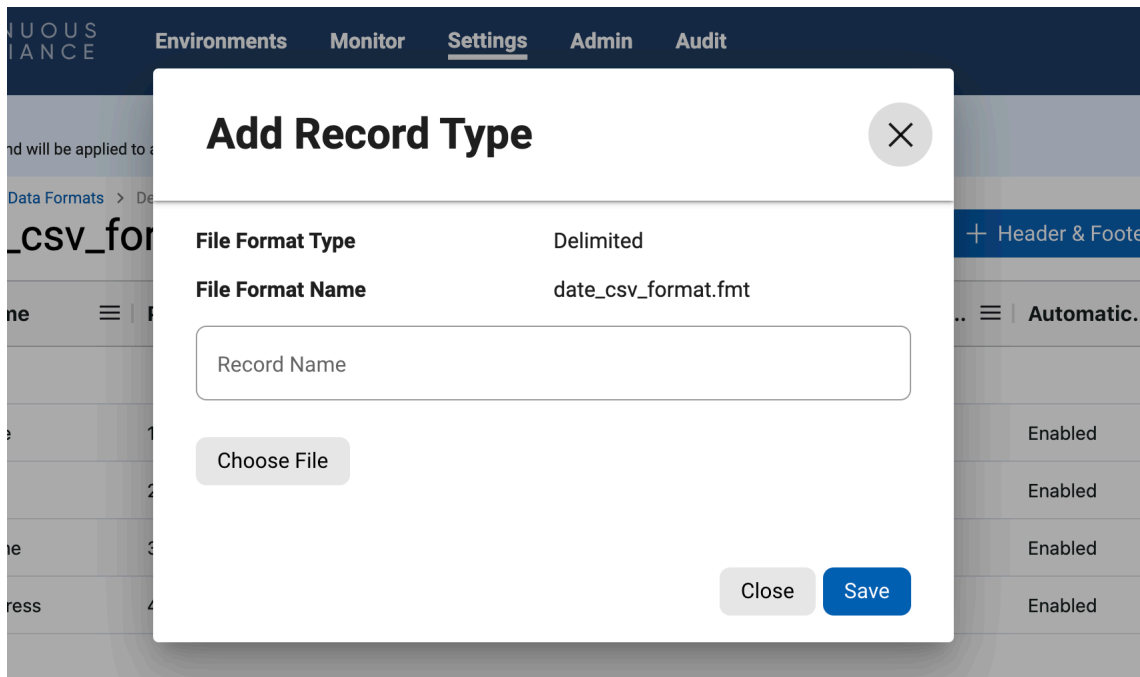
Record Type	Name	Position	Domain	Algorithm	File Format	Automatic Up...	Actions
All Records (6)							
	ID	1				Enabled	
	UNMASKED_00	2				Enabled	
	FIRSTNAME_00	3	FIRST_NAME	dlpx-core:FirstName		Disabled	
	LASTNAME_00	4	LAST_NAME	dlpx-core:LastName		Disabled	
	DATA_00	5			CTJIOQ0W,json	Disabled	
	DATA_01	6				Enabled	
new_record (2)							
	Name	1				Enabled	
	Email	2	EMAIL	dlpx-core:Email Uni...		Disabled	

i The **Environments > Ruleset** screen only allows viewing the record type, utilize the **Edit File Format** button for adding, editing, and deleting the record type which is a format level change.


6.10.2.1 Adding record types

Create a record type for each distinct record format by uploading a format file.
Perform the following steps to add a record type to a file format:

1. Click **+ Record Type** button. The **Add Record Type** window appears.



- In the **Add Record Type** window, enter values for the following fields:
 - Record Name** - A free-form name for this record type.
 - Choose File** - Browse for the file from which to import fields.

 The contents of the imported file vary for Delimited and Fixed-width. Refer to [Managing file formats](#)¹⁶⁹ for formatting examples.

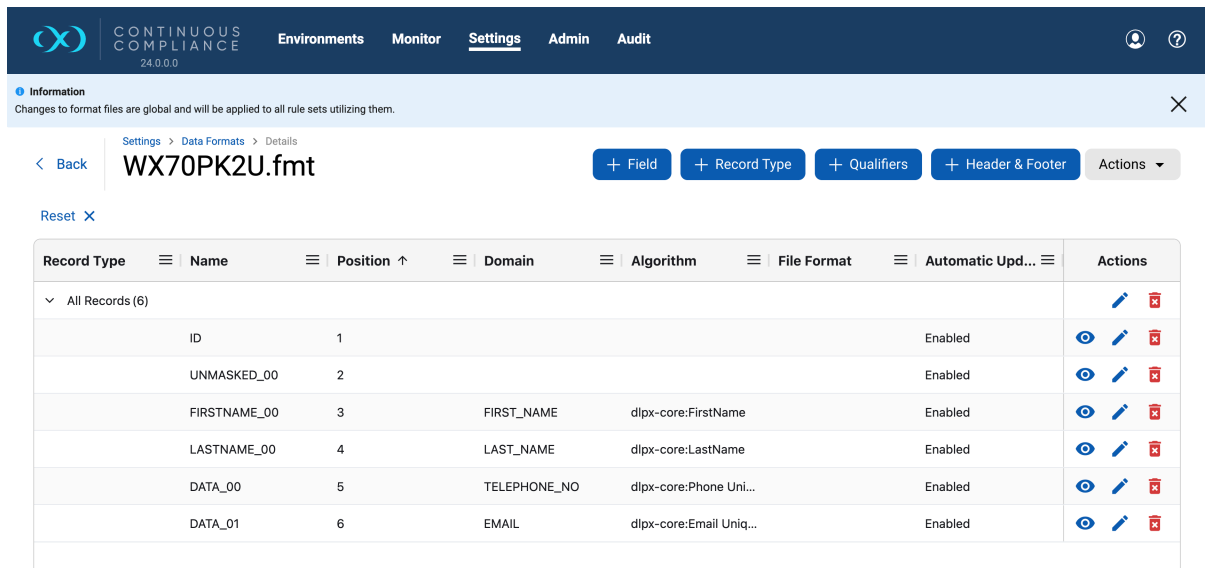
- Click **Save**.
- The added record type with corresponding fields will be displayed in the grid.

6.10.2.2 Editing record types

To edit an added record type name:

- Select the **Edit**  option from the **Actions** column of the record type to rename.

¹⁶⁹ <https://masking.delphix.com/docs/latest/managing-file-formats>



2. A pre-filled window (similar to the above) will appear.
3. Rename and Click on **Save**.

6.10.2.3 Deleting record types

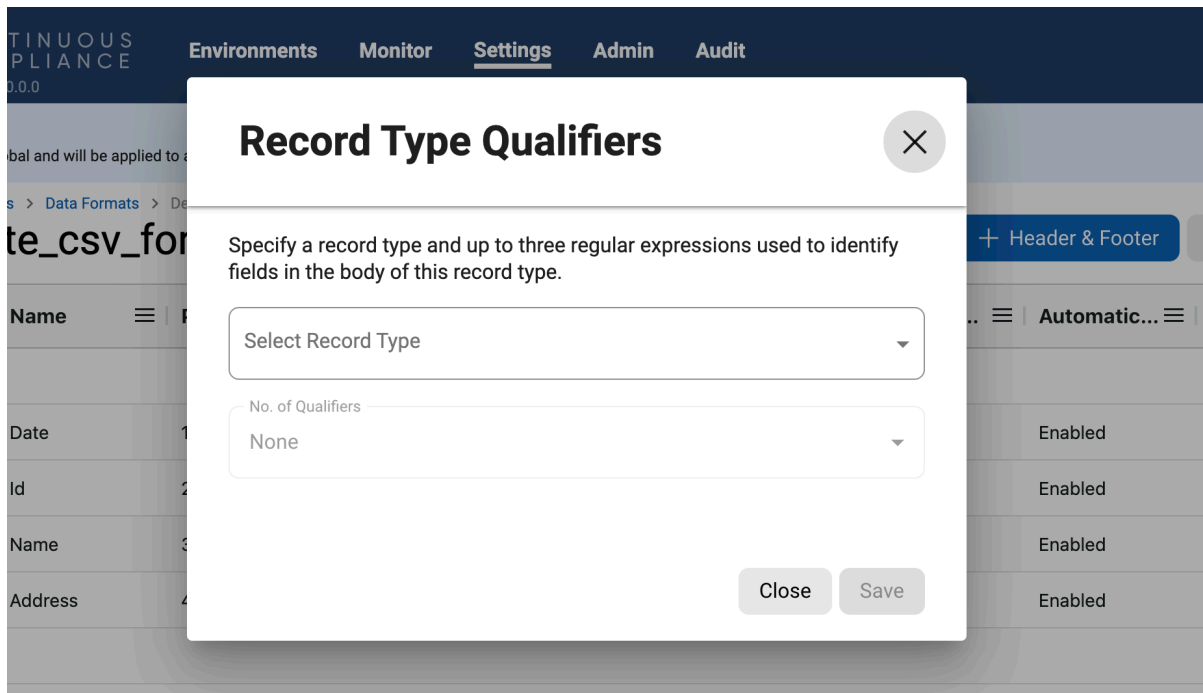
To delete an added record type:

1. Select the **Delete** option from the **Actions** column of the record type to be deleted.
2. A confirmation window will appear, confirm to Delete.

6.10.2.4 Managing qualifiers

In order to associate qualifiers with record types:

1. Click on **+ Qualifiers** Button, the **Record Type Qualifier** window will open.



2. Select the corresponding **Record Type** to add, edit, or view qualifiers.
3. **No. of Qualifiers** - Select the number of qualifiers. There can be a maximum of 3 qualifiers.

Record Type Qualifiers ✕

Specify a record type and up to three regular expressions used to identify fields in the body of this record type.

Select Record Type
New Record

No. of Qualifiers
3

Qualifier 1

Regular Expression
([A-Z])w+

Field Name
id

Qualifier 2

Regular Expression

Field Name

Qualifier 3

Regular Expression

Field Name

Close Save

i Fields to add Qualifier 1, Qualifier 2 and Qualifier 3 will render based on selected “No. of Qualifiers”.

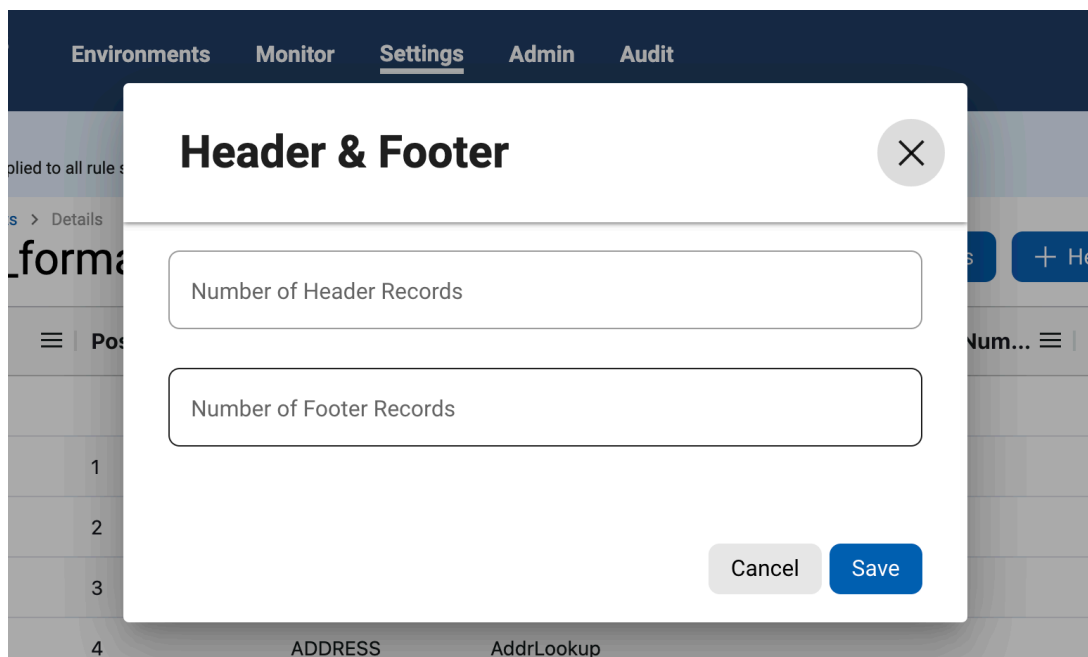
4. **Regular Expression** - This value is a regular expression that the Compliance Engine uses to match the specified field, to determine whether the record is of this type. A record type applies if its regular expression matches its specified identifier fields.
5. **Field Name** - Select the field name within the record type.

- Click **Save** once complete.

6.10.2.5 Configure header and footer

The **Header** or **Footer** associated with a format is used to specify the number of records that are not masked at the beginning and end of a file.

- Click the **+ Header & Footer** button and a small window will appear.
- Add or update the already configured **Number of Records** for the header or footer, or both, then save.



- Click on **Save**.

6.11 Managing jobs

This section describes how users can manage jobs, including Masking, Profiling, Tokenization, and Re-Identification jobs.

6.11.1 Jobs list

The **Environments** → **Jobs** page displays all the jobs present in the given environment based on the environment type.

6.11.1.1 Jobs - Masking Environment

The following job types are supported in a **Masking** environment:

1. Profiling
2. Masking

Job ID	Name	Rule Set	Type	Status	Completed Time	Run Time	Data Processed	Actions
96	sample_mask	sample_rs	MASK	SUCCEEDED	4 Apr 2024 13:30 IST	00:00:13	100%	...
95	sample_profile	sample_rs	PROFILE	RUNNING		00:00:01	0%	...
1	db_mask	dbRs	MASK	WARNING	3 Apr 2024 11:09 IST	00:00:11	100%	...

6.11.1.2 Jobs - Tokenize Environment

The following job types are supported in a **Tokenize** environment:

1. Profiling
2. Tokenization
3. Re-Identification

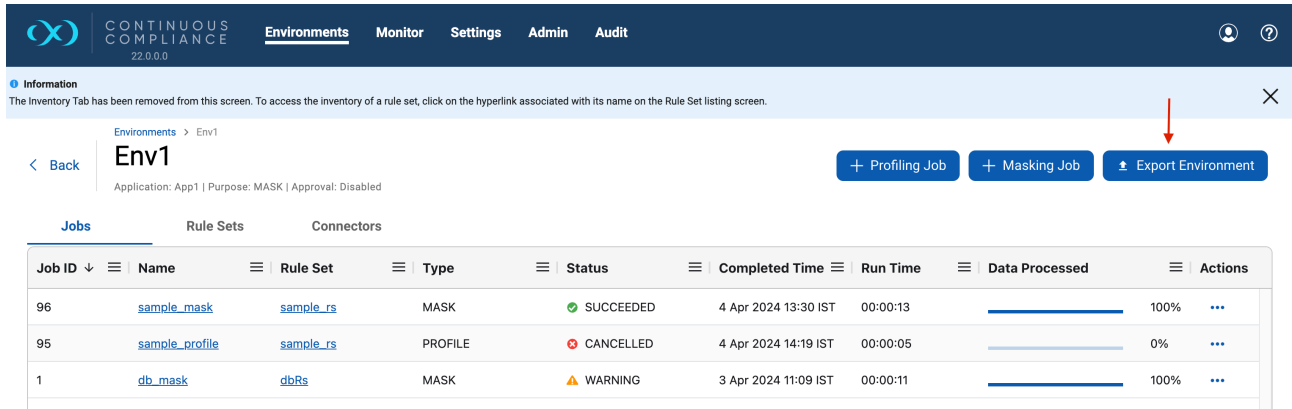
Job ID	Name	Rule Set	Type	Status	Completed Time	Run Time	Data Processed	Actions
97	sample_profile	con_VVW530WQ	PROFILE	CREATED				...
82	reidentify_2XMNSZRT	con_VVW530WQ	RE-IDENTIFY	SUCCEEDED	3 Apr 2024 19:16 IST	00:00:12	100%	...
81	tokenize_H9W1AIHV	con_VVW530WQ	TOKENIZE	SUCCEEDED	3 Apr 2024 19:16 IST	00:00:12	100%	...

i Sortable and filterable columns are Job ID, Name, Rule Set, Type, Status and Completed Time.

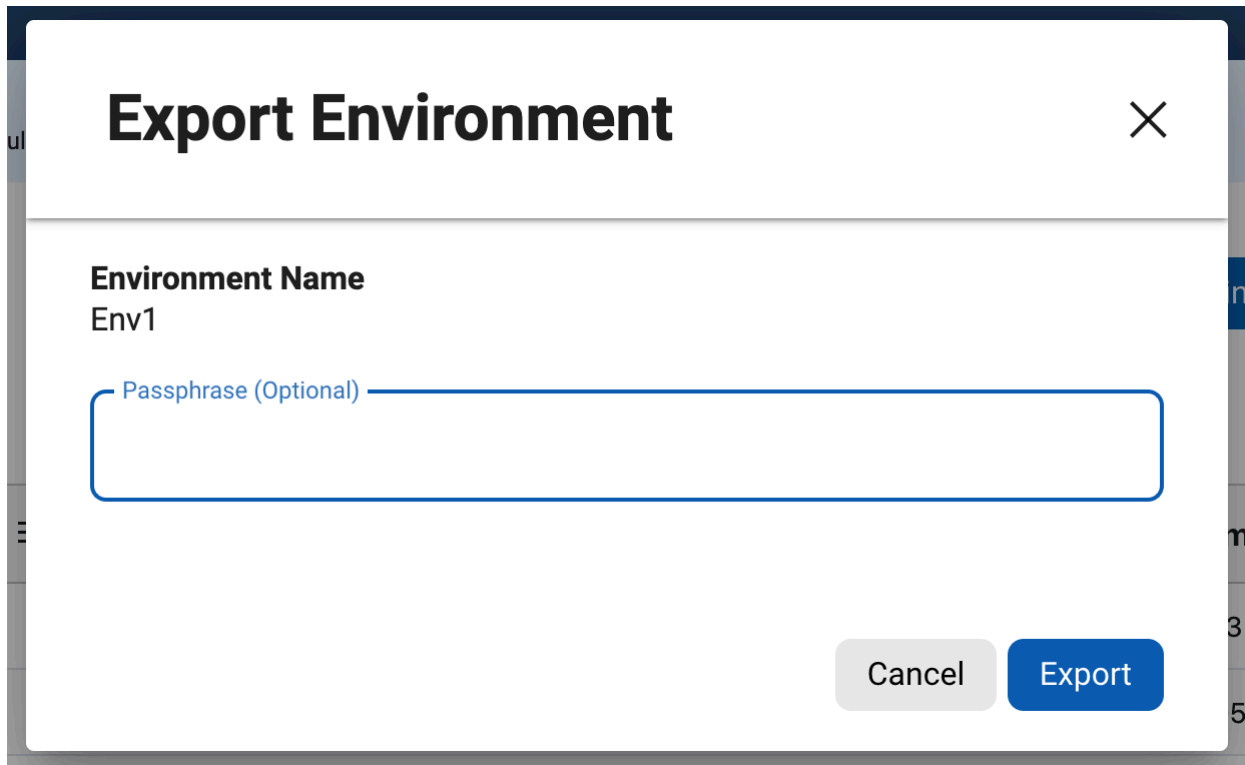
i To edit, view, or delete any of the jobs, click the (...) button to the right of the corresponding row under the **Actions** column and then click the corresponding **Edit**, **View**, or **Delete** action. If the user does not have permission to perform the selected action on the job, the action request will be disabled.

6.11.2 Export Environment

The **Environments** → **Jobs** page also provides a way to Export the current environment.

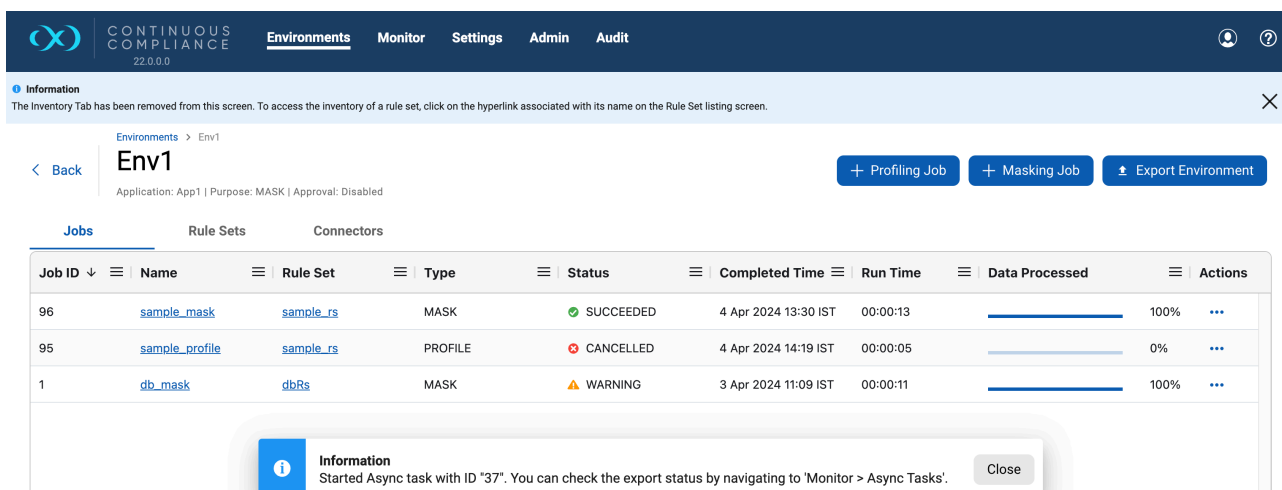


Clicking on **Export Environment** will open the below dialog.



An optional passphrase for encrypting the export file can be specified. Click **Export** to export the environment. All the information for the current environment (connectors, rule sets, jobs, and so on) is exported to a file.

Clicking on Export will create an Aysnc task and information will appear related to the created task at the bottom of the page.



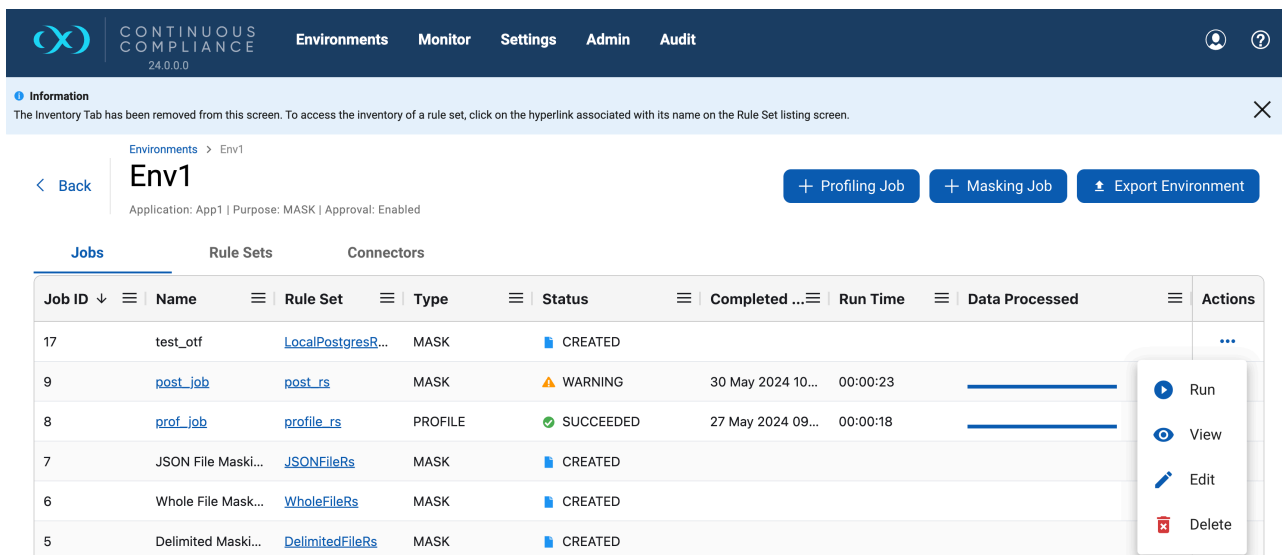
Users can also track export status from [Async Task Status](#) (see page 275).

6.11.3 Running and Stopping Jobs

This section describes how users can run and stop jobs.

6.11.3.1 Submitting a job

To submit or resubmit a job, Click the (...) button to the right of the corresponding job row under the **Actions** column and click **Run** . If the user doesn't have permission to run the job, then it will be disabled.



Upon submitting the job, the masking engine will check if there are enough resources allocated to simultaneously running jobs to determine whether to run or queue the submitted job. There are two resources that the submitted job will be verified against.

1. Maximum memory for all running jobs.

- This limit defaults to a dynamic calculation of 75% of the entire system's available memory minus 6GB, which is reserved for the masking web application. This calculation can be

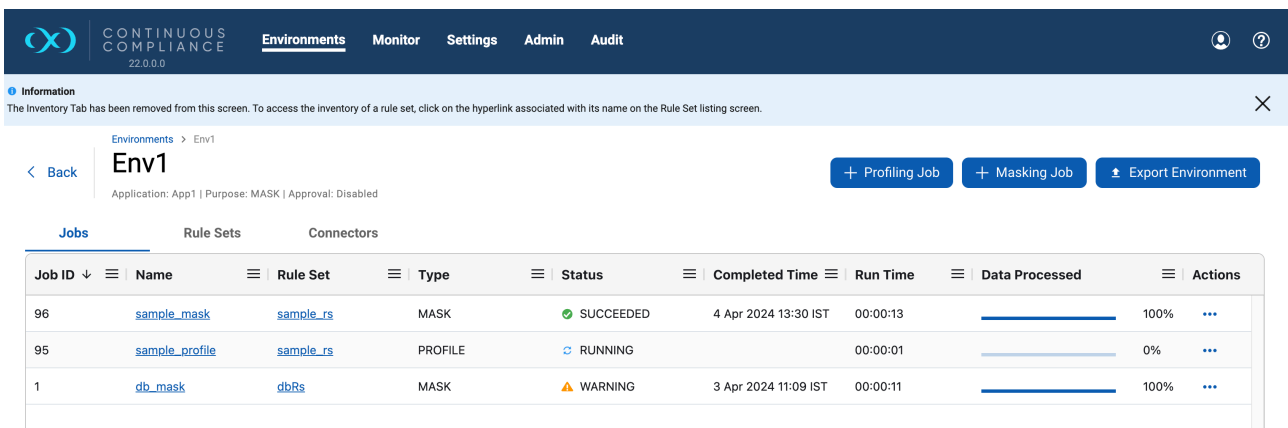
manually overridden by setting the general application setting `MaximumMemoryForJobs`. To revert a manually overridden limit back to the dynamically calculated limit, set the `MaximumMemoryForJobs` to 0.

2. Maximum number of simultaneously running jobs.

- This limit defaults to 7 simultaneously running jobs. However, this default value can be overridden by setting the general application setting `NumSimulJobsAllowed` to a different value. Users will need to modify this setting via the Masking API or the User Interface (Application Settings).

ⓘ If the submitted job causes all of the currently running jobs to exceed either of those limits, the job will be queued and run at a later time when enough of the other jobs stop running to free up resources. To view the the position of the job in the queue, navigate to the [Monitor Screen](#).¹⁷⁰

Once the job starts Running, the status will be changed. Editing/Deleting a Job is not allowed while it is running. When the job is completed, the status will change automatically.



6.11.3.1.1 Submitting a Multi-tenant Jobs

When the **Multi tenant** option is selected while creating a job, the User can select connector at the time of running a job. This option allows existing rulesets to be reused to mask identical schemas via different connectors.

1. **In-Place Job** → The User can select the Target connector at the job execution time

¹⁷⁰ <https://delphixdocs.atlassian.net/wiki/spaces/CC/pages/116927033/21.0.0.0+Job+monitoring>

The image shows a 'Select Target' dialog box. The title bar contains the text 'Select Target' and a close button (X). Below the title bar, there are two dropdown menus. The first dropdown is labeled 'Select Target Environment' and has 'Env1' selected. The second dropdown is labeled 'Select Target Connector' and has 'dbConnector' selected. At the bottom right of the dialog, there are two buttons: 'Cancel' and 'Run'.

2. **On-The-Fly Job** → User can select Source and Target connectors at the job execution time

Select Source and Target ✕

Select Target Environment

Env1 ✕ ▼

Select Target Connector

dbConnector ✕ ▼

Select Source Environment


env_1712061011703_ ✕ ▼

Select Source Connector

LocalhostPostgresDB ✕ ▼

Cancel Run

6.11.3.2 Stopping a Job

To stop a job, Click the (...) button to the right of the corresponding job row under the **Actions** column and click **Cancel**  . If the user doesn't have permission to run the job, then it will be disabled.

The screenshot shows the 'Env1' environment page with a table of jobs. The 'prof_job' row is selected, and a context menu is open over it. The table columns are Job ID, Name, Rule Set, Type, Status, Completed, Run Time, Data Processed, and Actions. The 'prof_job' row has a status of 'SUCCEEDED' and a completion date of '27 May 2024 09...'. The context menu options are Cancel, View, Edit, and Delete.

Job ID	Name	Rule Set	Type	Status	Completed	Run Time	Data Processed	Actions
17	test_otf	LocalPostgresR...	MASK	CREATED				...
9	post_job	post_rs	MASK	RUNNING		00:00:01	0%	...
8	prof_job	profile_rs	PROFILE	SUCCEEDED	27 May 2024 09...	00:00:18		Cancel, View, Edit, Delete
7	JSON File Maski...	JSONFileRs	MASK	CREATED				...
6	Whole File Mask...	WholeFileRs	MASK	CREATED				...
5	Delimited Maski...	DelimitedFileRs	MASK	CREATED				...
4	Fixed Masking J...	FixedFileRs	MASK	CREATED				...

On clicking **Cancel** Action, it will prompt for confirmation. Click on **Confirm** to cancel the job.

The dialog box has a title 'Cancel Job' and a close button (X). The main text asks 'Are you sure you want to cancel "sample_profile" job?'. At the bottom, there are two buttons: 'Cancel' (grey) and 'Confirm' (blue).

On clicking **Confirm**, the job will be cancelled and the status will be changed to CANCELLED.

Job ID	Name	Rule Set	Type	Status	Completed Time	Run Time	Data Processed	Actions
96	sample_mask	sample_rs	MASK	SUCCEEDED	4 Apr 2024 13:30 IST	00:00:13	100%	...
95	sample_profile	sample_rs	PROFILE	CANCELLED	4 Apr 2024 14:19 IST	00:00:05	0%	...
1	db_mask	dbRs	MASK	WARNING	3 Apr 2024 11:09 IST	00:00:11	100%	...

Stopping a RUNNING job will result in semi-masked data and terminate all subsequent transactions related to this job execution. Any dropped indexes, constraints, or triggers will not be recreated.. Stopping a QUEUED job will have no impact on the data source since the execution of the job has not yet begun. If email notifications are enabled, stopping a QUEUED job will send an email to the user who created the job indicating that it has been canceled by the user who stopped the job.

6.11.3.3 Enabling and disabling database constraints in Jobs

Depending on the type of target database you are using, the Delphix Engine can automatically enable and disable database constraints.

The ability to enable and disable constraints ensures that the Delphix Engine can update columns that have primary key or foreign key relationships. You can set Delphix to handle constraints automatically by enabling the **Disable Constraints** checkbox on a Masking job. If the built-in or extended connector is using a driver support plugin, **Disable Constraints** can be enabled via **Enable Tasks**. For a full list of supported built-in connectors and information on specific built-in driver support plugins, see [Built-in Driver Supports](#).¹⁷¹



Delphix does not support the enable/disable constraints feature for all databases. To see which databases are supported, see the [Data Source Support](#)¹⁷² page.

6.11.3.4 Creating SQL statements to run before and after jobs

When you create a masking job or a certification job, you can specify standard, static SQL statements to run before (prescript) you run a job and/or after (postscript) the job has been completed. For example, if you want to mask a column that has a foreign key constraint to another table, you could use a prescript to disable the constraint and a postscript to re-enable the constraint.

You create prescripts and postscripts by creating a text document with the SQL statement(s) to execute. If the text file contains more than one SQL statement, each statement must be separated by a semicolon [;]. For example, to remove records with date_column before December 12th, 2017 before masking a table

¹⁷¹ <https://delphixdocs.atlassian.net/wiki/spaces/CC/pages/116892981/21.0.0.0+Builtin+Driver+Supports>

¹⁷² <https://delphixdocs.atlassian.net/wiki/spaces/CC/pages/116894154/21.0.0.0+Data+source+support>

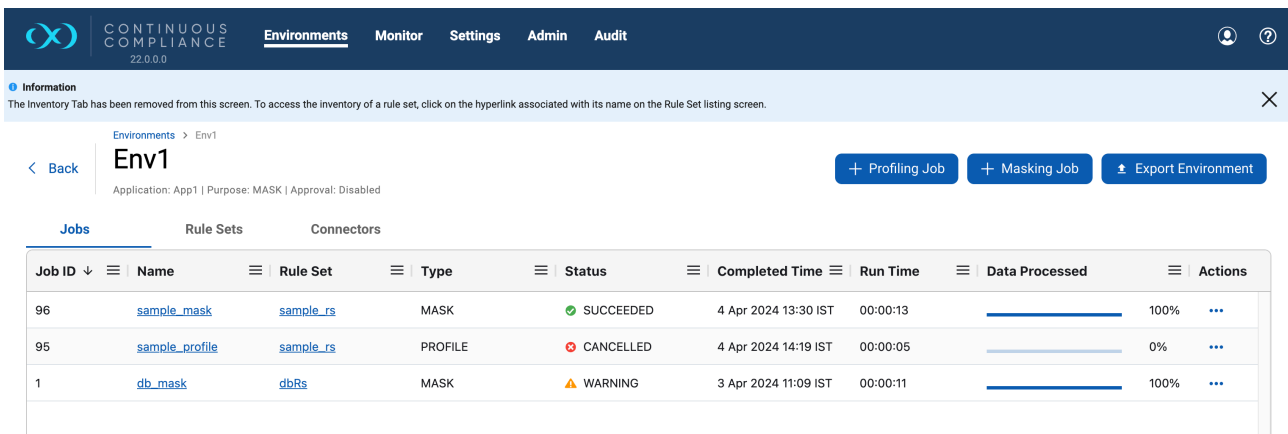
(owner.table), one would create a prescript file containing the following and associate the prescript file to the masking job that includes the table in its ruleset:

```
DELETE FROM owner.table WHERE date_column < '20171207';
```

Database-specific, SQL programming extensions (such as PL/SQL and Transact-SQL) and dynamic SQL statements are not supported in prescripts and postscripts. However, you can create procedures and functions using your database tooling of choice and call them using standard SQL statements from a prescript or postscript.

6.11.4 Create, view, edit, and delete jobs

This page covers how to create various jobs and manage them with actions like view, edit, and delete.



6.11.4.1 Create Masking Job

1. Click the **+ Masking Job** button at the top of the page, then the **Create Masking Job** window will appear.

Create Masking Job



- Details
- Configuration
- Summary

Details

Enter the Job Name, Select Masking Method and Rule Set. Click the Next button to continue.

Job Name

Target: Env1

Multi Tenant

Select Masking Method

Select Rule Set

Non-conforming Data

Stop job on first occurrence

This field applies only when the user chooses 'Mark job as Failed' for Non-Conforming Data behavior on Settings > Algorithms page or when Application Setting 'DefaultNonConformantDataHandling' is set to 'FAIL'.

Cancel
Back
Next
Save

2. The **Details** step prompts for the following information:

- **Job Name:** A free-form name for the job you are creating. Must be unique across the entire application.
- **Multi-Tenant:** This option allows existing rule sets to be reused to mask identical schemas via different connectors. The connector is selected at job execution time.
 - **Selective Data Distribution (SDD):** This option must be enabled for the masking job to be eligible for use in VDB masked provisioning as part of the Continuous Data Engine's SDD feature.
- **Masking Method:** Select either **In-Place** or **On-The-Fly**.
 - **In-Place** jobs update the source environment with the masked values.
 - **On-The-Fly** jobs read unmasked data from the source environment and write the masked data to the target environment.
- **Rule Set:** Select a rule set that this job will execute against.
- **Non-conforming Data**
 - **Stop job on first occurrence** (optional): To abort a job on the first occurrence of non-conformant data. The default is for this checkbox to be clear.
 - The job behavior depends on the settings specified in the **Algorithm Settings** page and on the individual algorithm pages that define how you view the presence of non-conforming data.

- The setting on the **Algorithm Settings** page is global and can be overridden by the setting on the algorithm page for that algorithm. These settings declare if the presence of Nonconforming data is a failure or a success for the job.
 - If **Mark job as Failed** is selected as a result of the above settings then the job would be aborted on the first occurrence of nonconforming data. If **Mark job as Succeeded** is selected as a result of the above settings then the job will not be aborted.
3. If **On-The-Fly** masking method is selected, then You will be prompted for Source information.
- **Source Environment:** Select the Source Environment from which this job will get the data.
 - **Source Connector:** Select the Source Connector that provides the connection to the chosen Source Environment.

Create Masking Job ×

- Details
- Configuration
- Summary

Details

Enter the Job Name, Select Masking Method and Rule Set. Click the Next button to continue.

Target: Env1

Multi Tenant

Select Masking Method

Select Rule Set

Select Source Environment

Select Source Connector

Non-conforming Data

Stop job on first occurrence

This field applies only when the user chooses 'Mark job as Failed' for Non-Conforming Data behavior on Settings > Algorithms page or when Application Setting 'DefaultNonConformantDataHandling' is set to 'FAIL'.

Cancel
Back
Next
Save

- **Database to File Job**

- When an **On-The-Fly** job is created with source as Database and Target as File, then check this option. This option will make sure you find all the database connectors in the Source Connector dropdown. Otherwise, the Source Connector dropdown will only display File Connectors.

Create Masking Job



Details

Configuration

Summary

Details

Enter the Job Name, Select Masking Method and Rule Set. Click the Next button to continue.

Job Name

Target: env_6RL79NMY

 Multi Tenant

Select Masking Method

On-The-Fly

Select Rule Set

delimited

Select Source Environment

 Database to File job

Select Source Connector

Non-conforming Data

 Stop job on first occurrence

This field applies only when the user chooses 'Mark job as Failed' for Non-Conforming Data behavior on Settings > Algorithms page or when Application Setting 'DefaultNonConformantDataHandling' is set to 'FAIL'.

Cancel

Back

Next

Save

**On-The-Fly Masking Jobs**

- Only certain combinations of connector types are supported.
- On-The-Fly jobs where the source and target connectors are of the same type (e.g. Oracle to Oracle, delimited file to delimited file), and jobs with a database source (e.g. Oracle, MS SQL) and the target is delimited files are supported.
- While creating Database (Source) to File (Target) job, make sure to check "Database to File Job" option on the UI, which will display all the database connectors in the source connector dropdown.
- The target tables or files must be created in advance and the names must match the names of the source tables or files. In the case of a database to delimited file job, the file names should match the table names.

4. If the selected Rule Set is a type of Database then one extra step will be added for configuring database options.

Create Masking Job



- Details
- **Database Options**
- Configuration
- Summary

Database Options

Enter the Database options for the job. Click the Next button to continue.

Commit Size

Update Threads
1

- Batch Update
- Drop Indexes
- Disable Triggers
- Disable Constraints

Pre SQL Script

Select File

Post SQL Script

Select File

Cancel Back **Next** Save

25 Database Options

Create Masking Job



- Details
- Database Options
- Configuration
- Summary

Database Options

Enter the Database options for the job. Click the Next button to continue.

Batch Update

Truncate

Enable Tasks

<input type="checkbox"/> Task Name	Execution Order
<input type="checkbox"/> Drop Constraints	1
<input type="checkbox"/> Drop Indexes	2
<input type="checkbox"/> Disable Triggers	3

Pre SQL Script





Post SQL Script

Cancel Back **Next** Save

26 Database Options with Enabled Tasks

5. You will be prompted for the following information:

- **Commit Size** (optional): The number of rows to process before issuing a commit to the database.
- **Update Threads**: The number of update threads to run in parallel to update the target database.
 - Multiple threads should not be used if the masking job contains any table without an index. Multi-threaded masking jobs can lead to deadlocks in the database engine.
 - Multiple threads can cause database engine deadlocks for databases using T-SQL. If masking jobs fail and a deadlock error exists on the database engine, then reduce the number of threads.
 - By default, it sets the **DefaultUpdateThreads** value from Application settings under the group "mask".
- **Batch Update** (optional): Enable or disable whether the database load phase to output the masked data will be performed in batches or not. The size of the batches is determined by the **Commit Size** field value. This option is recommended because it typically improves the performance of the masking job.

- **Truncate** (Optional and Only in Case of On-The-Fly jobs): To set whether the target database tables should be truncated before loading the masked data into the target database (after the masking phase is done).
- **Drop Indexes** (optional): Whether to automatically drop indexes on columns that are being masked and automatically re-create the index when the masking job is completed. The default is for this checkbox to not be selected and therefore not perform automatic dropping of indexes.
- **Disable Triggers** (optional): Whether to automatically disable database triggers. The default is for this checkbox to not be selected and therefore not perform automatic disabling of triggers.
- **Disable Constraints** (optional): Whether to automatically disable database constraints. The default is for this checkbox to not be selected and therefore not perform automatic disabling of constraints. For more information about database constraints see [Enabling and Disabling Database Constraints](#).¹⁷³
- **Enable Tasks** (optional): It displays tasks implemented by the driver support plugin being used. The default is for each checkbox to not be selected and therefore not perform any of the tasks. If the masking job being created is for a built-in connector with a built-in driver support plugin, the options displayed will be Disable Constraints, Disable Triggers, and Drop Indexes. For a full list of supported built-in connectors and information on specific built-in driver support plugins, see [Built-in Driver Supports](#).¹⁷⁴
- **Pre SQL Script** (optional): Specify a file that contains SQL statements to be run before the job starts. Click **Browse** to specify a file.
 - If you are editing the job and a Pre SQL Script file is already specified, you can click the Delete button  to remove the file.
 - If you want to download the file, click the Download button  .
 - The Delete and Download buttons only appear if a Pre SQL Script file was already specified.
 - For information about creating your Pre SQL Script files see [Creating SQL statements to run before and after Jobs](#)¹⁷⁵.
- **Post SQL Script** (optional): Specify a file that contains SQL statements to be run after the job finishes. Click **Browse** to specify a file.
 - If you are editing the job and a Post SQL Script file is already specified, you can click the Delete button  to remove the file.
 - If you want to download the file, click the Download button  .

¹⁷³ <https://delphixdocs.atlassian.net/wiki/spaces/CC/pages/116895362/%2821.0.0.0%29+Managing+jobs#Enabling-and-disabling-database-constraints>

¹⁷⁴ <https://delphixdocs.atlassian.net/wiki/spaces/CC/pages/116892981/21.0.0.0+Built-in+Driver+Supports>

¹⁷⁵ <https://delphixdocs.atlassian.net/wiki/spaces/CC/pages/116895362/%2821.0.0.0%29+Managing+jobs#Creating-SQL-statements-to-run-before-and-after-Jobs>

- The Delete and Download buttons only appear if a Post SQL Script file was already specified.
- For information about creating your Post SQL Script files see [Creating SQL statements to run before and after Jobs](#)¹⁷⁶.

i Pre/Post SQL Script will always connect to the database. If no Script is loaded, the Pre/Post SQL will make a connection with an empty statement.

6. After clicking next, on the next step of the wizard, You will be prompted for the following information.

Create Masking Job ✕

- Details
- Database Options
- Configuration**
- Summary

Configuration

Enter Configuration details for the job. Click the Next button to continue.

Number of Streams
1

Row Limit

Feedback Size

Min Memory (MB)

Max Memory (MB)

Comments

Email
abc@example.com

Cancel Back **Next** Save

176 <https://delphixdocs.atlassian.net/wiki/spaces/CC/pages/116895362/%2821.0.0.0%29+Managing+jobs#Creating-SQL-statements-to-run-before-and-after-Jobs>

- a. **Number of Streams:** The number of parallel streams to use when running the job. For example, you can select two streams to mask two tables in the Rule Set concurrently in the job instead of one table at a time.
- **Choosing the number of streams**
 - Jobs, even with a single stream, will have separate execution threads for input, masking, and output logic.
 - While it is not necessary to increase the number of streams to engage multiple CPU cores in a job, doing so may increase overall job performance dramatically, depending on a number of factors.
 - These factors include the performance characteristics of the data source and target, the number of processor cores available to the Delphix Masking Engine, and the number and types of masking algorithms applied in the Rule Set. The memory requirements for a job increase proportionately with the number of streams.
 - By default, it sets the **DefaultStreams** value from Application settings under the group “mask”.
- b. **Row Limit:** The number of data rows that may be in process simultaneously for each masking stream. For file jobs, this controls the number of delimited or fixed-width lines, mainframe records, or XML elements in process at one time.
- Setting this value to 0 allows unlimited rows into each stream.
 - Setting this value to -1 or leaving it blank will select a default limit based on rule set type.
 - **Choosing the Row Limit**
 - The default Row Limit values have been selected to allow typical jobs to run successfully with the default job memory and streams number settings.
 - This assumes a maximum row or record size of approximately 2000 bytes with 100 masked columns. If masked row or record size, or column count, exceed these values, it may be necessary to either allocate more memory to the job by increasing Max Memory, or reduce the Row Limit to a smaller value. Conversely, if the masked rows are quite small and have few masking assignments, increasing the Row Limit may improve job performance. Remember to consider the worst case (the largest rows, the most masking assignments) table or file format in the Rule Set when making this determination.
- c. **Feedback Size** (optional): The number of rows to process before writing a message to the logs. Set this parameter to the appropriate level of detail required for the logs.
- d. **Min Memory (MB)** (optional): Minimum amount of memory to allocate for the job, in megabytes.
- e. **Max Memory (MB)** (optional): Maximum amount of memory to allocate for the job, in megabytes.

- It is recommended that the **Min/Max Memory** should be set to at least **1024**.
 - f. **Comments** (optional): Add comments related to this masking job.
 - g. **Email** (optional): Add e-mail address(es) to which to send status messages.
7. After clicking next, on the last step of the wizard, the Summary will be displayed with the entered details.

Create Masking Job ✕

- Details
- Database Options
- Configuration
- **Summary**

Summary

View the job details below. Click the Back button to make changes or click the Save button to save the job.

Job Details

Job Name
tets-job

Target
Env1

Multi Tenant
No

Stop job on first occurrence of Nonconforming Data
Yes

Enable Tasks

Drop Indexes

Disable Triggers

Rule Set Details

Rule Set ID
1

Rule Set Name
dbRs

Rule Set Type
Database

Database Options

Update Threads
1

Batch Update
Yes

Configuration Details

Number of Streams
1

Row Limit
2000

Feedback Size
60000

Min Memory (MB)
1024

Max Memory (MB)
1024

Comments
Creating Masking Job

Email
abc@example.com

Cancel Back Next Save

8. When you are finished, click **Save**.



Database Options

- Some built-in connectors support the **Disable Constraints**, **Disable Triggers**, and **Drop Indexes** features (see the [Data Source Support](https://delphixdocs.atlassian.net/wiki/spaces/CC/pages/116894154/21.0.0.0+Data+source+support)¹⁷⁷ page).
- For built-in connectors implemented using driver support plugins, these options are available via the **Enable Tasks** section. For a full list of built-in connectors using driver support plugins, see [Built-in Driver Supports](https://delphixdocs.atlassian.net/wiki/spaces/CC/pages/116892981/21.0.0.0+Builtin+Driver+Supports)¹⁷⁸).
- For all other built-in connectors, these features will appear as checkboxes.

¹⁷⁷ <https://delphixdocs.atlassian.net/wiki/spaces/CC/pages/116894154/21.0.0.0+Data+source+support>

¹⁷⁸ <https://delphixdocs.atlassian.net/wiki/spaces/CC/pages/116892981/21.0.0.0+Builtin+Driver+Supports>

6.11.4.2 Create Tokenization Job

To Create Tokenization Jobs, click on the **+ Tokenization Job** button on the upper side of the page in the **Tokenize** environment. The flow and fields will be the same as the Masking job.

Create Tokenization Job



- Details
- Configuration
- Summary

Details

Enter the Job Name, Select Tokenization Method and Rule Set. Click the Next button to continue.

Target: env_T29CAEFQ

Multi Tenant

Select Tokenization Method

Select Tokenization Method

Select Rule Set

Non-conforming Data

Stop job on first occurrence

This field applies only when the user chooses 'Mark job as Failed' for Non-Conforming Data behavior on Settings > Algorithms page or when Application Setting 'DefaultNonConformantDataHandling' is set to 'FAIL'.

Cancel

Back

Next

Save

6.11.4.3 Create Re-Identification Job

To create a re-identification job, click on the **+ Re-Identification Job** button on the upper side of the page. The flow and fields will be the same as the Masking job.

Create Re-Identification Job



- Details
- Configuration
- Summary

Details

Enter the Job Name, Select Re-Identification Method and Rule Set. Click the Next button to continue.

Target: env_T29CAEFQ

Multi Tenant

Select Re-Identification Method

Select Rule Set

Non-conforming Data

Stop job on first occurrence

This field applies only when the user chooses 'Mark job as Failed' for Non-Conforming Data behavior on Settings > Algorithms page or when Application Setting 'DefaultNonConformantDataHandling' is set to 'FAIL'.

Cancel Back **Next** Save

6.11.4.4 Edit job

To perform an edit action on any of the jobs, Click the (...) button to the right of the corresponding row under the **Actions** column, **Edit** - action will be visible. If the user doesn't have permission to edit the job, then it will be disabled.

CONTINUOUS COMPLIANCE
Environments Monitor Settings Admin Audit

Information
The Inventory Tab has been removed from this screen. To access the inventory of a rule set, click on the hyperlink associated with its name on the Rule Set listing screen.

[Environments](#) > [Env1](#)

Application: App1 | Purpose: MASK | Approval: Enabled

+ Profiling Job
+ Masking Job
Export Environment

Job ID	Name	Rule Set	Type	Status	Completed ...	Run Time	Data Processed	Actions
17	test_of	LocalPostgresR...	MASK	CREATED				...
9	post_job	post_rs	MASK	WARNING	30 May 2024 10...	00:00:23	<div style="width: 100%; height: 10px; background-color: #1a3d4d;"></div>	<ul style="list-style-type: none"> Run View Edit Delete
8	prof_job	profile_rs	PROFILE	SUCCEEDED	27 May 2024 09...	00:00:18	<div style="width: 100%; height: 10px; background-color: #1a3d4d;"></div>	
7	JSON File Maski...	JSONFileRs	MASK	CREATED				
6	Whole File Mask...	WholeFileRs	MASK	CREATED				
5	Delimited Maski...	DelimitedFileRs	MASK	CREATED				

On clicking **Edit** action, a wizard will appear for editing the job. The job name is not editable after creation hence it will be disabled for Editing. For other fields, the user can edit and save.

Edit Masking Job



- Details
- Database Options
- Configuration
- Summary

Details

Enter the Job Name, Select Masking Method and Rule Set. Click the Next button to continue.

Job Name
sample_mask

Target: Env1

Multi Tenant

Select Masking Method
In-Place

Select Rule Set
sample_rs


Non-conforming Data

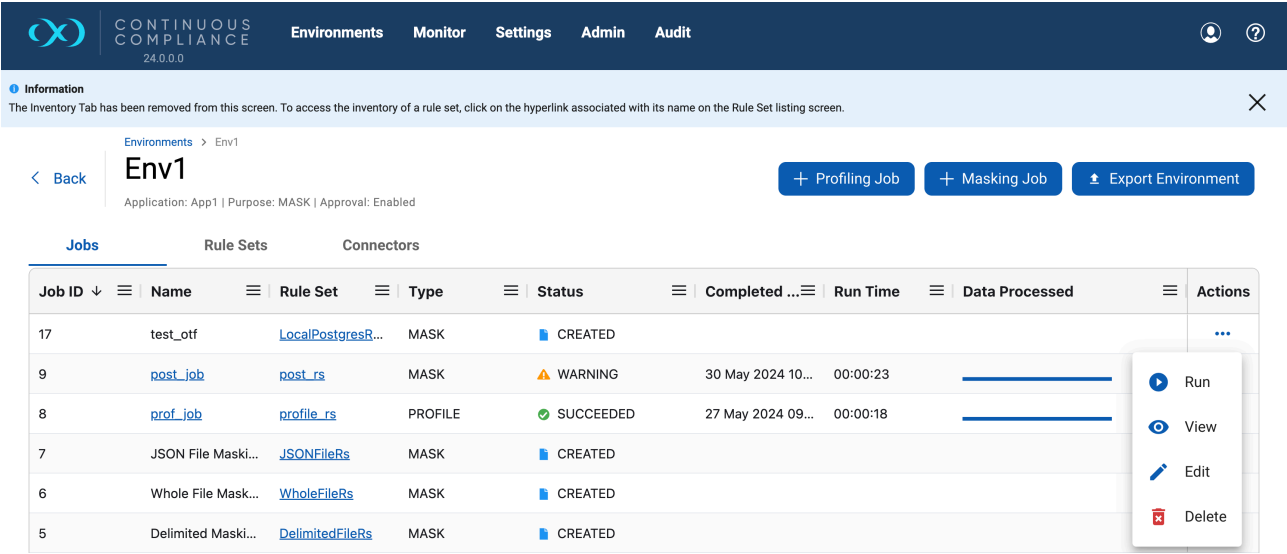
Stop job on first occurrence

This field applies only when the user chooses 'Mark job as Failed' for Non-Conforming Data behavior on Settings > Algorithms page or when Application Setting 'DefaultNonConformantDataHandling' is set to 'FAIL'.


Cancel Back Next Save

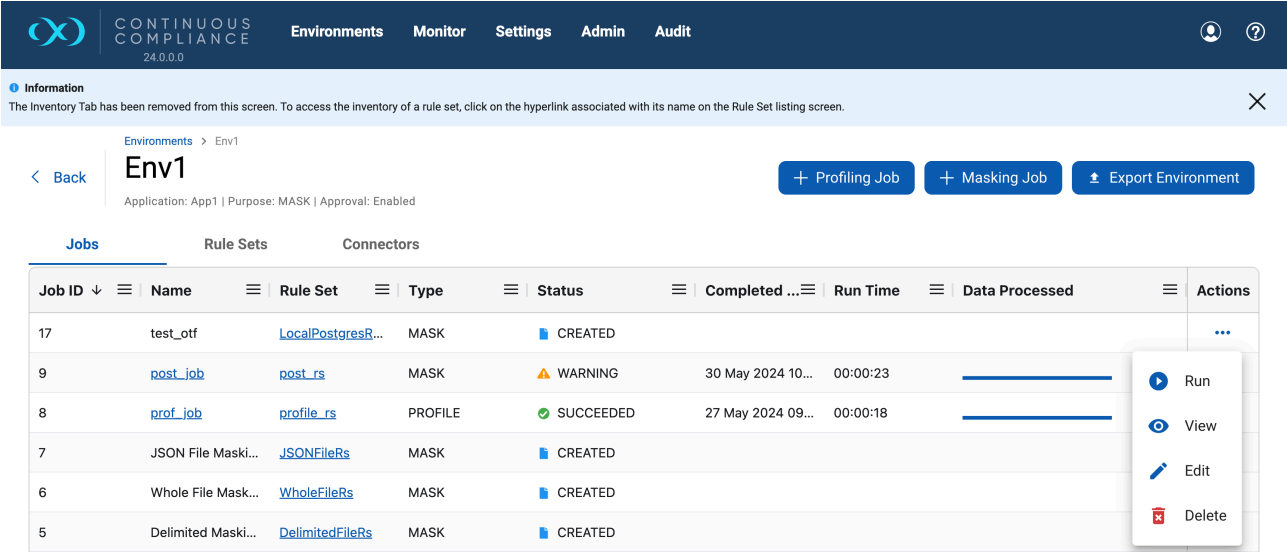
6.11.4.5 View job

To perform a view action on any of the jobs, Click the (...) button to the right of the corresponding row under the **Actions** column, **View** -  action will be visible.

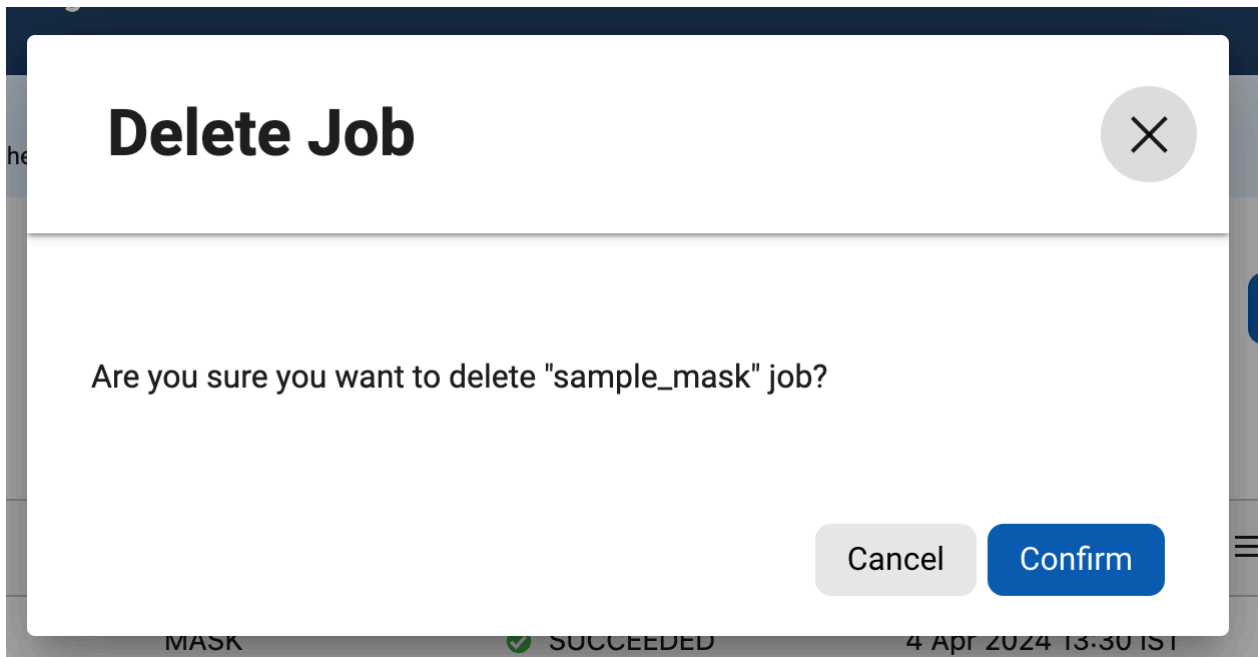


6.11.4.6 Delete job

To perform a delete action on any of the jobs, Click the (...) button to the right of the corresponding row under the **Actions** column, **Delete** -  action will be visible. If the user doesn't have permission to delete the job, then it will be disabled.



Clicking Delete Action will prompt for confirmation. Click on **Confirm** to delete the job.



6.11.5 Managing profiling jobs

This section describes how users can create, edit, view, and delete Profiling jobs.

You can create Profiling jobs for databases, XML, JSON, mainframe files, delimited files, and fixed-width file rule sets. It is not currently possible to profile XML or JSON documents stored in database columns.

When a profiling job runs, it applies all of the recognition logic specified in the profile set to each data element present in the rule set. The behavior of the profiler is also influenced by several application settings, refer to the **Profile group settings** section of [this article](#)¹⁷⁹.

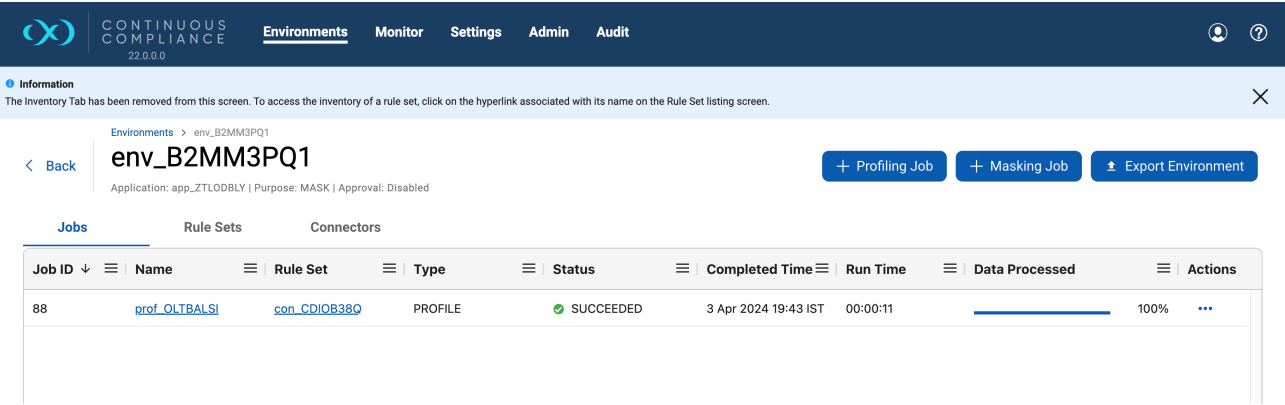
The Profiler assigns each sensitive data element to a domain, with each domain having a default masking algorithm. Then, in the inventory, masking algorithms can be manually updated as needed to establish the masking rulesets for your data sources.

Disabling Profiler Updates

If you wish to prevent the profiler from updating the domain and algorithm assignments for a particular column or file field, disable (uncheck) automatic updates for the column or file on the inventory screen.

Profiling Jobs are grouped within environments on the **Environments > Jobs** page along with all masking jobs. To navigate to the **Jobs** screen, click on an environment. It will land on the **Jobs** page.

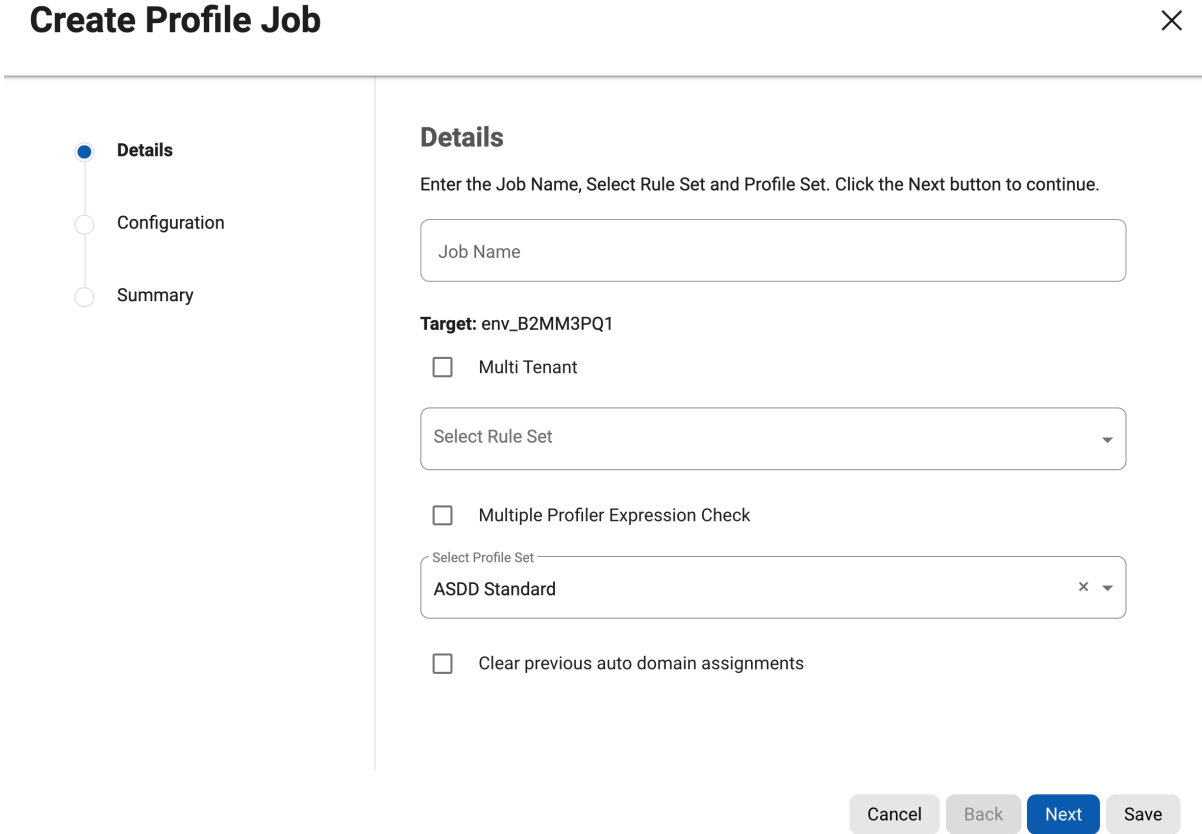
¹⁷⁹ <https://delphixdocs.atlassian.net/wiki/spaces/CC/pages/60654556/17.0.0.0+Masking+API+client>



6.11.5.1 Create Profile Job

To create a new Profile job:

1. Click the **+ Profiling Job** button on the upper side of the page.
2. The **Create Profile Job** wizard appears.



3. On the first step, You will be prompted for the following information:
 - **Job Name:** A free-form name for the job you are creating. Must be unique.

- **Multi Tenant:** Check the box if the job is for a multi-tenant database. This option allows existing rulesets to be re-used to mask identical schemas via different connectors. The connector is selected at job execution time.
 - **Rule Set:** Select the rule set that this job will profile.
 - **Multiple Profiler Expression Check:** By default, the profiler stops testing Profiler Expressions on a column or data value after the first expression matches. Check this box if the job should check all Profiler Expressions. If multiple Profiler Expressions match, the Profiler report will indicate multiple matches and the algorithm specified by the `DefaultMultiPhiAlgorithm` application setting will be assigned. This setting applies to both the legacy and ASDD profilers.
 - **Profile Sets:** The name of the Profile Set to use. A Profile Set is a set of Profile Expressions (for example, a set of financial expressions) or classifiers. The profile set selected determines whether the legacy or ASDD profiler will run. If the current data source is not supported by the ASDD profiler, selecting an ASDD profile set will result in an error and another profile set must be selected. Refer to [this section \(see page 653\)](#) for information regarding which connectors are supported by ASDD.
 - Depending upon the Rule Set selection, UI will show the **DefaultProfileSetName** value as selected for Profile Set from Application Settings under the group “asdd”.
 - **Clear previous auto domain assignments:** Checking this setting would reset or clear the domain assignments for the columns (with 'Enable Automatic Updates' enabled) from previous runs. It will be visible only in the case when the Profile set selected is ASDD-supported.
4. After clicking next, on the second step of the wizard, You will be prompted for the following information.

Create Profile Job



○ Details

● **Configuration**

○ Summary

Configuration

Enter Configuration details for the job. Click the Next button to continue.

Number of Streams

1

Feedback Size

Min Memory (MB)

Max Memory (MB)

Comments

Email

abc@example.com

Cancel
Back
Next
Save

- **Number of Streams:** The number of parallel streams to use when running the jobs. For example, you can select two streams to profile two tables in the ruleset concurrently in the job instead of one table at a time. By default, it sets **DefaultStreams** value from Application settings under group “profile”.
- **Feedback Size** (optional): The number of rows to process before writing a message to the logs. Set this parameter to the appropriate level of detail required for monitoring your job. For example, if you set this number significantly higher than the actual number of rows in a job, the progress for that job will only show 0 or 100%.
- **Min Memory (MB)** (optional): Minimum amount of memory to allocate for the job, in megabytes.
- **Max Memory (MB)** (optional): Maximum amount of memory to allocate for the job, in megabytes. When an ASDD profile set is selected, the max memory for the job must be at least 1024MB for each stream. For example, if No. of Streams is 4, this value would need to be 4096 or higher.
- **Comments** (optional): Add comments that are related to this job.
- **Email** (optional): Add e-mail address(es) to which to send status messages. Separate addresses with a comma (,).

- After clicking next, on the third step of the wizard, the Summary will be displayed with the entered details.

Create Profile Job

✕

- Details
- Configuration
- Summary

Summary

View the job details below. Click the Back button to make changes or click the Save button to save the job.

Job Details

Job Name
test_profile_job

Target
env_B2MM3PQ1

Multi Tenant
No

Configuration Details

Number of Streams
1

Min Memory (MB)
2048

Max Memory (MB)
2048

Feedback Size
10000

Comments
Creating a profile job

Email
abc@example.com

Rule Set Details

Rule Set ID
78

Rule Set Name
con_CD10B38Q

Rule Set Type
Fixed Width

Profile Set Details

Profile Set ID
4

Profile Set Name
ASDD Standard

Multiple Profiler Expression Check
No

Clear previous auto domain assignments
No

Cancel
Back
Next
Save

- When you are finished, click **Save**.

6.11.5.2 Edit Profile Job

To perform an edit action on any of the Profile jobs, Click the (...) button to the right of the corresponding row under the **Actions** column, **Edit** - action will be visible. If the user doesn't have permission to edit the job, then it will be disabled.

The screenshot shows the Continuous Compliance interface. At the top, there is a navigation bar with the logo and menu items: Environments, Monitor, Settings, Admin, Audit. Below the navigation bar, there is an information banner stating: "The Inventory Tab has been removed from this screen. To access the inventory of a rule set, click on the hyperlink associated with its name on the Rule Set listing screen." The main content area is titled "Env1" and includes a "Back" button and three action buttons: "+ Profiling Job", "+ Masking Job", and "Export Environment". Below this, there are tabs for "Jobs", "Rule Sets", and "Connectors". The "Jobs" tab is active, displaying a table with columns: Job ID, Name, Rule Set, Type, Status, Completed, Run Time, Data Processed, and Actions. The table contains several rows of job data. An actions menu is open over the table, showing options: Run, View, Edit, and Delete.


Job ID	Name	Rule Set	Type	Status	Completed	Run Time	Data Processed	Actions
17	test_otf	LocalPostgresR...	MASK	CREATED				Run, View, Edit, Delete
9	post_job	post_rs	MASK	WARNING	30 May 2024 10...	00:00:23		Run, View, Edit, Delete
8	prof_job	profile_rs	PROFILE	SUCCEEDED	27 May 2024 09...	00:00:18		Run, View, Edit, Delete
7	JSON File Maski...	JSONFileRs	MASK	CREATED				Run, View, Edit, Delete
6	Whole File Mask...	WholeFileRs	MASK	CREATED				Run, View, Edit, Delete
5	Delimited Maski...	DelimitedFileRs	MASK	CREATED				Run, View, Edit, Delete

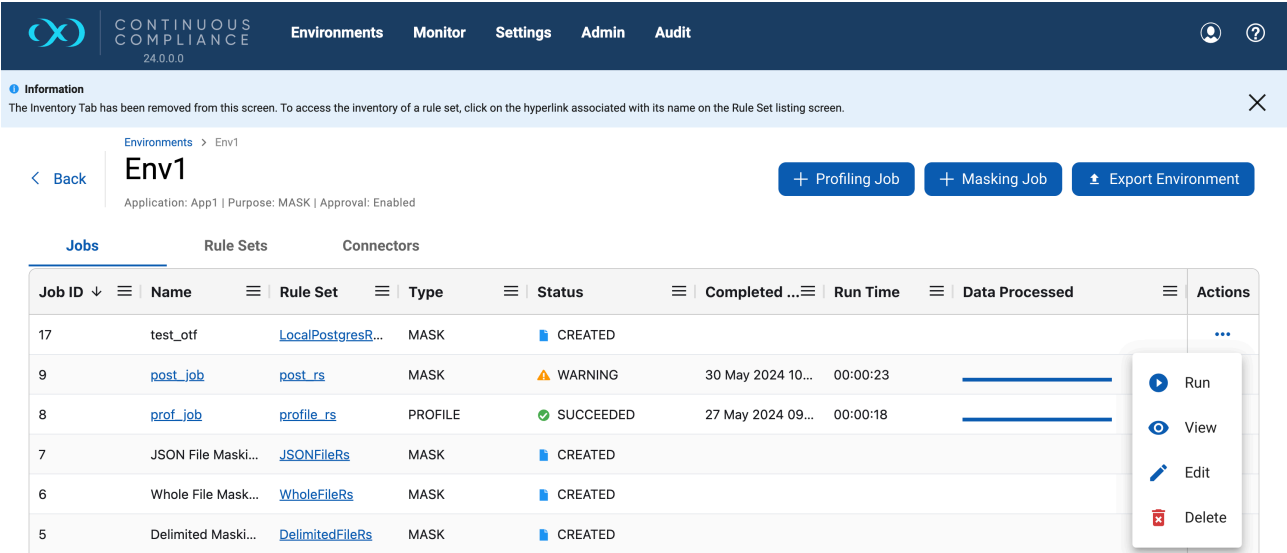
On clicking **Edit** action, a wizard will appear for editing the Profile job. The job name is not editable after creation hence it will be disabled for Editing. For other fields, the user can edit and save.

Edit Profile Job

The screenshot shows the "Edit Profile Job" wizard. On the left, there is a navigation pane with three steps: "Details" (selected), "Configuration", and "Summary". The main content area is titled "Details" and contains the following text: "Enter the Job Name, Select Rule Set and Profile Set. Click the Next button to continue." Below this text, there is a text input field for "Job Name" containing the value "sample_profile". There are two checkboxes: "Multi Tenant" (unchecked) and "Multiple Profiler Expression Check" (unchecked). Below these, there are two dropdown menus: "Select Rule Set" with the value "sample_rs" and "Select Profile Set" with the value "ASDD Standard". At the bottom, there is a checkbox for "Clear previous auto domain assignments" (unchecked). At the bottom right, there are four buttons: "Cancel", "Back", "Next", and "Save".

6.11.5.3 View Profile Job

To perform a view action on any of the Profile jobs, Click the (...) button to the right of the corresponding row under the **Actions** column, **View** -  action will be visible.



The screenshot shows the Continuous Compliance web interface. At the top, there is a navigation bar with the logo and menu items: Environments, Monitor, Settings, Admin, and Audit. Below the navigation bar, there is an information banner stating: "The Inventory Tab has been removed from this screen. To access the inventory of a rule set, click on the hyperlink associated with its name on the Rule Set listing screen." Below the banner, the page title is "Env1" with a "Back" button. There are three buttons: "+ Profiling Job", "+ Masking Job", and "Export Environment". Below these buttons, there are tabs for "Jobs", "Rule Sets", and "Connectors". The "Jobs" tab is active, showing a table with columns: Job ID, Name, Rule Set, Type, Status, Completed, Run Time, Data Processed, and Actions. The table contains several rows of jobs. The row for "prof_job" (Job ID 8) has a status of "SUCCEEDED" and a "View" action highlighted in the Actions column. The "View" action is represented by an eye icon.

Job ID	Name	Rule Set	Type	Status	Completed	Run Time	Data Processed	Actions
17	test_of	LocalPostgresR...	MASK	CREATED				...
9	post_job	post_rs	MASK	WARNING	30 May 2024 10...	00:00:23		Run
8	prof_job	profile_rs	PROFILE	SUCCEEDED	27 May 2024 09...	00:00:18		View
7	JSON File Maski...	JSONFileRs	MASK	CREATED				Edit
6	Whole File Mask...	WholeFileRs	MASK	CREATED				Delete
5	Delimited Maski...	DelimitedFileRs	MASK	CREATED				

On clicking **View** action, a wizard will open on the summary step.

View Profile Job



- Details
- Configuration
- Summary


Summary

View the job details below. Click the Back button to make changes or click the Save button to save the job.

Job Details Job Name sample_profile Target Env1 Multi Tenant No	Profile Set Details Profile Set ID 4 Profile Set Name ASDD Standard Multiple Profiler Expression Check No Clear previous auto domain assignments No
Rule Set Details Rule Set ID 90 Rule Set Name sample_rs Rule Set Type Database	Configuration Details Number of Streams 1 Min Memory (MB) 1024 Max Memory (MB) 1024 Feedback Size 50000 Email abc@example.com

Cancel Back Next Save

6.11.5.4 Delete Profile Job

To perform a delete action on any of the Profile jobs, Click the (...) button to the right of the corresponding row under the **Actions** column, **Delete** -  action will be visible. If the user doesn't have permission to delete the job, then it will be disabled.

Information
The Inventory Tab has been removed from this screen. To access the inventory of a rule set, click on the hyperlink associated with its name on the Rule Set listing screen.

Environments > Env1
Application: App1 | Purpose: MASK | Approval: Enabled

Jobs Rule Sets Connectors

Job ID	Name	Rule Set	Type	Status	Completed ...	Run Time	Data Processed	Actions
17	test_of	LocalPostgresR...	MASK	CREATED				...
9	post_job	post_rs	MASK	WARNING	30 May 2024 10...	00:00:23	<div style="width: 100%;"></div>	Run, View, Edit, Delete
8	prof_job	profile_rs	PROFILE	SUCCEEDED	27 May 2024 09...	00:00:18	<div style="width: 100%;"></div>	Run, View, Edit, Delete
7	JSON File Maski...	JSONFileRs	MASK	CREATED				...
6	Whole File Mask...	WholeFileRs	MASK	CREATED				...
5	Delimited Maski...	DelimitedFileRs	MASK	CREATED				...

Clicking Delete Action will prompt for confirmation. Click on **Confirm** to delete the job.

Delete Job

Are you sure you want to delete "sample_profile" job?

Cancel Confirm

6.11.6 Job monitoring


This section describes how users can monitor the progress and completion state of a job.

6.11.6.1 Monitoring jobs


Once a job has been created and started, you can monitor its progress by navigating to the **Monitor > Job Executions** or by clicking on the name of the job on any screen. The Job Executions page shows you a list of executed jobs, their progress as well and their current status. To get even more detail on the progress of an individual job, click on the Job Name.

The screenshot shows the 'Job Executions' page in the Continuous Compliance interface. The top navigation bar includes 'Environments', 'Monitor', 'Settings', 'Admin', and 'Audit'. The 'Monitor' section is active, showing 'Job Executions' with a 'Reset' button. The status bar at the top right displays '0 Jobs Queued', '1/7 Jobs Running', and '1024/19625 Memory Usage(MB)'. The main table lists several job executions with columns for Environment, Job Name, Status, Progress, Submit Time, Start Time, Type, and Actions. The jobs shown include 'post_job' and 'prof_job' in 'Env1' environment, with statuses ranging from 'RUNNING' to 'SUCCEEDED'.

Jobs Queued refers to the number of jobs in the Jobs Queue. When the limit is reached for running the maximum number of jobs and any job is triggered then it moves to Jobs Queue.





Click on **View Logs** -  from the Actions columns to the right of the corresponding execution, if you want to view the logs of completed execution.




The executions on the screen can be filtered or sorted by the various informational fields by clicking on the respective fields. More information on grid filtering and sorting can be found [here \(see page 232\)](#).

 Sortable and filterable columns are Environment, Job Name, Status, Submit Time, Start Time, and Type.

6.11.6.1.1 Event status

The following table lists the states of a job/event.

Job status icon	Status	Description
	Cancelled	It appears when a user cancel a running task/execution.
	Failed	It appears when there is an error in the execution of an event in the task/execution.
	Queued	It means the events of the task/execution are yet to start.
	Running	It appears when the event is in progress.

Job status icon	Status	Description
	Succeeded	It means the event is completed successfully. It appears when the event is successful.
	Skipped	It appears when the event is skipped and process moved to the next event of the task/execution.
	Warning (Non-Conformant)	It appears when an event of the task/execution is successful but with warning.

6.11.6.2 Monitoring a single job

In addition to viewing high-level stats about the status/progress of all your jobs, you can also deep dive into each job to get more details by clicking on the job name from the Job executions grid.

The screenshot displays the Continuous Compliance interface for a masking job. The top navigation bar includes 'Environments', 'Monitor', 'Settings', 'Admin', and 'Audit'. The breadcrumb trail shows 'Environments > env_DSWQZCJZ > mask_UZACO9VV'. The main heading is 'mask_UZACO9VV' with 'Execution ID: 34'. A green checkmark indicates 'Masking Completed', and a 'Report and Logs' button is present.

Execution Summary

- Execution ID: 34
- Environment: [env_DSWQZCJZ](#)
- Job Name: mask_UZACO9VV
- Job Type: Mask
- Job ID: 28
- Rule Set: [rs_7PJ12BW8](#)
- Connection Type: Table
- Source: -
- Target: DLPXDBORA
- Previous Run Time: 00:00:19
- Streams: 1
- Total # of Tables: 2
- Tables Masked: 2
- Tables to be Masked: 0
- Rows Masked: 20
- Rows Remaining: 0
- Tables with Nonconforming Data: 0
- Columns with Nonconforming Data: 0
- Updates Running: 1

Success Start Time: 28 May 2024 13:03:53 IST End Time: 28 May 2024 13:04:08 IST Run Time: 00:00:15

- Initializing
- Collecting Configurations
- Preparing
- Running Pre-execution Custom Driver Tasks
- Starting
- Running PreSQL Scripts
- Running PostSQL Scripts
- Running Post-execution Custom Driver Tasks
- Collecting Information
- Job Completed

Execution Components

ID ↑	Table Name	Progress	Status	Duration (HH:mm:ss)	Actions
55	testdata_FKCONSUMER	<div style="width: 100%;"></div>	100% ✓ SUCCEEDED	00:00:01	👁
56	testdata_FKPRODUCER	<div style="width: 100%;"></div>	100% ✓ SUCCEEDED	00:00:00	👁

Displaying 1 to 2 of 2

6.11.6.2.1 Execution Summary

This section displays more granular information about execution including; Environment name, Connector name, Job Name, Previous run time, Number of tables/files in the job, Number of tables/files masked, Number of tables/files to be masked, Type of job, Rows remaining to mask, Rows masked, Number of streams, etc.

6.11.6.2.2 Events


This section displays the Start time, Run Time, and End time of the job and also the sequence in which the events are executed.




The events are executed in the following sequence:

1. **Init Execution:** The execution has begun.
2. **Collecting Job Configurations:** Collect the Job details stored in the MDS/DB.
3. **Preparing Execution:** Create transformation XML for a kettle that includes, pre-script, post-script, create and drop identity XML.
4. **Execute Pre-Execution Custom Driver Task:** Execute custom drive pre-execution tasks.
5. **Start Execution:** Starts the masking Job
6. **Pre SQL Script:** execution prescript if available
7. **Post-SQL Script:** execution postscript if available
8. **Execute Post Execution Custom Driver Task:** Execute post-SQL operations from the custom driver
9. **Collect Job Information:** Collect and store execution information in the database.
10. **Execution Finished:** The execution is finished and removed from monitoring

6.11.6.2.3 Execution Components


In addition to seeing this information about each job, you can look into the status/progress of each table/file defined in the job. Each table/file will be displayed under **Execution Components**. It includes information such as Time taken for processing, Rows Masked per Minute, Rows Masked, Rows Remaining, Number of columns/fields having Non-conformant data, etc.

Users can maximize this section by using the icon  on the top right corner of this section.

Execution Components 							
ID ↑	Table Name	Progress	Status	Duration (HH:mm:ss)	Enter full screen Actions		
55	testdata_FKCONSUMER	<div style="width: 100%;"><div style="width: 100%;"></div></div>	100% ✔ SUCCEEDED	00:00:01			
56	testdata_FKPRODUCER	<div style="width: 100%;"><div style="width: 100%;"></div></div>	100% ✔ SUCCEEDED	00:00:00			

Displaying 1 to 2 of 2

27 Execution Components Minimized

Logs for individual tables/files can be viewed by clicking the **View Logs** -  option under the **Actions** column.

ID ↑	Table Name	Progress	Status	Duration (HH:mm:ss)	Rows Per Min	Rows Masked	Rows Remaining	Exit full screen Actions
55	testdata_FKCONSUMER	<div style="width: 100%;"></div>	100% ✔ SUCCEEDED	00:00:01	434	10	0	
56	testdata_FKPRODUCER	<div style="width: 100%;"></div>	100% ✔ SUCCEEDED	00:00:00	2690	10	0	

28 Execution Components Maximized

i

- On hovering on the column name, the information about column is displayed in the tooltip.
- The execution components grid supports sorting and filtering on ID, Table Name, Status and Rows Masked columns.
- This grid doesn't support displaying filter chips after the filter is applied and filters are not persisted in cache.
- More information on grid filtering and sorting can be found [here \(see page 232\)](#).

6.11.6.2.4 Report and Logs

Use the **Report and Logs** button to retrieve Job and Inventory Reports in PDF formats. From this screen, the user can also view logs for this execution by clicking on **“Execution Log”**.

The screenshot shows the 'Report and Logs' section for a job named 'mask_UZAC09VV'. On the left, an 'Execution Summary' sidebar lists details: Execution ID 34, Environment env_DSWQZCJZ, Job Name mask_UZAC09VV, Job Type Mask, Job ID 28, and Rule Set rs_7PJ12BW8. The main area shows a 'Success' status with a timeline from 13:03:53 IST to 13:04:08 IST. A list of tasks includes: Initializing, Collecting Configurations, Preparing, Running Pre-execution Custom Driver Tasks, Starting, Running PreSQL Scripts, Running PostgreSQL Scripts, Running Post-execution Custom Driver Tasks, Collecting Information, and Job Completed. A 'Report and Logs' menu is open, showing options for 'Execution Log', 'Inventory Report', and 'Job Report'.

6.11.6.2.5 Profile Results

This section comes for Profiling jobs. It displays the sensitive data findings on a table-column by table-column or file-field by file-field basis.

CONTINUOUS COMPLIANCE
24.0.0.0

[Environments](#)
[Monitor](#)
[Settings](#)
[Admin](#)
[Audit](#)

Environments > env_W3466FEU > prof_1MVP92EA

< Back **prof_1MVP92EA**

Execution ID: 56

✔ Profiling Completed Report and Logs

^ Execution Summary

Execution ID
56

Environment
[env_W3466FEU](#)

Job Name
prof_1MVP92EA

Job Type
Profile

Job ID
49

Rule Set
[con_7Z1OKJH0](#)

Connection Type
Delimited

Source
-

Target
con_2MDCBE8W

Previous Run Time
-

Streams
1

Total # of Files
2

Files Profiled
2

Files to be Profiled
0

^ Success Start Time: 31 May 2024 11:48:31 IST End Time: 31 May 2024 11:48:43 IST Run Time: 00:00:12

- ✔ Initializing
- ✔ Collecting Configurations
- ✔ Preparing
- ✔ Starting
- ✔ Profiling Completed

^ Execution Components ↗

ID ↑	File Name	Progress	Status	Duration (HH:mm:ss)	Actions
105	delimited3xa0eiuw.txt	<div style="width: 100%;"></div>	100% ✔ SUCCEEDED	00:00:10	👁
106	delimite7qfj4.txt	<div style="width: 100%;"></div>	100% ✔ SUCCEEDED	00:00:11	👁

Displaying 1 to 2 of 2

^ Profile Results ↗

File Name	Field Name ↑	Algorithm	Domain
delimited3xa0eiuw.txt	DATA_00	dlpx-core:Phone Unique	TELEPHONE_NO
delimite7qfj4.txt	DATA_00	dlpx-core:Phone Unique	TELEPHONE_NO
delimite7qfj4.txt	DATA_01	dlpx-core:Email Unique	EMAIL
delimited3xa0eiuw.txt	DATA_01	dlpx-core:Email Unique	EMAIL
delimited3xa0eiuw.txt	FIRSTNAME_00	dlpx-core:FirstName	FIRST_NAME
delimite7qfj4.txt	FIRSTNAME_00	dlpx-core:FirstName	FIRST_NAME

Displaying 1 to 7 of 8

i

- Profile results grid supports sorting and filtering on all the columns.
- This grid doesn't support displaying filter chips after the filter is applied and filters are not persisted in cache.
- More information on grid filtering and sorting can be found [here](#) (see page 513).

6.11.6.3 Displaying non-conformant data

When a masking job encounters non-conformant data, the job will either Fail or Succeed with a warning, depending on how the algorithms associated with the ruleset for the job are configured. As depicted in the screenshot, the non-conformant data can be accessed via the **Execution Components** on the Monitor page for the job, which can be accessed by clicking on the Job name from the Monitor Grid. In the **Execution Summary** section, a summary of the **Tables with Nonconforming Data** and **Columns with Nonconforming**

Data is reported. Further details on the non-conformant data encountered can be accessed by clicking the **“View Logs - [Eye Icon]”** on each table or file listed in the **Execution Components** grid.

Success Report



Events for This Execution Component Learn More			
Event	Cause	Approximate Count	Description
UNMASKED_DATA	PATTERN_MATCH_FAILURE	1	<p>Column EXCEPTION_00 value did not conform to the expected format algorithm plg_AALNM2I8:DelphixMaskingException: Expected QA Exception: NonConformantDataException</p> <p>The top nonconforming data samples were:</p> <ul style="list-style-type: none"> • LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL

18_17_testdata_EXCEPTION.txt

```
'author': 'Delphix QA', 'apiVersion': '1.0.0-SNAPSHOT'}
2023/12/07 07:56:01 - Table input.0 - Attempting to set statement fetch size to 10000
2023/12/07 07:56:01 - Table input.0 - Initializing ThroughputLogger
2023/12/07 07:56:02 - env IRZUEEZ4.con.580TRPZD - Statement fetch size set to 10000
2023/12/07 07:56:02 - Table input.0 - JobMemoryManager: Total Pause 0/1173ms (0%) Heap 16086864b of 1045954560b (2%) GC Count 1
2023/12/07 07:56:02 - Table input.0 - ThroughputLogger:interval 0.068 MB (1000 rows) counted in 1.168 sec; throughput = 0.058 MB/sec
2023/12/07 07:56:02 - Table input.0 - ThroughputLogger:total 0.068 MB (1000 rows) counted in 1.168 sec; throughput = 0.058 MB/sec
2023/12/07 07:56:02 - Table input.0 - Finished reading query, closing connection.
2023/12/07 07:56:02 - Select values.0 - Finished processing (I=0, O=0, R=1000, W=1000, U=0, E=0)
2023/12/07 07:56:02 - Table input.0 - Finished processing (I=1000, O=0, R=0, W=1000, U=0, E=0)
2023/12/07 07:56:02 - Get All Lookups Values.0 - Finished processing (I=0, O=0, R=1000, W=1000, U=0, E=0)
2023/12/07 07:56:02 - Delphix Algorithm.0 - Batch size is: 4000
2023/12/07 07:56:02 - Delphix Algorithm.0 - Extended Algorithm Usage (EXALG) for Table or File:
2023/12/07 07:56:02 - Delphix Algorithm.0 - EXALG: fields=EXCEPTION_00 () algorithm=plg_AALNM2I8:DelphixMaskingException plugin=plg_AALNM2I8
2023/12/07 07:56:02 - Delphix Algorithm.0 - Algorithm step ended 0 worker threads
2023/12/07 07:56:02 - Delphix Algorithm.0 - Finished processing (I=0, O=0, R=1000, W=1000, U=0, E=0)
2023/12/07 07:56:02 - SelectValues_MetaData.0 - Finished processing (I=0, O=0, R=1000, W=1000, U=0, E=0)
2023/12/07 07:56:02 - String Cut.0 - Finished processing (I=0, O=0, R=1000, W=1000, U=0, E=0)
2023/12/07 07:56:02 - DelphixTableUpdate.0 - Finished processing (I=1000, O=0, R=1000, W=1000, U=1000, E=0)
```

The non-conformant data events are displayed followed by the masking log for the table or file. If there were no non-conformant data events, "No Events to Display" is displayed, otherwise, for each type of non-conformant data, a row will be displayed with the following information:

- **Event type:** Either JOB_ABORTED or UNMASKED_DATA if the job was not aborted.
- **Cause:** Always PATTERN_MATCH_FAILURE.
- **Approximate Row Count:** Approximate number of rows with non-conformant data (at least within an order of magnitude).
- **Description:** Details the name of the column or field with non-conformant data


6.11.6.4 Interpreting samples of non-conformant data patterns

Each character in the non-conformant data is sampled per its [Unicode Character Property](https://en.wikipedia.org/wiki/Unicode_character_property)¹⁸⁰.


- N for digits
- L for letters
- M for marks
- P for punctuation
- S for symbols
- Z for separator
- O for other
- U for unknown

¹⁸⁰ https://en.wikipedia.org/wiki/Unicode_character_property

6.11.6.5 Tracking Non-conformant Data

 Please note that actual personal data is never displayed, only the samples (a.k.a. patterns) of non-conformant data are displayed on this page

Using the database-specific SQL query, it is possible to locate data corresponding to the non-conformant data sample. The table and column names can be found on the table report. In the example above, the table name is "testdata_EXCEPTION" and the column name is "EXCEPTION_00".

 **Note**
The pattern might be not an exact representation of the data in the field, but a part of the data. For instance, white spaces at the beginning or at the end of the data might be truncated.

6.11.6.5.1 Oracle example

Below are the [Oracle character classes](#)¹⁸¹, used in the regular expression:

Character Class Syntax	Meaning
[[:alnum:]]	All alphanumeric characters
[[:alpha:]]	All alphabetic characters
[[:blank:]]	All blank space characters.
[[:cntrl:]]	All control characters (nonprinting)
[[:digit:]]	All numeric digits
[[:graph:]]	All [[:punct:]], [[:upper:]], [[:lower:]], and [[:digit:]] characters.
[[:lower:]]	All lowercase alphabetic characters

¹⁸¹ https://docs.oracle.com/cd/B12037_01/server.101/b10759/ap_posix001.htm

Character Class Syntax	Meaning
[[:print:]]	All printable characters
[[:punct:]]	All punctuation characters
[[:space:]]	All space characters (nonprinting)
[[:upper:]]	All uppercase alphabetic characters
[[:xdigit:]]	All valid hexadecimal characters

For the LLLLL sample in the example above, the Oracle DB SQL query would look like this:

```
SELECT EXCEPTION_00 FROM testdata_EXCEPTION WHERE regexp_like(EXCEPTION_00,
'[[[:alpha:]]]{5}');
```

For the LLLLZLLLZLLLL sample, the Oracle DB SQL query would look like:

```
SELECT EXCEPTION_00 FROM testdata_EXCEPTION WHERE regexp_like(EXCEPTION_00,
'[[[:alpha:]]]{4}[[[:space:]]][[:alpha:]]{3}[[[:space:]]][[:alpha:]]{4}');
```

6.11.6.5.2 Limitation for the multi-column extensible algorithm

If a Non-conformant data pattern is encountered - it is displayed for all the masked columns of the MC Algorithm, not only for the column where that event has occurred. In that case, the manual analysis of the error message will be required to find the actual column(s) with the Non-conformant data.

6.12 Whole file masking

You can now configure the masking engine to mask the complete file and pass the content of that file as a single input to an algorithm.


6.12.1 Pre-requisites

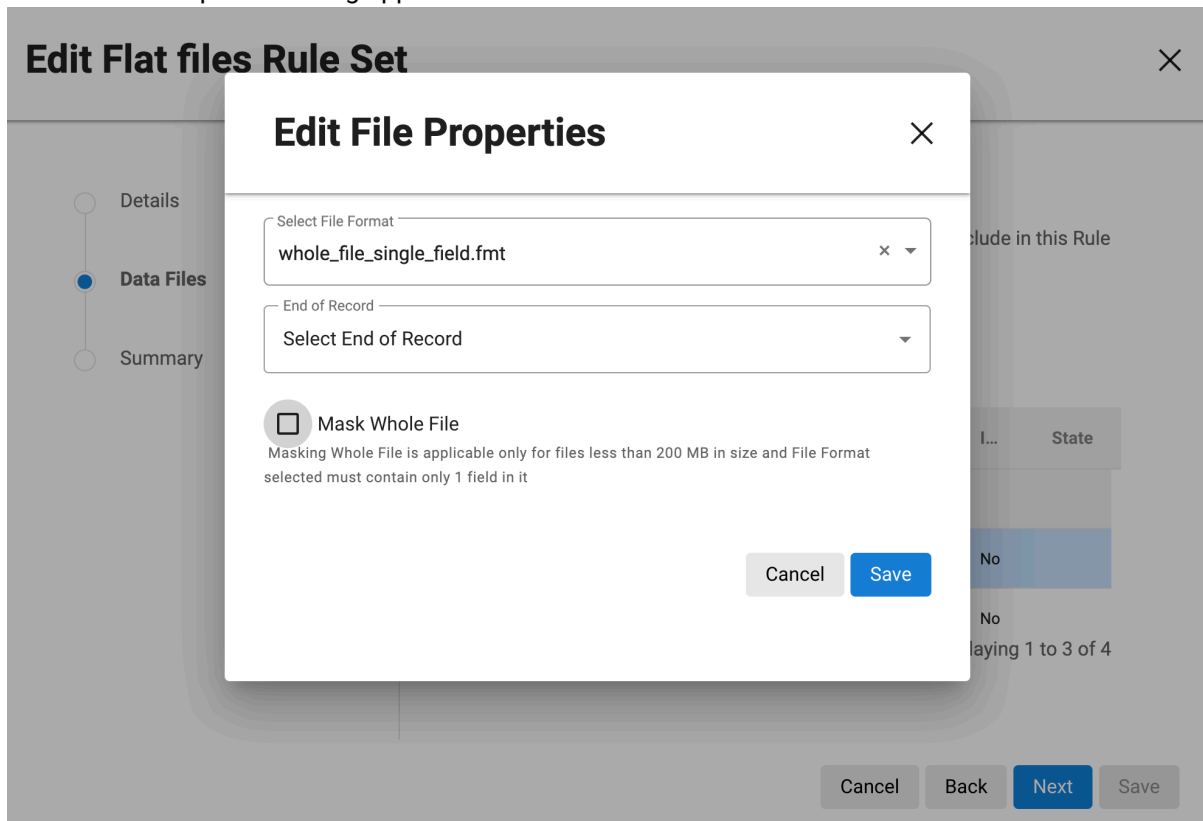
- You must create a fixed-width file connector.
- You must create a Rule Set that has:
 - a. Only one field.
 - b. Of Length 0.

For more information, see [Managing connectors](#) (see page 355), [Managing file formats](#) (see page 427), and [Managing rule sets](#)¹⁸².

6.12.2 Masking a whole file

Here are the steps to process the entire file's content using a single algorithm. This functionality is limited to fixed-width ruleset types only.

1. Navigate to **Environments > Rule Sets**.
2. Click the Actions (...) drop-down to the right of a rule set on the **rule set** screen, and then select the **Edit** -  option.
3. In the rule set wizard's second step "**Data Files**", you can select one or more files or regular expression patterns either by clicking on the checkboxes or by clicking anywhere on the rows.
4. Click on the "**Edit Selected**" button, located on top of the grid.
5. The Edit File Properties dialog appears.



6. From the **Select File Format** drop-down, select a file format that has only one field defined in it. Selecting any other file format will result in an error.


¹⁸² <https://masking.delphix.com/docs/latest/managing-rule-sets>

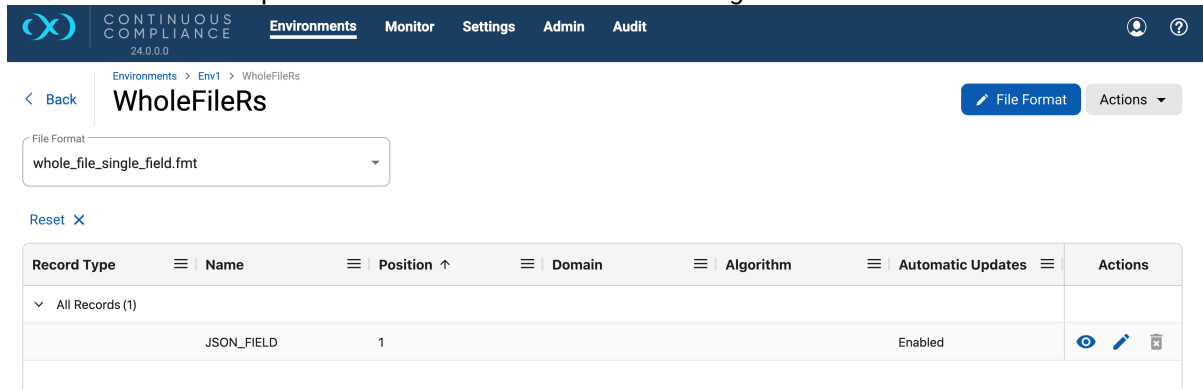
7. Select the **Mask whole file** checkbox to enable whole file masking. Selecting this option will make the "End of Record Field" disappear. This configuration is no longer necessary because the masking engine will now read the entire file and send it to the algorithm.

Note:

Masking the whole file is applicable only for:

- Files that are less than 200 MB in size. However, you can modify this limit via API by configuring **Whole File Masking Max File Size In MB** key in the Application Settings.
- The file format that has only one field defined in it. The masking whole file is applicable only for: Files that are less than 200 MB in size. However, you can modify this limit via API by configuring **Whole File Masking Max File Size In MB** key in the Application Settings. File format that has only one field defined in it.

8. Click on **Save**.
9. Go to **Environments > Rule Set** and click on the hyperlink of the ruleset name associated with the above ruleset. Alternatively, you can navigate to **Settings > Data Format** and click on the hyperlink of the name of the single-field file format created above. This will bring up an inventory-like screen listing an entry for the single-file field.
10. The **Add Record Types**, **Add Fields**, and **Delete** options will not be available for file formats marked with whole file masking.
11. Click the **Edit** -  option under the **Actions** column to the right of the field.



12. From the **Algorithm** drop-down list, select the matching extended algorithms that must be applied to the file.

Edit Field ✕

Information
If the algorithm selected does not support Tokenization and is used in a Tokenize/Re-Identify environment, it would not be possible to reidentify the data.

Field Name
JSON_FIELD

Notes (Optional) ↗

Formatting

Record Type: All Records ✕ ▼ Position: 1 Length: 0

Masking

Uncheck 'Enable Automatic Updates' to make Masking assignments editable.

Enable Automatic Updates

Select Domain ▼ Select Algorithm ▼

Cancel Save

i You are not allowed to adjust the length or position of the field defined in a file format if the file format is assigned to any data file in fixed-width ruleset having "Mask whole file" option enabled.

13. Click **Save**.

7 Identifying sensitive data

This section contains the following topics:

- [Discovering your sensitive data \(see page 526\)](#)
- [Out of the box profiling settings \(see page 528\)](#)
- [ASDD standard profile set \(see page 529\)](#)
- [Standard profile set expressions \(see page 586\)](#)
- [Legacy profile set expressions \(see page 592\)](#)
- [Configuring profile sets \(see page 603\)](#)
- [Managing domains \(see page 611\)](#)
- [Managing classifiers \(see page 616\)](#)
- [Managing expressions \(see page 630\)](#)
- [ASDD profile set import and export \(see page 646\)](#)
- [Reporting profiling results \(see page 647\)](#)
- [ASDD features and support \(see page 653\)](#)

7.1 Discovering your sensitive data

7.1.1 Overview

After connecting data to the masking service, the next step is to discover which of the data should be secured. This process is referred as *sensitive data discovery*, or *profiling* throughout the product documentation.

Once a rule set has been [created \(see page 400\)](#), profiling is done by [Managing rule sets \(see page 400\)](#) and [running](#)¹⁸³ a profiling job for that rule set. A profiling job examines the metadata, such as column names and types, and potentially the data itself, to determine which columns or fields contain sensitive information. Upon determining that a data item is sensitive, the profiler assigns the matching domain and associated masking algorithm to the column or field. A profiling job covers only those tables and files present in the rule set; any new objects accessible through the defined connector will not be discovered and must be manually added to the rule set.

The Continuous Compliance product currently ships with two distinct profiling implementations: the new Automated Sensitive Data Discovery (ASDD) profiler and the legacy profiler. The content of the profile set determines which implementation will be chosen when a profiling job is run. The [ASDD profiler supports \(see page 653\)](#) a wider range of logic for detecting sensitive fields and improved data inspection logic for databases. However, at this time, ASDD profiling is limited to only specific database variants.

¹⁸³ <https://delphixdocs.atlassian.net/wiki/spaces/CC/pages/9930081/Running+a+profiling+job>

7.1.2 Concepts

7.1.2.1 Profile set

The *Profile Set* chosen defines the logic that will be used to determine which columns or fields in the rule set contain sensitive information. A profile set may contain a set of *search expression* and *type expressions*, or a set of *classifiers*, that define the recognition logic for the legacy or ASDD profiler, respectively. As each expression or classifier is associated with a *domain*, the composition of the profile set determines which types of sensitive data may be detected by a profiling job use a particular profile set. Several [built-in profile sets](#) (see page 528) are available by default.

7.1.2.2 Domain

A domain represents a particular type of sensitive information, such as first name or tax ID number. Based on the detection logic in the profile set, a profile job may assign a domain to a particular field or column in the rule set; when this occurs, the default masking algorithm defined for that domain will also be assigned. The domain mechanism helps to ensure that the same masking algorithm is applied consistently across rule sets whenever a particular type of sensitive data is discovered.

7.1.2.3 Level - column or data

The term Level is used for search expressions to indicate whether the data itself is examined, or if profiling is done based only on the field or column name and type. Examining the data is more time-consuming than examining metadata alone, as the profiling job must retrieve data from the data source.

7.1.2.4 Classifier

A classifier defines a specific piece of logic for recognizing sensitive data. Classifiers may only be used with the ASDD profiler. Classifiers use a framework and instance model, similar to algorithms. A framework represents a particular software module for detecting sensitive information, while an instance provides the configuration for a framework and associates it with a particular domain. The pre-built *ASDD Standard* profile set includes a number of classifier instance definitions. It is possible to create additional instances using the API client.

The following classifier frameworks are available:

- **PATH** - Examines the path to the data in question and applies regular expression and/or exact match logic to match domains. For databases, the path includes the table and column name.
- **TYPE** - Uses the data type and length of a field or column to reject possible domain matches. Supported types are String, Number, Date and Binary.
- **REGEX** - Matches the data itself using regular expressions to match or reject domains.
- **LIST** - Checks whether data values are present in a list of value to match or reject domains.

Of these frameworks, **PATH** and **TYPE** operate at the column level, while **REGEX** and **LIST** operate at the data level. It is not currently possible to install additional classifier frameworks.

7.1.2.5 Search expression

A search expression defines a regular expression (regex) that will be used to match data to a domain. How the regex is applied depends on the value chosen for level - column-level expressions are matched against the field or column name, while data-level expressions are matched against the data values themselves. Every legacy, built-in profile set includes a number of column-level search expressions

designed to identify common sensitive data types (SSN, Name, Addresses, etc). The pre-built profile sets do not include any data level expressions by default, but some [data level expressions](#) (see page 630) are included (but not part of any profile set) that may be added to user-created profile sets. You also have the ability to create additional search expressions.

7.1.2.6 Type expression

A type expression defines a constraint limiting matches for a particular domain to a particular set of data types, with an optional minimum length for each type. For example, matches for the FIRST_NAME domain may be limited to only string columns with a length of 8 characters or more. Supported types are STRING, NUMBER, DATE, and BINARY. The [Standard](#) (see page 586) profile set includes type expressions for most domains, and more may be created if desired.

7.2 Out of the box profiling settings

The Delphix Platform comes out of the box with recognition logic to help you discover over 30 types of sensitive data (account numbers, addresses, etc.). This logic is organized into a number of pre-built profile sets that can be easily applied to a rule set when a profile job is created.

7.2.1 ASDD standard profile set

This is the recommended profiler set for the ASDD profiler and should be preferred for all [data sources supported](#) (see page 653) by the ASDD Profiler. This profile set has the widest range of classification logic, including classifiers for all logic in the legacy **Standard** profile set, as well as data-level classifiers for a number of domains. It includes value list classifiers capable of detecting several domains, such as FIRST_NAME and LAST_NAME, even when column names are not meaningful. Data level detection is limited to English language values.

The classifiers present in the ASDD Standard profile set are described in the [ASDD Standard Profile Set](#) (see page 529) section.

7.2.2 Standard profile set

This is the recommended profile set for the legacy profiler. It contains column-level search and type expressions appropriate for detecting a wide range of sensitive information.

The column and type expressions used in this profile set are described in the [Standard Profile Set Expressions](#) (see page 586) section.

7.2.3 Legacy profile sets

The legacy profile sets are provided for backward compatibility, specifically, to provide consistent results for pre-existing profiling jobs. For other uses, the *Standard* profile set described above is preferred. The legacy profile sets do not contain any type expressions to restrict matching based on the column type.

These profile sets are:

- Financial - Legacy
- HIPAA - Legacy

The expressions used by these profile sets are described in the [Legacy Profile Set Expressions](#) (see page 592) section.

7.3 ASDD standard profile set

As of the 11.0.0.0 release, administrators control when upgrades of the **ASDD Standard Profile Set** occur. For every Delphix Engine release, the ASDD Standard Profile Set will be made available to users in the **Automated Sensitive Data Discovery** directory folder on the [Delphix Download site](#)¹⁸⁴.

A README is also present, which includes a change.log and usage instructions. Additionally, as of 11.0.0.0, the contents of the ASDD Standard Profile Set may be customized. To recover or upgrade the ASDD Standard profile set, see the [ASDD Profile Set Import and Export article](#) (see page 646) for more information on the import API endpoint.

For reference, you can download or expand the JSON listing below, which contains the full configuration for all classifiers present in the ASDD Standard Profile Set as of the 21.0.0.0 release. The profile set for individual releases can be found in the [downloads section](#).¹⁸⁵

[ASDD standard profile set JSON classifiers list.json](#)¹⁸⁶

ASDD Standard Profile Set JSON classifiers listing

```
[
  {
    "domain": "ACCOUNT_NO",
    "name": "Account Number - Path",
    "type": "PATH",
    "properties": {
      "paths": [
        {
          "matchType": "REGEX",
          "matchStrength": 0.67,

```

¹⁸⁴ <https://download.delphix.com/folder/4385/Delphix%20Product%20Releases/Automated%20Sensitive%20Data%20Discovery>

¹⁸⁵ <https://download.delphix.com/folder/4385/Delphix%20Product%20Releases/Automated%20Sensitive%20Data%20Discovery>

¹⁸⁶ <https://delphixdocs.atlassian.net/wiki/download/attachments/205360246/ASDD%20standard%20profile%20set%20JSON%20classifiers%20list.json?api=v2&cacheVersion=1&modificationDate=1724058992726&version=1>

```

        "fieldValue": "(?i)(?>(account|acct|acct)_?-? ?(number|num|nbr|
no|user))($|[ _-])"
    },
    "rejectStrength": 0.0
}
},
{
    "domain": "ACCOUNT_NO",
    "name": "Account Number - Type",
    "type": "TYPE",
    "properties": {
        "allowedTypes": [
            {
                "minimumLength": "5",
                "typeName": "String"
            },
            {
                "minimumLength": "5",
                "typeName": "Number"
            }
        ],
        "matchAutoIncrementingColumn": true
    }
},
{
    "domain": "ADDRESS",
    "name": "Address Line 1 - Path",
    "type": "PATH",
    "properties": {
        "paths": [
            {
                "matchType": "REGEX",
                "matchStrength": 0.67,
                "fieldValue": "(?i)(?>((st(reet)?_?-? ?addr(ess)?)|street?|(?<!
email[_- ]?|ip[_- ]?|web[_- ]?|url[_- ]?)addr(ess)?_?-? ?((l(i)?n(e)? ?_?(1|))?)?"
            }
        ],
        "rejectStrength": 0.0
    }
},
{
    "domain": "ADDRESS",
    "name": "Address Line 1 - Type",
    "type": "TYPE",
    "properties": {
        "allowedTypes": [
            {
                "minimumLength": "20",
                "typeName": "String"
            }
        ]
    }
}

```



```

},
{
  "domain": "ADDRESS",
  "name": "Address Line 1 - Regex",
  "type": "REGEX",
  "properties": {
    "dataPatterns": [
      {
        "regex": "(?i)(.*[\\s]+b(ou)?l(e)?v(ar)?d[\\s]*.*)|(.*[\\s]+st(reet)?[\\s]*.*)|(.*[\\s]+ave[.]?(nue)?[\\s]*.*)|(.*[\\s]+r(oa)?d[\\s]*.*)|(.*[\\s]+l(a)?n(e)?[\\s]*.*)|(.*[\\s]+cir(cle)?[\\s]*.*)|(.*[\\s]+dr[.](ive)?[\\s]*.*)|(.*[\\s]+h(igh)?w(a)?y[\\s]*.*)|(.*[\\s]+r(ou)?t(e)?[\\s]*.*)|(.*[\\s]+c(our)?t[\\s]*.*)|(.*[\\s]+way[\\s]*.*)",
        "matchStrength": 0.7,
        "allowPartialMatch": false
      }
    ],
    "rejectStrength": 0.1
  }
},
{
  "domain": "ADDRESS_LINE2",
  "name": "Address Line 2 - Path",
  "type": "PATH",
  "properties": {
    "paths": [
      {
        "matchType": "REGEX",
        "matchStrength": 0.67,
        "fieldValue": "(?i)(?>((st(reet)?_?-? ?addr(ess)?)|street?|(?<!email[-_ ]?|ip[-_ ]?|web[-_ ]?|url[-_ ]?)addr(ess)?_?-? ?((l(i)?n(e)? ?_?([2-9])))?)"
      }
    ],
    "rejectStrength": 0.0
  }
},
{
  "domain": "ADDRESS_LINE2",
  "name": "Address Line 2 - Type",
  "type": "TYPE",
  "properties": {
    "allowedTypes": [
      {
        "minimumLength": "20",
        "typeName": "String"
      }
    ]
  }
},
{
  "domain": "ADDRESS_LINE2",
  "name": "Address Line 2 - Regex",

```

```

    "type": "REGEX",
    "properties": {
      "dataPatterns": [
        {
          "regex": "(?i)(.*[\\s]*ap(ar)?t(ment)?[.\\s]+.*)|(.*[\\s]*s(ui)?te[.\\s]+.*)|(c(are)?[\\s]*[\\]\\]?[/]?o(f)?[.\\s]+.*)|(.*[\\s]*b(ui)?ld(in)?g[.\\s]+.*)|(.*[\\s]*fl(oor)?[.\\s]+.*)|(.*[\\s]*r(oo)?m[.\\s]+.*)|(.*[\\s]*unit[.\\s]+.*)",
          "matchStrength": 0.7,
          "allowPartialMatch": false
        }
      ],
      "rejectStrength": 0.1
    }
  },
  {
    "domain": "AGE",
    "name": "Age - Path",
    "type": "PATH",
    "properties": {
      "paths": [
        {
          "matchType": "REGEX",
          "matchStrength": 0.67,
          "fieldValue": "(?i)(?>(^[ _-])(age)[ _-]? (group|grp|number|number|no)?)(${[ _-]}")
        }
      ],
      "rejectStrength": 1.0
    }
  },
  {
    "domain": "AGE",
    "name": "Age - Regex",
    "type": "REGEX",
    "properties": {
      "dataPatterns": [
        {
          "note": "Number range between 01 --> 99",
          "regex": "\\d{1,2}",
          "matchStrength": 0.1,
          "allowPartialMatch": false
        }
      ],
      "rejectStrength": 0.1
    }
  },
  {
    "domain": "AGE",
    "name": "Age - Type",
    "type": "TYPE",
    "properties": {
      "allowedTypes": [
        {

```

```

        "minimumLength": "2",
        "typeName": "Number"
    },
    {
        "minimumLength": "2",
        "typeName": "String"
    }
]
},
{
    "domain": "BANK_ACCOUNT_NO",
    "name": "Bank Account Number - Path",
    "type": "PATH",
    "properties": {
        "paths": [
            {
                "matchType": "REGEX",
                "matchStrength": 0.67,
                "fieldValue": "(?i)(?>(bank_?-? )?(account|acct|acct)_?-? ?
(number|num|nbr|no))($|[ _-])"
            }
        ],
        "rejectStrength": 0.0
    }
},
{
    "domain": "BANK_ACCOUNT_NO",
    "name": "Bank Account Number - Regex",
    "type": "REGEX",
    "properties": {
        "dataPatterns": [
            {
                "regex": "\\d{5,17}",
                "matchStrength": 0.05,
                "allowPartialMatch": false
            }
        ],
        "rejectStrength": 0.1
    }
},
{
    "domain": "BANK_ACCOUNT_NO",
    "name": "Bank Account Number - Type",
    "type": "TYPE",
    "properties": {
        "allowedTypes": [
            {
                "minimumLength": "5",
                "typeName": "Number"
            }
        ],
        "matchAutoIncrementingColumn": true
    }
}

```

```

    }
  },
  {
    "domain": "BENEFICIARY_NO",
    "name": "Beneficiary Number - Path",
    "type": "PATH",
    "properties": {
      "paths": [
        {
          "matchType": "REGEX",
          "matchStrength": 0.67,
          "fieldValue": "(?i)(?>(bene(ficiary)?_?-? ?(number|num|nbr|no|
id)))"
        }
      ],
      "rejectStrength": 0.0
    }
  },
  {
    "domain": "BENEFICIARY_NO",
    "name": "Beneficiary Number - Type",
    "type": "TYPE",
    "properties": {
      "allowedTypes": [
        {
          "minimumLength": "5",
          "typeName": "Number"
        },
        {
          "minimumLength": "10",
          "typeName": "String"
        }
      ]
    }
  },
  {
    "domain": "BIOMETRIC",
    "name": "Biometric - Path",
    "type": "PATH",
    "properties": {
      "paths": [
        {
          "matchType": "REGEX",
          "matchStrength": 0.67,
          "fieldValue": "(?i)(?>(biometric))"
        }
      ],
      "rejectStrength": 0.0
    }
  },
  {
    "domain": "BIOMETRIC",
    "name": "Biometric - Type",

```

```

"type": "TYPE",
"properties": {
  "allowedTypes": [
    {
      "minimumLength": "10",
      "typeName": "String"
    },
    {
      "minimumLength": "0",
      "typeName": "Binary"
    }
  ]
}
},
{
  "domain": "BLOOD_TYPE",
  "name": "Blood Type - List",
  "type": "LIST",
  "properties": {
    "valueLists": [
      {
        "file": "file://blood_types.txt",
        "matchStrength": 1.0
      }
    ]
  }
},
{
  "domain": "BLOOD_TYPE",
  "name": "Blood Type - Path",
  "type": "PATH",
  "properties": {
    "paths": [
      {
        "matchType": "REGEX",
        "matchStrength": 0.67,
        "fieldValue": "(?>blood[-_ ]?(group|grp|type))"
      }
    ],
    "rejectStrength": 0.0
  }
},
{
  "domain": "BLOOD_TYPE",
  "name": "Blood Type - Type",
  "type": "TYPE",
  "properties": {
    "allowedTypes": [
      {
        "minimumLength": "2",
        "typeName": "String"
      }
    ]
  }
}

```

```

    }
  },
  {
    "domain": "CERTIFICATE_NO",
    "name": "Certificate Number - Path",
    "type": "PATH",
    "properties": {
      "paths": [
        {
          "matchType": "REGEX",
          "matchStrength": 0.67,
          "fieldValue": "(?i)(?>(^|[-_ ])cert(ificate)?_?-? ?(number|num|
nbr|no|id))"
        }
      ],
      "rejectStrength": 0.0
    }
  },
  {
    "domain": "CERTIFICATE_NO",
    "name": "Certificate Number - Type",
    "type": "TYPE",
    "properties": {
      "allowedTypes": [
        {
          "minimumLength": "10",
          "typeName": "String"
        },
        {
          "minimumLength": "5",
          "typeName": "Number"
        }
      ]
    }
  },
  {
    "domain": "CITY",
    "name": "City - List",
    "type": "LIST",
    "properties": {
      "valueLists": [
        {
          "file": "file://us_cities.txt",
          "matchStrength": 1.0
        }
      ]
    }
  },
  {
    "domain": "CITY",
    "name": "City - Path",
    "type": "PATH",
    "properties": {

```

```

    "paths": [
      {
        "matchType": "REGEX",
        "matchStrength": 0.67,
        "fieldValue": "(?i)(?>^(home_?-? ?city|city))"
      },
      {
        "matchType": "REGEX",
        "matchStrength": 0.67,
        "fieldValue": "(?i)(?>^(address_?-? ?city|city|city_?-? ?
address))"
      }
    ],
    "rejectStrength": 0.0
  }
},
{
  "domain": "CITY",
  "name": "City - Type",
  "type": "TYPE",
  "properties": {
    "allowedTypes": [
      {
        "minimumLength": "10",
        "typeName": "String"
      }
    ]
  }
},
{
  "domain": "COMPANY_NAME",
  "name": "Company Name - List",
  "type": "LIST",
  "properties": {
    "valueLists": [
      {
        "file": "file://companies.txt",
        "matchStrength": 1.0
      }
    ]
  }
},
{
  "domain": "COMPANY_NAME",
  "name": "Company Name - Path",
  "type": "PATH",
  "properties": {
    "paths": [
      {
        "matchType": "REGEX",
        "matchStrength": 0.67,
        "fieldValue": "(?>(comp|company|org|organization|employer)[-_ ]?
(name|nm))"
      }
    ]
  }
}

```

```

        }
      ],
      "rejectStrength": 0.0
    }
  },
  {
    "domain": "COMPANY_NAME",
    "name": "Company Name - Type",
    "type": "TYPE",
    "properties": {
      "allowedTypes": [
        {
          "minimumLength": "6",
          "typeName": "String"
        }
      ]
    }
  },
  {
    "domain": "COUNTRY",
    "name": "Country - Path",
    "type": "PATH",
    "properties": {
      "paths": [
        {
          "matchType": "REGEX",
          "matchStrength": 0.67,
          "fieldValue": "(?i)country",
          "allowPartialMatch": false
        }
      ],
      "rejectStrength": 0.0
    }
  },
  {
    "domain": "COUNTRY",
    "name": "Country - Type",
    "type": "TYPE",
    "properties": {
      "allowedTypes": [
        {
          "minimumLength": "15",
          "typeName": "String"
        }
      ]
    }
  },
  {
    "domain": "COUNTRY",
    "name": "Country - List",
    "type": "LIST",
    "properties": {
      "valueLists": [

```



```

        {
            "file": "file://countries.txt",
            "matchStrength": 1.0
        }
    ],
    "rejectStrength": 0.5
}
},
{
    "domain": "COUNTY",
    "name": "County - Path",
    "type": "PATH",
    "properties": {
        "paths": [
            {
                "matchType": "REGEX",
                "matchStrength": 0.67,
                "fieldValue": "(?i)(?>(county))"
            }
        ],
        "rejectStrength": 0.0
    }
},
{
    "domain": "COUNTY",
    "name": "County - Type",
    "type": "TYPE",
    "properties": {
        "allowedTypes": [
            {
                "minimumLength": "10",
                "typeName": "String"
            }
        ]
    }
},
{
    "domain": "CREDIT CARD",
    "name": "Credit Card Number - Path",
    "type": "PATH",
    "properties": {
        "paths": [
            {
                "matchType": "REGEX",
                "matchStrength": 0.67,
                "fieldValue": "(?i)(?>(^cc|credit[ -_]?card)[ -_]?((number|num|
nbr|no)?)"
            },
            {
                "matchType": "REGEX",
                "matchStrength": 0.67,
                "fieldValue": "(?i)(?>card[ -_]?((number|num|nbr|no))"
            }
        ]
    }
}

```

```

    ],
    "rejectStrength": 0.0
  }
},
{
  "domain": "CREDIT CARD",
  "name": "Credit Card Number - Regex",
  "type": "REGEX",
  "properties": {
    "dataPatterns": [
      {
        "regex": "(?:3[47][0-9]{2}[0-9]{6}[0-9]{4})",
        "allowPartialMatch": false,
        "checksumType": "LUHN",
        "dataCleanRegex": "[ -.]",
        "matchStrength": 0.9
      },
      {
        "regex": "(?:4[0-9]{12}(?:[0-9]{3})?(?:[0-9]{3})?)",
        "allowPartialMatch": false,
        "checksumType": "LUHN",
        "dataCleanRegex": "[ -.]",
        "matchStrength": 0.9
      },
      {
        "regex": "(?:5[1-5][0-9]{2}|222[1-9]|22[3-9][0-9]|2[3-6][0-9]{2}|
27[01][0-9]|2720)[0-9]{12}",
        "allowPartialMatch": false,
        "checksumType": "LUHN",
        "dataCleanRegex": "[ -.]",
        "matchStrength": 0.9
      },
      {
        "regex": "(?:2131|1800|35[0-9]{3})[0-9]{11}",
        "allowPartialMatch": false,
        "checksumType": "LUHN",
        "dataCleanRegex": "[ -.]",
        "matchStrength": 0.9
      },
      {
        "regex": "3(?:0[0-5,9]|6[0-9])[0-9]{11}|3[89][0-9]{12}?(?:[0-9]
{1,3})?",
        "allowPartialMatch": false,
        "checksumType": "LUHN",
        "dataCleanRegex": "[ -.]",
        "matchStrength": 0.9
      },
      {
        "regex": "6(?:011|5[0-9][0-9])[0-9]{2}|4[4-9][0-9]{3}|2212[6-9]|
221[3-9][0-9]|22[2-8][0-9]{2}|229[0-1][0-9]|2292[0-5])[0-9]{10}?(?:[0-9]{3})?",
        "allowPartialMatch": false,
        "checksumType": "LUHN",
        "dataCleanRegex": "[ -.]",

```

```

        "matchStrength": 0.9
      }
    ],
    "rejectStrength": 0.1
  }
},
{
  "domain": "CREDIT_CARD",
  "name": "Credit Card Number - Type",
  "type": "TYPE",
  "properties": {
    "allowedTypes": [
      {
        "minimumLength": "15",
        "typeName": "Number"
      },
      {
        "minimumLength": "15",
        "typeName": "String"
      }
    ]
  }
},
{
  "domain": "CUSTOMER_NO",
  "name": "Customer Number - Path",
  "type": "PATH",
  "properties": {
    "paths": [
      {
        "matchType": "REGEX",
        "matchStrength": 0.67,
        "fieldValue": "(?i)(?>(cust(omer|mr)?) ?_?-?(num(ber)?|nbr|no))"
      }
    ],
    "rejectStrength": 0.0
  }
},
{
  "domain": "CUSTOMER_NO",
  "name": "Customer Number - Type",
  "type": "TYPE",
  "properties": {
    "allowedTypes": [
      {
        "minimumLength": "5",
        "typeName": "String"
      },
      {
        "minimumLength": "5",
        "typeName": "Number"
      }
    ]
  }
}
]

```

```

    }
  },
  {
    "domain": "DOB",
    "name": "Date of Birth - Path",
    "type": "PATH",
    "properties": {
      "paths": [
        {
          "matchType": "REGEX",
          "matchStrength": 0.67,
          "fieldValue": "(?i)(?>dob|dtofb|(day|date?|dt)_?-?(of)?_?
(birth))"
        },
        {
          "matchType": "REGEX",
          "matchStrength": 0.67,
          "fieldValue": "(?i)(?>b(irth)?_?-? ?(date|day|dt))"
        }
      ],
      "rejectStrength": 0.0
    }
  },
  {
    "domain": "DOB",
    "name": "Date of Birth - Type",
    "type": "TYPE",
    "properties": {
      "allowedTypes": [
        {
          "minimumLength": "6",
          "typeName": "String"
        },
        {
          "typeName": "Date"
        }
      ]
    }
  },
  {
    "domain": "DEPARTMENT",
    "name": "Department - List",
    "type": "LIST",
    "properties": {
      "valueLists": [
        {
          "file": "file://departments.txt",
          "matchStrength": 1.0
        }
      ],
      "tokenizeInput": true,
      "tokenizationDelimiter": " "
    }
  }
}

```

```

},
{
  "domain": "DEPARTMENT",
  "name": "Department - Path",
  "type": "PATH",
  "properties": {
    "paths": [
      {
        "matchType": "REGEX",
        "matchStrength": 0.67,
        "fieldValue": "(?>((dept|depa?rtme?nt)[-_ ]?(name|nm)))"
      }
    ],
    "rejectStrength": 0.0
  }
},
{
  "domain": "DEPARTMENT",
  "name": "Department - Type",
  "type": "TYPE",
  "properties": {
    "allowedTypes": [
      {
        "minimumLength": "5",
        "typeName": "String"
      }
    ]
  }
},
{
  "domain": "DRIVING_LC",
  "name": "Drivers License - Path",
  "type": "PATH",
  "properties": {
    "paths": [
      {
        "matchType": "REGEX",
        "matchStrength": 0.67,
        "fieldValue": "(?i)(?>(drivers?|lic(ense)?)_-? ?(number|num|nbr|
no))"
      }
    ],
    "rejectStrength": 0.0
  }
},
{
  "domain": "DRIVING_LC",
  "name": "Drivers License - Type",
  "type": "TYPE",
  "properties": {
    "allowedTypes": [
      {
        "minimumLength": "10",

```

```

        "typeName": "Number"
      },
      {
        "minimumLength": "10",
        "typeName": "String"
      }
    ]
  },
  {
    "domain": "DRIVING_LC",
    "name": "Drivers License - Regex",
    "type": "REGEX",
    "properties": {
      "dataPatterns": [
        {
          "note": "One alpha followed by digits (short): AZ 1+8, CA 1+7, HI
1+8, IN 1+9, KS 1+8, MA 1+9, MO 1+5..9, MT 1+9, NE 1+6..8, NV 1+8, OH 1+4..8, OK 1+9,
OR 1+6..7, RI 1+6, UT 4-10, VA 1+8..11, VT 1+8, WV 1+6",
          "regex": "[A-Z][0-9]{4,9}",
          "matchStrength": 0.15,
          "allowPartialMatch": false
        },
        {
          "note": "One alpha followed by digits (long): FL 1+12, MD 1+12,
MI 1+12, MN 1+12, NJ 1+14, UT 4-10, VA 1+8..11",
          "regex": "[A-Z][0-9]{10,14}",
          "matchStrength": 0.3,
          "allowPartialMatch": false
        },
        {
          "note": "One alpha followed by digits with dashes (short): KS
1+8, KY 1+8, VA 1+8..11",
          "dataCleanRegex": "[-]",
          "regex": "[A-Z][0-9]{8,9}",
          "matchStrength": 0.15,
          "allowPartialMatch": false
        },
        {
          "note": "One alpha followed by digits with dashes (long): FL
1+12, IL 1+11 or 1+12, MD 1+12, MN 1+12, VA 1+8..11, WI 1+13",
          "dataCleanRegex": "[-]",
          "regex": "[A-Z][0-9]{10,13}",
          "matchStrength": 0.3,
          "allowPartialMatch": false
        },
        {
          "note": "Two alpha followed by digits: OH 2+3..7, WV 2+7",
          "regex": "[A-Z]{2}[0-9]{3,7}",
          "matchStrength": 0.2,
          "allowPartialMatch": false
        }
      ]
    }
  }

```

```

    "note": "Digits followed by alpha: ME 7+1, MO 8+2 or 9+1, VT 7+1",
    "regex": "[0-9]{7}[A-Z]|[0-9]{8}[A-Z]{2}|[0-9]{9}[A-Z]",
    "matchStrength": 0.2,
    "allowPartialMatch": false
  },
  {
    "note": "Digits and spaces: NM 3-3-3, NY 3-3-3, PA 2-3-3",
    "regex": "[0-9]{2,3} [0-9]{3} [0-9]{3}",
    "matchStrength": 0.1,
    "allowPartialMatch": false
  },
  {
    "note": "Digit-alpha-digit: IA 3+2+4, NH 2+3+5",
    "regex": "[0-9]{3}[A-Z]{2}[0-9]{4}|[0-9]{2}[A-Z]{3}[0-9]{5}",
    "matchStrength": 0.3,
    "allowPartialMatch": false
  },
  {
    "note": "Digits only: AL 7-8, AK 7, AZ 9, AR 9, CT 9, DE 7, DC 7,
GA 9, HI 9, IA 9, LA 8, ME 7-8, MA 9, MO 9, MT 9 or 13-14, NV 9-10 or 12, NY 9, NM
8-9, NC 1-12, ND 9, OH 8, OK 9, RI 7, SC 5-11, SD 6-10 or 12, TN 7-9, TX 7-8, VA 9,
WV 7, WY 9",
    "regex": "([0-9]{6,14})",
    "matchStrength": 0.0,
    "allowPartialMatch": false
  },
  {
    "note": "CO",
    "regex": "[0-9]{2}-[0-9]{3}-[0-9]{4}",
    "matchStrength": 0.1,
    "allowPartialMatch": false
  },
  {
    "note": "ID",
    "regex": "[A-Z]{2}[0-9]{6}[A-Z]",
    "matchStrength": 0.3,
    "allowPartialMatch": false
  },
  {
    "note": "IN",
    "regex": "[0-9]{2}-[0-9]{4}-[0-9]{4}",
    "matchStrength": 0.1,
    "allowPartialMatch": false
  },
  {
    "note": "MA - SA + 7 digits",
    "regex": "SA[0-9]{7}",
    "matchStrength": 0.5,
    "allowPartialMatch": false
  },
  {
    "note": "MI - 1 alpha + 12 digits with spaces",
    "regex": "[A-Z]( [0-9]{3}){4}",

```

```

    "matchStrength": 0.5,
    "allowPartialMatch": false
  },
  {
    "note": "MS - looks just like SSN with dashes",
    "regex": "([0-9]{3}-[0-9]{2}-[0-9]{4})",
    "matchStrength": 0.0,
    "allowPartialMatch": false
  },
  {
    "note": "MO - not covered elsewhere",
    "regex": "[0-9]{3}[A-Z][0-9]{6}|[A-Z][0-9]{6}R",
    "matchStrength": 0.3,
    "allowPartialMatch": false
  },
  {
    "note": "NH - NHL + 8 digits",
    "regex": "NHL[0-9]{8}",
    "matchStrength": 0.5,
    "allowPartialMatch": false
  },
  {
    "note": "ND - 3 letters, 6 digits with spaces",
    "regex": "[A-Z]{3} ?[0-9]{2} ?[0-9]{4}",
    "matchStrength": 0.3,
    "allowPartialMatch": false
  },
  {
    "note": "NJ 1 alpha + 14 digits with spaces",
    "dataCleanRegex": "[-]",
    "regex": "[A-Z][0-9]{4} [0-9]{5} [0-9]{5}",
    "matchStrength": 0.4,
    "allowPartialMatch": false
  },
  {
    "note": "WA - old - very broad will match any 12 position alpha
string",
    "regex": "[A-Z]([A-Z]{4}|[A-Z]{3}[*]|[A-Z]{2}[*]{2}|[A-Z]{1}[*]
{3})[A-Z]{2}[0-9A-Z]{5}",
    "matchStrength": 0.0,
    "allowPartialMatch": false
  },
  {
    "note": "WA - new",
    "regex": "WDL[0-9A-Z]{9}",
    "matchStrength": 0.3,
    "allowPartialMatch": false
  },
  {
    "note": "WY - 9 digits with a dash",
    "regex": "[0-9]{6}-[0-9]{3}",
    "matchStrength": 0.1,
    "allowPartialMatch": false
  }

```



```

        }
      ],
      "rejectStrength": 0.75
    }
  },
  {
    "domain": "EMAIL",
    "name": "Email Address - Path",
    "type": "PATH",
    "properties": {
      "paths": [
        {
          "matchType": "REGEX",
          "matchStrength": 0.67,
          "fieldValue": "(?i)(cust|customer|partner|home|private|def|
default)_-? ?(email)_-? ?(address|)"
        },
        {
          "matchType": "REGEX",
          "matchStrength": 0.67,
          "fieldValue": "(?i)(?>(email_-? ?)(addr?e?s?s?))?"
        }
      ],
      "rejectStrength": 0.0
    }
  },
  {
    "domain": "EMAIL",
    "name": "Email Address - Regex",
    "type": "REGEX",
    "properties": {
      "dataPatterns": [
        {
          "regex": "[A-Z0-9.!#$%&'*/=?^_{}~-]{1,64}@(?=[1,255}$)[A-
Z0-9-]+(?:\\. [A-Z0-9-]+)*",
          "matchStrength": 0.9,
          "allowPartialMatch": false
        }
      ],
      "rejectStrength": 0.1
    }
  },
  {
    "domain": "EMAIL",
    "name": "Email Address - Type",
    "type": "TYPE",
    "properties": {
      "allowedTypes": [
        {
          "minimumLength": "20",
          "typeName": "String"
        }
      ]
    }
  }
]

```

```
    }
  },
  {
    "domain": "ETHNICITY",
    "name": "Ethnicity - List",
    "type": "LIST",
    "properties": {
      "valueLists": [
        {
          "file": "file://ethnicities.txt",
          "matchStrength": 1.0
        }
      ]
    }
  },
  {
    "domain": "ETHNICITY",
    "name": "Ethnicity - Path",
    "type": "PATH",
    "properties": {
      "paths": [
        {
          "matchType": "REGEX",
          "matchStrength": 0.67,
          "fieldValue": "(?>ethnicity|ethnicities|^race)"
        }
      ],
      "rejectStrength": 0.0
    }
  },
  {
    "domain": "ETHNICITY",
    "name": "Ethnicity - Type",
    "type": "TYPE",
    "properties": {
      "allowedTypes": [
        {
          "minimumLength": "7",
          "typeName": "String"
        }
      ]
    }
  },
  {
    "domain": "FIRST_NAME",
    "name": "First Name - List",
    "type": "LIST",
    "properties": {
      "valueLists": [
        {
          "file": "file://us_first.txt",
          "matchStrength": 1.0
        }
      ],
    },
  },
```

```

        {
            "file": "file://de_first.txt",
            "matchStrength": 1.0
        },
        {
            "file": "file://ch_first.txt",
            "matchStrength": 1.0
        }
    ]
}
},
{
    "domain": "FIRST_NAME",
    "name": "First Name - Path",
    "type": "PATH",
    "properties": {
        "paths": [
            {
                "matchType": "REGEX",
                "matchStrength": 0.67,
                "fieldValue": "(?i)(?>(mid(dle)?_?-? ?(na?me?))(_?-?user)?)"
            },
            {
                "matchType": "REGEX",
                "matchStrength": 0.67,
                "fieldValue": "(?i)(?>(f(first)?_?-? ?(na?me?))(_?-?user)?)"
            }
        ],
        "rejectStrength": 0.0
    }
},
{
    "domain": "FIRST_NAME",
    "name": "First Name - Type",
    "type": "TYPE",
    "properties": {
        "allowedTypes": [
            {
                "minimumLength": "10",
                "typeName": "String"
            }
        ]
    }
},
{
    "domain": "TEXT",
    "name": "Text - Path",
    "type": "PATH",
    "properties": {
        "paths": [
            {
                "matchType": "REGEX",
                "matchStrength": 0.67,

```

```

        "fieldValue": "(?>(free[-_ ]?text|comment|description|note|
remark|text|user[-_ ]?input|user[-_ ]?text))"
    }
    ],
    "rejectStrength": 0.0
}
},
{
    "domain": "TEXT",
    "name": "Text - Type",
    "type": "TYPE",
    "properties": {
        "allowedTypes": [
            {
                "minimumLength": "20",
                "typeName": "String"
            }
        ]
    }
},
{
    "domain": "FULL_NAME",
    "name": "Full Name - Path",
    "type": "PATH",
    "properties": {
        "paths": [
            {
                "matchType": "REGEX",
                "matchStrength": 0.67,
                "fieldValue": "(?i)(?>((fu?l?l|whole|user)([-_ ]*)?(na?me?)))"
            }
        ],
        "rejectStrength": 0.0
    }
},
{
    "domain": "FULL_NAME",
    "name": "Full Name - Type",
    "type": "TYPE",
    "properties": {
        "allowedTypes": [
            {
                "minimumLength": "20",
                "typeName": "String"
            }
        ]
    }
},
{
    "domain": "FULL_NAME",
    "name": "Full Name - List",
    "type": "LIST",
    "properties": {

```

```

    "valueLists": [
      {
        "file": "file://us_first.txt",
        "matchStrength": 0.7
      },
      {
        "file": "file://de_first.txt",
        "matchStrength": 0.7
      },
      {
        "file": "file://ch_first.txt",
        "matchStrength": 0.7
      },
      {
        "file": "file://us_last.txt",
        "matchStrength": 0.7
      },
      {
        "file": "file://de_last.txt",
        "matchStrength": 0.7
      },
      {
        "file": "file://ch_last.txt",
        "matchStrength": 0.7
      }
    ],
    "tokenizeInput": true,
    "tokenizationDelimiter": " "
  },
  {
    "domain": "GENDER",
    "name": "Gender - Path",
    "type": "PATH",
    "properties": {
      "paths": [
        {
          "matchType": "REGEX",
          "matchStrength": 0.67,
          "fieldValue": "(?i)(?>(ge?nde?r|sex)[-_ ]?(type|identity)?)(${[-_ ]})"
        }
      ]
    },
    "rejectStrength": 0.0
  },
  {
    "domain": "GENDER",
    "name": "Gender - Type",
    "type": "TYPE",
    "properties": {
      "allowedTypes": [
        {

```

```

        "minimumLength": "5",
        "typeName": "String"
      }
    ]
  },
  {
    "domain": "GENDER",
    "name": "Gender - List",
    "type": "LIST",
    "properties": {
      "valueLists": [
        {
          "file": "file://genders.txt",
          "matchStrength": 0.9
        }
      ],
      "rejectStrength": 0.1
    }
  },
  {
    "domain": "HIPAA_DATE",
    "name": "HIPAA Date - Path",
    "type": "PATH",
    "properties": {
      "paths": [
        {
          "matchType": "REGEX",
          "matchStrength": 0.67,
          "fieldValue": "(?i)(?>(adm(it|i)ssion)?|tr(ea)?t(ment)?_?-?|ds|disc(h|harge))-?_? ?(date|day|dt))"
        }
      ],
      "rejectStrength": 0.0
    }
  },
  {
    "domain": "HIPAA_DATE",
    "name": "HIPAA Date - Type",
    "type": "TYPE",
    "properties": {
      "allowedTypes": [
        {
          "minimumLength": "6",
          "typeName": "String"
        },
        {
          "typeName": "Date"
        }
      ]
    }
  },
  {

```

```

"domain": "HOUSE_NO",
"name": "House Number - Path",
"type": "PATH",
"properties": {
  "paths": [
    {
      "matchType": "REGEX",
      "matchStrength": 0.67,
      "fieldValue": "(?>(house?|plot)[_ -]??(number|num|nbr|no|id))"
    }
  ],
  "rejectStrength": 0.0
}
},
{
  "domain": "HOUSE_NO",
  "name": "House Number - Type",
  "type": "TYPE",
  "properties": {
    "allowedTypes": [
      {
        "minimumLength": "1",
        "typeName": "String"
      },
      {
        "minimumLength": "1",
        "typeName": "Number"
      }
    ]
  }
},
{
  "domain": "IBAN",
  "name": "IBAN - Path",
  "type": "PATH",
  "properties": {
    "paths": [
      {
        "matchType": "REGEX",
        "matchStrength": 0.67,
        "fieldValue": "(?>(^[_ -])iban|(international|int|intl)[_ -]?
bank[_ -]?acc(ount)?[_ -]??(number|num|nbr|no))"
      }
    ],
    "rejectStrength": 0.0
  }
},
{
  "domain": "IBAN",
  "name": "IBAN - Regex",
  "type": "REGEX",
  "properties": {
    "dataPatterns": [

```

```

    {
      "note": "Fetched from: 'https://www.swift.com/standards/data-standards/iban-international-bank-account-number', List of countries in order for regex formats listed below: Andorra, United Arab Emirates (The), Albania, Austria, Azerbaijan, Bosnia and Herzegovina, Belgium, Bulgaria, Bahrain, Burundi, Brazil, Republic of Belarus, Switzerland, Costa Rica, Cyprus, Czechia, Germany, Djibouti, Denmark, Dominican Republic, Estonia, Egypt, Spain, Finland, Faroe Islands, France, United Kingdom, Georgia, Gibraltar, Greenland, Greece, Guatemala, Croatia, Hungary, Ireland, Israel, Iraq, Iceland, Italy, Jordan, Kuwait, Kazakhstan, Lebanon, Saint Lucia, Liechtenstein, Lithuania, Luxembourg, Latvia, Libya, Monaco, Moldova, Montenegro, Macedonia, Mauritania, Malta, Mauritius, Netherlands, Norway, Pakistan, Poland, Palestine, State of, Portugal, Qatar, Romania, Serbia, Russia, Saudi Arabia, Seychelles, Sudan, Sweden, Slovenia, Slovakia, San Marino, Somalia, Sao Tome and Principe, El Salvador, Timor-Leste, Tunisia, Turkey, Ukraine, Vatican City State, Virgin Islands, Kosovo",
      "regex": "(AD[0-9]{10}[A-Z0-9]{12})|(AE[0-9]{21})|(AL[0-9]{10}[A-Z0-9]{16})|(AT[0-9]{18})|(AZ[0-9]{2}[A-Z]{4}[A-Z0-9]{20})|(BA[0-9]{18})|(BE[0-9]{14})|(BG[0-9]{2}[A-Z]{4}[0-9]{6}[A-Z0-9]{8})|(BH[0-9]{2}[A-Z]{4}[A-Z0-9]{14})|(BI[0-9]{25})|(BR[0-9]{25}[A-Z]{1}[A-Z0-9]{1})|(BY[0-9]{2}[A-Z0-9]{4}[0-9]{4}[A-Z0-9]{16})|(CH[0-9]{7}[A-Z0-9]{12})|(CR[0-9]{20})|(CY[0-9]{10}[A-Z0-9]{16})|(CZ[0-9]{22})|(DE[0-9]{20})|(DJ[0-9]{25})|(DK[0-9]{16})|(DO[0-9]{2}[A-Z0-9]{4}[0-9]{20})|(EE[0-9]{18})|(EG[0-9]{27})|(ES[0-9]{22})|(FI[0-9]{16})|(FO[0-9]{16})|(FR[0-9]{12}[A-Z0-9]{11}[0-9]{2})|(GB[0-9]{2}[A-Z]{4}[0-9]{14})|(GE[0-9]{2}[A-Z]{2}[0-9]{16})|(GI[0-9]{2}[A-Z]{4}[A-Z0-9]{15})|(GL[0-9]{16})|(GR[0-9]{9}[A-Z0-9]{16})|(GT[0-9]{2}[A-Z0-9]{24})|(HR[0-9]{19})|(HU[0-9]{26})|(IE[0-9]{2}[A-Z]{4}[0-9]{14})|(IL[0-9]{21})|(IQ[0-9]{2}[A-Z]{4}[0-9]{15})|(IS[0-9]{24})|(IT[0-9]{2}[A-Z]{1}[0-9]{10}[A-Z0-9]{12})|(JO[0-9]{2}[A-Z]{4}[0-9]{4}[A-Z0-9]{18})|(KW[0-9]{2}[A-Z]{4}[A-Z0-9]{22})|(KZ[0-9]{5}[A-Z0-9]{13})|(LB[0-9]{6}[A-Z0-9]{20})|(LC[0-9]{2}[A-Z]{4}[A-Z0-9]{24})|(LI[0-9]{7}[A-Z0-9]{12})|(LT[0-9]{18})|(LU[0-9]{5}[A-Z0-9]{13})|(LV[0-9]{2}[A-Z]{4}[A-Z0-9]{13})|(LY[0-9]{23})|(MC[0-9]{12}[A-Z0-9]{11}[0-9]{2})|(MD[0-9]{2}[A-Z0-9]{20})|(ME[0-9]{20})|(MK[0-9]{5}[A-Z0-9]{10}[0-9]{2})|(MR[0-9]{25})|(MT[0-9]{2}[A-Z]{4}[0-9]{5}[A-Z0-9]{18})|(MU[0-9]{2}[A-Z]{4}[0-9]{19}[A-Z]{3})|(NL[0-9]{2}[A-Z]{4}[0-9]{10})|(NO[0-9]{13})|(PK[0-9]{2}[A-Z]{4}[A-Z0-9]{16})|(PL[0-9]{26})|(PS[0-9]{2}[A-Z]{4}[A-Z0-9]{21})|(PT[0-9]{23})|(QA[0-9]{2}[A-Z]{4}[A-Z0-9]{21})|(RO[0-9]{2}[A-Z]{4}[A-Z0-9]{16})|(RS[0-9]{20})|(RU[0-9]{31})|(SA[0-9]{4}[A-Z0-9]{18})|(SC[0-9]{2}[A-Z]{4}[0-9]{20}[A-Z]{3})|(SD[0-9]{16})|(SE[0-9]{22})|(SI[0-9]{17})|(SK[0-9]{22})|(SM[0-9]{2}[A-Z]{1}[0-9]{10}[A-Z0-9]{12})|(SO[0-9]{21})|(ST[0-9]{23})|(SV[0-9]{2}[A-Z]{4}[0-9]{20})|(TL[0-9]{21})|(TN[0-9]{22})|(TR[0-9]{8}[A-Z0-9]{16})|(UA[0-9]{8}[A-Z0-9]{19})|(VA[0-9]{20})|(VG[0-9]{2}[A-Z]{4}[0-9]{16})|(XK[0-9]{18})",
      "allowPartialMatch": false,
      "checksumType": "MOD97",
      "matchStrength": 0.9
    },
    ],
    "rejectStrength": 0.1
  }
},
{
  "domain": "IBAN",
  "name": "IBAN - Type",
  "type": "TYPE",
  "properties": {
    "allowedTypes": [

```



```

        {
            "minimumLength": "15",
            "typeName": "String"
        }
    ]
}
},
{
    "domain": "IP ADDRESS",
    "name": "IP Address - Path",
    "type": "PATH",
    "properties": {
        "paths": [
            {
                "matchType": "REGEX",
                "matchStrength": 0.67,
                "fieldValue": "(?i)(?>(ip_?-? ?address?s?s?))"
            }
        ],
        "rejectStrength": 0.0
    }
},
{
    "domain": "IP ADDRESS",
    "name": "IP Address - Type",
    "type": "TYPE",
    "properties": {
        "allowedTypes": [
            {
                "minimumLength": "10",
                "typeName": "String"
            }
        ]
    }
},
{
    "domain": "IP ADDRESS",
    "name": "IP Address - Regex",
    "type": "REGEX",
    "properties": {
        "dataPatterns": [
            {
                "regex": "(?>((25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?))\\.){3}(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)",
                "note": "IPv4",
                "allowPartialMatch": false,
                "matchStrength": 0.9
            },
            {
                "regex": "(?>([A-F0-9]{0,4}:){2,7}[A-F0-9]{0,4})",
                "note": "IPv6 standard addresses",
                "allowPartialMatch": false,
                "matchStrength": 0.9
            }
        ]
    }
}

```

```

    },
    {
      "regex": "(?>(([A-F0-9]{0,4}:){2,6})(25[0-5]|2[0-4][0-9]|[01]?
[0-9][0-9]?)\.\.){3}(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?))",
      "note": "IPv6 dual addresses",
      "allowPartialMatch": false,
      "matchStrength": 0.9
    }
  ],
  "rejectStrength": 0.1
}
},
{
  "domain": "JOB_TITLE",
  "name": "Job Title - List",
  "type": "LIST",
  "properties": {
    "valueLists": [
      {
        "file": "file://job_titles.txt",
        "matchStrength": 1.0
      }
    ]
  }
},
{
  "domain": "JOB_TITLE",
  "name": "Job Title - Path",
  "type": "PATH",
  "properties": {
    "paths": [
      {
        "matchType": "REGEX",
        "matchStrength": 0.67,
        "fieldValue": "(?>(job)[-_ ]?(title[-_ ])?(name|nm)|job[-_ ]?
title|title[-_ ]?(name|nm))"
      }
    ]
  },
  "rejectStrength": 0.0
}
},
{
  "domain": "JOB_TITLE",
  "name": "Job Title - Type",
  "type": "TYPE",
  "properties": {
    "allowedTypes": [
      {
        "minimumLength": "7",
        "typeName": "String"
      }
    ]
  }
}
}

```

```
},
{
  "domain": "LANGUAGE",
  "name": "Language - List",
  "type": "LIST",
  "properties": {
    "valueLists": [
      {
        "file": "file://languages.txt",
        "matchStrength": 1.0
      }
    ]
  }
},
{
  "domain": "LANGUAGE",
  "name": "Language - Path",
  "type": "PATH",
  "properties": {
    "paths": [
      {
        "matchType": "REGEX",
        "matchStrength": 0.67,
        "fieldValue": "(?>la?nguage)"
      }
    ],
    "rejectStrength": 0.0
  }
},
{
  "domain": "LANGUAGE",
  "name": "Language - Type",
  "type": "TYPE",
  "properties": {
    "allowedTypes": [
      {
        "minimumLength": "7",
        "typeName": "String"
      }
    ]
  }
},
{
  "domain": "LAST_NAME",
  "name": "Last Name - List",
  "type": "LIST",
  "properties": {
    "valueLists": [
      {
        "file": "file://us_last.txt",
        "matchStrength": 1.0
      },
      {
```

```

        "file": "file://de_last.txt",
        "matchStrength": 1.0
    },
    {
        "file": "file://ch_last.txt",
        "matchStrength": 1.0
    }
]
}
},
{
    "domain": "LAST_NAME",
    "name": "Last Name - Path",
    "type": "PATH",
    "properties": {
        "paths": [
            {
                "matchType": "REGEX",
                "matchStrength": 0.67,
                "fieldValue": "(?i)(?>((l(as)?t)_?-? ?(na?me?))(_?-?user)?)"
            },
            {
                "matchType": "REGEX",
                "matchStrength": 0.67,
                "fieldValue": "(?i)(?>(sur) _?-? ?(name)?_?-? ?(no|id|str|
value|))"
            }
        ],
        "rejectStrength": 0.0
    }
},
{
    "domain": "LAST_NAME",
    "name": "Last Name - Type",
    "type": "TYPE",
    "properties": {
        "allowedTypes": [
            {
                "minimumLength": "10",
                "typeName": "String"
            }
        ]
    }
},
{
    "domain": "PLATE_N0",
    "name": "License Plate - Path",
    "type": "PATH",
    "properties": {
        "paths": [
            {
                "matchType": "REGEX",
                "matchStrength": 0.67,

```

```

        "fieldValue": "(?i)(?>(license|li?c)?[-_ ]?plate[-_ ]?(number|
num|nbr|no?)"
    }
  ],
  "rejectStrength": 0.0
}
},
{
  "domain": "PLATE_NO",
  "name": "License Plate - Type",
  "type": "TYPE",
  "properties": {
    "allowedTypes": [
      {
        "minimumLength": "1",
        "typeName": "String"
      }
    ]
  }
},
{
  "domain": "LOCATION_COORDINATES",
  "name": "Location Coordinates - Path",
  "type": "PATH",
  "properties": {
    "paths": [
      {
        "matchType": "REGEX",
        "matchStrength": 0.67,
        "fieldValue": "(?i)(?>((map|geographica?l?|areal|location|
lat(itude)?|lo?ng(itude)?)[-_ ]?(co-?o?rd(inates?))|(latitude|longitude|co-?o?
rdinates?)([-_ ]value?))"
      }
    ],
    "rejectStrength": 0.0
  }
},
{
  "domain": "LOCATION_COORDINATES",
  "name": "Location Coordinates - Regex",
  "type": "REGEX",
  "properties": {
    "dataPatterns": [
      {
        "note": "Latitude and Longitude in decimal degrees format",
        "regex": "(?!^[+-]?\\d+[. ]\\d+([. ]\\d+)?$)(?:[+-]?([1-8]?[0-9])
[. °](\\d{4,8})°?[NS]? ,? ?)?[+-]?([1-9]?[0-9]|1[0-7][0-9])[. °](\\d{4,8})°?[NEWS]?",
        "allowPartialMatch": false,
        "matchStrength": 0.9
      },
      {
        "note": "Latitude or Longitude in decimal degrees format (float
value format with low confidence)",

```

```

        "regex": "(?:[+-]?([1-9]?[0-9]|1[0-7][0-9])[. ]\\d{4,8})",
        "allowPartialMatch": false,
        "matchStrength": 0.05
    },
    {
        "note": "Latitude & longitude in degrees & minutes format",
        "regex": "(?:[+-]?([1-8]?[0-9])[. °]([0-5]?[0-9])\\.\\.\\.\\d{3,4})
(?:'|')?[NS]?,(? ?)?[+-]?([1-9]?[0-9]|1[0-7][0-9])[. °]([0-5]?[0-9])\\.\\.\\.\\d{3,4})(?:'|
')?[NEWS]?",
        "allowPartialMatch": false,
        "matchStrength": 0.9
    },
    {
        "note": "Latitude and Longitude in degrees, minutes & seconds
format",
        "regex": "(?:[+-]?([1-8]?[0-9])[. °]([0-5]?[0-9])(?:'|')([0-5]?
[0-9])\\.\\.\\.\\d{1,2})?(?:\\\"|\\')?[NS]?,(? ?)?[+-]?([1-9]?[0-9]|1[0-7][0-9])[. °]([0-5]?
[0-9])(?:'|')([0-5]?[0-9])\\.\\.\\.\\d{1,2})?(?:\\\"|\\')?[NEWS]?",
        "allowPartialMatch": false,
        "matchStrength": 0.9
    }
    ],
    "rejectStrength": 0.1
}
},
{
    "domain": "LOCATION_COORDINATES",
    "name": "Location Coordinates - Type",
    "type": "TYPE",
    "properties": {
        "allowedTypes": [
            {
                "minimumLength": "8",
                "typeName": "String"
            },
            {
                "minimumLength": "8",
                "typeName": "Number"
            }
        ]
    }
}
},
{
    "domain": "MARITAL_STATUS",
    "name": "Marital Status - List",
    "type": "LIST",
    "properties": {
        "valueLists": [
            {
                "file": "file://marital_status.txt",
                "matchStrength": 1.0
            }
        ]
    }
}
]

```

```

    }
  },
  {
    "domain": "MARITAL_STATUS",
    "name": "Marital Status - Path",
    "type": "PATH",
    "properties": {
      "paths": [
        {
          "matchType": "REGEX",
          "matchStrength": 0.67,
          "fieldValue": "(?>(marital|marriage|married)[-_ ]?sta?tu?s)"
        }
      ],
      "rejectStrength": 0.0
    }
  },
  {
    "domain": "MARITAL_STATUS",
    "name": "Marital Status - Type",
    "type": "TYPE",
    "properties": {
      "allowedTypes": [
        {
          "minimumLength": "6",
          "typeName": "String"
        }
      ]
    }
  },
  {
    "domain": "MEDICAL_CODES",
    "name": "Medical Codes - Path",
    "type": "PATH",
    "properties": {
      "paths": [
        {
          "matchType": "REGEX",
          "matchStrength": 0.67,
          "fieldValue": "(?>(med(ica?l)?|(med(ica?l)?|hosp(ita?l)?)[-_ ]?(condition|cond|procedure|proc|bill(ing)?|diag(nosis)?|diagnostic|pcs|cpt)))[-_ ]?(code|cd))"
        },
        {
          "matchType": "REGEX",
          "matchStrength": 0.67,
          "fieldValue": "(?>(^[^_ ])(icd[-_ ]?(9|10|11)?[-_ ]?(cm|pcs)?|cpt)[-_ ]?(code|cd)?)"
        }
      ],
      "rejectStrength": 0.0
    }
  },

```

```

{
  "domain": "MEDICAL_CODES",
  "name": "Medical Codes - Regex",
  "type": "REGEX",
  "properties": {
    "dataPatterns": [
      {
        "note": "ICD-9 codes: Legacy medical disease and diagnostic codes
used world-wide",
        "regex": "\\d{3}\\.\\d{0,2}|E\\d{3}\\.?.\\d?|V\\d{2}\\.?.\\d{0,2}",
        "allowPartialMatch": false,
        "matchStrength": 0.05
      },
      {
        "note": "ICD-10-CM codes: Newer medical disease and diagnostic
codes used world-wide",
        "regex": "[A-Z][0-9][0-9AB]\\.[0-9A-Z]{0,4}",
        "allowPartialMatch": false,
        "matchStrength": 0.15
      },
      {
        "note": "ICD-10-PCS (Procedure Coding System) codes: Medical
inpatient procedure codes used world-wide.",
        "regex": "[0-9A-HJ-NP-Z]{7}",
        "allowPartialMatch": false,
        "matchStrength": 0.05
      },
      {
        "note": "CPT (Current Procedural Terminology) codes (end with
alpha regex): Medical procedure codes used mainly in USA and a few handful of
countries.",
        "regex": "[0-9]{4}[A-Z]",
        "allowPartialMatch": false,
        "matchStrength": 0.2
      },
      {
        "note": "CPT (Current Procedural Terminology) codes (no alpha
regex): Medical procedure codes used mainly in USA and a few handful of countries.",
        "regex": "[0-9]{4}[0-9]",
        "allowPartialMatch": false,
        "matchStrength": 0.05
      }
    ],
    "rejectStrength": 0.1
  }
},
{
  "domain": "MEDICAL_CODES",
  "name": "Medical Codes - Type",
  "type": "TYPE",
  "properties": {
    "allowedTypes": [
      {

```



```

        "minimumLength": "3",
        "typeName": "String"
    },
    {
        "minimumLength": "3",
        "typeName": "Number"
    }
]
}
},
{
    "domain": "MEDICAL_DRUG",
    "name": "Medical Drug - List",
    "type": "LIST",
    "properties": {
        "valueLists": [
            {
                "file": "file://medical_drugs_generic.txt",
                "matchStrength": 1.0
            },
            {
                "file": "file://medical_drugs_brand.txt",
                "matchStrength": 1.0
            }
        ],
        "tokenizeInput": true,
        "tokenizationDelimiter": " "
    }
},
{
    "domain": "MEDICAL_DRUG",
    "name": "Medical Drug - Path",
    "type": "PATH",
    "properties": {
        "paths": [
            {
                "matchType": "REGEX",
                "matchStrength": 0.67,
                "fieldValue": "(?>(med|medical|prescription)?[-_ ]?drug([-_ ]?(prescription)?)(name|nm)?|(medicine?|medication)[-_ ]?(name|nm))"
            }
        ],
        "rejectStrength": 0.0
    }
},
{
    "domain": "MEDICAL_DRUG",
    "name": "Medical Drug - Type",
    "type": "TYPE",
    "properties": {
        "allowedTypes": [
            {
                "minimumLength": "6",

```

```

        "typeName": "String"
      }
    ]
  }
},
{
  "domain": "NATIONAL_ORIGIN",
  "name": "National Origin - List",
  "type": "LIST",
  "properties": {
    "valueLists": [
      {
        "file": "file://national_origins.txt",
        "matchStrength": 1.0
      }
    ]
  }
},
{
  "domain": "NATIONAL_ORIGIN",
  "name": "National Origin - Path",
  "type": "PATH",
  "properties": {
    "paths": [
      {
        "matchType": "REGEX",
        "matchStrength": 0.67,
        "fieldValue": "(?>nationa?li?ty|nat(ion)?a?l[-_ ]?(ori?gi?ns?))"
      }
    ],
    "rejectStrength": 0.0
  }
},
{
  "domain": "NATIONAL_ORIGIN",
  "name": "National Origin - Type",
  "type": "TYPE",
  "properties": {
    "allowedTypes": [
      {
        "minimumLength": "2",
        "typeName": "String"
      }
    ]
  }
},
{
  "domain": "PO_BOX",
  "name": "PO Box - Path",
  "type": "PATH",
  "properties": {
    "paths": [
      {

```

```

                "matchType": "REGEX",
                "matchStrength": 0.67,
                "fieldValue": "(?i)(?>(p.?o.?_?-? ?box|post_?-? ?office_?-? ?
box ?_?-?) (number|num|nbr|no)?)"
            }
        ],
        "rejectStrength": 0.0
    }
},
{
    "domain": "PO_BOX",
    "name": "PO Box - Type",
    "type": "TYPE",
    "properties": {
        "allowedTypes": [
            {
                "minimumLength": "4",
                "typeName": "String"
            },
            {
                "minimumLength": "4",
                "typeName": "Number"
            }
        ]
    }
},
{
    "domain": "PASSWORD",
    "name": "Password - Path",
    "type": "PATH",
    "properties": {
        "paths": [
            {
                "matchType": "REGEX",
                "matchStrength": 0.67,
                "fieldValue": "(?i)(?>(pass(?!po?rt)) ?_?-??(word)?_?-? ?(word|
nbr|no|id|value|))"
            }
        ],
        "rejectStrength": 0.0
    }
},
{
    "domain": "PASSWORD",
    "name": "Password - Type",
    "type": "TYPE",
    "properties": {
        "allowedTypes": [
            {
                "minimumLength": "6",
                "typeName": "String"
            }
        ]
    }
}
]

```



```

BIF|CVE|KMF|CDF|DJF|EGP|ERN|ETB|SZL|GMD|GHS|GNF|KES|LSL|LRD|LYD|MGA|MWK|MUR|MRU|MAD|
MZN|NAD|NGN|RWF|STN|SCR|SLL|SOS|ZAR|SSP|SDG|TZS|TND|UGX|USD|AUD|BDT|BTN|BND|KHR|CNY|
HKD|IDR|INR|JPY|KZT|KGS|LAK|MOP|MYR|MVR|MNT|MMK|NPR|NZD|KPW|PKR|PHP|SGD|KRW|LKR|TWD|
TJS|USD|THB|TMT|UZS|VND|BTC|XBT|ETH|LTC|XMR|XRP|USDT)|(|€|L|Br|KM|лB|kn|Kč|kn|£|ft|kr|
Íkr|CHF|L|дeH|zł|lei|₪|RSD|CHF|฿|₪|£|\\$|f|B\\$|BZ\\$|Bs|R\\$|CA\\$|CI\\$|CUC\\$|f|
RD\\$|FK£|Q|G\\$|G|L|J\\$|C\\$|B/\\.|¢|S/\\.|Sr\\$|TT\\$|\\$U|Bs\\.|đ|₴|₹|₺|\\.\.د
ب\\|€|لارڻي|ل\\ل.ينار|ك.ل|ب|د\\د.ع|ل|£S|AED|ل|ق\\ل.ر|ع\\ل.ر|SR|ب|FCFA|CFA|ج|Kz|P|FBu|
CVE|CF|FC|Fdj|E£|Nkf|Br|L|D|GH¢|FG|KSh|L|LD\\$|LD|Ar|K|Rs|UM|DH|MT|N\\$|฿|RF|Db|SR|Le|
Sh\\.So\\.|R|SS£|SDG|TSh|ت\\ل.د|Ush|\\$|A\\$|฿|Nu|B\\$|₪|¥|元|\\$|HK\\$|Rp|₹|₪|som|₣|
MOP\\$|RM|MRf|₹|K|Rs|\\$|₩|Rs|₪|S\\$|₩|Rs|NT\\$|TJS|US\\$|฿|m|som|đ|฿|₺|₪|XRP|₹)",
    "allowPartialMatch": false,
    "matchStrength": 0.9
  },
  {
    "note": "Currency with Numeric digits and without prefix or
suffix currency symbols, but ending in decimals",
    "regex": "[+-]?(\\d{1,3}(,\\d{3})*|(\\d+))(\\.\\d{2})",
    "allowPartialMatch": false,
    "matchStrength": 0.3
  }
],
"rejectStrength": 0.1
}
},
{
  "domain": "PAYMENT_AMOUNT",
  "name": "Payment Amount - Type",
  "type": "TYPE",
  "properties": {
    "allowedTypes": [
      {
        "minimumLength": "4",
        "typeName": "String"
      },
      {
        "minimumLength": "4",
        "typeName": "Number"
      }
    ]
  }
}
},
{
  "domain": "POLITICAL_ORIEN",
  "name": "Political Orientation - List",
  "type": "LIST",
  "properties": {
    "valueLists": [
      {
        "file": "file://political_orientations.txt",
        "matchStrength": 1.0
      }
    ]
  }
},
"rejectStrength": 0.3

```

```

    }
  },
  {
    "domain": "POLITICAL_ORIEN",
    "name": "Political Orientation - Path",
    "type": "PATH",
    "properties": {
      "paths": [
        {
          "matchType": "REGEX",
          "matchStrength": 0.67,
          "fieldValue": "(?>politi?ca?l[-_ ]?(orie?nt?(ation)?s?|pref(er)?
(ence)?s?|ideolo?gy))"
        }
      ],
      "rejectStrength": 0.0
    }
  },
  {
    "domain": "POLITICAL_ORIEN",
    "name": "Political Orientation - Type",
    "type": "TYPE",
    "properties": {
      "allowedTypes": [
        {
          "minimumLength": "3",
          "typeName": "String"
        }
      ]
    }
  },
  {
    "domain": "ZIP",
    "name": "Postcode - Path",
    "type": "PATH",
    "properties": {
      "paths": [
        {
          "matchType": "REGEX",
          "matchStrength": 0.67,
          "fieldValue": "(?i)(?>(zip|post|postal)_?-? ?(co?de?)|(zip))"
        }
      ],
      "rejectStrength": 0.0
    }
  },
  {
    "domain": "ZIP",
    "name": "Postcode - Regex",
    "type": "REGEX",
    "properties": {
      "dataPatterns": [
        {

```

```

        "regex": "[0-9]{5}(?:-[0-9]{4})$",
        "matchStrength": 0.7
    },
    {
        "regex": "[0-9]{5}$",
        "matchStrength": 0.2
    }
],
"rejectStrength": 0.1
}
},
{
    "domain": "ZIP",
    "name": "Postcode - Type",
    "type": "TYPE",
    "properties": {
        "allowedTypes": [
            {
                "minimumLength": "4",
                "typeName": "Number"
            },
            {
                "minimumLength": "4",
                "typeName": "String"
            }
        ]
    }
},
{
    "domain": "PRECINCT",
    "name": "Precinct - Path",
    "type": "PATH",
    "properties": {
        "paths": [
            {
                "matchType": "REGEX",
                "matchStrength": 0.67,
                "fieldValue": "(?i)(?>precinct|prcnct)"
            }
        ],
        "rejectStrength": 0.0
    }
},
{
    "domain": "PRECINCT",
    "name": "Precinct - Type",
    "type": "TYPE",
    "properties": {
        "allowedTypes": [
            {
                "minimumLength": "0",
                "typeName": "String"
            }
        ],
    },

```

```

        {
            "minimumLength": "0",
            "typeName": "Number"
        }
    ]
}
},
{
    "domain": "RECORD_NO",
    "name": "Record Number - Path",
    "type": "PATH",
    "properties": {
        "paths": [
            {
                "matchType": "REGEX",
                "matchStrength": 0.67,
                "fieldValue": "(?>(med(ical)?|rec(ord)?|(med(ical)?)[-_ ]?
rec(ord)?)[-_ ]?(num(ber)?|nbr|no|id|key))"
            },
            {
                "matchType": "REGEX",
                "matchStrength": 0.67,
                "fieldValue": "(?>(med(ical)?)[-_ ]?)?health([-_ ]?ca?re?)?([-_ ]?
med(ical)?)?([-_ ]?rec(ord)?)[-_ ]?(num(ber)?|nbr|no|id|key))"
            }
        ],
        "rejectStrength": 0.0
    }
},
{
    "domain": "RECORD_NO",
    "name": "Record Number - Type",
    "type": "TYPE",
    "properties": {
        "allowedTypes": [
            {
                "minimumLength": "5",
                "typeName": "String"
            },
            {
                "minimumLength": "5",
                "typeName": "Number"
            }
        ]
    }
},
{
    "domain": "RELIGIOUS_ORIEN",
    "name": "Religious Orientation - List",
    "type": "LIST",
    "properties": {
        "valueLists": [
            {

```



```

        "file": "file://religious_orientations.txt",
        "matchStrength": 1.0
    }
],
    "rejectStrength": 0.3
}
},
{
    "domain": "RELIGIOUS_ORIEN",
    "name": "Religious Orientation - Path",
    "type": "PATH",
    "properties": {
        "paths": [
            {
                "matchType": "REGEX",
                "matchStrength": 0.67,
                "fieldValue": "(?>religion|religio?us[-_ ]?(orie?nt?(ation)?s?|
pref(er)?(ence)?s?|practice|faith|type|identity))"
            }
        ],
        "rejectStrength": 0.0
    }
},
{
    "domain": "RELIGIOUS_ORIEN",
    "name": "Religious Orientation - Type",
    "type": "TYPE",
    "properties": {
        "allowedTypes": [
            {
                "minimumLength": "3",
                "typeName": "String"
            }
        ]
    }
},
{
    "domain": "ROUTING_NO",
    "name": "Routing Number - Path",
    "type": "PATH",
    "properties": {
        "paths": [
            {
                "matchType": "REGEX",
                "matchStrength": 0.67,
                "fieldValue": "(?>(^[^-_ ])(routing|aba|ach|rtn|routing[-_ ]?
transit)[-_ ]?(number|num|nbr|no))"
            }
        ],
        "rejectStrength": 0.0
    }
},
{

```

```

"domain": "ROUTING_NO",
"name": "Routing Number - Regex",
"type": "REGEX",
"properties": {
  "dataPatterns": [
    {
      "regex": "\\d{9}",
      "allowPartialMatch": false,
      "checksumType": "MOD10_ABA",
      "matchStrength": 0.9
    }
  ],
  "rejectStrength": 0.5
}
},
{
  "domain": "ROUTING_NO",
  "name": "Routing Number - Type",
  "type": "TYPE",
  "properties": {
    "allowedTypes": [
      {
        "minimumLength": "9",
        "typeName": "String"
      },
      {
        "minimumLength": "9",
        "typeName": "Number"
      }
    ]
  }
},
{
  "domain": "SCHOOL_NM",
  "name": "School Name - Path",
  "type": "PATH",
  "properties": {
    "paths": [
      {
        "matchType": "REGEX",
        "matchStrength": 0.67,
        "fieldValue": "(?i)(?>school_?-?na?me?)"
      }
    ],
    "rejectStrength": 0.0
  }
},
{
  "domain": "SCHOOL_NM",
  "name": "School Name - Type",
  "type": "TYPE",
  "properties": {

```

```

        "allowedTypes": [
            {
                "minimumLength": "20",
                "typeName": "String"
            }
        ]
    },
    {
        "domain": "SECURITY_CODE",
        "name": "Security Code - Path",
        "type": "PATH",
        "properties": {
            "paths": [
                {
                    "matchType": "REGEX",
                    "matchStrength": 0.67,
                    "fieldValue": "(?i)(?>se?cu?r(i?ty)??_?co?de?)"
                }
            ],
            "rejectStrength": 0.0
        }
    },
    {
        "domain": "SECURITY_CODE",
        "name": "Security Code - Type",
        "type": "TYPE",
        "properties": {
            "allowedTypes": [
                {
                    "minimumLength": "5",
                    "typeName": "String"
                },
                {
                    "minimumLength": "5",
                    "typeName": "Number"
                },
                {
                    "minimumLength": "0",
                    "typeName": "Binary"
                }
            ]
        }
    },
    {
        "domain": "SERIAL_NO",
        "name": "Serial Number - Path",
        "type": "PATH",
        "properties": {
            "paths": [
                {
                    "matchType": "REGEX",
                    "matchStrength": 0.67,

```

```

        "fieldValue": "(?i)(?>(ser(ial)?_?-? ?(number|num|nbr|no))"
    }
  ],
  "rejectStrength": 0.0
}
},
{
  "domain": "SERIAL_NO",
  "name": "Serial Number - Type",
  "type": "TYPE",
  "properties": {
    "allowedTypes": [
      {
        "minimumLength": "0",
        "typeName": "Number"
      },
      {
        "minimumLength": "0",
        "typeName": "String"
      }
    ]
  }
},
{
  "domain": "SEXUAL_ORIENTATION",
  "name": "Sexual Orientation - Path",
  "type": "PATH",
  "properties": {
    "paths": [
      {
        "matchType": "REGEX",
        "matchStrength": 0.67,
        "fieldValue": "(?i)(?>(sexuali?ty|sexual[-_ ]?(orie?nt?(ation)?
s?)?(pref(er)?(ence)?s?)?[-_ ]?(type|identity?))"
      }
    ],
    "rejectStrength": 0.0
  }
},
{
  "domain": "SEXUAL_ORIENTATION",
  "name": "Sexual Orientation - Type",
  "type": "TYPE",
  "properties": {
    "allowedTypes": [
      {
        "minimumLength": "5",
        "typeName": "String"
      }
    ]
  }
},
{

```

```

"domain": "SEXUAL_ORIENTATION",
"name": "Sexual Orientation - List",
"type": "LIST",
"properties": {
  "valueLists": [
    {
      "file": "file://sexual_orientations.txt",
      "matchStrength": 0.9
    }
  ],
  "rejectStrength": 0.1
}
},
{
"domain": "SIGNATURE",
"name": "Signature - Path",
"type": "PATH",
"properties": {
  "paths": [
    {
      "matchType": "REGEX",
      "matchStrength": 0.67,
      "fieldValue": "(?i)(?>(signature))"
    }
  ],
  "rejectStrength": 0.0
}
},
{
"domain": "SIGNATURE",
"name": "Signature - Type",
"type": "TYPE",
"properties": {
  "allowedTypes": [
    {
      "minimumLength": "0",
      "typeName": "String"
    },
    {
      "minimumLength": "0",
      "typeName": "Binary"
    }
  ]
}
},
{
"domain": "SSN",
"name": "Social Security Number - Path",
"type": "PATH",
"properties": {
  "paths": [
    {
      "matchType": "REGEX",

```

```

        "matchStrength": 0.67,
        "fieldValue": "(?i)(?>(ssn$|social_?-? ?security_?-? ?(number|
num|nbr|no|code|id)))"
    },
    "rejectStrength": 0.0
}
},
{
    "domain": "SSN",
    "name": "Social Security Number - Regex",
    "type": "REGEX",
    "properties": {
        "dataPatterns": [
            {
                "regex": "(?!000)(?!666)[0-8]\\d{2}([- ])(?!00)\\d{2}\\1(?!0000)\\
\d{4}",
                "allowPartialMatch": false,
                "matchStrength": 0.7
            },
            {
                "regex": "[9]\\d{2}([- ])(?!6[6-9]))(?!89)([5-9][0-9])\\1\\d{4}",
                "allowPartialMatch": false,
                "matchStrength": 0.7
            },
            {
                "regex": "(?!000)(?!666)[0-8]\\d{2}(?!00)\\d{2}(?!0000)\\d{4}",
                "allowPartialMatch": false,
                "matchStrength": 0.1
            },
            {
                "regex": "[9]\\d{2}(?!6[6-9]))(?!89)([5-9][0-9])\\d{4}",
                "allowPartialMatch": false,
                "matchStrength": 0.1
            }
        ],
        "rejectStrength": 0.1
    }
},
{
    "domain": "SSN",
    "name": "Social Security Number - Type",
    "type": "TYPE",
    "properties": {
        "allowedTypes": [
            {
                "minimumLength": "9",
                "typeName": "Number"
            },
            {
                "minimumLength": "9",
                "typeName": "String"
            }
        ]
    }
}

```

```

    ]
  }
},
{
  "domain": "SWIFT_CODE",
  "name": "Swift Code - Path",
  "type": "PATH",
  "properties": {
    "paths": [
      {
        "matchType": "REGEX",
        "matchStrength": 0.67,
        "fieldValue": "(?>(swift|(^|[-_ ])bic)[-_ ]?(number|num|nbr|no|
code|cd|id)|bank[-_ ]?(identification|identifier)[-_ ]?(code|cd))"
      }
    ],
    "rejectStrength": 0.0
  }
},
{
  "domain": "SWIFT_CODE",
  "name": "Swift Code - Regex",
  "type": "REGEX",
  "properties": {
    "dataPatterns": [
      {
        "note": "Swift-code format: 8-11 digit code with country-code at
5,6 positions.",
        "regex": "[A-Z]{4}-?[AD|AE|AF|AG|AI|AL|AM|AO|AQ|AR|AS|AT|AU|AW|
AX|AZ|BA|BB|BD|BE|BF|BG|BH|BI|BJ|BL|BM|BN|BO|BQ|BR|BS|BT|BV|BW|BY|BZ|CA|CC|CD|CF|CG|
CH|CI|CK|CL|CM|CN|CO|CR|CU|CV|CW|CX|CY|CZ|DE|DJ|DK|DM|DO|DZ|EC|EE|EG|EH|ER|ES|ET|FI|
FJ|FK|FM|FO|FR|GA|GB|GD|GE|GF|GG|GH|GI|GL|GM|GN|GP|GQ|GR|GS|GT|GU|GW|GY|HK|HM|HN|HR|
HT|HU|ID|IE|IL|IM|IN|IO|IQ|IR|IS|IT|JE|JM|JO|JP|KE|KG|KH|KI|KM|KN|KP|KR|KW|KY|KZ|LA|
LB|LC|LI|LK|LR|LS|LT|LU|LV|LY|MA|MC|MD|ME|MF|MG|MH|MK|ML|MM|MN|MO|MP|MQ|MR|MS|MT|MU|
MV|MW|MX|MY|MZ|NA|NC|NE|NF|NG|NI|NL|NO|NP|NR|NU|NZ|OM|PA|PE|PF|PG|PH|PK|PL|PM|PN|PR|
PS|PT|PW|PY|QA|RE|RO|RS|RU|RW|SA|SB|SC|SD|SE|SG|SH|SI|SJ|SK|SL|SM|SN|SO|SR|SS|ST|SV|
SX|SY|SZ|TC|TD|TF|TG|TH|TJ|TK|TL|TM|TN|TO|TR|TT|TV|TW|TZ|UA|UG|UM|US|UY|UZ|VA|VC|VE|
VG|VI|VN|VU|WF|WS|YE|YT|ZA|ZM|ZW]-?[A-Z0-9]{2}(-[A-Z0-9]{3})?",
        "allowPartialMatch": false,
        "matchStrength": 0.2
      }
    ],
    "rejectStrength": 0.3
  }
},
{
  "domain": "SWIFT_CODE",
  "name": "Swift Code - Type",
  "type": "TYPE",
  "properties": {
    "allowedTypes": [
      {
        "minimumLength": "8",

```

```

        "typeName": "String"
      }
    ]
  }
},
{
  "domain": "TAX_ID",
  "name": "Tax ID - Path",
  "type": "PATH",
  "properties": {
    "paths": [
      {
        "matchType": "REGEX",
        "matchStrength": 0.67,
        "fieldValue": "(?i)(?(tax)_?-? ?(id(ent)?_)-? ?((co?de?)|
(number|num|nbr|no)?)"
      },
      {
        "matchType": "REGEX",
        "matchStrength": 0.67,
        "fieldValue": "(?i)(?>tin$)"
      }
    ],
    "rejectStrength": 0.0
  }
},
{
  "domain": "TAX_ID",
  "name": "Tax ID - Type",
  "type": "TYPE",
  "properties": {
    "allowedTypes": [
      {
        "minimumLength": "6",
        "typeName": "String"
      },
      {
        "minimumLength": "6",
        "typeName": "Number"
      }
    ]
  }
},
{
  "domain": "TELEPHONE_NO",
  "name": "Telephone Number - Path",
  "type": "PATH",
  "properties": {
    "paths": [
      {
        "matchType": "REGEX",
        "matchStrength": 0.67,

```



```

                "fieldValue": "(?i)(?>(phone|contact|^tel($|[ _-])|fax|mobile|
telephone)_-? ?)(number|num|nbr|no)?"
            }
        ],
        "rejectStrength": 0.0
    }
},
{
    "domain": "TELEPHONE_NO",
    "name": "Telephone Number - Regex",
    "type": "REGEX",
    "properties": {
        "dataPatterns": [
            {
                "regex": "((\\+?1|001)[- . ]?)?(\\([0-9]{3}\\)[- . ]?[0-9]{3}
[- . ])[0-9]{3}[- . ]?[0-9]{4}",
                "allowPartialMatch": false,
                "matchStrength": 0.7
            },
            {
                "regex": "\\+(?:[0-9][ .()\\/-]?){6,14}[0-9]",
                "allowPartialMatch": false,
                "matchStrength": 0.5
            },
            {
                "regex": "[0-9]{5,17}",
                "allowPartialMatch": false,
                "dataCleanRegex": "[+ -.()\\/-]",
                "matchStrength": 0.05
            }
        ],
        "rejectStrength": 0.1
    }
},
{
    "domain": "TELEPHONE_NO",
    "name": "Telephone Number - Type",
    "type": "TYPE",
    "properties": {
        "allowedTypes": [
            {
                "minimumLength": "7",
                "typeName": "Number"
            },
            {
                "minimumLength": "7",
                "typeName": "String"
            }
        ]
    }
},
{
    "domain": "UNION_MEMBERSHIP_NO",

```

```

    "name": "Union Membership Number - Path",
    "type": "PATH",
    "properties": {
      "paths": [
        {
          "matchType": "REGEX",
          "matchStrength": 0.67,
          "fieldValue": "(?i)(?(union)[_ -?](membe?r(ship)?)?[_ -]?
(number|num|nbr|no|id)(?!\\w*(name|group|grp))"
        }
      ],
      "rejectStrength": 0.0
    }
  },
  {
    "domain": "UNION_MEMBERSHIP_NO",
    "name": "Union Membership Number - Type",
    "type": "TYPE",
    "properties": {
      "allowedTypes": [
        {
          "minimumLength": "3",
          "typeName": "String"
        },
        {
          "minimumLength": "3",
          "typeName": "Number"
        }
      ]
    }
  },
  {
    "domain": "US_PASSPORT",
    "name": "US Passport - Path",
    "type": "PATH",
    "properties": {
      "paths": [
        {
          "matchType": "REGEX",
          "matchStrength": 0.67,
          "fieldValue": "(?i)(?(passport)[_ -?](number|num|nbr|no)?"
        }
      ],
      "rejectStrength": 0.0
    }
  },
  {
    "domain": "US_PASSPORT",
    "name": "US Passport - Regex",
    "type": "REGEX",
    "properties": {
      "dataPatterns": [
        {

```

```

        "regex": "[A-Z]\\d{8}",
        "allowPartialMatch": false,
        "matchStrength": 0.2
      },
      {
        "regex": "(?!0{9})\\d{9}",
        "allowPartialMatch": false,
        "matchStrength": 0.1
      }
    ],
    "rejectStrength": 0.1
  }
},
{
  "domain": "US_PASSPORT",
  "name": "US Passport - Type",
  "type": "TYPE",
  "properties": {
    "allowedTypes": [
      {
        "minimumLength": "9",
        "typeName": "String"
      },
      {
        "minimumLength": "9",
        "typeName": "Number"
      }
    ]
  }
},
{
  "domain": "US_STATE",
  "name": "US State - Path",
  "type": "PATH",
  "properties": {
    "paths": [
      {
        "matchType": "REGEX",
        "matchStrength": 0.67,
        "fieldValue": "(?i)state",
        "allowPartialMatch": false
      }
    ],
    "rejectStrength": 0.0
  }
},
{
  "domain": "US_STATE",
  "name": "US State - Type",
  "type": "TYPE",
  "properties": {
    "allowedTypes": [
      {

```

```

        "minimumLength": "14",
        "typeName": "String"
    }
  ]
}
},
{
  "domain": "US_STATE",
  "name": "US State - List",
  "type": "LIST",
  "properties": {
    "valueLists": [
      {
        "file": "file://us_states_full.txt",
        "matchStrength": 1.0
      }
    ],
    "rejectStrength": 0.1
  }
},
{
  "domain": "USPS_STATE_CODE",
  "name": "USPS State Code - Path",
  "type": "PATH",
  "properties": {
    "paths": [
      {
        "matchType": "REGEX",
        "matchStrength": 0.67,
        "fieldValue": "(?i)state[ _-]? (cd|code|abbrev)?",
        "allowPartialMatch": false
      }
    ],
    "rejectStrength": 0.0
  }
},
{
  "domain": "USPS_STATE_CODE",
  "name": "USPS State Code - Type",
  "type": "TYPE",
  "properties": {
    "allowedTypes": [
      {
        "minimumLength": "2",
        "typeName": "String"
      }
    ]
  }
},
{
  "domain": "USPS_STATE_CODE",
  "name": "USPS State Code - List",
  "type": "LIST",

```

```

    "properties": {
      "valueLists": [
        {
          "file": "file://us_states.txt",
          "matchStrength": 1.0
        }
      ],
      "rejectStrength": 0.5
    }
  },
  {
    "domain": "VIN_NO",
    "name": "Vehicle Identification Number - Path",
    "type": "PATH",
    "properties": {
      "paths": [
        {
          "matchType": "REGEX",
          "matchStrength": 0.67,
          "fieldValue": "(?i)(?(>(^vin$|vehicle_?-? ?id(entification)?_?-? ?
(number|num|nbr|no)))"
        },
        {
          "matchType": "REGEX",
          "matchStrength": 0.67,
          "fieldValue": "(?i)(?(>(vehicle))"
        }
      ],
      "rejectStrength": 0.0
    }
  },
  {
    "domain": "VIN_NO",
    "name": "Vehicle Identification Number - Type",
    "type": "TYPE",
    "properties": {
      "allowedTypes": [
        {
          "minimumLength": "10",
          "typeName": "String"
        }
      ]
    }
  },
  {
    "domain": "WEB",
    "name": "Web URL - Path",
    "type": "PATH",
    "properties": {
      "paths": [
        {
          "matchType": "REGEX",
          "matchStrength": 0.67,

```

```

        "fieldValue": "(?i)(?(>(^url_?-? ?|web_? ?)(addr?e?s?s?)?)")
    }
  ],
  "rejectStrength": 0.0
}
},
{
  "domain": "WEB",
  "name": "Web URL - Type",
  "type": "TYPE",
  "properties": {
    "allowedTypes": [
      {
        "minimumLength": "10",
        "typeName": "String"
      }
    ]
  }
},
{
  "domain": "USER_ID",
  "name": "User ID - Path",
  "type": "PATH",
  "properties": {
    "paths": [
      {
        "matchType": "REGEX",
        "matchStrength": 0.67,
        "fieldValue": "(?i)(?(>(use?r)([-_ ])?(na?me?|id)?)$"
      }
    ],
    "rejectStrength": 0.0
  }
},
{
  "domain": "USER_ID",
  "name": "User ID - Type",
  "type": "TYPE",
  "properties": {
    "allowedTypes": [
      {
        "minimumLength": "8",
        "typeName": "String"
      },
      {
        "minimumLength": "8",
        "typeName": "Number"
      }
    ]
  }
}
]

```



The ASDD Standard profile set can identify the following PII:

ASDD Standard identifiable PII

Account Number
Address Line 1
Address Line 2
Age
Bank Account Number
Beneficiary Number
Biometric
Blood Type
Certificate Number
City
Company Name
Country
County
Credit Card Number
Customer Number
Date of Birth
Department
Drivers License
Email Address
Ethnicity
First Name
Text
Full Name
Gender
HIPAA Date
House Number
IBAN
IP Address
Job Title
Language
Last Name
License Plate
Location Coordinate
Marital Status
Medical Codes
Medical Drug
National Origin
PO Box
Password
Payment Amount
Political Orientation
Postcode
Precinct
Record Number
Religious Orientation

Routing Number
 School Name
 Security Code
 Serial Number
 Sexual Orientation
 Signature
 Social Security Number
 Swift Code
 Tax ID
 Telephone Number
 Union Membership Number
 US Passport
 US State
 USPS State Code
 Vehicle Identification Number
 Web URL
 User ID

7.4 Standard profile set expressions

This section lists all the column and type profile expressions used by the standard profile set included with the product.

7.4.1 Column level expressions

Expression Name	Domain	Expression
Account_Number_V2	ACCOUNT_NO	(?>(account acct acnt acct)_?-? ?(number num nbr no user))\$
Address_Line1_V2	ADDRESS	(?>((street_?-? ?address) street address)_?-? ?((line)? ?_?(1))?)\$
Address_Line2_V2	ADDRESS_LINE2	(?>((street_?-? ?address) street address)_?-? ?((line)? ?_?(2 3 4 5))?)\$
Beneficiary_NO_V2	BENEFICIARY_NO	(?>(bene(ficiary)?_?-? ?(Number Num Nbr No Id)))\$
Biometric_V2	BIOMETRIC	(?>(biometric))\$

Expression Name	Domain	Expression
Certificate_Number_V2	CERTIFICATE_NO	(?>(Cert(ificate)?_?-? ?(Number Num Nbr No)))\$
Certificate_ID_V2	CERTIFICATE_NO	(?>cert(ificate)?_?-? ?id)
City_V2	CITY	(?>^(address_?-? ?city city city_?-? ?address)\$)
City_V2_2	CITY	(?>^(address home_?-? ?city city city_?-? ?address?e?)\$)
County_V2	COUNTY	(?>(county)\$)
Card_Number_V2	CREDIT CARD	(?>card_?-? ?(number num nbr no))\$
Credit_Card_Number_V2	CREDIT CARD	(?>credit_?-? ?card_?-? ?(number num nbr no))\$
Customer_Number_V2	CUSTOMER_NO	(?>(cust(omer mr)?) ?_?-?(num(ber)? nbr no))\$
Birth_Date_V2	DOB	(?>b(irth)?_?-? ?(date day dt))\$
DOB_Date_V2	DOB	(?>dob dtofb (day date? dt)_?-?(of)?_?(birth))\$
Admission_Date_V2	DOB	(?>(adm(it ission)? tr(ea)?t(ment)?_?-? ds disc(h harge))_? ?(date day dt))\$
Drivers_License_Number_V2	DRIVING_LICENSE	(?>(drivers? lic(ense)?)_?-? ?(number num nbr no))\$
Email_V2	EMAIL	(?>(email_?-? ?)(addr?e?s?s?)?)\$

Expression Name	Domain	Expression
Email_V2_2	EMAIL	(cust customer partner home private def default)_?-? ?(email)_?-? ?(address)
First_Name_V2	FIRST_NAME	(?>(f(first)?_?-? ?(na?me?))(_?-?user)?)\$
Middle_Name_V2	FIRST_NAME	(?>(mid(dle)?_?-? ?(na?me?))(_?-?user)?)\$
Full_Name_V2	FULL_NAME	(?>((fu?l?l_?-? ? whole_?-? ?)?_?-?(na?me?))(_?-?user)?)\$
IP_Address_V2	IP ADDRESS	(?>(ip_?-? ?addre?s?s?))\$
Last_Name_V2	LAST_NAME	(?>((l(as)?t)_?-? ?(na?me?))(_?-?user)?)\$
Surname_V2	LAST_NAME	(?>(sur) _?-? ?(name)?_?-? ?(no id str value))
Password_V2	PASSWORD	(?>(pass) _?-??(word)?_?-? ?(word nbr no id value))
PO_Box_V2	PO_BOX	(?>(p.?o.?_?-? ?box post_?-? ?office_?-? ?box _?-?) (number num nbr no)?\$)
Precinct_V2	PRECINCT	(?>precinct prcncct)\$
Record_Number_V2	RECORD_NUMBER	(?>(rec record)_?(number num nbr no))\$
School_Name_V2	SCHOOL_NAME	(?>school_?-?na?me?)\$

Expression Name	Domain	Expression
Security_Code_V2	SECURITY_CODE	(?>se?cu?r(i?ty)?_?co?de?)\$
Serial_Number_V2	SERIAL_NO	(?>(ser(ial)?)_?-? ?(number num nbr no))\$
Signature_V2	SIGNATURE	(?>(signature)\$)
Social_Security_Number_V2	SSN	(?>(ssn\$ social_?-? ?security_?-? ?(number num nbr no code id))\$)
TaxID_Code_or_Number_V2	TAX_ID	(?>(tax)_?-? ?(id(ent)?)_?-? ?((co?de?) (number num nbr no)))?)\$
TaxID_Number_V2	TAX_ID	(?>tin\$)
Telephone_or_Contact_Number_V2	TELEPHONE_NO	(?>(phone contact tel fax)_?-? ?)(number num nbr no)?\$
Vehicle_V2	VIN_NO	(?>(vehicle)\$)
VIN_NO_V2	VIN_NO	(?>(^vin\$ Vehicle_?-? ?Id(entification)?_?-? ?(Number Num Nbr No))\$)
Web_URL_Address_V2	WEB	(?>(^url_?-? ? web_? ?)(addr?e?s?s?)?)\$
Zip_or_Postal_Code_V2	ZIP	(?>(zip post postal)_?-? ?(co?de?) (zip))

7.4.2 Type expressions

The column data type, if recognized, must match one of the specified types for the domain to be assigned.

Expression Name	Domain	Expression	Minimum Length
ACCOUNT_NO_type_V2	ACCOUNT_NO	Number	5
ACCOUNT_NO_type_V2_2	ACCOUNT_NO	String	5
ADDRESS_type_V2	ADDRESS	String	20
ADDRESS_LINE2_type_V2	ADDRESS_LINE2	String	20
BENEFICIARY_NO_type_V2	BENEFICIARY_NO	String	10
BENEFICIARY_NO_type_V2_2	BENEFICIARY_NO	Number	5
BIOMETRIC_type_V2	BIOMETRIC	String	10
BIOMETRIC_type_V2_2	BIOMETRIC	Binary	None
CERTIFICATE_NO_type_V2	CERTIFICATE_NO	String	10
CERTIFICATE_NO_type_V2_2	CERTIFICATE_NO	Number	5
CITY_type_V2	CITY	String	10
COUNTY_type_V2	COUNTY	String	10
CREDIT_CARD_type_V2	CREDIT_CARD	String	15
CREDIT_CARD_type_V2_2	CREDIT_CARD	Number	15
CUSTOMER_NO_type_V2	CUSTOMER_NO	String	5
CUSTOMER_NO_type_V2_2	CUSTOMER_NO	Number	5
DOB_type_V2	DOB	String	6
DOB_type_V2_2	DOB	Date	None

Expression Name	Domain	Expression	Minimum Length
DRIVING_LC_type_V2	DRIVING_LC	String	10
DRIVING_LC_type_V2_2	DRIVING_LC	Number	10
EMAIL_type_V2	EMAIL	String	20
FIRST_NAME_type_V2	FIRST_NAME	String	10
FULL_NAME_type_V2	FULL_NAME	String	20
IP_ADDRESS_type_V2	IP_ADDRESS	String	10
LAST_NAME_type_V2	LAST_NAME	String	10
PASSWORD_type_V2	PASSWORD	String	6
PO_BOX_type_V2	PO_BOX	String	4
PO_BOX_type_V2_2	PO_BOX	Number	4
PRECINCT_type_V2	PRECINCT	String	None
PRECINCT_type_V2_2	PRECINCT	Number	None
RECORD_NO_type_V2	RECORD_NO	String	5
RECORD_NO_type_V2_2	RECORD_NO	Number	5
SCHOOL_NM_type_V2	SCHOOL_NM	String	20
SECURITY_CODE_type_V2	SECURITY_CODE	String	5
SECURITY_CODE_type_V2_2	SECURITY_CODE	Number	5
SERIAL_NO_type_V2	SERIAL_NO	Number	None

Expression Name	Domain	Expression	Minimum Length
SERIAL_NO_type_V2_2	SERIAL_NO	String	None
SIGNATURE_type_V2	SIGNATURE	String	None
SIGNATURE_type_V2_2	SIGNATURE	Binary	None
SSN_type_V2	SSN	String	None
SSN_type_V2_2	SSN	Number	None
TAX_ID_type_V2	TAX_ID	String	6
TAX_ID_type_V2_2	TAX_ID	Number	6
TELEPHONE_NO_type_V2	TELEPHONE_NO	String	7
TELEPHONE_NO_type_V2_2	TELEPHONE_NO	Number	7
VIN_NO_type_V2	VIN_NO	String	10
WEB_type_V2	WEB	String	10
ZIP_type_V2	ZIP	Number	4
ZIP_type_V2_2	ZIP	String	4

7.5 Legacy profile set expressions

This section describes all the column level profile expressions used by the two legacy Profile Sets included with the product, as well as several data level expressions included with the product but not in any of the pre-constructed Profile Sets.

7.5.1 Account numbers

An account number is the primary identifier for ownership of an account, whether a vendor account, a checking or brokerage account, or a loan account. An account number is used whether or not the identifier uses letters or numbers. Below are the profile Expressions Delphix uses to identify account numbers:

Expression Name	Domain	Expression Level	Expression
Account number	ACCOUNT_NO	Column	(?>(acc(oun\ n)?t)?_?(num(ber)?\ nbrjno)?)(?!\\w*(ID\\ type))

7.5.2 Physical addresses

Below are the profile Expressions Delphix uses to identify physical addresses:

Expression Name	Domain	Expression Level	Expression
Address	ADDRESSES	Column	^(?:(?!postalcode\\ city\\ state\\ country\\ email\\ (l\\ ln\\ lin\\ line)?_?2{1}\\ ID).)*addre?s?s?_?(?:(?!city\\ state\\ country\\ email\\ (l\\ ln\\ lin\\ line)?_?2{1}\\ ID).)*\$
Street Address	ADDRESSES	Column	(?>(str(eet)?_?addre?s?s?\\ street))(?!\\w*(ID\\ type))
Data - Address	ADDRESSES	Data	(.[\\s]+b(ou)?*l(e)?v(ar)?d[\\s].) (. [\\s]+st[.]?(reet)?[\\s].) (. [\\s]+ave[.]?(nue)?[\\s].) (. [\\s]+r(oa)?d[\\s].) (. [\\s]+l(a)?n(e)?[\\s].) (. [\\s]+cir(cle)?[\\s].*)
Address Line2 - before	ADDRESSES_LINE2	Column	^(?:(?!email\\ ID).)*(l\\ ln\\ lin\\ line)?2{1}_?addre?s?s?(?:(?!email\\ ID).)*\$
Address Line2 - after	ADDRESSES_LINE2	Column	^(?:(?!email\\ ID).)*addre?s?s?_(l\\ ln\\ lin\\ line)?_?2{1}(?:(?!email\\ ID).)*\$

Expression Name	Domain	Expression Level	Expression
Data - Address Line 2	ADDRESSES_LI NE2	Data	(.*[\s]*ap(ar)?t(ment)?[\s]+.*) (*[\s]*s(ui)?te[\s]+.*) (c(are)?[\s]*[\\\/]?o(f)?[\s]+.*)

7.5.3 Beneficiary ID

Below are the profile Expressions Delphix uses to identify beneficiary IDs:

Expression Name	Domain	Expression Level	Expression
Beneficiary number	BENEFICIARY_NO	Column	(?>(bene(ficiary)?)_?(num(ber)? nbr\ no))(?!w*ID)1
Beneficiary ID	BENEFICIARY_NO	Column	(?>(bene(ficiary)?)_?id)

7.5.4 Biometrics

Below are the profile Expressions Delphix uses to biometric data:

Expression Name	Domain	Expression Level	Expression
Biometric	BIOMETRIC	Column	biometric

7.5.5 Certificate ID

Below are the profile Expressions Delphix uses to identify certificate IDs:

Expression Name	Domain	Expression Level	Expression
Certificate number	CERTIFICATE_NO	Column	(?>cert(ificate)?_?(num(ber)?\ nbr\ no\ id))
Certificate ID	CERTIFICATE_NO	Column	(?>cert(ificate)?_?id)

7.5.6 City

Below are the profile Expressions Delphix uses to identify cities:

Expression Name	Domain	Expression Level	Expression
City	CITY	Column	ci?ty(?!\\w*ID)

7.5.7 Country

Below are the profile Expressions Delphix uses to identify countries:

Expression Name	Domain	Expression Level	Expression
Country	COUNTRY	Column	c(ou)?nty(?!\\w*ID)

7.5.8 Credit card

Below are the profile Expressions Delphix uses to identify credit cards:

Expression Name	Domain	Expression Level	Expression
Card number	CREDIT CARD	Column	(?>ca?rd_?(num(ber)?\ nbr\ no)?)(?!\\w*ID)

Expression Name	Domain	Expression Level	Expression
Credit Card number	CREDIT CARD	Column	(?>cre?di?t_(ca?rd)?_?(num(ber)?\ nbr\ no)?)(?!\\w*ID)
Data - Credit Card	CREDIT CARD	Data	^(?:3[47][0-9]{13} 4[0-9]{12}(?:[0-9]{3})?(?:[0-9]{3})?\ (?:5[1-5][0-9]{2}\ 22[1-9]\ 22[3-9][0-9]\ 2[3-6][0-9]{2}\ 27[01][0-9]\ 2720)[0-9]{12}\ 6(?:011\ 5[0-9][0-9])[0-9]{2}\ 4[4-9][0-9]{3}\ 2212[6-9]\ 221[3-9][0-9]\ 22[2-8][0-9]{2}\ 229[0-1][0-9] 2292[0-5])[0-9]{10}?(?:[0-9]{3})?\ 3(?:0[0-5,9]\ 6[0-9])[0-9]{11}\ 3[89][0-9]{14}?(?:[0-9]{1,3})?)\$

7.5.9 Customer number

Below are the profile Expressions Delphix uses to identify customer IDs:

Expression Name	Domain	Expression Level	Expression
Customer number	CUSTOMER_NUM	Column	(?>(cu?st(omer\ mr)?)_?(num(ber)?\ nbr\ no)?)(?!\\w*ID)

7.5.10 Date of birth

Below are the profile Expressions Delphix uses to identify dates of birth:

Expression Name	Domain	Expression Level	Expression
Birth Date	DOB	Column	(?>(bi?rth)_?(date?\ day\ dt))(?!\\w*ID)

Expression Name	Domain	Expression Level	Expression
Birth Date1	DOB	Column	(?>dob\ dtofb\ (day\ date?\ dt)_?(of)?_?(bi?rth))(?!\\w*ID)
Birth Date2	DOB	Column	(?>b_(date?\ day))(?!\\w*ID)
Admission Date	DOB	Column	(?>(adm(it\ ission)?)_?(date?\ day\ dt))(?!\\w*ID)
Treatment Date	DOB	Column	(?>(tr(ea)?t(ment)?)_?(date?\ day\ dt))(?!\\w*ID)
Discharge Date	DOB	Column	(?>(ds\ disc(h\ harge)?)_?(date?\ day\ dt))(?!\\w*ID)

7.5.11 Driver license number

Below are the profile Expressions Delphix uses to identify driver license numbers:

Expression Name	Domain	Expression Level	Expression
Drivers License number	DRIVIN G_LC	Column	(?>(dri?v(e?rs?e)?)_?(license li?c)?_?(num(ber)?\ nbr no?))(?!\\w*ID)
Drivers License number1	DRIVIN G_LC	Column	(^license\$\\ (license\\ li?c)?_?(num(ber)?\ nbr\ no?))(?!\\w*ID)

7.5.12 Email

Below are the profile Expressions Delphix uses to identify emails:

Expression Name	Domain	Expression Level	Expression
Email	EMAIL	Column	<code>^(?:(!invalid).)*email(?:\w*ID)</code>
Data - Email	EMAIL	Column	<code>\b[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,6}\b</code>

7.5.13 First name

Below are the profile Expressions Delphix uses to identify first names:

Expression Name	Domain	Expression Level	Expression
First Name	FIRST_NAME	Column	<code>(?>(fi?rst)?(na?me?)\ f_?name)(?! \w*ID)</code>
Middle Name	FIRST_NAME	Column	<code>(?>(mid(dle)?)?(na?me?)\ m_?name)(?! \w*ID)</code>

7.5.14 IP address

Below are the profile Expressions Delphix uses to IP addresses:

Expression Name	Domain	Expression Level	Expression
IP Address	IP ADDRESS	Column	<code>(?>(ip_?address?s?s?))(?! \w*(ID\ type))</code>
Data - IP Address	IP ADDRESS	Data	<code>\b(?:(:?25[0-5]\ 2[0-4][0-9]\ 1[0-9][0-9]\ [1-9]?[0-9])\.)\{3\}(?:25[0-5]\ 2[0-4][0-9]\ 1[0-9][0-9]\ [1-9]?[0-9])\b</code>

7.5.15 Last name

Below are the profile Expressions Delphix uses to identify last names:

Expression Name	Domain	Expression Level	Expression
Last Name	LAST_NAME	Column	<code>^(?:(!portal\ ID .)*((la?st)?(na?me?)\ l_?name)(?:(!portalname\ ID .)*\$</code>

7.5.16 Plate number

Below are the profile Expressions Delphix uses to identify plate numbers:

Expression Name	Domain	Expression Level	Expression
License Plate	PLATE_NO	Column	<code>^(?:(!template ID type .)*(license\ li?c)?_?plate_(number)?\ nbr\ no)?(?:(!template\ ID type .)*\$</code>

7.5.17 PO Box numbers

Below are the profile Expressions Delphix uses to identify PO box numbers:

Expression Name	Domain	Expression Level	Expression
PO Box	PO_BOX	Column	<code>po_?box</code>
Data - PO Box	PO_BOX	Data	<code>po box\ p\.o\</code>

7.5.18 Precinct

Below are the profile Expressions Delphix uses to identify precincts:

Expression Name	Domain	Expression Level	Expression
Precinct	PRECINCT	Column	(>?precinct\ prcnct)(?!\\w*ID)

7.5.19 Record number

Below are the profile Expressions Delphix uses to identify record numbers:

Expression Name	Domain	Expression Level	Expression
Record number	RECORD_NO	Column	(?>rec(ord)?_?(num(ber)?\\ nbr\\ no))(?!\\w*(ID\\ type))

7.5.20 School name

Below are the profile Expressions Delphix uses to identify school names:

Expression Name	Domain	Expression Level	Expression
School Name	SCHOOL_NM	Column	(?>school_?na?me?)(?!\\w*ID)

7.5.21 Security code

Below are the profile Expressions Delphix uses to identify security codes:

Expression Name	Domain	Expression Level	Expression
Security Code	SECURITY_CODE	Column	(?>se?cu?r(i?ty)?_?co?de?)(?!\\w*ID)

7.5.22 Serial number

Below are the profile Expressions Delphix uses to identify serial numbers:

Expression Name	Domain	Expression Level	Expression
Serial number	SERIAL_NUMBER	Column	(?>(ser(ial)?)_?(num(ber)?\ nbr no)) (?!\\w*ID)

7.5.23 Signature

Below are the profile Expressions Delphix uses to identify signatures:

Expression Name	Domain	Expression Level	Expression
Signature	SIGNATURE	Column	signature(?!\\w*(ID\\ type))

7.5.24 Social security number

Below are the profile Expressions Delphix uses to social security numbers:

Expression Name	Domain	Expression Level	Expression
Social Security number	SSN	Column	ssn(?!\\w*ID)
Data - SSN	SSN	Data	\\b(?:000)(?:666)[0-8]\\d{2}[-](?:00)\\d{2}[-](?:0000)\\d{4}\\b

7.5.25 Tax ID

Below are the profile Expressions Delphix uses to identify tax IDs:

Expression Name	Domain	Expression Level	Expression
Tax ID number	TAX_ID	Column	tin\$ ^tin\\ _tin\\ tin_

Expression Name	Domain	Expression Level	Expression
Tax ID Code or number	TAX_ID	Column	<code>(ta?x)?(id(ent)?)?_?((co?de?)\ num(ber)?\ nbr\ no)?</code>

7.5.26 Telephone number

Below are the profile Expressions Delphix uses to identify telephone numbers:

Expression Name	Domain	Expression Level	Expression
Telephone or Contact number	TELEPHONE_NO	Column	<code>(?>((tele?)?phone)\ (co?nta?ct\ tel)_?(num(ber)?\ nbr\ no))(?!w*(ID\ type))</code>
Data - Phone number	TELEPHONE_NO	Data	<code>\(?:\b[0-9]{3}\)?[-.]?[0-9]{3}[-.]?[0-9]{4}\b</code>
Fax number	TELEPHONE_NO	Data	<code>(?>fax_?(num(ber)?\ nbr\ no)?)(?!w*(ID\ type))</code>

7.5.27 Vin number

Below are the profile Expressions Delphix uses to identify vin numbers:

Expression Name	Domain	Expression Level	Expression
Vehicle	VIN_NO	Column	vehicle
VIN	VIN_NO	Column	<code>vin\$\ ^vin\ _vin\ vin_</code>

7.5.28 Web address

Below are the profile Expressions Delphix uses to identify web addresses:

Expression Name	Domain	Expression Level	Expression
Web or URL Address	WEB	Column	<code>(?>(url\ web_?address?s?s?))(!\w*(ID\ type))</code>
Data - Web Address	WEB	Data	<code>\b(?:(:https?\ ftp\ file)://\ www\. \ ftp\.)[-AZ0-9+&-@#/%=~_\ \$?!:,.]*[A-Z0-9+&-@#/%=~_\ \$\]</code>

7.5.29 ZIP code

Below are the profile Expressions Delphix uses to identify zip codes:

Expression Name	Domain	Expression Level	Expression
zip or Postal Code	ZIP	Column	<code>(?>(zip\ post(al)?_?((code?)?4?))(!\w*ID)</code>
Data - Zip Code	ZIP	Data	<code>1\b([0-9]{5})-([0-9]{4})\b</code>

7.6 Configuring profile sets

A profile set defines the set of classifiers or expressions that will be used to identify sensitive information in the rule set when a profiling job is run. Refer to [Discovering Your Sensitive Data \(see page 526\)](#) for an overview of profile sets and related concepts.

To display, view, and manage the profile sets, navigate to **Settings > Profile Sets**.

Settings > Profile Sets

Profile Sets

+ Profile Set

Name	Description	ASDD	Owner	Created	Actions
set_B6DLFHB3	IL4T92EC	No	admin	2 Apr 2024 11:22 IST	...
set_4U7ZIRRB	YYQY6GIE	No	admin	2 Apr 2024 10:05 IST	...
set_PEP45RXN	4IOIRD8G	No	admin	2 Apr 2024 10:03 IST	...
ASDD Standard		Yes	admin	16 Jun 2023 01:27 IST	...
Standard		No	admin	16 Jun 2023 01:27 IST	...
Financial - Legacy		No	admin	15 Jun 2023 05:30 IST	...
HIPAA - Legacy		No	admin	15 Jun 2023 05:30 IST	...

Displaying 1 to 7 of 7

The Profile sets on the screen can be filtered or sorted by the various informational fields by clicking on the respective field. More information on grid filtering and sorting can be found [here](#) (see page 232).



- Sortable fields are Name, Description and Created.
- Filterable fields are Name, Description, Owner and Created.

7.6.1 Creating Profile Sets

The content of a profiler set depends on the profiler implementation with which it is intended to be used. Profile sets for the legacy profile contain search expressions and type expressions, while profile sets for the ASDD profiler contain classifiers.

7.6.1.1 Adding a Profiler set for the legacy profiler

1. Click the **+ Profile Set** button from the top-right corner just above the Profile Set grid.
2. The Add Profiler Set dialog appears.

Add Profile Set

Legacy profile sets are deprecated. They will reach end of life and be removed in an upcoming release. Read more [here](#)

Set Name
Demo

Description (Optional)

ASDD Support

Search Expressions Type Expressions

Select Search Expressions

<input type="checkbox"/> Select All
<input type="checkbox"/> Account Number
<input type="checkbox"/> Account_Number_V2
<input type="checkbox"/> Address
<input type="checkbox"/> Address_Line1_V2
<input type="checkbox"/> Address Line 2 - after
<input type="checkbox"/> Address Line2 - before
<input type="checkbox"/> Address_Line2_V2
<input type="checkbox"/> Admission Date

Cancel Save

3. Enter a profiler **Set Name**. This name must be unique among all profile set names.
4. Optionally, enter a **Description** for this Profiler Set.
5. Select the **ASDD Support** checkbox.
6. Select the **Search Expressions** tab (it is selected by default).
7. Click the checkbox to the left of each search expression you wish to add to the profile set. It is not recommended that All Search Expressions be selected, as there is significant duplication among built-in search expressions.
8. Select the **Type Expressions** tab.
9. Click the box to the left of each type of expression you wish to add to the profile set.

Add Profile Set

Legacy profile sets are deprecated. They will reach end of life and be removed in an upcoming release. Read more [here](#)

Set Name
Demo

Description (Optional)

ASDD Support

Search Expressions **Type Expressions**

Select Type Expressions

<input type="checkbox"/> Select All
<input type="checkbox"/> ACCOUNT_NO_type_V2
<input type="checkbox"/> ACCOUNT_NO_type_V2_2
<input type="checkbox"/> ADDRESS_LINE2_type_V2
<input type="checkbox"/> ADDRESS_type_V2
<input type="checkbox"/> BENEFICIARY_NO_type_V2
<input type="checkbox"/> BENEFICIARY_NO_type_V2_2
<input type="checkbox"/> BIOMETRIC_type_V2
<input type="checkbox"/> BIOMETRIC_type_V2_2

Cancel Save

- When you are finished adding expressions, click **Save** to save the new profile set.

7.6.1.2 Adding a Profiler set for the ASDD profiler

- Click the **+ Profile Set** button from the top-right corner just above the Profile Set grid.
- The Add Profiler Set dialog appears.
- Enter a profiler **Set Name**. This name must be unique among all profile set names.
- Optionally, enter a **Description** for this Profiler Set.
- Check the **ASDD Support** box(Checked by default).

Add Profile Set

Set Name
Demo

Description (Optional)

Assignment Threshold (Optional)

ASDD Support

Classifiers

Select Classifiers

<input type="checkbox"/> Select All
<input type="checkbox"/> Account Number - Path
<input type="checkbox"/> Account Number - Type
<input type="checkbox"/> Address Line 1 - Path
<input type="checkbox"/> Address Line 1 - Regex
<input type="checkbox"/> Address Line 1 - Type
<input type="checkbox"/> Address Line 2 - Path
<input type="checkbox"/> Address Line 2 - Regex
<input type="checkbox"/> Address Line 2 - Type

Cancel Save

- Optionally, enter the **Assignment Threshold** for this Profiler Set. The value must be met or exceeded for the ASDD profiler to make a domain and algorithm assignment. The value must be an integer from 1 to 100. If not specified, the value defaults to the application setting for ASDD called **DefaultAssignmentThreshold**.
- Click the box to the left of each classifier you wish to add to the profile set.
- When finished with adding classifiers, click **Save** to save the new profile set.

Add Profile Set

Set Name
Demo

Description (Optional)

Assignment Threshold (Optional)


ASDD Support

Classifiers

Select Classifiers

<input type="checkbox"/> Select All
<input checked="" type="checkbox"/> Account Number - Path
<input type="checkbox"/> Account Number - Type
<input type="checkbox"/> Address Line 1 - Path
<input checked="" type="checkbox"/> Address Line 1 - Regex
<input type="checkbox"/> Address Line 1 - Type
<input checked="" type="checkbox"/> Address Line 2 - Path
<input type="checkbox"/> Address Line 2 - Regex
<input type="checkbox"/> Address Line 2 - Type

Cancel Save

 Creating or modifying profile sets can also be done via the API and requires only two API calls. See [ASDD Profile Set Import and Export](#) (see page 646) for usage instructions.

7.6.2 Modifying Profile Set

Users can perform 4 types of action on this screen by clicking the (...) button to the right of the corresponding row under the **Actions** column.

CONTINUOUS COMPLIANCE 24.0.0.0

Environments Monitor Settings Admin Audit

Algorithms
Classifiers
Data Formats
Domains
Expressions
JDBC Drivers
Profile Sets

Settings > Profile Sets

Profile Sets + Profile Set

Name	Description	ASDD	Owner	Created	Actions
set_JTE0WJYN	L4TBN82J	No	admin	31 May 2024 11:47 IST	...
set_61ACCTH	P0BS8V18	Yes	admin	31 May 2024 11:44	View Edit Duplicate Delete
set_A42DPSKH	LW4A5K05	No	admin	31 May 2024 11:43	...
set_E4460BJI	RKZ29SMG	No	admin	31 May 2024 11:40	...
set_L04Z1Y6V	IINDR6T8	No	admin	31 May 2024 00:00	...
set_U47Q81HY	76L9RGAP	No	admin	28 May 2024 12:33	...
set_L6NV44SW	88TDNTLV	Yes	admin	28 May 2024 12:32 IST	...
set_ZO02ZQPO	1410PAPY	No	admin	28 May 2024 12:31 IST	...
set_W7H2BESA	ZZ86RWHD	No	admin	28 May 2024 12:31 IST	...
set_98UC5SHC	QM19N8XI	No	admin	28 May 2024 12:05 IST	...
set_KP57ET4R	YO3XPO4Z	No	admin	28 May 2024 11:09 IST	...
set_1XVCCS9K	Y2F4CJTP	No	admin	28 May 2024 11:08 IST	...

Displaying 1 to 13 of 18

7.6.2.1 View Profile Set

Everything in the dialog will be disabled when the View - action is selected.

View Profile Set

Set Name: ASDD Standard

Description (Optional):

Assignment Threshold (Optional): 1

ASDD Support

Classifiers

- Select All
- Account Number - Path
- Account Number - Type
- Address Line 1 - Path
- Address Line 1 - Regex
- Address Line 1 - Type
- Address Line 2 - Path
- Address Line 2 - Regex
- Address Line 2 - Type

Close

7.6.2.2 Edit Profile Set

The Edit Profile Set dialog will appear with existing details. Check or uncheck the box to the left of each object to add/remove them from the profile set as desired.

Edit Profile Set

Set Name
Demo

Description (Optional)
Editing Demo

Assignment Threshold (Optional)
1


ASDD Support

Classifiers

Select Classifiers

<input type="checkbox"/> Select All
<input checked="" type="checkbox"/> Account Number - Path
<input checked="" type="checkbox"/> Address Line 1 - Path
<input checked="" type="checkbox"/> Address Line 2 - Path
<input type="checkbox"/> Account Number - Type
<input type="checkbox"/> Address Line 1 - Regex
<input type="checkbox"/> Address Line 1 - Type
<input type="checkbox"/> Address Line 2 - Regex
<input type="checkbox"/> Address Line 2 - Type

Cancel Save

 **ASDD Support**
It is not possible to change the value of the **ASDD Support** setting for an existing profile set. A new profile set must be created.

7.6.2.3 Duplicate Profile Set

On selecting a duplicate option, details will be pre-filled to the new profile set dialog, and the user can give a new name and duplicate the profile set.

Duplicate Profile Set

Set Name
Duplicate_demo

Description (Optional)
Creating Demo

Assignment Threshold (Optional)
1

ASDD Support

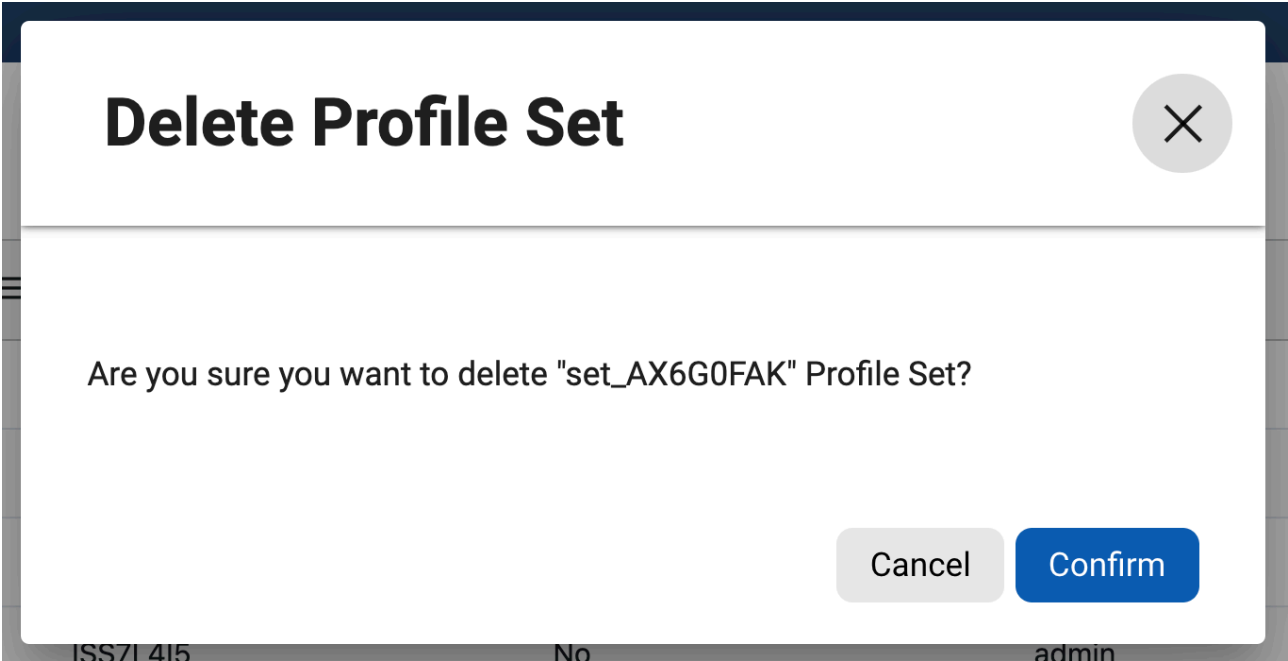
Classifiers

<input type="checkbox"/> Select All
<input checked="" type="checkbox"/> Account Number - Path
<input checked="" type="checkbox"/> Address Line 1 - Path
<input checked="" type="checkbox"/> Address Line 2 - Path
<input type="checkbox"/> Account Number - Type
<input type="checkbox"/> Address Line 1 - Regex
<input type="checkbox"/> Address Line 1 - Type
<input type="checkbox"/> Address Line 2 - Regex
<input type="checkbox"/> Address Line 2 - Type

Cancel Save

7.6.2.4 Delete Profile Set

Clicking Delete Action will prompt for confirmation. Click on **Confirm** to delete the profile set. You will be blocked from deleting a profile set if it is currently assigned to any jobs.



7.7 Managing domains

7.7.1 Overview

This article describes how to create and manage domains. Refer to [Discovering Your Sensitive Data](#) (see page 526) for an overview of domains and related concepts.

7.7.2 Domains

Domains identify a specific type of sensitive data, along with the masking algorithm to use for that data.

Delphix Continuous Compliance includes built-in domains and algorithms for many common types of sensitive data. Users can choose to select a different default masking algorithm for a domain, and/or create additional domains with their default algorithms.

Additional created algorithms appear in the **Algorithms** drop-down menu. Because each domain has a single default masking algorithm, a distinct domain (along with recognition logic) must exist or be created for each distinct algorithm in order for the profiler to assign that algorithm in rule sets.

If the purpose for the environment where a profile job is run is set to **Tokenization/Re-Identify**, the tokenization algorithm associated with the domain will be assigned instead of the masking algorithm. Each domain referenced by any profile set used in tokenization environments should have a tokenization algorithm value defined.

Navigate to **Settings > Domains** and the list of domains will be displayed. From here, you can add, edit, or delete domains. It also shows their default masking and tokenization algorithms.

The screenshot shows the 'Domains' page in the Delphix Continuous Compliance interface. The page title is 'Domains' and there is a '+ Domain' button in the top right corner. The table below lists various domains with their respective owners, masking methods, and tokenization methods.

Name ↑	Owner	Masking Method	Tokenization Method	Actions
ACCOUNT_NO	System	dlpx-core:CM Alpha-Numeric		...
ACCOUNT_TK	System	NullValueLookup	AccountTK	...
ADDRESS	System	AddrLookup		...
ADDRESS_LINE2	System	AddrLine2Lookup		...
AGE	System	dlpx-core:Age SL		...
BANK_ACCOUNT_NO	System	dlpx-core:CM Numeric		...
BENEFICIARY_NO	System	dlpx-core:CM Alpha-Numeric		...
BIOMETRIC	System	NullValueLookup		...
BLOOD_TYPE	System	dlpx-core:BloodType SL		...
CERTIFICATE_NO	System	dlpx-core:CM Alpha-Numeric		...

Displaying 1 to 10 of 69

The domains on the screen can be filtered or sorted by the various informational fields by clicking on the respective fields. More information on grid filtering and sorting can be found [here](#) (see page 232).



- Sortable fields are Name, Masking Method and Tokenization Method.
- Filterable fields are Name, Owner, Masking Method and Tokenization Method.

7.7.3 Adding a new domain

1. Click the **+ Domain** button from the top-right corner above the Domains grid.
2. Enter the new **Domain Name**. The domain name specified will appear as a menu option on the **Inventory** screen elsewhere in the Delphix Masking Engine. Domain names must be unique.

Add Domain ×

- Domain Name
- Algorithm
- Tokenized Algorithm
- Summary

Domain Name

Specify details to identify this domain.

3. Select a default **Masking Algorithm** for the new domain, and click Next.

Add Domain ✕

- Domain Name
- Algorithm**
- Tokenized Algorithm
- Summary

Algorithm

Specify the masking algorithm to apply data matching this domain.

Select Algorithm
ACCOUNT_SL ✕ ▾

Algorithm Details

Name
ACCOUNT_SL

Description
Random number strings

Algorithm Framework
Secure Lookup

Cancel Back Next Save

4. Select a default **Tokenization Algorithm** for the new domain if desired, and click Next.

Add Domain ✕

- Domain Name
- Algorithm
- Tokenized Algorithm**
- Summary

Tokenized Algorithm

Specify the tokenization algorithm to apply data matching this domain.

Select Tokenization Algorithm
ACCOUNT_TK ✕ ▾

Algorithm Details

Name
ACCOUNT_TK

Algorithm Framework
Tokenization

Cancel Back Next Save

5. Click **Save**.

Add Domain



- Domain Name
- Algorithm
- Tokenized Algorithm
- Summary

Summary

View the domain details below. Click the Back button to make changes or click the Save button to save the domain.

Details

Domain Name
Test_domain

Masking Algorithm

Name
ACCOUNT SL

Description
Random number strings

Algorithm Framework
Secure Lookup

Tokenized Algorithm

Name
ACCOUNT_TK

Algorithm Framework
Tokenization

Cancel Back Next Save

7.7.4 Modifying Domains

Users can perform 3 types of action on this screen by clicking the (...) button to the right of the corresponding row under the **Actions** column.

CONTINUOUS COMPLIANCE 24.0.0.0
Environments Monitor Settings Admin Audit

Settings > Domains

Domains + Domain

Name ↑	Owner	Masking Method	Tokenization Method	Actions
ACCOUNT_NO	System	dlpx-core:CM Alpha-Numeric		...
ACCOUNT_TK	System	NullValueLookup	AccountTK	<div style="border: 1px solid #ccc; background-color: white; padding: 5px; width: fit-content;"> <ul style="list-style-type: none"> View Edit Delete </div>
ADDRESS	System	AddrLookup		
ADDRESS_LINE2	System	AddrLine2Lookup		
AGE	System	dlpx-core:Age SL		
BANK_ACCOUNT_NO	System	dlpx-core:CM Numeric		...
BENEFICIARY_NO	System	dlpx-core:CM Alpha-Numeric		...
BIOMETRIC	System	NullValueLookup		...
BLOOD_TYPE	System	dlpx-core:BloodType SL		...

Displaying 1 to 10 of 69

7.7.4.1 View Domain

It will open the summary step directly. Every field on the wizard will be disabled when the View action is selected.

View Domain ✕

- Domain Name
- Algorithm
- Tokenized Algorithm
- Summary**

Summary

View the domain details below. Click cancel to exit.

Details

Domain Name
ACCOUNT_TK

Masking Algorithm

Name
NULL SL

Description
This algorithm replaces the input with a null or empty value, depending on the context. This algorithm's name is chosen for backward compatibility only. It does not perform any kind of lookup and is not related to the Secure Lookup framework.

Tokenized Algorithm

Name
ACCOUNT_TK

Algorithm Framework
Tokenization

Cancel Back Next Save

7.7.4.2 Edit Domain

On clicking **Edit** action, a wizard will appear for editing the domain. The domain name is not editable after creation hence it will be disabled for Editing. Users can update Masking and Tokenized algorithms.

Edit Domain ✕

- Domain Name**
- Algorithm
- Tokenized Algorithm
- Summary

Domain Name

Specify details to identify this domain.

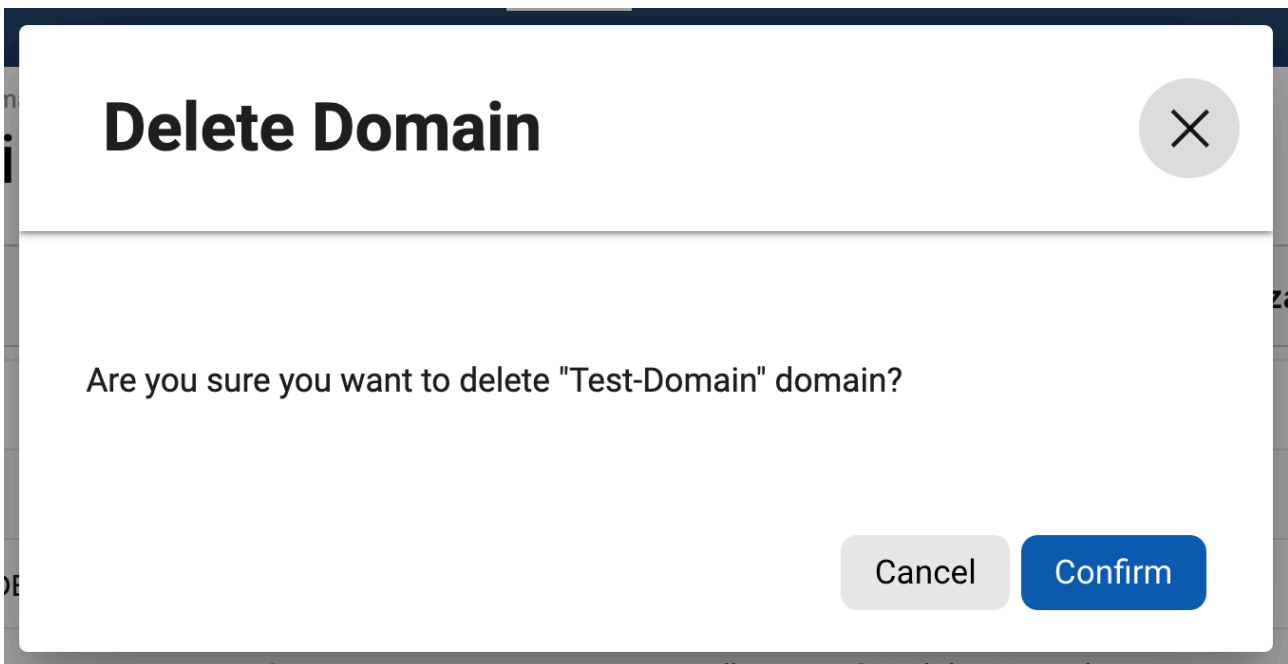
Domain Name
ACCOUNT_NO

Cancel Back Next Save

- Note that updating the default masking or tokenization algorithm assigned to a domain only impacts algorithm assignments made by future profiling job executions, it does **not** have any immediate affect on algorithm assignments in existing rule sets.

7.7.4.3 Delete Domain

Clicking Delete Action will prompt for confirmation. Click on **Confirm** to delete the domain.



7.8 Managing classifiers

Classifier instances define the logic that the ASDD profiler uses to identify sensitive information. For an overview of classifiers and related concepts, refer to [Discovering Your Sensitive Data \(see page 526\)](#). Each classifier instance is based on a **classifier framework** that implements the recognition logic.

An overview of the available frameworks is available in the [classifier concept \(see page 526\)](#) section. The [API Calls for Managing Classifiers \(see page 945\)](#) article describes how to use the API Client to retrieve a detailed description of all classifier frameworks on the system, along with their configuration schemas.

To view a list of all classifier instances available, Navigate to **Settings > Classifiers**.

Settings > Classifiers


Classifiers

[+ Classifier](#)

Name ↑	Domain	Owner	Type	Description	Actions
Credit Card Number - Regex	CREDIT CARD	System	REGEX		...
Credit Card Number - Type	CREDIT CARD	System	TYPE		...
Customer Number - Path	CUSTOMER_NO	System	PATH		...
Customer Number - Type	CUSTOMER_NO	System	TYPE		...
Data Classifier - Email	EMAIL	admin	REGEX	UMR9L97B	...
Data Classifier - Telephone...	TELEPHONE_NO	admin	REGEX	541T7XLB	...
Data Classifier - Zip Code	ZIP	admin	REGEX	6H677W88	...
Date of Birth - Path	DOB	System	PATH		...
Date of Birth - Type	DOB	System	TYPE		...
Department - List	DEPARTMENT	System	LIST		...
Department - Path	DEPARTMENT	System	PATH		...
Department - Type	DEPARTMENT	System	TYPE		...

Displaying 36 to 48 of 165

The classifiers on the screen can be filtered or sorted by the various informational fields by clicking on the respective fields. More information on grid filtering and sorting can be found [here \(see page 232\)](#).



- Sortable fields are Name, Domain and Description.
- Filterable fields are Name, Domain, Owner, Type, and Description.

7.8.1 Adding a new Classifier

1. Click the **+ Classifier** button from the top-right corner above the Classifiers grid.

Add Classifier

Classifier Name:

Domain:

Classifier Framework:

Description (Optional):

Regular Expressions

Match Strength	Checksum Type	Case Sensitive	Partial Match	Actions

Reject Strength (Optional):

Add Regex Configuration

Profiling Regex:

Data Clean Regex (Optional):

Checksum Type (Optional):

Buttons: Cancel, Add

Buttons: Cancel, Save

2. Select a Domain from the **Domain** dropdown.
 - Domains are used by profiling jobs to determine the masking algorithm to apply to the sensitive data. When a classifier is matched, the profiling job will associate the specified domain with the sensitive data.
 - The Masking Engine comes out of the box with over 30 pre-defined domains. For more information on domains, refer to the [Managing Domains \(see page 611\)](#) article.
3. Enter the following information for the classifier:
 - **Classifier Name** to create.
 - **Description** for the classifier (optional).
4. Select the **Classifier Framework** to create the classifier with (i.e. Data Type, Regex, Path, or List). The bottom part of the form will change based on the classifier framework selected.
5. To Add a **Regex** Classifier:
 - Fill out the **Profiling Regex** field (mandatory), then click **Add**. For the rest of the fields, if left unfilled/unselected, the default value will be selected and added to the table.

Add Classifier

Classifier Name: Domain: Classifier Framework:

Description (Optional):

Regular Expressions

Match Strength	Checksum Type	Case Sensitive	Partial Match	Actions
Profiling Regex : \d(1,2)				
100%	NONE	No	Yes	⋮

Add Regex Configuration

Profiling Regex:

Data Clean Regex (Optional):

Checksum Type (Optional):

Reject Strength (Optional):

6. To Add a **Data Type** Classifier:

- Choose a **Data Type** from the dropdown (mandatory), then click **Add**. For the rest of the fields, if left unfilled/unselected, the default value will be selected and added to the table.

Add Classifier ✕

Classifier Name Domain Classifier Framework

Description (Optional)

Data Types

Data Type	Minimum Length	Actions
String	2	⋮
Number	2	⋮

Add Data Type Configuration

Data Type

Minimum Length (Optional)

Include auto-incrementing columns when searching the database for this domain

- 7. To Add a **List** Classifier:
 - Upload a **File** (mandatory), then click **Add**. For the rest of the fields, if left unfilled/unselected, the default value will be selected and added to the table.

Add Classifier ✕

Classifier Name
Test

Domain
AGE ✕ ▾

Classifier Framework
List ▾

Description (Optional) //

Lists

Name	Match Strength	Actions

Add List Configuration

Choose File

Match Strength % (Optional)

Cancel
Add

Reject Strength (Optional)

Tokenize input values (Optional)
Tokenization delimiter(s) (Optional)

Cancel
Save

8. To Add a **Path** Classifier:

- Add a **Field Value** (mandatory), then click **Add**. For the rest of the fields, if left unfilled/unselected, the default value will be selected and added to the table.

Add Classifier

Classifier Name: test Domain: AGE Classifier Framework: Path

Description (Optional):

Paths

Match Strength	Match Type	Case Sensitive	Partial Match	Actions
Field Value : (?i)(?>("[_-])(age)[_-]?)(group grp number num nbr no)?(\$[_-])				
50%	REGEX	No	Yes	...

Reject Strength (Optional): 0

Add Path Configuration

Field Value: [input]

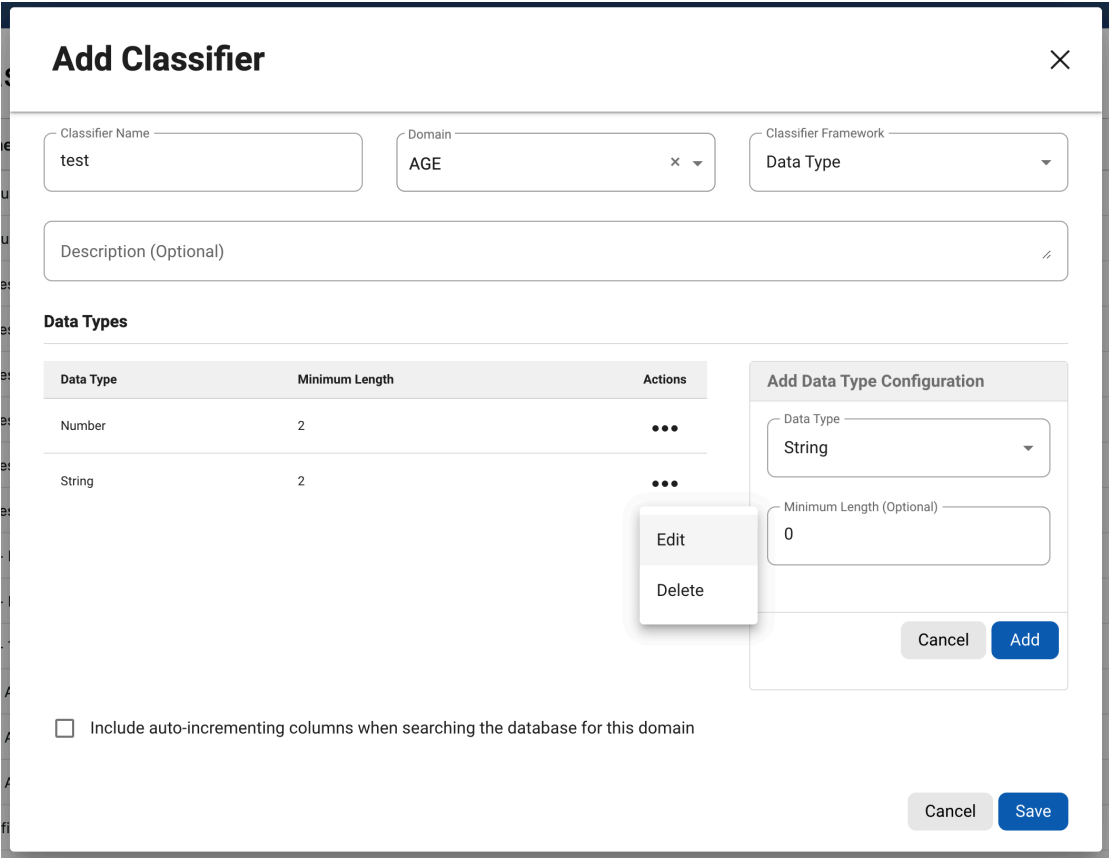
Match Type: [dropdown: Regex]

Parent Value (Optional): [input]

[Cancel] [Add]

[Cancel] [Save]

9. It also allows in-place editing/deleting of configurations.
- To edit/delete, click the Actions button (...) next to the configuration.
 - Clicking Edit will take all the information from the corresponding row to the right side form, where it can then be modified.



- 10. Click **Save** to create the classifier. Users can add multiple configurations of the same framework type to a classifier.

7.8.2 Modifying Classifiers

Users can perform 3 types of action on this screen by clicking the (...) button to the right of the corresponding row under the **Actions** column.

Settings > Classifiers

Classifiers

+ Classifier

Name ↑	Domain	Owner	Type	Description	Actions
Account Number - Path	ACCOUNT_NO	System	PATH		...
Account Number - Type	ACCOUNT_NO	System	TYPE		...
Address Line 1 - Path	ADDRESS	System	PATH		...
Address Line 1 - Regex	ADDRESS	System	REGEX		...
Address Line 1 - Type	ADDRESS	System	TYPE		...
Address Line 2 - Path	ADDRESS_LINE2	System	PATH		...
Address Line 2 - Regex	ADDRESS_LINE2	System	REGEX		...
Address Line 2 - Type	ADDRESS_LINE2	System	TYPE		...
Age - Path	AGE	System	PATH		...

Displaying 1 to 10 of 165

7.8.2.1 View Classifier

Every field on the dialog will be disabled when the View action is selected.

View Classifier ✕

Classifier Name: Data Classifier - Email Domain: EMAIL Classifier Framework: Regex

Description (Optional): UMR9L97B

Regular Expressions

Match Strength	Checksum Type	Case Sensitive	Partial Match	Actions
Profiling Regex : [A-Z0-9.!#\$%&*+=?^_{}~\-\]+@[A-Z0-9-]+(?:\.[A-Z0-9-]+)*				
80%	NONE	No	Yes	⋮

[Close](#)

7.8.2.2 Edit Classifier

On clicking **Edit** action, a dialog will appear for editing the classifier. The Classifier name and framework type are not editable after creation hence it will be disabled for Editing.

Edit Classifier

Classifier Name: Data Classifier - Email Domain: EMAIL Classifier Framework: Regex

Description (Optional): UMR9L97B

Regular Expressions

Match Strength	Checksum Type	Case Sensitive	Partial Match	Actions
Profiling Regex : [A-Z0-9.!#\$%&*+=?^_{}~\]+@[A-Z0-9-]+(?:\.[A-Z0-9-]+)*				
80%	NONE	No	Yes	⋮

Add Regex Configuration

Profiling Regex

Data Clean Regex (Optional)

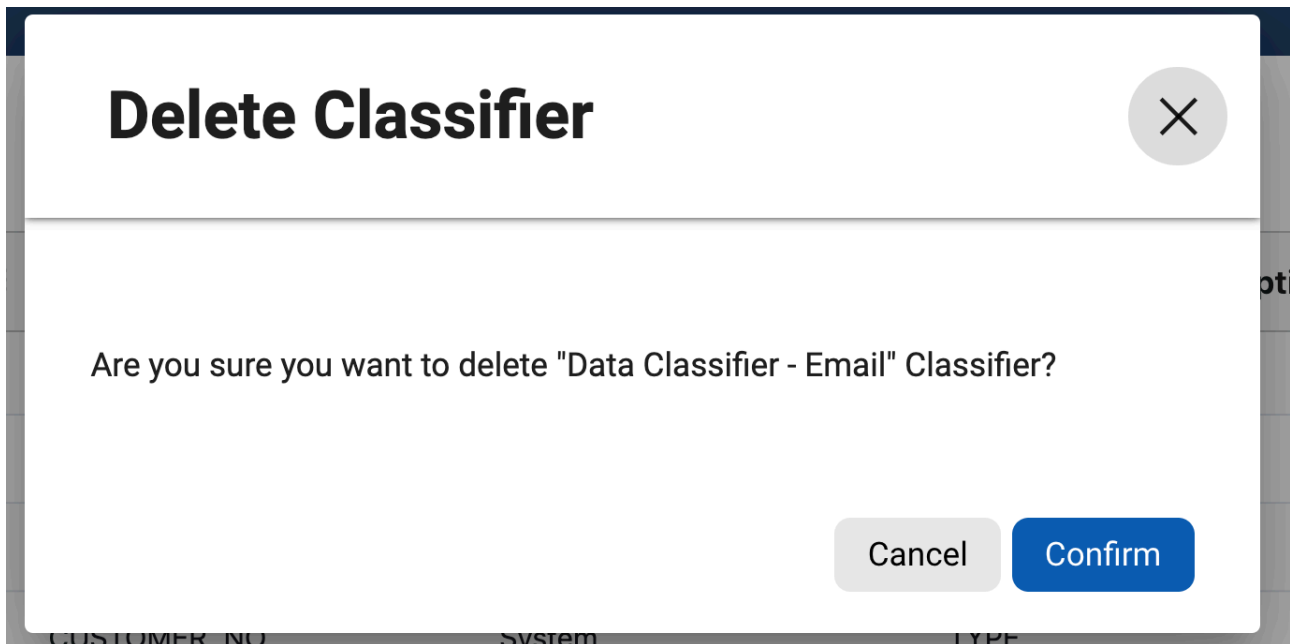
Checksum Type (Optional): None

Cancel Add

Cancel Save

7.8.2.3 Delete Classifier

Clicking Delete Action will prompt for confirmation. Click on **Confirm** to delete the classifier.



7.8.3 Configuration considerations for classifiers

Classifier design is more complex than search or type expressions because classifiers offer more flexibility in matching logic and configuration around match strength. Classifiers contain more configuration, typically encompassing all logic of the framework's type for a particular domain. For example, a legacy profile set might have three different column-level search expressions, but these would all be consolidated into a single PATH type classifier. Classifiers also add the notion of rejection strength, which allows the profiling logic to eliminate domains from consideration earlier in the profiling process.

7.8.3.1 Strength values

Classifiers use a scale of -1.0 to 1.0 to represent match and reject strength, which correspond to the -100 to 100 confidence values used in the UI. The product currently does not display confidence values for non-matches, so only values between 0 and 100 are typically visible.

A value of -1.0 indicates absolute rejection, and the domain in question is immediately eliminated from the set of possible matches. Similarly, a value of 1.0 indicates absolute confirmation, and the domain is assigned as a match without checking any other classifiers for that domain.

When multiple classifiers produce match or reject strength values, those results may be combined to get a final confidence. If the results conflict, with opposite signs, the result with the highest absolute value takes precedence. If the results have the same sign, the final result for that domain is a stronger match. The exact values and formula applied are under development and may change in the future. Currently, only the strongest column level result and strongest data level result are combined in this fashion.

7.8.3.1.1 Examples

1. A column named "ssn_present" matches a PATH classifier for the SSN domain with a match strength of 0.67. However, the column is boolean type and does not match the TYPE classifier for the SSN domain, which returns a -1.0 result. The verdict is -1.0 and the SSN domain is not assigned.
2. A column named "passport_no" contains 9-digit numeric values, which match the REGEX classifiers for both the SSN and PASSPORT_NO domains. Both REGEX classifiers return a confidence of 0.5 for this match. However, while the PATH classifier for the PASSPORT_NO domain matches and returns a match strength of 0.67, the PATH classifier for the SSN domain does not match, returning 0. The final confidence values are PASSPORT_NO at 0.84, and SSN at 0.5, so the PASSPORT_NO domain is the best match and the PASSPORT_NO domain and associated masking algorithm are assigned to the column.



Default assignment threshold

The ASDD profiler has a default minimum confidence value of 1, which means that any positive match, no matter how weak, will trigger an assignment. This is significantly different from the legacy profiler, which by default requires an 80% match for data level expressions. The application setting ASDD/DefaultAssignmentThreshold controls this value. For more details, refer to this section.

7.8.3.1.2 Choosing values for match strength

The match strength value (typically called **matchStrength** in the classifier configuration) reflects how confident the classifier is that a particular data element exclusively matches the associated domain. A match strength of 0.01 indicates that the data element may belong to the domain, but might also belong to any number of other domains or not be sensitive at all, while a value of 1.0 reflects absolute certainty that this data matches this domain and no other domain. A value of 0 provides no information.

Not all classifiers have a match strength greater than 0. One example of this is TYPE classifiers, which typically have a high reject strength, but 0 match strength (since it is impossible to match any of the built-in domains based on the data type of a column alone).

PATH classifiers built-in to the product typically have a match strength of 0.67, so in order for a REGEX or LIST classifier to override a PATH result, that classifier's match strength or reject strength would have to be higher than this value. This can help eliminate false positive results from the PATH classifiers, but be wary of the next recommendation before setting match strength to a high value.

When choosing match strength for REGEX classifiers, consider whether the pattern is unique to the type of sensitive data being detected. If it is not, it is safer to give a relatively low match strength in the range of 0.1 to 0.5, so that PATH level results can contribute information. Consider this example of REGEX detection of US Social Security numbers. These might be stored as a string value with a more distinct pattern like "001-23-4567", or simply as a 9-digit number "001234567". A 9-digit number might be any number of other numeric identifiers, like account number, passport number, a row identifier for rows in another table, etc. so the match strength for the [0-9]{9} regex should be quite low. The distinct text pattern with dashes has a much higher match strength since it is unlikely to be any other kind of information.

For LIST classifiers that utilize tokenization of inputs and reuse the same lists as other classifiers, consider lowering the match strength for each of these files used for tokenization. For example, list classifiers for First Name and Full Name may reference the same list of values, but during discovery with tokenization, the same values that match the First Name classifier will also match the Full Name classifier. Ideally, this configuration should allow the first name column to have a higher match to the First Name classifier.

Here are some additional tips for choosing match strength:

- Use a higher match strength for patterns that are more likely to be unique to the type of sensitive data being detected.
- Use a lower match strength for patterns that are less likely to be unique to the type of sensitive data being detected.
- Use a match strength of 0 for patterns that are not unique to the type of sensitive data being detected.

7.8.3.1.3 Choosing values for reject strength

The reject strength (typically called **rejectStrength** in the classifier configuration) value reflects how likely it is that a value matches the classifier's domain when the classifier does not match. If you are certain that your classifier configurations will match every possible value for the domain, the reject strength should be set to 1.0; however, this degree of certainty is rare. Similar to match strength, not all classifiers provide any rejection capability. This is true of PATH classifiers, for example, as we cannot rely on an unknown database schema to use predictable or human-readable names for columns.

The reject strength for classifiers applies any time there is no match. For example, if a REGEX classifier contains 4 regular expressions, each expression would be tested against the column data value, and if none match, the reject strength defines the result. For this reason, it can be useful to add a pattern that matches quite broadly, even if it's not particularly selective for the domain in question, with a low match strength. This prevents a full rejection for values that might match this classifier's domain as well as one or more other domains.

For LIST and REGEX classifiers where the set of patterns or list values is known to be only a subset of possible values for the domain, reject strength should be below 0.5 to allow column-level matches to take precedence, even if none of the data values match. For example, the value lists built-in for first and last name LIST classifier only contain English values, and names might be in other languages. These classifiers have "reject strength" set relatively low, to prevent the LIST classifiers from overriding the PATH classifier match if, for example, the column contains only Japanese names.

Here are some additional tips for choosing reject strength:

- Use a higher reject strength for classifiers that are more likely to match all values in the domain.
- Use a lower reject strength for classifiers that are less likely to match all values in the domain.
- Use a reject strength of 0 for classifiers that are not designed to match all values in the domain.

7.8.3.2 Regex configuration

The PATH and REGEX classifier types consume regular expressions using Java 8 regex syntax and matching logic. These classifiers have additional configuration options to control whether these patterns should match the entire input, and whether they are case-sensitive. For this reason, avoid using regex constructs such as "**^(pattern)\$**" for these purposes.

7.8.3.3 Type classifiers

The TYPE classifier framework uses the same four types as Type Expressions, as described in the [Managing expressions \(see page 630\)](#) section. However, the type-matching system is more versatile and provides better type identification across all database variants. This framework also supports the identification of auto-incrementing columns. The property **matchAutoIncrement** defines whether auto-incrementing columns should be considered for profiling (**true**) or completely skipped (**false**) for the domain. The default value is **false**.

7.8.3.4 Tokenization in list classifiers

List classifiers to identify complex data can be created by combining multiple sets of list files and enabling **tokenizeInput**. In such list classifiers, the input is tokenized to individual tokens which are split by delimiter values specified by the **tokenizationDelimiter** field. The default value for this field is a space " ", but can be a string of individual characters such as `-_.\n` where each character is a delimiter. The confidence value for each input value is calculated by summing the match strengths of all matching tokens divided by the total number of tokens in the input.

For example, to identify a FULL_NAME which is a combination of FIRST_NAME & LAST_NAME, it's unrealistic to create a list with all possible combinations of first and last names, but by providing separate lists and turning on tokenization, discovery can match each token of the input to the value lists.

7.9 Managing expressions

The **search** and **type** expressions define logic used by the legacy profiler to identify sensitive information. Refer to [Discovering Your Sensitive Data \(see page 526\)](#) for an overview of expressions and related concepts.

To view a list of all expressions, Navigate to **Settings > Expressions**. This screen has **Search Expression** and **Type Expression** tabs that select which type of expression is visible.

Settings > Expressions

Expressions

+ Search Expression

Search Expressions Type Expressions

Name ↑	Domain	Owner	Level	Actions
Account Number	ACCOUNT_NO	System	Column	...
Account_Number_V2	ACCOUNT_NO	System	Column	...
Address	ADDRESS	System	Column	...
Address_Line1_V2	ADDRESS	System	Column	...
Address Line 2 - after	ADDRESS_LINE2	System	Column	...
Address Line2 - before	ADDRESS_LINE2	System	Column	...
Address_Line2_V2	ADDRESS_LINE2	System	Column	...
Admission Date	DOB	System	Column	...

Displaying 1 to 8 of 96

7.9.1 Profile expressions

The **column** and **data** level expressions use regex text patterns on column meta-data and the data within the column, respectively, to identify sensitive data.



Column and data level expressions are case insensitive.

Type expressions can limit matches with column-level expressions by data type. A Type Expression consists of a user-chosen name, a data type, an optional minimum field length, and a domain to which the constraint applies. The supported data types are String, Number, Date, and Binary. Each type represents a number of native datatypes in the database.

- For **String** type, all character types supported by the database such as VARCHAR, NVARCHAR, CLOB, and NCLOB are considered String types for profiling. The minLength parameter considers the length specification of the column type, which may be characters or bytes. For example, Oracle supports VARCHAR2 fields measuring in either characters or in bytes. A VARCHAR2(20) column can hold 20 characters, whereas a VARCHAR2(20 BYTE) column can hold 20 bytes, which may be fewer than 20 characters if multibyte characters are present. A type expression with a minLength of 20 will match to both.
- For **Number** type, all numeric types are considered Number types by the profiling logic, including INTEGER, FLOAT, BIG_INTEGER, etc. The minLength parameter considers the number of base-10 digits supported by the type. For floating-point values, minLength refers to the integral part of the number.
- For **Date** type, the Date type includes all calendar date and date/time types, such as DATE and LOCAL_DATE_TIME types. The minLength parameter is not permitted for Date Type Expressions.

- For **Binary** type, the Binary type includes large object types such as BLOB and BINARY. The minLength parameter considers the maximum storage size of the column in bytes.

If there is more than one Type Expression assigned to a domain, then a column will match for the domain if the regular expression matches, and at least one of the type expressions match. For example, dates of birth are often stored in string types instead of dates, so you might have a string type expression and a date type expression assigned to the Date of Birth domain to allow columns of either type to match. Two Type Expressions of the same type cannot be assigned to the same domain in the same profile set. If there are no Type Expressions assigned to a domain, then the profile expression alone will determine matching without regard to data type.

Like Profile Expressions, Profile Type Expressions must be part of a profile set to be effective. Profile Type Expressions have no effect on Data Level Profiling.



Profile Type Expressions are only supported for database profiling. They have no effect on profiling of file data.



As of version 9.0, only Oracle and MSSQL Server are fully supported. On other platforms, Type Expressions may result in unexpected matches.

More information on expressions can be found at [Managing Search Expressions \(see page 632\)](#) and [Managing Type Expressions \(see page 639\)](#).

7.9.2 Managing search expressions

To view a list of all search expressions, Navigate to **Settings > Expressions**. By default, the **Search Expressions** tab is selected and all the search expressions will be displayed.

Settings > Expressions

Expressions

+ Search Expression

Search Expressions Type Expressions

Name ↑	Domain	Owner	Level	Actions
Account Number	ACCOUNT_NO	System	Column	...
Account_Number_V2	ACCOUNT_NO	System	Column	...
Address	ADDRESS	System	Column	...
Address_Line1_V2	ADDRESS	System	Column	...
Address Line 2 - after	ADDRESS_LINE2	System	Column	...
Address Line2 - before	ADDRESS_LINE2	System	Column	...
Address_Line2_V2	ADDRESS_LINE2	System	Column	...
Admission Date	DOB	System	Column	...

Displaying 1 to 8 of 96

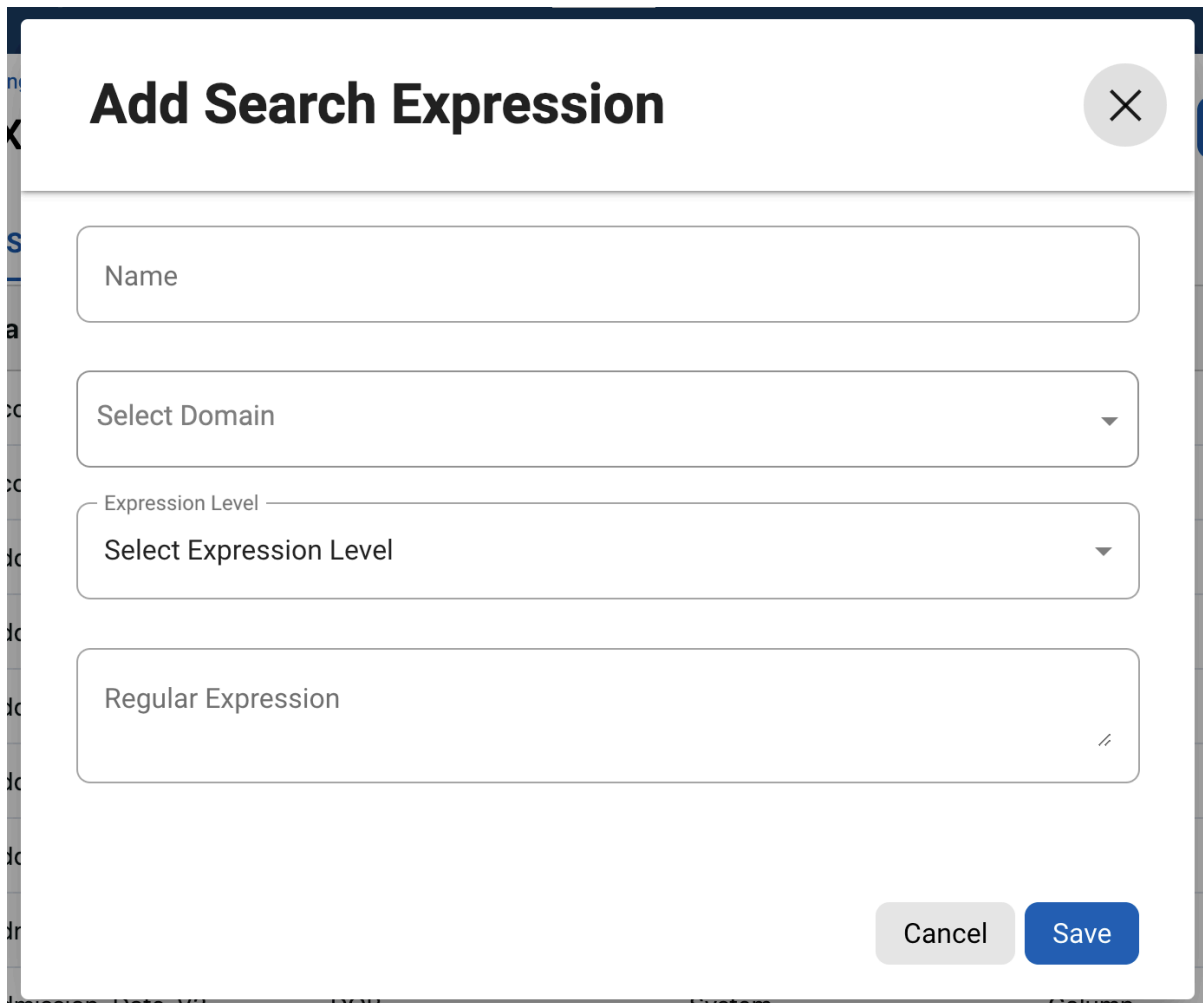
The search expressions on the screen can be filtered or sorted by the various informational fields by clicking on the respective field. More information on grid filtering and sorting can be found [here \(see page 232\)](#).



- Sortable fields are Name, Domain, and Level.
- Filterable fields are Name, Domain, Owner and Level.
- The `profile-expressions/search` endpoint also allow for searching and filtering of profile expressions. More information on syntax can be found at [API Calls for Searching and Filtering \(see page 952\)](#).

7.9.2.1 Adding new Search Expression

1. Click the **+ Search Expression** button from the top-right corner above the Search Expressions grid.



Add Search Expression

Name

Select Domain

Expression Level

Select Expression Level

Regular Expression

Cancel Save

2. Enter the **Expression Name**. The name used to identify this expression as part of a Profiler Set.
3. Select a Domain from the **Domain** dropdown.
 - Domains are used by Profiling jobs to determine the masking algorithm to apply to your sensitive data. When an Expression is matched, the Profiling job will associate the specified Domain with the sensitive data.
 - The Masking Engine comes out of the box with over 30 pre-defined domains. For more information on domains, refer to the [Managing Domains \(see page 611\)](#) article.
4. Select an **Expression Level** for the Expression
 - **Column Level:** To identify sensitive data based on column names.
 - **Data Level:** To identify sensitive data based on data values, not column names.
5. Enter the **Regular Expression**. The regular expression used to identify sensitive data.
6. Click **Save** to create the expression.

7.9.2.2 Modifying Search Expressions

Users can perform 4 types of action on this screen by clicking the (...) button to the right of the corresponding row under the **Actions** column.

Settings > Expressions

Expressions

+ Search Expression

Search Expressions Type Expressions

Name ↑	Domain	Owner	Level	Actions
Account Number	ACCOUNT_NO	System	Column	...
Account_Number_V2	ACCOUNT_NO	System	Column	...
Address	ADDRESS	System	Column	...
Address_Line1_V2	ADDRESS	System	Column	...
Address Line 2 - after	ADDRESS_LINE2	System	Column	...
Address Line2 - before	ADDRESS_LINE2	System	Column	...
Address_Line2_V2	ADDRESS_LINE2	System	Column	...
Admission Date	DOB	System	Column	...

Displaying 1 to 8 of 96

7.9.2.2.1 View Search Expression

Every field on the dialog will be disabled when the View action is selected.

View Search Expression ✕

Name
Account Number

Select Domain
ACCOUNT_NO

Expression Level
Column Level

Regular Expression
(?>(acc(oun|n)?t)?(num(ber)?|nbr|no)?)(?!w*(ID|type))

Close

7.9.2.2.2 Edit Search Expression

On Clicking **Edit** action, a dialog will appear for editing the search expression.

Edit Search Expression ✕

Name

Select Domain

Expression Level

Regular Expression

7.9.2.2.3 Duplicate Search Expression

On selecting a duplicate option, details will be pre-filled to the new search expression dialog, and the user can give a new name and duplicate the search expression.

Duplicate Search Expression ✕

Name

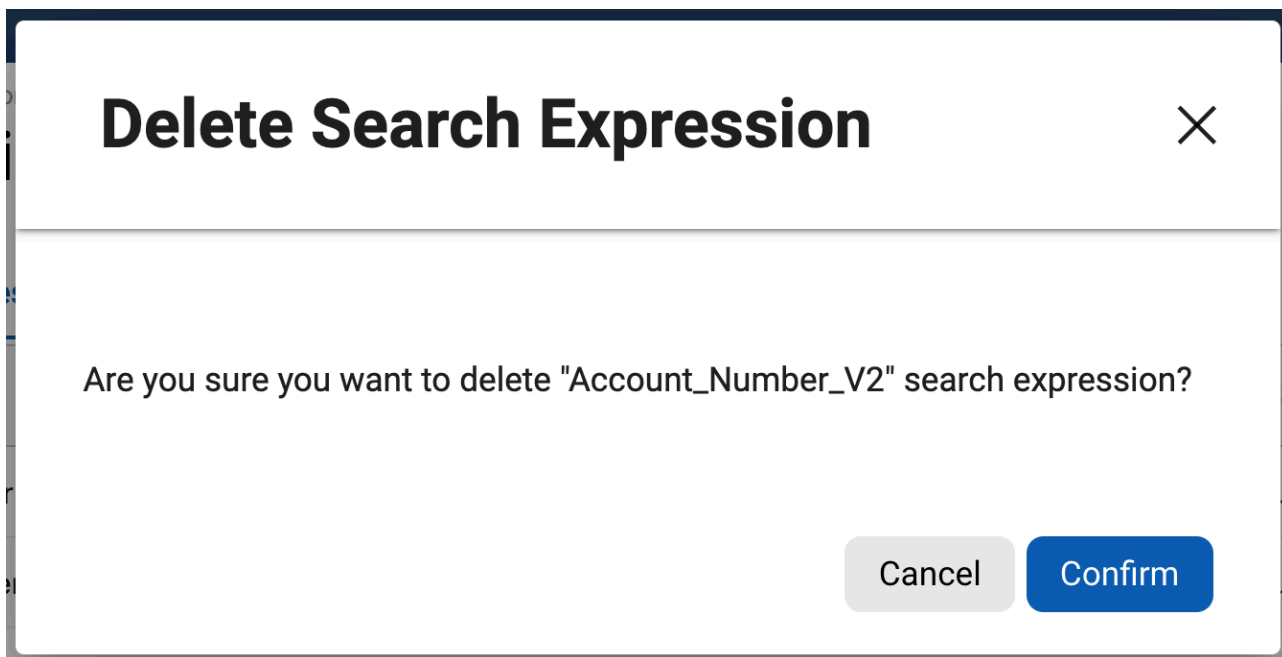
Select Domain

Expression Level

Regular Expression

7.9.2.2.4 Delete Search Expression

Clicking the Delete Action will prompt for confirmation. Click on **Confirm** to delete the expression. Deletion will be blocked if the expression is currently assigned to one or more profile sets.



7.9.3 Managing type expressions

To view a list of all type expressions, Navigate to **Settings > Expressions** and select the **Type Expressions** tab.

The screenshot shows the "Expressions" page in the Continuous Compliance application. The breadcrumb is "Settings > Expressions". The page title is "Expressions" with a "+ Type Expression" button. There are two tabs: "Search Expressions" and "Type Expressions" (which is selected). Below the tabs is a table with columns: Name, Domain, Owner, Level, and Actions. The table lists 8 type expressions. At the bottom right of the table area, it says "Displaying 1 to 8 of 49".

Name ↑	Domain	Owner	Level	Actions
ACCOUNT_NO_type_V2	ACCOUNT_NO	System	Type	...
ACCOUNT_NO_type_V2_2	ACCOUNT_NO	System	Type	...
ADDRESS_LINE2_type_V2	ADDRESS_LINE2	System	Type	...
ADDRESS_type_V2	ADDRESS	System	Type	...
BENEFICIARY_NO_type_V2	BENEFICIARY_NO	System	Type	...
BENEFICIARY_NO_type_V2_2	BENEFICIARY_NO	System	Type	...
BIOMETRIC_type_V2	BIOMETRIC	System	Type	...
BIOMETRIC_type_V2_2	BIOMETRIC	System	Type	...

The type expressions on the screen can be filtered or sorted by the various informational fields by clicking on the respective field. More information on grid filtering and sorting can be found [here](#) (see page 232).



- Sortable fields are Name and Domain.
- Filterable fields are Name, Domain, and Owner.
- The `profile-type-expressions/search` endpoint also allow for searching and filtering of profile-type expressions. More information on syntax can be found at [API Calls for Searching and Filtering](#) (see page 952).

7.9.3.1 Adding new Type Expression

1. Click the **+ Type Expression** button from the top-right corner above the Type Expressions grid.

Add Type Expression ✕

Name

Select Domain

Expression Level
Type Level

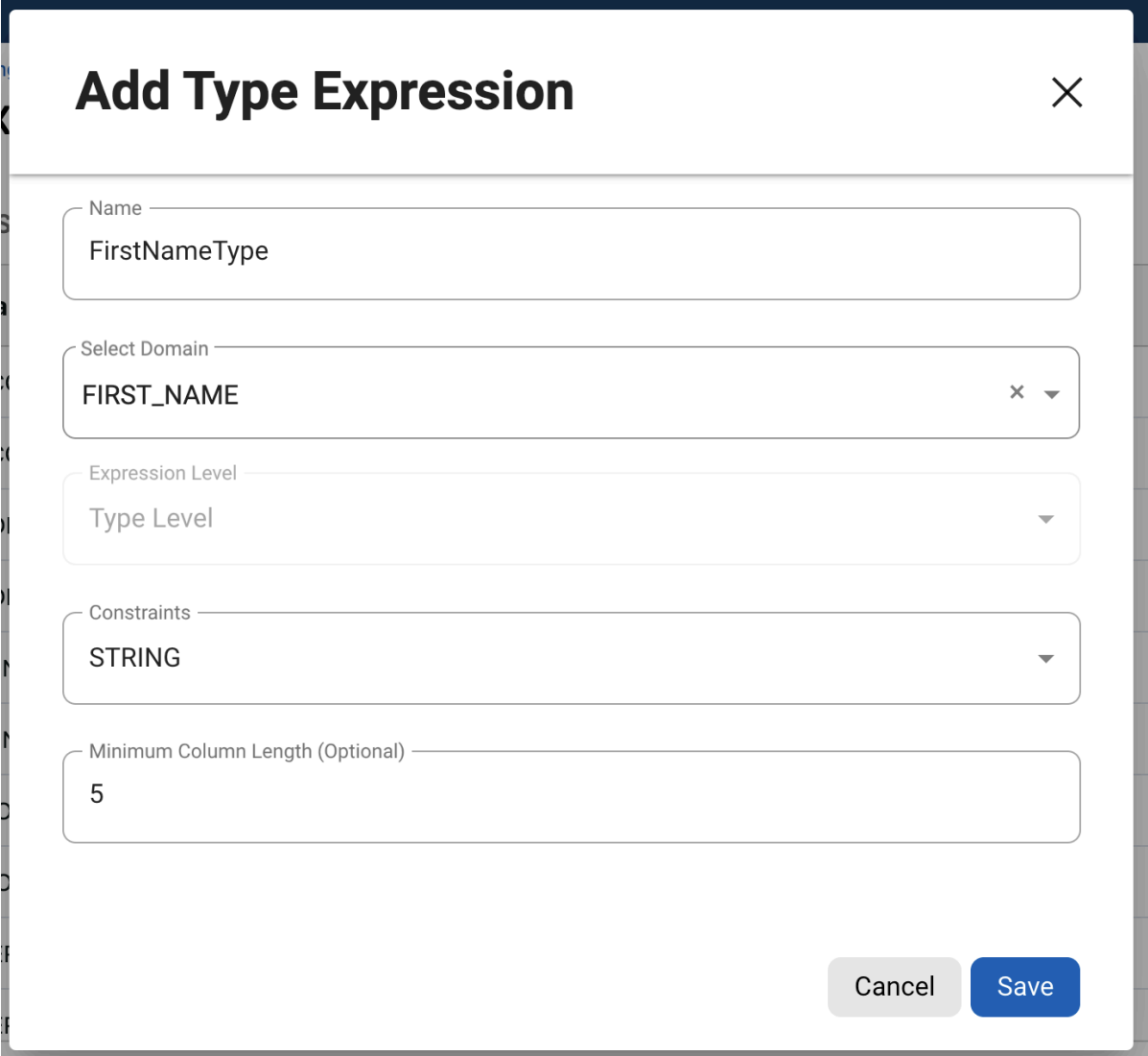
Constraints
Constraints

Minimum Column Length (Optional)

Cancel Save

2. Enter the **Expression Name** and select a **Domain**.

3. Select **Constraints** (Data Type) for the expression: String, Numeric, Binary, Date.
4. Set a **Minimum Column Length** for the data type if desired.
 - **Note:** Length constraints are not applied to large object types such as CLOBs and BLOBs.
5. For example, to ensure that column-level profiling only identifies a column with the FIRST_NAME domain, if the column is a string type and has a capacity of at least 5 characters, add the type constraint shown below.



Add Type Expression [X]

Name
FirstNameType

Select Domain
FIRST_NAME [X] ▼

Expression Level
Type Level ▼

Constraints
STRING ▼

Minimum Column Length (Optional)
5

Cancel Save

6. When you are finished, click **Save**.

7.9.3.2 Modifying Type Expressions

Users can perform 4 types of action on this screen by clicking the (...) button to the right of the corresponding row under the **Actions** column.

Settings > Expressions

Expressions

+ Type Expression

Search Expressions Type Expressions

Name ↑	Domain	Owner	Level	Actions
ACCOUNT_NO_type_V2	ACCOUNT_NO	System	Type	...
ACCOUNT_NO_type_V2_2	ACCOUNT_NO	System	Type	View Edit Duplicate Delete
ADDRESS_LINE2_type_V2	ADDRESS_LINE2	System	Type	...
ADDRESS_type_V2	ADDRESS	System	Type	...
BENEFICIARY_NO_type_V2	BENEFICIARY_NO	System	Type	...
BENEFICIARY_NO_type_V2...	BENEFICIARY_NO	System	Type	...
BIOMETRIC_type_V2	BIOMETRIC	System	Type	...
BIOMETRIC_type_V2_2	BIOMETRIC	System	Type	...

Displaying 1 to 8 of 49

7.9.3.2.1 View Type Expression

Every field on the dialog will be disabled when the View action is selected.

View Type Expression ✕

Name
ACCOUNT_NO_type_V2

Select Domain
ACCOUNT_NO

Expression Level
Type Level

Constraints
NUMBER

Minimum Column Length (Optional)
5

Close

7.9.3.2.2 Edit Type Expression

On Clicking **Edit** action, a dialog will appear for editing the type expression.

Edit Type Expression ✕

Name
ACCOUNT_NO_type_V2

Select Domain
ACCOUNT_NO ✕ ▼

Expression Level
Type Level ▼

Constraints
NUMBER ▼

Minimum Column Length (Optional)
5

Cancel Save

7.9.3.2.3 Duplicate Type Expression

On selecting a duplicate option, details will be pre-filled to the new type expression dialog, and the user can give a new name and duplicate the type expression.

Duplicate Type Expression ✕

Name

Select Domain
ACCOUNT_NO

Expression Level
Type Level

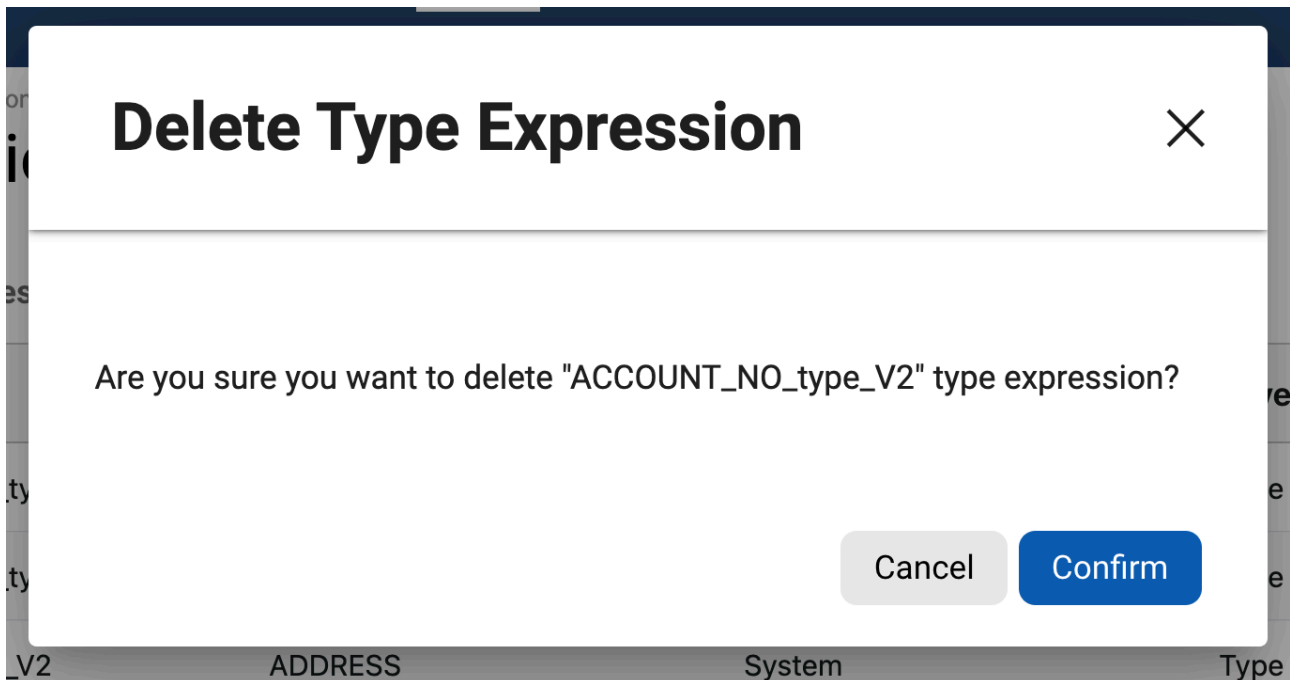
Constraints
NUMBER

Minimum Column Length (Optional)
5

Cancel Duplicate

7.9.3.2.4 Delete Type Expression

Clicking the Delete Action will prompt for confirmation. Click on **Confirm** to delete the expression. Deletion will be blocked if the expression is currently assigned to one or more profile sets.



7.10 ASDD profile set import and export

The ASDD Automatic Discovery set upgrade allows users to seamlessly create and update an ASDD profile set and its associated domains and classifiers. For usage instructions, see [API Calls for ASDD Profile Set Import and Export](#) (see page 941).

7.10.1 ASDD profile set import

ASDD profile set import facilitates a quicker and simpler option for the creation or update of the following masking objects:

1. A profile set
 - A new profile set can be created.
 - An existing profile set can be updated **if there are no active profiling jobs using the specified profile set.**
2. Domains
 - New domains can be created along with their assigned masking and tokenization algorithms.
 - Existing domains can have their assigned masking and tokenization algorithms updated.
3. Classifiers
 - New classifiers can be created.
 - Existing classifiers' configuration can be modified **if there are no active profiling jobs using a profile set that includes the specified classifiers.**

The profile set configuration is exported in the form of a dynamically generated JSON file, formatted just like the [ASDD Standard profile set JSON](#) (see page 529), including any files used by the classifiers that are a part of the profile set, all packaged in a single zip file.

7.10.1.1 Limitations

ASDD profile set import does not support importing multiple profile sets at a time; it only allows for a single profile set configuration to be installed at a time.

7.10.2 ASDD profile set export

Like the import function, the ASDD profile set export allows for an easy option to export a given profile set configuration in a zip file. This can be used to easily import the profile set onto another engine, or modifications can be made to the profile set JSON and then imported to the same engine to make updates to the given profile set.

7.11 Reporting profiling results

This section describes the different ways of sharing/exploring the results of a profiling job.

7.11.1 Job Execution page

After a Job has been started from the **Environment > Jobs** screen, clicking on the Job Name will result in the display of the profiling job details. After the Job has been completed, it will display the sensitive data findings on a table-column by table-column or file-field by file-field basis under **Profile Results**.

Environments > env_W3466FEU > prof_1MVP92EA
prof_1MVP92EA
Execution ID: 56

Profiling Completed

Report and Logs

Execution Summary

Execution ID
56

Environment
[env_W3466FEU](#)

Job Name
prof_1MVP92EA

Job Type
Profile

Job ID
49

Rule Set
[con_7Z1OKJH0](#)

Connection Type
Delimited

Source
-

Target
con_2MDCBE8W

Previous Run Time
-

Streams
1

Total # of Files
2

Files Profiled
2

Files to be Profiled
0

Success Start Time: 31 May 2024 11:48:31 IST End Time: 31 May 2024 11:48:43 IST Run Time: 00:00:12

- Initializing
- Collecting Configurations
- Preparing
- Starting
- Profiling Completed

Execution Components

ID	File Name	Progress	Status	Duration (HH:mm:ss)	Actions
105	delimited3xa0eiuw.txt	<div style="width: 100%;"></div>	100% ✓ SUCCEEDED	00:00:10	View
106	delimitede7qlfj4.txt	<div style="width: 100%;"></div>	100% ✓ SUCCEEDED	00:00:11	View

Displaying 1 to 2 of 2

Profile Results

File Name	Field Name	Algorithm	Domain
delimited3xa0eiuw.txt	DATA_00	dlpx-core:Phone Unique	TELEPHONE_NO
delimitede7qlfj4.txt	DATA_00	dlpx-core:Phone Unique	TELEPHONE_NO
delimitede7qlfj4.txt	DATA_01	dlpx-core:Email Unique	EMAIL
delimited3xa0eiuw.txt	DATA_01	dlpx-core:Email Unique	EMAIL
delimited3xa0eiuw.txt	FIRSTNAME_00	dlpx-core:FirstName	FIRST_NAME
delimitede7qlfj4.txt	FIRSTNAME_00	dlpx-core:FirstName	FIRST_NAME

Displaying 1 to 7 of 8

i When profiling multiple files (mainframe data sets), ASDD makes inventory assignments by averaging results across all files using the same format in the rule set.

7.11.2 PDF report

To retrieve a PDF report, click on the **Report and Logs** → **Job Report** at the top of the page.

The screenshot shows the Continuous Compliance web interface. At the top, there is a navigation bar with 'Environments', 'Monitor', 'Settings', 'Admin', and 'Audit'. Below this, the breadcrumb path is 'Environments > env_9BSR894V > prof_T5A7SLU6'. The main heading is 'prof_T5A7SLU6' with 'Execution ID: 62'. A green checkmark indicates 'Profiling Completed'. An 'Execution Summary' box on the left lists: Execution ID: 62, Environment: env_9BSR894V, Job Name: prof_T5A7SLU6. A central box shows the job status as 'Success' with a timeline from 3 Apr 2024 17:02:09 IST to 17:02:17 IST. A list of steps includes: Initializing, Collecting Configurations, Preparing, Starting, and Profiling Completed. A 'Report and Logs' dropdown menu is open, showing 'Execution Log' and 'Job Report' options.

The screenshot shows the 'Profiling Report' page. At the top, there is a header with the Delphix logo and the title 'Profiling Report'. Below this, there are two summary boxes: 'Job Profile' and 'Job Status'. The 'Job Profile' box contains: Name: prof_T5A7SLU6, Type: Profiling, Environment: env_9BSR894V, Schema: DLPXDEORA. The 'Job Status' box contains: Current Status: SUCCEEDED, Start Time: 2024-04-03 11:32:09,906, End Time: 2024-04-03 11:32:17,14. Below these boxes is a table with 7 columns: Table, Column, Domain, Algorithm, Data Type, Confidence, and Auto ID. The table contains 7 rows of profiling results.

Table	Column	Domain	Algorithm	Data Type	Confidence	Auto ID
testdata_PROFILE_00000	DATA_00	TELEPHONE_NO	dlpx-core:Phone Unique	VARCHAR2	80	true
testdata_PROFILE_00000	DATA_000000_00	EMAIL	dlpx-core:Email Unique	VARCHAR2	80	true
testdata_PROFILE_00001	DATA_00	TELEPHONE_NO	dlpx-core:Phone Unique	VARCHAR2	80	true
testdata_PROFILE_00001	DATA_01	TELEPHONE_NO	dlpx-core:Phone Unique	VARCHAR2	80	true
testdata_PROFILE_00001	DATA_000001_01	EMAIL	dlpx-core:Email Unique	VARCHAR2	80	true
testdata_PROFILE_00001	DATA_000001_00	EMAIL	dlpx-core:Email Unique	VARCHAR2	80	true
testdata_PROFILE_00002	DATA_000002_00	EMAIL	dlpx-core:Email Unique	VARCHAR2	80	true

7.11.3 Rule set page

Alternatively, after a job is completed successfully, the profiling results can be displayed through the **Rule Set** screen. The view differs by connection type as shown below.

7.11.3.1 Database Rule Set

Profiling results can be determined by examining the assigned **Algorithm** for the table(s) in the Rule Set.

Environments > ENV > test-profile-rs

test-profile-rs

All Logical Keys All Filters Actions

Table	Column	Data Type	Domain	Algorithm	File Format	Actions
▼ testdata_PROFILE... (7)						
	000000_00_FIRSTNAME	VARCHAR2 (64)	FIRST_NAME	dlpx-core:FirstName		
	000000_00_LASTNAME	VARCHAR2 (64)	LAST_NAME	dlpx-core:LastName		
	DATA_00	VARCHAR2 (100)	TELEPHONE_NO	dlpx-core:Phone Unique		
	DATA_000000_00	VARCHAR2 (64)	EMAIL	dlpx-core:Email Unique		
	ID Primary Key, Index	NUMBER (38)				
	UNMASKED_00	VARCHAR2 (64)				
	UNMASKED_000000_00	VARCHAR2 (64)				
▼ testdata_PROFIL... (13)						
	000001_00_FIRSTNAME	VARCHAR2 (64)	FIRST_NAME	dlpx-core:FirstName		
	000001_00_LASTNAME	VARCHAR2 (64)	LAST_NAME	dlpx-core:LastName		
	000001_01_FIRSTNAME	VARCHAR2 (64)	FIRST_NAME	dlpx-core:FirstName		

Displaying 1 to 12 of 46

7.11.3.2 File Rule Set

Profiling results can be determined by examining the assigned **Domain** and **Algorithm** for the files(s) in the Rule Set.

Environments > env_W3466FEU > con_7Z10KJH0

con_7Z10KJH0

File Format: WX70PK2U.fmt

Reset

Record Type	Name	Position	Domain	Algorithm	File Format	Automatic Updates	Actions
▼ All Records (6)							
	ID	1				Enabled	
	UNMASKED_00	2				Enabled	
	FIRSTNAME_00	3	FIRST_NAME	dlpx-core:FirstName		Enabled	
	LASTNAME_00	4	LAST_NAME	dlpx-core:LastName		Enabled	
	DATA_00	5	TELEPHONE_NO	dlpx-core:Phone Unique		Enabled	
	DATA_01	6	EMAIL	dlpx-core:Email Unique		Enabled	

7.11.3.3 Mainframe Rule Set

Profiling results can be determined by examining the assigned **Domain** and **Algorithm** for the files(s) in the Rule Set.

Information
Manage redefine conditions for the fields at the format level by selecting 'Edit Mainframe Format'. You can handle global masking assignments directly on this page.

Environments > env_QAXIIN8R > rs_MD0S420A

File Format: Demo.cbl

Name	Domain	Algorithm	Redefine Condition	Redefines	Actions
CS-CUSTOMER-RECORD (16)					
CUST-TYPE					
PERSON-DET (7) REDEFINED			[CUST-TYPE]="P"		
PERSON-FIRSTNAME	FIRST_NAME	dlpx-core:FirstName			
PERSON-LASTNAME	LAST_NAME	dlpx-core:LastName			
PERSON-ADDRESS1	ADDRESS	AddrLookup			
PERSON-CITY	CITY	USCitiesLookup			
PERSON-STATE					
PERSON-ZIP	BENEFICIARY_NO	dlpx-core:CM Digits			
PERSON-SSN					
COMP-DET (6) REDEF				PERSON-DET	
COMP-ENTITYNM					

7.11.4 CSV

To get a spreadsheet capturing the profiling results for the rule set, click on **Actions** → **Export Rule Set** from Actions near the top of the page, and a CSV file will be created.

Environments > env_XH3M04HZ > rs_XWAJ4M6K

All Logical Keys All Filters Actions

Table	Column	Data Type	Domain	Algorithm	File Format
testdata_PROFILE_000... (5)					
	DATA_00	VARCHAR2 (100)	TELEPHONE_NO	dlpx-core:Phone Unique	
	DATA_000000_00	VARCHAR2 (64)	EMAIL	dlpx-core:Email Unique	
	ID Primary Key, Index	NUMBER (38)			
	UNMASKED_00	VARCHAR2 (64)			
	UNMASKED_000000_00	VARCHAR2 (64)			
testdata_PROFILE_000... (9)					
	DATA_00	VARCHAR2 (100)	TELEPHONE_NO	dlpx-core:Phone Unique	
	DATA_000001_00	VARCHAR2 (64)	EMAIL	dlpx-core:Email Unique	
	DATA_000001_01	VARCHAR2 (64)	EMAIL	dlpx-core:Email Unique	
	DATA_01	VARCHAR2 (100)	TELEPHONE_NO	dlpx-core:Phone Unique	
	ID Primary Key, Index	NUMBER (38)			
	UNMASKED_00	VARCHAR2 (64)			

rs_XWAJ4M6K

Environment Name	Rule Set	Table Name	Type	Parent Column Name	Column Name	Data Type	Domain	Algorithm	Document Store Type	File Format	Is Masked	ID Method	Row Type	Date Format	Notes	Multi-Column	Logical
env_XH3M04HZ	rs_XWAJ4M6K	testdata_PROFILE_000000	-	-	DATA_00	VARCHAR2 (100)	TELEPHONE_NO	dipx-core:Phone Unique	-	-	TRUE	Auto	All Row	-	-	-	-
env_XH3M04HZ	rs_XWAJ4M6K	testdata_PROFILE_000000	-	-	DATA_000000_00	VARCHAR2 (64)	EMAIL	dipx-core:Email Unique	-	-	TRUE	Auto	All Row	-	-	-	-
env_XH3M04HZ	rs_XWAJ4M6K	testdata_PROFILE_000000	PK IX	-	ID	NUMBER (38)	-	-	-	-	FALSE	Auto	All Row	-	-	-	-
env_XH3M04HZ	rs_XWAJ4M6K	testdata_PROFILE_000000	-	-	UNMASKED_00	VARCHAR2 (64)	-	-	-	-	FALSE	Auto	All Row	-	-	-	-
env_XH3M04HZ	rs_XWAJ4M6K	testdata_PROFILE_000000	-	-	UNMASKED_000000_00	VARCHAR2 (64)	-	-	-	-	FALSE	Auto	All Row	-	-	-	-
env_XH3M04HZ	rs_XWAJ4M6K	testdata_PROFILE_000001	-	-	DATA_00	VARCHAR2 (100)	TELEPHONE_NO	dipx-core:Phone Unique	-	-	TRUE	Auto	All Row	-	-	-	-
env_XH3M04HZ	rs_XWAJ4M6K	testdata_PROFILE_000001	-	-	DATA_000001_00	VARCHAR2 (64)	EMAIL	dipx-core:Email Unique	-	-	TRUE	Auto	All Row	-	-	-	-
env_XH3M04HZ	rs_XWAJ4M6K	testdata_PROFILE_000001	-	-	DATA_000001_01	VARCHAR2 (64)	EMAIL	dipx-core:Email Unique	-	-	TRUE	Auto	All Row	-	-	-	-
env_XH3M04HZ	rs_XWAJ4M6K	testdata_PROFILE_000001	-	-	DATA_01	VARCHAR2 (100)	TELEPHONE_NO	dipx-core:Phone Unique	-	-	TRUE	Auto	All Row	-	-	-	-
env_XH3M04HZ	rs_XWAJ4M6K	testdata_PROFILE_000001	PK IX	-	ID	NUMBER (38)	-	-	-	-	FALSE	Auto	All Row	-	-	-	-
env_XH3M04HZ	rs_XWAJ4M6K	testdata_PROFILE_000001	-	-	UNMASKED_00	VARCHAR2 (64)	-	-	-	-	FALSE	Auto	All Row	-	-	-	-
env_XH3M04HZ	rs_XWAJ4M6K	testdata_PROFILE_000001	-	-	UNMASKED_000001_00	VARCHAR2 (64)	-	-	-	-	FALSE	Auto	All Row	-	-	-	-
env_XH3M04HZ	rs_XWAJ4M6K	testdata_PROFILE_000001	-	-	UNMASKED_000001_01	VARCHAR2 (64)	-	-	-	-	FALSE	Auto	All Row	-	-	-	-
env_XH3M04HZ	rs_XWAJ4M6K	testdata_PROFILE_000001	-	-	UNMASKED_01	VARCHAR2 (64)	-	-	-	-	FALSE	Auto	All Row	-	-	-	-

The spreadsheet can then be shared and manually modified to correct the sensitive data findings by:

1. Changing the **Is Masked**, **Algorithm**, and/or **Domains** fields for the respective Table/Column or File/Field in the CSV file accordingly.
2. Importing the modified spreadsheet by clicking on **Actions** → **Import Rule Set** near the top of the page and specifying the modified CSV file name.

7.11.5 API endpoint

Using the API endpoint `/profileResultDatabase/{executionId}`, profiling results can be retrieved by providing the executionId. This method is only for database connections and will not work with other connection types. Results will be returned in JSON format.

```
{
  "_pageInfo": {
    "numberOnPage": 4,
    "total": 4
  },
  "responseList": [
    {
      "columnMetadataId": 1,
      "columnName": "CITY",
      "tableName": "PROFILE_TEST",
      "domainName": "CITY",
      "algorithmName": "NullValueLookup",
      "dataType": "VARCHAR2",
      "isProfilerWritable": false
    },
    {
      "columnMetadataId": 2,
      "columnName": "COUNTRY",
      "tableName": "PROFILE_TEST",
      "dataType": "VARCHAR2",
      "isProfilerWritable": false
    },
    {
      "columnMetadataId": 3,
      "columnName": "DOB",
      "tableName": "PROFILE_TEST",
      "domainName": "DOB",

```

```

    "algorithmName": "DateShiftDiscrete",
    "dataType": "DATE",
    "confidence": 100,
    "isProfilerWritable": true
  },
  {
    "columnMetadataId": 4,
    "columnName": "ADDRESS",
    "tableName": "PROFILE_TEST",
    "domainName": "ADDRESS",
    "algorithmName": "AddrLookup",
    "dataType": "VARCHAR2",
    "confidence": 100,
    "isProfilerWritable": true
  }
]
}

```

7.12 ASDD features and support

The ASDD profiler was introduced in Continuous Compliance version 9.0.0.0 and represents the future direction for sensitive data discovery. It offers a number of advantages compared to the legacy profiler, but currently has some limitations as well.

The introduction of the ASDD profiler does not make any changes to the legacy profiler. Existing profiling jobs should continue to function as they have in the past.

The ASDD profiler currently supports the following:

- Databases
- Delimited files
- Fixed width files
- JSON files (not available for the legacy profiler)
- XML files
- Mainframe

7.12.1 ASDD features

- The ASDD profiler uses classifiers rather than search and type expressions. Classifiers support more features and configuration options than expressions.
 - The `LIST` classifier framework is new and has no equivalent functionality in the legacy profiler.
 - The `TYPE` classifier framework uses standard Java `SQLType` values to identify data types, which should provide broad support for type detection across all database variants.
 - The `PATH` classifier supports exact matching and can be configured to consider table name in addition to column name when matching.

- The **REGEX** classifier supports the following checksums for data-level recognition:
 - a. **LUHN** : Luhn checksum for credit card numbers.
 - b. **MOD10_ABA** : Modulus 10 checksum for ABA Routing numbers (USA).
 - c. **MOD11_NHS** : Modulus 11 checksum for 10-digit NHS (National Health Services) numbers.
 - d. **MOD11_TFN** : Modulus 11 checksum for 9-digit TFN (Australian Tax File Numbers).
 - e. **MOD97** : Modulus 97 checksum for IBAN (International Bank Account Numbers).
- The ASDD profiler provides better matching when the number of rows in a table is less than the target number of rows for profiling, and in general provides more nuanced confidence value in profiling results.
- The ASDD profiler attempts to retrieve more data values when a large fraction of the data values for a column are null or empty. The threshold to trigger an additional query is controlled by the application setting **ASDD/DefaultNullFilterThreshold** – more information can be found in the **Application settings** section of the [Masking API client](#)¹⁸⁷ page.
- The ASDD profiler supports statistical sampling for Oracle, SQL Server, and SAP ASE databases, so the data sampled will better reflect the full range of values for each column across the entire table.



Sample percentage

When data sampling is employed, the sample percentage is always set to 1% – if this percentage does not yield enough rows, the query is performed again without sampling.

- The **ASDD Standard**¹⁸⁸ profile set contains data level logic by default, allowing some columns containing sensitive information to be identified even if the column names are not meaningful.
 - New or improved **REGEX** classifiers for Zip Code and Email Address domains.
 - New **LIST** classifiers are present for First and Last Name, Full Name, US City, US State, and Country domains.
- Classifiers and profile sets using them may be exported and imported using the Engine Sync feature. Classifiers are included when the **Export Settings** action is performed from the **Environments** tab.
- Prior to the 22.0.0.0 release, an ASDD profile set assignment threshold was exclusively determined by the application setting for ASDD called **DefaultAssignmentThreshold**. Assignment threshold can now be set on a per profile set basis. See [Configuring profile sets](#)¹⁸⁹ for more information.

7.12.2 ASDD limitations

The primary limitation of the ASDD Profiler is that it is not yet supported for all connectors. The UI will report an error if the user attempts to save a job using an ASDD profile set with an unsupported connector.

Currently, the following limitations apply to the ASDD Profiler:

¹⁸⁷ <https://masking.delphix.com/docs/latest/continuous-compliance-api-client>

¹⁸⁸ <https://masking.delphix.com/docs/latest/asdd-standard-profile-set>

¹⁸⁹ <https://masking.delphix.com/docs/latest/configuring-profile-sets>

- Profiling XML or JSON documents stored in database columns are not supported.
- Discovery may fail or produce lower-quality results for some extended connectors due to known issues:
 - The SQL syntax used for column truncation is not compatible with all database variants. This is known to cause failures in discovery jobs for the Informix database.

8 Securing sensitive data

This section contains the following topics:

- [Algorithms](#) (see page 656)
- [Builtin Driver Supports](#) (see page 820)

8.1 Algorithms

8.1.1 Introduction to Masking Algorithms

This article provides a brief outline of the different available algorithm options, along with other general algorithm information. More specific algorithm details can be explored in the Out-Of-The-Box Algorithm Instances or Algorithm Frameworks sections.

An algorithm plugin can be configured through the graphical user interface by entering the plugin's required configuration in JSON format. For more information, visit the [General UI for Extended Algorithms](#)¹⁹⁰ page.

8.1.2 Algorithm options

8.1.2.1 Out-of-the-box algorithm instances

Out-of-the-box algorithm instances are pre-configured ready-to-use algorithms. The out-of-the-box algorithms with related frameworks can be customized using the corresponding extensible frameworks. For more information on algorithm instance extensibility, see the [Extensible Algorithms](#)¹⁹¹ page.

Algorithm Instances	Extensible?	Related Framework
dlpx-core:ABARoutingNumber SL (see page 687)	X	Secure Lookup ¹⁹²
AccNoLookup ¹⁹³	X	Secure Lookup ¹⁹⁴
AddrLookup ¹⁹⁵	X	Secure Lookup ¹⁹⁶

¹⁹⁰ <https://masking.delphix.com/docs/latest/general-ui-for-extended-algorithms>

¹⁹¹ <https://masking.delphix.com/docs/latest/algorithms-authoring-extensible-plugins>

¹⁹² <https://masking.delphix.com/docs/latest/secure-lookup-algorithm-frameworks>

¹⁹³ <https://masking.delphix.com/docs/latest/accnlookup>

¹⁹⁴ <https://masking.delphix.com/docs/latest/secure-lookup-algorithm-frameworks>

¹⁹⁵ <https://masking.delphix.com/docs/latest/addrlookup>

¹⁹⁶ <https://masking.delphix.com/docs/latest/secure-lookup-algorithm-frameworks>

Algorithm Instances	Extensible?	Related Framework
AddrLine2Lookup ¹⁹⁷	X	Secure Lookup ¹⁹⁸
dlpx-core:Age SL (see page 688)	X	Secure Lookup ¹⁹⁹
BusinessLegalEntityLookup ²⁰⁰	X	Secure Lookup ²⁰¹
dlpx-core:CM Alpha-Numeric ²⁰²	X	Character Mapping ²⁰³
dlpx-core:CM Digits ²⁰⁴	X	Character Mapping ²⁰⁵
dlpx-core:CM Numeric ²⁰⁶	X	
CommentLookup ²⁰⁷	X	Secure Lookup ²⁰⁸
Credit Card ²⁰⁹	X	Payment Card ²¹⁰
Date Shift Discrete ²¹¹	X	
Date Shift Fixed ²¹²	X	Date Shift ²¹³
Date Shift Variable ²¹⁴	X	

197 <https://masking.delphix.com/docs/latest/addrline2lookup>

198 <https://masking.delphix.com/docs/latest/secure-lookup-algorithm-frameworks>

199 <https://masking.delphix.com/docs/latest/secure-lookup-algorithm-frameworks>

200 <https://masking.delphix.com/docs/latest/businesslegalentitylookup>

201 <https://masking.delphix.com/docs/latest/secure-lookup-algorithm-frameworks>

202 <https://masking.delphix.com/docs/latest/dlpx-core-cm-alpha-numeric>

203 <https://masking.delphix.com/docs/latest/character-mapping-algorithm-frameworks>

204 <https://masking.delphix.com/docs/latest/dlpx-core-cm-digits>

205 <https://masking.delphix.com/docs/latest/character-mapping-algorithm-frameworks>

206 <https://masking.delphix.com/docs/latest/dlpx-core-cm-numeric>

207 <https://masking.delphix.com/docs/latest/commentlookup>

208 <https://masking.delphix.com/docs/latest/secure-lookup-algorithm-frameworks>

209 <https://masking.delphix.com/docs/latest/credit-card>

210 <https://masking.delphix.com/docs/latest/payment-card-algorithm-frameworks>

211 <https://masking.delphix.com/docs/latest/date-shift-discrete>

212 <https://masking.delphix.com/docs/latest/date-shift-fixed>

213 <https://masking.delphix.com/docs/latest/date-shift-algorithm-frameworks>

214 <https://masking.delphix.com/docs/latest/date-shift-variable>

Algorithm Instances	Extensible?	Related Framework
DrivingLicenseNoLookup ²¹⁵	X	Secure Lookup ²¹⁶
DummyHospitalNameLookup ²¹⁷	X	Secure Lookup ²¹⁸
EmailLookup ²¹⁹	X	Secure Lookup ²²⁰
dlpx-core:Email SL ²²¹	X	Email ²²²
dlpx-core:Email Unique ²²³	X	Email ²²⁴
dlpx-core:FirstName ²²⁵	X	Name ²²⁶
FirstNameLookup ²²⁷	X	Secure Lookup ²²⁸
dlpx-core:FullName ²²⁹	X	Full Name ²³⁰
FullNMLookup ²³¹	X	Secure Lookup ²³²
LastCommaFirstLookup ²³³	X	Secure Lookup ²³⁴
dlpx-core:LastName ²³⁵	X	Name ²³⁶

215 <https://masking.delphix.com/docs/latest/drivinglicensenolookup>

216 <https://masking.delphix.com/docs/latest/secure-lookup-algorithm-frameworks>

217 <https://masking.delphix.com/docs/latest/dummyhospitalnamelookup>

218 <https://masking.delphix.com/docs/latest/secure-lookup-algorithm-frameworks>

219 <https://masking.delphix.com/docs/latest/emaillookup>

220 <https://masking.delphix.com/docs/latest/secure-lookup-algorithm-frameworks>

221 <https://masking.delphix.com/docs/latest/dlpx-core-email-sl>

222 <https://masking.delphix.com/docs/latest/email-algorithm-frameworks>

223 <https://masking.delphix.com/docs/latest/dlpx-core-email-unique>

224 <https://masking.delphix.com/docs/latest/email-algorithm-frameworks>

225 <https://masking.delphix.com/docs/latest/dlpx-core-firstname>

226 <https://masking.delphix.com/docs/latest/name-algorithm-frameworks>

227 <https://masking.delphix.com/docs/latest/firstnamelookup>

228 <https://masking.delphix.com/docs/latest/secure-lookup-algorithm-frameworks>

229 <https://masking.delphix.com/docs/latest/dlpx-core-fullname>

230 <https://masking.delphix.com/docs/latest/full-name-algorithm-frameworks>

231 <https://masking.delphix.com/docs/latest/fullnmlookup>

232 <https://masking.delphix.com/docs/latest/secure-lookup-algorithm-frameworks>

233 <https://masking.delphix.com/docs/latest/lastcommafirstlookup>

234 <https://masking.delphix.com/docs/latest/secure-lookup-algorithm-frameworks>

235 <https://masking.delphix.com/docs/latest/dlpx-core-lastname>

236 <https://masking.delphix.com/docs/latest/name-algorithm-frameworks>

Algorithm Instances	Extensible?	Related Framework
LastNameLookup ²³⁷	X	Secure Lookup ²³⁸
dlpx-core:Lat_Long Coordinates (see page 681)	X	Regex Decompose ²³⁹
NullValueLookup ²⁴⁰	X	
dlpx-core:Phone Unique ²⁴¹	X	
dlpx-core:Phone US ²⁴²	X	
RandomValueLookup ²⁴³	X	Secure Lookup ²⁴⁴
dlpx-core:Redact Digits-Zero (see page 685)	X	
RepeatFirstDigit ²⁴⁵	X	
SchoolNameLookup ²⁴⁶	X	Secure Lookup ²⁴⁷
SecureShuffle ²⁴⁸	X	
dlpx-core:TimeRange (see page 702)	X	Segment Mapping ²⁴⁹
dlpx-core:SwiftCode SL (see page 699)	X	Secure Lookup ²⁵⁰
USCitiesLookup ²⁵¹	X	Secure Lookup ²⁵²

237 <https://masking.delphix.com/docs/latest/lastnamelookup>

238 <https://masking.delphix.com/docs/latest/secure-lookup-algorithm-frameworks>

239 <https://masking.delphix.com/docs/latest/regex-decompose-algorithm-frameworks>

240 <https://masking.delphix.com/docs/latest/nullvaluelookup>

241 <https://masking.delphix.com/docs/latest/dlpx-core-phone-unique>

242 <https://masking.delphix.com/docs/latest/dlpx-core-phone-us>

243 <https://masking.delphix.com/docs/latest/randomvaluelookup>

244 <https://masking.delphix.com/docs/latest/secure-lookup-algorithm-frameworks>

245 <https://masking.delphix.com/docs/latest/repeatfirstdigit>

246 <https://masking.delphix.com/docs/latest/schoolnamelookup>

247 <https://masking.delphix.com/docs/latest/secure-lookup-algorithm-frameworks>

248 <https://masking.delphix.com/docs/latest/secureshuffle>

249 <https://masking.delphix.com/docs/latest/segment-mapping-algorithm-frameworks>

250 <https://masking.delphix.com/docs/latest/secure-lookup-algorithm-frameworks>

251 <https://masking.delphix.com/docs/latest/uscitieslookup>

252 <https://masking.delphix.com/docs/latest/secure-lookup-algorithm-frameworks>

Algorithm Instances	Extensible?	Related Framework
USstatecodesLookup ²⁵³	X	Secure Lookup ²⁵⁴
USstatesLookup ²⁵⁵	X	Secure Lookup ²⁵⁶
WebURLsLookup ²⁵⁷	X	Secure Lookup ²⁵⁸

8.1.2.2 Algorithm frameworks

Algorithm frameworks allow for the creation of algorithm instances with a custom configuration. For more information on algorithm framework extensibility, see the [Extensible Algorithms](#)²⁵⁹ page. More information on multi-column algorithms can be found in the [Using Multi-Column Algorithms](#)²⁶⁰ page.

Algorithm Framework	Extensible?	Multi-Column?	Out of the Box Instances
Binary Lookup ²⁶¹	X		
Character Mapping ²⁶²	X		dlpx-core:CM Alpha-Numeric ²⁶³ dlpx-core:CM Digits ²⁶⁴
Character Replacemen (see page 712) ^t	X		
Data Cleansing ²⁶⁵	X		
Date Replacement ²⁶⁶	X		

²⁵³ <https://masking.delphix.com/docs/latest/usstatecodeslookup>

²⁵⁴ <https://masking.delphix.com/docs/latest/secure-lookup-algorithm-frameworks>

²⁵⁵ <https://masking.delphix.com/docs/latest/usstateslookup>

²⁵⁶ <https://masking.delphix.com/docs/latest/secure-lookup-algorithm-frameworks>

²⁵⁷ <https://masking.delphix.com/docs/latest/weburlslookup>

²⁵⁸ <https://masking.delphix.com/docs/latest/secure-lookup-algorithm-frameworks>

²⁵⁹ <https://masking.delphix.com/docs/latest/algorithms-authoring-extensible-plugins>

²⁶⁰ <https://masking.delphix.com/docs/latest/using-multi-column-algorithms>

²⁶¹ <https://masking.delphix.com/docs/latest/binary-lookup-algorithm-frameworks>

²⁶² <https://masking.delphix.com/docs/latest/character-mapping-algorithm-frameworks>

²⁶³ <https://masking.delphix.com/docs/latest/dlpx-core-cm-alpha-numeric>

²⁶⁴ <https://masking.delphix.com/docs/latest/dlpx-core-cm-digits>

²⁶⁵ <https://masking.delphix.com/docs/latest/data-cleansing-algorithm-frameworks>

²⁶⁶ <https://masking.delphix.com/docs/latest/date-replacement-algorithm-frameworks>

Algorithm Framework	Extensible?	Multi-Column?	Out of the Box Instances
Date Shift ²⁶⁷	X		Date Shift Fixed ²⁶⁸
Dependent Date Shift ²⁶⁹	X	X	
Email ²⁷⁰	X		dlpx-core:Email Unique ²⁷¹ dlpx-core:Email SL ²⁷²
Free Text Redaction ²⁷³	X		
Full Name ²⁷⁴	X		dlpx-core:FullName ²⁷⁵
Mapping ²⁷⁶	X		
Min Max ²⁷⁷	X		
Name ²⁷⁸	X		dlpx-core:FirstName ²⁷⁹ dlpx-core:LastName ²⁸⁰
Numeric Expression ²⁸¹	X		
Payment Card ²⁸²	X		Credit Card ²⁸³
Regex Decompose ²⁸⁴	X		dlpx-core:Lat_Long Coordinates (see page 681)

²⁶⁷ <https://masking.delphix.com/docs/latest/date-shift-algorithm-frameworks>

²⁶⁸ <https://masking.delphix.com/docs/latest/date-shift-fixed>

²⁶⁹ <https://masking.delphix.com/docs/latest/dependent-date-shift-algorithm-frameworks>

²⁷⁰ <https://masking.delphix.com/docs/latest/email-algorithm-frameworks>

²⁷¹ <https://masking.delphix.com/docs/latest/dlpx-core-email-unique>

²⁷² <https://masking.delphix.com/docs/latest/dlpx-core-email-sl>

²⁷³ <https://masking.delphix.com/docs/latest/free-text-redaction-algorithm-frameworks>

²⁷⁴ <https://masking.delphix.com/docs/latest/full-name-algorithm-frameworks>

²⁷⁵ <https://masking.delphix.com/docs/latest/dlpx-core-fullname>

²⁷⁶ <https://masking.delphix.com/docs/latest/mapping-algorithm-frameworks>

²⁷⁷ <https://masking.delphix.com/docs/latest/min-max-algorithm-frameworks>

²⁷⁸ <https://masking.delphix.com/docs/latest/name-algorithm-frameworks>

²⁷⁹ <https://masking.delphix.com/docs/latest/dlpx-core-firstname>

²⁸⁰ <https://masking.delphix.com/docs/latest/dlpx-core-lastname>

²⁸¹ <https://masking.delphix.com/docs/latest/numeric-expression-algorithm-frameworks>

²⁸² <https://masking.delphix.com/docs/latest/payment-card-algorithm-frameworks>

²⁸³ <https://masking.delphix.com/docs/latest/credit-card>

²⁸⁴ <https://masking.delphix.com/docs/latest/regex-decompose-algorithm-frameworks>

Algorithm Framework	Extensible?	Multi-Column?	Out of the Box Instances
Secure Lookup ²⁸⁵	X		See the Secure Lookup (out of the box algorithm instances) ²⁸⁶ page.
Segment Mapping ²⁸⁷	X		<code>dlpx-core:TimeRange</code> (see page 702)
String Algorithm Chain ²⁸⁸	X		
Tokenization ²⁸⁹	X		

8.1.3 Configuring your algorithms

8.1.3.1 Algorithm settings

The **Settings > Algorithm** tab displays algorithm Names along with Type and Description. This is where you add (create) new algorithms. The default algorithms and any algorithms you have defined appear on this tab.

²⁸⁵ <https://masking.delphix.com/docs/latest/secure-lookup-algorithm-frameworks>

²⁸⁶ <https://masking.delphix.com/docs/latest/secure-lookup-out-of-the-box-algorithm-instances>

²⁸⁷ <https://masking.delphix.com/docs/latest/segment-mapping-algorithm-frameworks>

²⁸⁸ <https://masking.delphix.com/docs/latest/string-algorithm-chain-algorithm-frameworks>

²⁸⁹ <https://masking.delphix.com/docs/latest/tokenization-algorithm-frameworks>

Settings > Algorithms

Algorithms Extension Support Policy + Algorithm

Algorithm Name	Framework Name	Mask Type	Owner	Actions
01TEST	ChainedFramework1	STRING	admin	...
04	Email	STRING	admin	...
ACCOUNT_SL	Secure Lookup	STRING	System	...
ACCOUNT_TK	Tokenization	STRING	admin	...
ADDRESS LINE 2 SL	Secure Lookup	STRING	System	...
ADDRESS LINE SL	Secure Lookup	STRING	System	...
ALG_HN8HDOPC	Tokenization	STRING	admin	...
ALG_JRTIYFXL	Dependent Date Shift	GENERIC_DATA_ROW	admin	...
ALG_L7BV4W41	Tokenization	STRING	admin	...
ALG_NEHZLYY5	Dependent Date Shift	GENERIC_DATA_ROW	admin	...
ALG_PVQ7Z4W8	Tokenization	STRING	admin	...
ALG_S4K0LSRH	Tokenization	STRING	admin	...
ALG_TO9V2IYN	Tokenization	STRING	admin	...

Displaying 1 to 14 of 138

The algorithms on the screen can be filtered or sorted by the various informational fields by clicking on the respective fields. More information on grid filtering and sorting can be found [here \(see page 232\)](#).

• Sortable fields are Algorithm Name, Framework Name, and Mask Type.

• Filterable fields are Algorithm Name, Framework Name, Mask Type, and Owner.

8.1.3.2 Creating new algorithms

If none of the default algorithms meet your needs, you might want to create a new algorithm. An algorithm that you create is called a "user-defined algorithm".

Algorithm Frameworks give you the ability to quickly and easily define the algorithms you want, directly on the Settings page. After you create an algorithm, your algorithm will be available to all users.

To add an algorithm:

1. Click the **+ Algorithm** button from the top-right corner above the Algorithms grid.
2. Enter the Algorithm Name and description, Select Framework and click next.

Add Algorithm



Details

Specify the algorithm details.

Name: test_algo

Description:

Framework Name: Date Shift

Mask Type
LOCAL_DATE_TIME

Date Shift Framework: Allow users to configure a range in reference to the input that will output masked dates within the specified range. All outputs are formatted the same as the inputs, and all outputs are deterministically changed to a new value.

Cancel Back Next Save

3. Enter the configuration according to the Algorithm framework, and click next.

Add Algorithm



Configuration

Configure the algorithm.

Minimum Range: 1

Maximum Range: 4

Roll

Unit: Days

Date Shift Framework Description

Date Shift Framework: Allow users to configure a range in reference to the input that will output masked dates within the specified range. All outputs are formatted the same as the inputs, and all outputs are deterministically changed to a new value.

Cancel Back Next Save

4. View the summary on the last step to confirm the changes.

Add Algorithm



Navigation: Details, Configuration, **Summary**

Summary

View the algorithm details below. Click the Back button to make changes or click the Save button to save the algorithm configuration.

Details	Configuration
<p>Name test_algo</p> <p>Description Creating test date shift algo.</p> <p>Framework Name Date Shift</p> <p>Mask Type LOCAL_DATE_TIME</p>	<p>Minimum Range 1</p> <p>Maximum Range 4</p> <p>Roll false</p> <p>Unit Days</p>

Buttons: Cancel, Back, Next, **Save**

5. Click **Save**.

Algorithm creation and editing are types of Async Tasks. In some cases, it might take some time to create an Async task for creating or editing an algorithm. In that case, the User can select the **Run in background** option when it appears on the wizard which will close the wizard run the task in the background, and continue with other work.

Add Algorithm



Navigation: Details, Configuration, **Summary**

Summary

View the algorithm details below. Click the Back button to make changes or click the Save button to save the algorithm configuration.

Details	Configuration
<p>Name test</p> <p>Description</p> <p>Framework Name Date Shift</p> <p>Mask Type LOCAL_DATE_TIME</p>	<p>Minimum Range 1</p> <p>Maximum Range 4</p> <p>Roll false</p> <p>Unit Days</p>

Buttons: Cancel, Back, Next, Save

Run in background (button)

8.1.3.3 Editing algorithms

Administrators and users with EDIT Algorithm permission assigned in their Role may edit any user-defined algorithm on the system.

The following algorithm instances cannot be modified:

- Instances that ship with and are defined by the system
- Instances defined by algorithm plugins

For editing an algorithm, click the (...) button to the right of the corresponding row under the **Actions** column and select the **Edit** option.

The screenshot displays the 'Algorithms' management interface. At the top, there's a navigation bar with 'Settings > Algorithms' and a '+ Algorithm' button. Below this is a table listing various algorithms. The table has the following columns: Algorithm Name, Framework Name, Mask Type, Owner, and Actions. A dropdown menu is visible over the 'Actions' column for the 'ACCOUNT_SL' algorithm, showing 'Edit' and 'Delete' options. The table contains 13 rows of data, with the first row being 'ACCOUNT_SL' (Secure Lookup, STRING, System) and the last row being 'ALG_T7472U' (Secure Lookup, STRING, admin). At the bottom right of the table, it says 'Displaying 1 to 13 of 73'.

Algorithm Name	Framework Name	Mask Type	Owner	Actions
ACCOUNT_SL	Secure Lookup	STRING	System	...
ACCOUNT_TK	Tokenization	STRING	admin	...
ADDRESS LINE 2 SL	Secure Lookup	STRING	System	...
ADDRESS LINE SL	Secure Lookup	STRING	System	...
ALG_292BK018	MinMax Number	BIG_DECIMAL	admin	...
ALG_5U10QIC2	Secure Lookup	STRING	admin	...
ALG_F9638DUS	Tokenization	STRING	admin	...
ALG_JH3IEBM0	Tokenization	STRING	admin	...
ALG_KDGHVSM9	Tokenization	STRING	admin	...
ALG_MZ736AZE	Tokenization	STRING	admin	...
ALG_N1FN18OI	Tokenization	STRING	admin	...
ALG_NPY6V9HZ	MinMax Date	LOCAL_DATE_TIME	admin	...
ALG_T7472U	Secure Lookup	STRING	admin	...

Algorithm name and framework type cannot be changed after creation. Users can edit the configuration on the second step of the wizard and click save to modify changes.

Edit Algorithm



- Details
- Configuration
- Summary

Details

Specify the algorithm details.

Name: test

Description: dsfds

Framework Name: Date Shift

Mask Type
LOCAL_DATE_TIME

Date Shift Framework: Allow users to configure a range in reference to the input that will output masked dates within the specified range. All outputs are formatted the same as the inputs, and all outputs are deterministically changed to a new value.

Cancel Back Next Save

8.1.3.4 Viewing algorithms

Algorithm instances that are defined by the system and defined by algorithm plugins can only be viewed. Click the (...) button to the right of the corresponding row under the **Actions** column and select the **View** option.

Settings > Algorithms

Algorithms

Algorithm Name	Framework Name	Mask Type	Owner	Actions
ACCOUNT_SL	Secure Lookup	STRING	System	...
ACCOUNT_TK	Tokenization	STRING	admin	View
ADDRESS_LINE_2_SL	Secure Lookup	STRING	System	...
ADDRESS_LINE_SL	Secure Lookup	STRING	System	...
ALG_292BK018	MinMax Number	BIG_DECIMAL	admin	...
ALG_5U10QIC2	Secure Lookup	STRING	admin	...
ALG_F9638DUS	Tokenization	STRING	admin	...
ALG_JH3IEBM0	Tokenization	STRING	admin	...
ALG_KDGHVSM9	Tokenization	STRING	admin	...
ALG_MZ736AZE	Tokenization	STRING	admin	...
ALG_N1FN18OI	Tokenization	STRING	admin	...
ALG_NPY6V9HZ	MinMax Date	LOCAL_DATE_TIME	admin	...
ALG_T7472IU	Secure Lookup	STRING	admin	...

Displaying 1 to 13 of 73

View Algorithm ✕

- Details
- Configuration
- Summary**

Summary

View the algorithm details below. Click the Back button to make changes or click the Save button to save the algorithm configuration.

Details

Name
DateShiftDiscrete

Description
This algorithm masks all dates with the same year-month combination to the same day. A new day is returned for each year-month combination. This algorithm is deterministic based on an algorithm key.

Framework Name
Date Shift Discrete

Mask Type
LOCAL_DATE_TIME

Cancel Back Next Save

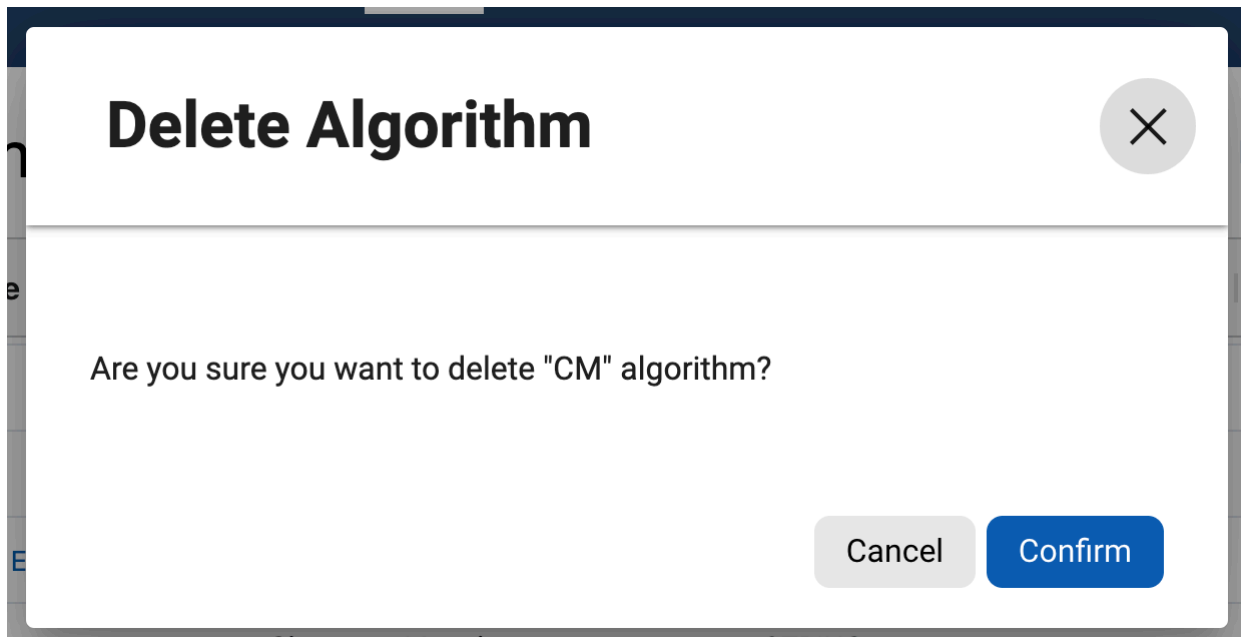
8.1.3.5 Deleting algorithms


Click the (...) button to the right of the corresponding row under the **Actions** column and select the **Delete** option to delete an algorithm.

The screenshot shows the 'Algorithms' management interface. The top navigation bar includes 'Environments', 'Monitor', 'Settings', 'Admin', and 'Audit'. The left sidebar lists various categories like 'Classifiers', 'Data Formats', 'Domains', 'Expressions', 'JDBC Drivers', and 'Profile Sets'. The main content area displays a table of algorithms. A dropdown menu is open over the 'Delete' option for the 'ALG_5U10QIC2' algorithm.

Algorithm Name	Framework Name	Mask Type	Owner	Actions
ACCOUNT_SL	Secure Lookup	STRING	System	...
ACCOUNT_TK	Tokenization	STRING	admin	...
ADDRESS_LINE_2_SL	Secure Lookup	STRING	System	...
ADDRESS_LINE_SL	Secure Lookup	STRING	System	...
ALG_292BK018	MinMax Number	BIG_DECIMAL	admin	...
ALG_5U10QIC2	Secure Lookup	STRING	admin	...
ALG_F9638DUS	Tokenization	STRING	admin	...
ALG_JH3IEBM0	Tokenization	STRING	admin	...
ALG_KDGHVSM9	Tokenization	STRING	admin	...
ALG_MZ736AZE	Tokenization	STRING	admin	...
ALG_N1FN18OI	Tokenization	STRING	admin	...
ALG_NPY6V9HZ	MinMax Date	LOCAL_DATE_TIME	admin	...
ALG_TB7AZ9U	Secure Lookup	STRING	admin	...

Displaying 1 to 13 of 73



 Algorithm instances that are defined by the system and defined by algorithm plugins cannot be deleted.

8.1.4 Algorithms Keys

Most masking algorithms include a key as part of their configuration. Changing this key changes the output of these algorithms. For example, if the [FirstNameLookup](#)²⁹⁰ algorithm masks "Michelle" to "Rachael," changing the algorithm's key might cause it to mask "Michelle" to "Ben".

An algorithm's key can be randomized using the following API endpoint:

```
PUT https://host.example.com/masking/api/algorithms/{algorithmName}/randomize-key
```

8.1.5 Multi-column algorithms

8.1.5.1 Overview

Multi-column algorithms are a special kind of algorithm that allows a single algorithm assignment to be made spanning multiple columns or fields in inventory. This allows coordinated masking of multiple fields - for example, masking two date-time values while preserving the interval between them.

²⁹⁰ <https://masking.delphix.com/docs/latest/firstnamelookup>

The [Dependent Date Shift](#)²⁹¹ algorithm is an example of a multi-column algorithm.

8.1.5.2 Usage

Each multi-column algorithm defines a set of **Logical Fields**; these logical fields are assigned to the actual fields or columns in inventory, defining how each value will be treated by the algorithm. A particular logical field may be *read-only*, indicating that it is considered as input but not masked by the multi-column algorithm, and/or *optional*, meaning the logical field is not required in order for the masking assignment to be complete. Furthermore, the **Algorithm Group** number allows a multi-column algorithm to be assigned multiple times in the same table or file-format, with the group number indicating which set(s) of logical fields should be processed together as a single assignment.

ⓘ Incomplete multi-column masking assignments in the inventory may not be detected until such time as a masking job is executed using that inventory. It is important to review each multi-column assignment carefully to ensure that for each *Algorithm Group*, each non-optional *Logical Field* is assigned to a column or field in the table or file-format.

8.1.6 Limitations

Multi-column algorithms may only be applied in inventories for data connectors where entire rows or records are processed as a unit.

Specific limitations:

- Multi-column algorithms are not supported for **XML** file masking.
- Multi-column algorithm assignments must be contained with a single Record Type for delimited and fixed-width files.
- Multi-column algorithm assignments must not cross redefines in VSAM copybooks.
- Multi-column algorithms may not be called by other algorithms through the algorithm chaining feature.

8.1.7 Algorithm frameworks overview

8.1.7.1 Choosing an algorithm framework

See the Algorithm Frameworks section for a detailed description of each Algorithm Framework. The algorithm framework you choose will depend on the format of the data and your internal data security guidelines.

²⁹¹ <https://masking.delphix.com/docs/latest/dependent-date-shift>

8.1.7.2 Choosing between character and segment mapping frameworks

The Character Mapping algorithm is intended to replace Segment Mapping for many use cases. That said, it does not replicate every feature of that algorithm, so the specific masking application will determine which one is appropriate.

Reasons to choose Character Mapping over Segment Mapping:

- Character Mapping can mask all characters in the first Unicode plane. Segment Mapping can only mask "[a-zA-Z]" + "[0-9]"
- Character Mapping automatically preserves all non-masked characters. Segment Mapping requires configuration of preserve characters. Character Mapping is much easier to use when the data is potentially "dirty" or not consistently formatted.
- Character Mapping can process preserve ranges in reverse, allowing the last positions of an input to be preserved when inputs have different lengths. Segment Mapping preserve ranges are always processed from the beginning of input.
- Character Mapping uses a more complex masking computation, so that every maskable position influences every other position in the masked value. Segment Mapping pre-computes the permutations for each segment independently.

Reasons to choose Segment Mapping over Character Mapping:

- Segment mapping can mask different parts of the input, determined by position, differently. Character Mapping always masks the same groups of characters regardless of position.
- Segment mapping can map inputs to different outputs at a position, like { A, B, C, D } -> { W, X, Y, Z } by specifying different *Input* and *Mask* values. This is not possible with Character Mapping.
- Segment mapping supports numeric segments, with up to 6-digit segments masked to a specific range. Character Mapping doesn't allow this kind of range limiting.

8.1.8 Out of the box algorithm instances

This section contains the following topics:


- [dlpx-core: CM Alpha-Numeric \(see page 672\)](#)
- [dlpx-core: CM Digits \(see page 673\)](#)
- [dlpx-core: CM Numeric \(see page 673\)](#)
- [Credit Card \(see page 674\)](#)
- [Date Shift Discrete \(see page 674\)](#)
- [Date Shift Fixed \(see page 675\)](#)
- [Date Shift Variable \(see page 675\)](#)
- [dlpx-core: Email SL \(see page 676\)](#)
- [dlpx-core: Email Unique \(see page 676\)](#)
- [dlpx-core: FirstName \(see page 677\)](#)
- [dlpx-core: FullName \(see page 678\)](#)
- [dlpx-core: LastName \(see page 680\)](#)
- [dlpx-core:Lat_Long Coordinates \(see page 681\)](#)
- [NullValueLookup \(see page 683\)](#)
- [dlpx-core: Phone Unique \(see page 684\)](#)

- [dlpx-core: Phone US](#) (see page 684)
- [dlpx-core:Redact Digits-Zero](#) (see page 685)
- [RepeatFirstDigit](#) (see page 685)
- [Secure Lookup \(Out of the box algorithm instances\)](#) (see page 686)
- [dlpx-core:TimeRange](#) (see page 702)
- [dlpx-core: IBAN](#) (see page 702)
- [SecureShuffle](#) (see page 704)

8.1.8.1 dlpx-core: CM Alpha-Numeric


Based > Extensible Algorithm Framework (see page 656)

The CM Alpha-Numeric algorithm is an instance of the [C²⁹²Character Mapping Algorithm Framework](#). (see page 707)

 CM Alpha-Numeric should only be used on non-numeric data types.


This algorithm masks all ASCII digit, lowercase, and uppercase characters, as well as some extended latin and cyrillic characters. Refer to the framework description for details of how masking is performed.

At least one character in the input must be masked, or Non-Conformant data handling will be triggered.

 The character mapping algorithm can be used for tokenization and reidentification jobs.

For example:

- "6379315274824970" → "0345698341375224"
- "ABCxyz123" → "HANwhp391"
- "Si" → "Cž"
- "999-12-3456." → "668-23-1138."
- "2000:a86f::1" → "3893:u55x::0"

 This algorithm may generate non-conformant data events.

8.1.8.2 dlp-core: CM Digits

| **Based >Extensible Algorithm Framework** (see page 656)

The CM Digits algorithm is an instance of the [Character Mapping Algorithm Framework](#). (see page 707)

This algorithm masks all ASCII digits. Refer to the framework description for details of how masking is performed. Be aware that this algorithm can produce value collisions when applied to Numeric data types. This is because leading zeros are not significant in numeric types, so while "7" → "8" and "304" → "008" may be different string results, when inserted into a numeric field, they represent the same value. If this behavior is undesirable, consider using the [CM Numeric](#) (see page 673) algorithm.

At least one character in the input must be masked, or Non-Conformant data handling will be triggered.

For example:

- "6379315274824970" → "8345698341375224"
- "99" → "05"
- "ABCxyz123" → "ABCxyz391"
- "0" → "6"



This algorithm may generate non-conformant data events.

8.1.8.3 dlp-core: CM Numeric

| **Based >Extensible Algorithm Framework** (see page 656)

The CM Numeric algorithm is an algorithm based on logic in the [Character Mapping Algorithm Framework](#). (see page 707)




CM Numeric algorithm should only be used for numeric data types.

The framework this algorithm is based on is not configurable and cannot be reused to create additional instances.

This algorithm masks all ASCII digit without the possibility of the first digit masking to "0". Leading and trailing zeros are preserved. The value "0" always masks to "0". Unlike the "CM digits" instance, the number of significant digits is always preserved for all numeric inputs.


Refer to the framework description for details of how masking is performed.

At least one character in the input must be masked, or Non-Conformant data handling will be triggered.

-  This algorithm can only be used for integer data in tokenization and reidentification jobs. Masking numbers with decimal points is not reversible.

For example:

- "6379315274824970" → "5210366768740261"
- "99" → "75"
- "000051.1230" → "000072.9040"
- "ABCxyz123" → "ABCxyz391"
- "0" → "0"

-  This algorithm may generate non-conformant data events.


8.1.8.4 Credit Card

Based > [Extensible Algorithm Framework](#) (see page 1007)

The Credit Card algorithm is an instance of the [Payment Card Algorithm Framework](#) (see page 786). The algorithm requires input values to have at least 8 digits in the character group [0-9]. If an input value has less than this, the algorithm will return an error. It preserves the first 6 digits of the input and requires at least one position to be masked for masking to be considered successful. The algorithm masks all subsequent digits by replacing them with a random value. All input characters that are not in the character group [0-9] are preserved. The algorithm maintains Luhn check validity through masking so input values with a valid Luhn check will mask to a value with a valid Luhn check. The out-of-the-box instance of this algorithm is called **CreditCard**.

For example:

- "6379315274824970" → "6379318341375224"
- "6379.3152.7482.4970" → "6379.3183.4137.5224"
- "abc5473defg04828hijkl0656253" → "abc5473defg04971hijkl6490341"

-  This algorithm may generate non-conformant data events.

8.1.8.5 Date Shift Discrete

The Date Shift Discrete algorithm masks all dates with the same year-month combination to the same day. A different day is returned for each year-month combination. As an example, any inputs with a year-month combination of February 2020 may return a day value of 23 while any inputs with a year-month combination

of January 2020 may return a day value of 5. All values of the input other than the day value are preserved. This algorithm is deterministic based on an algorithm key. The out-of-the-box instance of this algorithm is called **DateShiftDiscrete**.

For example:

- "1989-11-19 00:00:00" → "1989-11-30 00:00:00"
- "1989-12-19 04:15:00" → "1989-12-24 04:15:00"
- "2012-11-19 17:00:55" → "2012-11-08 17:00:55"
- "2012-11-09 00:23:59" → "2012-11-08 00:23:59"



This algorithm may generate non-conformant data events.

8.1.8.6 Date Shift Fixed

Based > Extensible Algorithm Framework (see page 656)

The Date Shift Fixed algorithm is an instance of the [Date Shift Algorithm Framework](#) (see page 720) masking the input to 5 days in the future with roll enabled so only the day of the month will change, all other units will remain the same. Dates at the end of the month will roll back to the beginning of the same month in the same year. The out-of-the-box instance of this algorithm is called **DateShiftFixed**.

For example:

- "2001-02-05 12:30:00" → "2001-02-10 12:30:00"
- "2001-02-27 15:45:00" → "2001-02-04 15:45:00"
- "2001-12-28 00:00:00" → "2001-12-02 00:00:00"



This algorithm may generate non-conformant data events.

8.1.8.7 Date Shift Variable

The Date Shift Variable algorithm returns a random date within the same month-year as the input date. Dates will not mask to the original input date. This algorithm may produce collisions. The out-of-the-box instance of this algorithm is called **DateShiftVariable**.

For example:

- "2019-02-05 10:00:00" → "2019-02-13 10:00:00"
- "2019-02-12 15:30:00" → "2019-02-13 15:30:00"
- "2019-02-27 00:45:30" → "2019-02-17 00:45:30"
- "2020-02-27 00:00:00" → "2020-02-22 00:00:00"



This algorithm may generate non-conformant data events.

8.1.8.8 dlp-core: Email SL

| *Based >* [Extensible Algorithm Framework](#) (see page 656)

The Email SL algorithm is an instance of the [Email Algorithm Framework](#) (see page 726). This algorithm splits the input on the '@' symbol. [Handling of malformed inputs](#) (see page 0) is detailed on the [Email Algorithm Framework](#) page (see page 726). This algorithm does not generate any non-conformant data events. The algorithm will split the input into two parts: **name** and **domain**. Name is the portion before the '@' symbol and domain is the portion after the '@' symbol.

A secure lookup is applied to the name portion of the input. The provided secure lookup file contains 20,000 unique lookup values in various formats. The following formats are used in the default lookup file:

- FirstName.LastName
- FirstName_LastName
- FirstInitial.LastName
- FirstNameLastName
- FirstNameLastInitialNumber

The domain portion is replaced by the fixed value "example.com". This value is a reserved domain with a valid DNS entry.

This algorithm is deterministic based on an algorithm key. It is possible that there may be collisions where two different values mask to the same value due to the nature of secure lookup. The out-of-the-box instance of this algorithm is called **dlp-core:Email SL**.

For example:

- "bob@gmail.com" → "E.Duboise@example.com"
- "bob@hotmail.com" → "E.Duboise@example.com"
- "alex@gmail.com" → "OrvinA436@example.com"
- "joe_123@yahoo.com" → "Amil.Steidinger@example.com"


8.1.8.9 dlp-core: Email Unique

| *Based >* [Extensible Algorithm Framework](#) (see page 656)

The Email Unique algorithm is an instance of the [Email Algorithm Framework](#). (see page 726) This algorithm splits the input on the '@' symbol. [Handling of malformed inputs](#) (see page 0) is detailed on the [Email Algorithm Framework](#) (see page 726) page. This algorithm does not generate any non-conformant data events. The algorithm will split the input into two parts: **name** and **domain**. Name is the portion before the '@' symbol and domain is the portion after the '@' symbol.

The name portion is masked by performing a SHA-256 hash of the entire input (including the domain). This means that inputs with the same name portion but different domain portions will mask to different values.

The hashed value is then encoded using Base32 encoding. The result of these transformations is the masked name portion.

 This instance produces masked name portions with lengths of 52 characters. The full masked length is 64 characters with the '@example.com' appended.

The domain portion is replaced by the fixed value "example.com". This value is a reserved domain with a valid DNS entry.

This algorithm is deterministic based on an algorithm key. This algorithm provides unique masked values for each input. The out-of-the-box instance of this algorithm is called **dlpx-core:Email Unique**.

For example:

- "bob@gmail.com" → "XF35TNMKPPTMQF4CX5264ZRXOMJJL2DQVE3KTZNIJ2NS6EUH7GLA@example.com"
- "bob@hotmail.com" → "M2U3LCC24MP5XDQ7DH4RSDW6QXCWRTSJVQF22C7IKBXDQ3LBM7NQ@example.com"
- "alex@gmail.com" → "CQKOVBP3VT42XHLBBUHEWIAJ26X3NROEBZHMSC7B4NFSZSTBIQ@example.com"
- "joe_123@yahoo.com" → "JTJNSLW4TWQ7VKG2KMRMMMH4M3FRIXUXFR7TIEL6VJR3G6AU2Q@example.com"

8.1.8.10 dlpx-core: FirstName

Based > Extensible Algorithm Framework (see page 1007)

The First Name algorithm is an instance of the [Name Algorithm Framework](#) (see page 773). The algorithm requires String type input values.

The expected format for the valid input contains at least one word, which consist of at least one non-whitespace character. If the input value does not match the expected format, the value will not be masked. I.e. if input contains null or empty string or white spaces only, then the algorithm returns unmasked input value.

No non-conformant data errors are thrown by that algorithm.

A single character is considered an abbreviation, i.e. it will be masked to a single character. Whether it is followed by the dot (.) or not. Words separated by the hyphen (-) are considered as a single word (even if divided from hyphen by spaces).

The default First Name instance is configured without particle files, so every input word is considered as a valid part of the name. The maximum number of names masked and returned is 2; additional names are dropped. Leading and trailing white spaces are not preserved.

For example:

Input	Masked Output
null	null
"" (empty string)	""
" " (white spaces only)	" "
& (single non alphanumeric character)	R
&?	Michael
M	S
M.	S.
Ann- Marie	Boris
Ann-Marie M.	Boris S.
Ann-Marie Jane Jill	Boris Shelby
von (particle not configured)	tim
von Jane	tim Shelby

8.1.8.11 dlp-core: FullName

Based >[Extensible Algorithm Framework](#) (see page 1007)

The Full Name algorithm is an instance of the [Full Name Algorithm Framework](#) (see page 735). The algorithm requires String type input values.

If input value is non-conformant (for example: null or white spaces), it is not masked. Words containing any character(s) are considered as a valid input and masked. Words separated by hyphen (-) are considered as a single word (even if divided from hyphen by spaces). No non-conformant data errors are thrown by that algorithm.

The default Full Name algorithm instance uses all default parameters, and chains "dlpx-core:FirstName" algorithm instance for first names masking, and "dlpx-core:LastName" for last name masking.

Below are few examples of the Full Name default algorithm instance masking:

Input	Masked Output
Manuel Maria Saxe-Coburgo-Gotha	Nimisha Kum Mcneish
Manuel - Boris Maria Saxe-Coburgo-Gotha	Simeon Kum Mcneish
Manuel Maria Saxe -Coburgo - Gotha	Nimisha Kum Mcneish
Manuel Maria de Saxe-Coburgo-Gotha	Nimisha Kum Mcneish
Manuel Maria de Saxe-Coburgo-Gotha (*)	Nimisha Kum Casteleyn
Manuel Maria - de ? Saxe-Coburgo-Gotha : #	Nimisha Muharrem E
Manuel Maria Saxe-Coburgo-Gotha (*)	Nimisha Kum Casteleyn
Mr. Manuel Maria de Saxe-Coburgo-Gotha #	Levar Nimisha E (dlpx-core:FirstName does not have any configured particles)
Saxe-Coburgo-Gotha, Manuel Maria	Mcneish, Nimisha Kum
saxe-coburgo-gotha, Manuel Maria	mcneish, Nimisha Kum
SAXE-COBURGO-GOTHA, MANUEL Maria	MCNEISH, NIMISHA Kum
Saxe-Coburgo-Gotha: M. M	Claudia T. S
M. G. Maria Saxe-Coburgo-Gotha	T. E. Mcneish
M M Saxe-Coburgo-Gotha	T T Mcneish
M M. Saxe-Coburgo-Gotha	T T. Mcneish
M M. S	T T. G
M M. S.	T T. G.

Input	Masked Output
m m. s.	t t. g.
Max	Grassi
Max	Grassi

8.1.8.12 dlpX-core: LastName

Based > [Extensible Algorithm Framework](#) (see page 1007)

The Last Name algorithm is an instance of the [Name Algorithm Framework](#)²⁹³. The algorithm requires String type input values.

The expected format for the valid input contains at least one word, which consists of at least one non-whitespace character. If the input value does not match the expected format, the value will not be masked. I.e. if input contains null or empty string or white spaces only then the algorithm returns the unmasked input value.

No non-conformant data errors are thrown by this algorithm.

Single-word input is not checked for configured particles. Single character is considered an abbreviation, i.e. it will be masked to a single character, whether it is followed by a dot (.) or not. Words separated by a hyphen (-) are considered as a single word (even if divided from the hyphen by spaces).

This algorithm uses the Name framework and is configured to mask only based on the first word. Certain punctuation marks, such as periods (.), are removed and do not affect the masked value; this behavior is not related to the removal or preservation of particles.

The default Last Name instance is configured with a particleToRemove file. After configured particles are removed from the input, a single word is masked and returned; additional words are dropped. Leading and trailing white spaces are not preserved.

For example:

Input	Masked Output
null	null
"" (empty string)	""
" " (white spaces only)	" "


²⁹³ <https://masking.delphix.com/docs/latest/name-algorithm-frameworks>


Input	Masked Output
? (single non-alphanumeric character)	K
Michael	M
S	S
S.	S.
Ann- Marie	Boris
von (particle) wilke	Froot
Smith	von Froot
Smith von Froot Weissman	Smith

8.1.8.13 dlp-core:Lat_Long Coordinates

Based > [Extensible Algorithm Framework](#) (see page 656)

The Lat_Long Coordinates algorithm is an instance of the [Regex Decompose Algorithm Framework](#). (see page 788)

 This algorithm recognizes and masks only the EPSG:4326 coordinate system (WGS84) containing latitude and longitude.

 Lat_Long Coordinates algorithm should only be used for string or numeric data types and does not support spatial or geometric column data-types.

The algorithm can mask latitude & longitude individually and also a combination of latitude and longitude in that order separated by delimiters- `comma` or `space` . The supported formats are:

Name	Format	Explanation	valid ranges	Valid examples that the algorithm can mask
DD - Decimal degrees	DD.DDDDDD D°<dir>	DD: Degrees	[-90, +90] for latitude [-180, +180] for longitude.	41.40338 41.40338°
		DDDDDD: Decimal degrees	[0000-99999999]	+39°98212343 -79.945790 89.1424342N 137.234543°E +39°98212343, -7 9.945790
DMM- Degrees, Decimal Minutes	DD°MM.MM MM'<dir>	DD: Degrees	[-90, +90] for latitude [-180, +180] for longitude.	41 24.2028 2 10.4418
		MM: Minutes	[0, 59]	65°24.342N
		MMMM': Decimal minutes	[000-9999]	+84°59.1234'
DMS- Degrees Minutes Seconds	DD°MM' SS.SS"<dir>	DD: Degrees	[-90, +90] for latitude [-180, +180] for longitude.	41°24'12.2"N 41°16'35.8"N
		MM': Minutes	[0, 59]	2°10'26"E
		SS: Seconds	[0, 59]	173°41'53.88"W
		SS": Decimal Seconds	[0, 99]	

<dir> can be one of the four directions - N, S, E & W

The algorithm will preserve the degrees (DD), the prefix direction (+/-), the direction character suffix (N, E, W, S) at the end, and any non-digits (symbols – ., °, ', ', ", "). It masks the decimal parts to valid decimals, minutes, and seconds (0-59) to valid minutes and seconds (0-59) respectively.

Valid input masking examples:

- +41.939390 → +41.684362

- 41°24'12.2"N → 41°16'35.8"N
- 41°24'12"N 2°10'26"E → 41°16'35"N 2°50'05"E
- 41 24.2028, 2 10.4418 → 41 16.3542, 2 50.5383
- 32°00'05"N 173°41'53"W → 32°15'07"N 173°30'02"W
- +41.939390 → +41.684362

Fallback action and non-conforming inputs:

The above coordinate formats of latitude and longitude are one of the most widely adopted coordinate systems. There are several other coordinate formats and spatial systems which can be found [online](#)²⁹⁴. When the current algorithm is used to mask such coordinates, they are treated as non-conforming inputs as they do not match any of the regex patterns defined in the algorithm. For all such data, a fallback action masks or redacts all the digits of the input value to "0" keeping the format, directions, degrees, minutes, seconds symbols, and any other characters intact. As a result, non-conformant events are not produced for this algorithm.

Non-conforming input masking examples:

- 41°65'35.8"N → 00°00'00.0"N
- 189.98212343°W → 000.00000000°W
- ABCxyz391 → ABCxyz000


To keep the algorithm regexes reasonably simple and readable (which also impacts performance), the outermost values of the allowed latitude and longitude ranges are also masked or redacted to 0.

- 90°00'00.0"N → 00°00'00.0"N
- 180°00'00.00"W → 000°00'00.00"W
- -90°00'00.0" → -00°00'00.0"

8.1.8.14 NullValueLookup

| Based > [Extensible Algorithm Framework](#) (see page 656)

This algorithm replaces the input with a null or empty value, depending on the context.

 The algorithm's name is chosen for backward compatibility only. It does not perform any kind of lookup and is not related to the Secure Lookup framework.

For example:

- "6379315274824970" → null
- "ABCxyz123" → null
- "Si" → null
- "999-12-3456." → null

²⁹⁴ <https://spatialreference.org/ref/epsg/>

- "2000:a86f::1" → null

8.1.8.15 dlpX-core: Phone Unique

| Based > [Extensible Algorithm Framework](#) (see page 656)

The Phone Unique algorithm masks the last 7 digits in the character group [0-9] with the hash value of the digits. All characters outside of this character group remain unmasked and are preserved in the masked value.

The maximum acceptable input length is 30 symbols, longer inputs will trigger non-conformant data handling. The input must contain at least one character in the character group [0-9], or non-conformant data handling will be triggered.

For example:

- "12-765" → "29-540"
- "(123)456-7890" → "(123)012-3901"
- "1(800) FLOWERS" → "2(746) FLOWERS"
- "+1-650-513-0514" → "+1-650-409-9747"
- "(512) 333-1234 ext 123" → "(512) 333-2905 ext 908"
- "CALL-ME-FLOWERS" → "CALL-ME-FLOWERS" (and generates a non-conformant data event)



This algorithm may generate non-conformant data events.

8.1.8.16 dlpX-core: Phone US

| Based > [Extensible Algorithm Framework](#) (see page 656)

The Phone US algorithm masks the last 4 digits in the character group [0-9] with the hash value of the digits and the 3 preceding digits are replaced with the value '555'. All characters outside of this character group remain unmasked and are preserved in the masked value.

The maximum acceptable input length is 30 symbols, longer inputs will trigger non-conformant data handling. The input must contain at least one character in the character group [0-9], or non-conformant data handling will be triggered.

For example:

- "12-765" → "58-504"
- "(123)456-7890" → "(123)555-3085"
- "1(800) FLOWERS" → "2(746) FLOWERS"
- "+1-650-513-0514" → "+1-650-555-9202"
- "(512) 333-1234 ext 123" → "(512) 333-5550 ext 497"
- "CALL-ME-FLOWERS" → "CALL-ME-FLOWERS" (and generates a non-conformant data event)

 This algorithm may generate non-conformant data events.

8.1.8.17 dlpx-core:Redact Digits-Zero

Based > Extensible Algorithm Framework (see page 656)

This algorithm replaces all the digits of the input with a 0.

For example:

- "98034209" → "000000000"
- "ABCxyz123" → "ABCxyz000"
- "Si" → "Si"
- "999-12-3456." → "000-00-0000."
- "" → ""


8.1.8.18 RepeatFirstDigit

Based > Extensible Algorithm Framework (see page 656)

This algorithm masks the "+4" component of a zip code by repeating its first digit four times, unless the first digit is zero, in which case '1' is repeated four times.

The input must contain a 4-character string "DDDD" (where each 'D' is a numeric digit). The following formats are valid inputs:

- 4-character string "DDDD" where 'D' is a digit
- 9-character string "ccccDDDD" where 'D' is a digit, and 'c' can be any character
- 10-character string "ccccccDDDD" where 'D' is a digit, and 'c' can be any character but must contain at least one hyphen or period
- 14-character string "ccccDDDDcccc" where 'D' is a digit, and 'c' can be any character

 14-character "ccccDDDDcccc" inputs will be truncated to "ccccDDDD"

For example:

- "6912" → "6666"
- "0123" → "1111"
- "941173564" → "941173333"
- "43556-9703" → "43556-9999"
- "009078377 SJPR" → "009078888"



This algorithm may generate non-conformant data events.

8.1.8.19 Secure Lookup (Out of the box algorithm instances)

This section covers the following topics:

- [ABARoutingNumber SL \(see page 687\)](#)
- [AccNoLookup \(see page 687\)](#)
- [AddrLookup \(see page 688\)](#)
- [AddrLine2Lookup \(see page 688\)](#)
- [Age SL \(see page 688\)](#)
- [BloodType SL \(see page 689\)](#)
- [BusinessLegalEntityLookup \(see page 689\)](#)
- [CommentLookup \(see page 690\)](#)
- [Department SL \(see page 690\)](#)
- [DrivingLicenseNoLookup \(see page 691\)](#)
- [DummyHospitalNameLookup \(see page 691\)](#)
- [EmailLookup \(see page 691\)](#)
- [Ethnicity SL \(see page 692\)](#)
- [FirstNameLookup \(see page 692\)](#)
- [FullNMLLookup \(see page 693\)](#)
- [Gender SL \(see page 694\)](#)
- [JobTitle SL \(see page 694\)](#)
- [Language SL \(see page 694\)](#)
- [LastCommaFirstLookup \(see page 695\)](#)
- [LastNameLookup \(see page 695\)](#)
- [MaritalStatus SL \(see page 696\)](#)
- [MedicalGenericDrug SL \(see page 696\)](#)
- [NationalOrigin SL \(see page 697\)](#)
- [PoliticalOrientation SL \(see page 697\)](#)
- [RandomValueLookup \(see page 698\)](#)
- [ReligiousOrientation SL \(see page 698\)](#)
- [SexualOrientation SL \(see page 698\)](#)
- [SchoolNameLookup \(see page 699\)](#)
- [SwiftCode SL \(see page 699\)](#)
- [USCitiesLookup \(see page 700\)](#)
- [USCountiesLookup \(see page 700\)](#)
- [USstatecodesLookup \(see page 701\)](#)
- [USstatesLookup \(see page 701\)](#)
- [WebURLsLookup \(see page 701\)](#)

8.1.8.19.1 ABARoutingNumber SL

| *Based >* [Extensible Algorithm Framework](#)²⁹⁵

The ABARoutingNumber SL algorithm is an instance of the [Secure Lookup Algorithm Framework](#) (see page 793).

This algorithm performs a lookup on the input value and returns a value from the provided lookup file. It is possible for this algorithm to produce the same output value for different input values. Inputs to this algorithm are case-sensitive, so two inputs with the same value in different cases may mask to different values. Leading and trailing whitespaces are preserved by this algorithm.

The lookup values for this algorithm are valid 9-digit routing numbers (ABA numbers/routing transit account numbers) of the banks in the USA.

For example:

- "271991207" → "053273981"
- "122242649" → "321171757"
- "081509106" → "112207209"
- "322275924" → "064208123"

8.1.8.19.2 AccNoLookup

| *Based >* [Extensible Algorithm Framework](#)²⁹⁶

The AccNoLookup algorithm is an instance of the [Secure Lookup Algorithm Framework](#). (see page 793)

This algorithm performs a lookup on the input value and returns a value from the provided lookup file. It is possible for this algorithm to produce the same output value for different input values. Inputs to this algorithm are case-sensitive so two inputs with the same value in different cases may mask to different values. Leading and trailing whitespaces are preserved by this algorithm.

The lookup values for this algorithm are 5 digit account numbers.

For example:

- "6379315274824970" → "64893"
- "ABCxyz123" → "72345"
- "ID3938491" → "72433"
- "999-12-3456" → "25326"
- "2000:a86f::1" → "86432"

²⁹⁵ <https://delphixdocs.atlassian.net/wiki/pages/createpage.action?fromPageId=205360380&linkCreation=true&spaceKey=CC&title=Introduction+to+masking+algorithms>

²⁹⁶ <https://delphixdocs.atlassian.net/wiki/pages/createpage.action?fromPageId=205360494&linkCreation=true&spaceKey=CC&title=Introduction+to+masking+algorithms>

8.1.8.19.3 AddrLookup

| *Based >* [Extensible Algorithm Framework](#)²⁹⁷

The AddrLookup algorithm is an instance of the [Secure Lookup Algorithm Framework](#). (see page 793)

This algorithm performs a lookup on the input value and returns a value from the provided lookup file. It is possible for this algorithm to produce the same output value for different input values. Inputs to this algorithm are case-sensitive so two inputs with the same value in different cases may mask to different values. Leading and trailing whitespaces are preserved by this algorithm.

The lookup values for this algorithm are line one address values.

For example:

- "49 Main St" → "55 BLUE DR"
- "1947 Highway 5" → "92 GREEN ST"
- "9 County Route 52.5" → "1049 ORANGE CIRCLE"

8.1.8.19.4 AddrLine2Lookup

| *Based >* [Extensible Algorithm Framework](#)²⁹⁸

The AddrLine2Lookup algorithm is an instance of the [Secure Lookup Algorithm Framework](#). (see page 793)

This algorithm performs a lookup on the input value and returns a value from the provided lookup file. It is possible for this algorithm to produce the same output value for different input values. Inputs to this algorithm are case-sensitive so two inputs with the same value in different cases may mask to different values. Leading and trailing whitespaces are preserved by this algorithm.

The lookup values for this algorithm are line two address values such as apartment number.

For example:

- "#483" → "UNIT 29"
- "APT 3D" → "P.O. BOX 934"
- "unit 13B" → "APARTMENT 1"

8.1.8.19.5 Age SL

| *Based >* [Extensible Algorithm Framework](#) (see page 656)

The AgeLookup algorithm is an instance of the [Secure Lookup Algorithm Framework](#) (see page 793).

This algorithm performs a lookup on the input value and returns a value from the provided lookup file. It is possible for this algorithm to produce the same output value for different input values. Leading and trailing whitespaces are preserved by this algorithm.

²⁹⁷ <https://delphixdocs.atlassian.net/wiki/pages/createpage.action?fromPageId=205360175&linkCreation=true&spaceKey=CC&title=Introduction+to+masking+algorithms>

²⁹⁸ <https://delphixdocs.atlassian.net/wiki/pages/createpage.action?fromPageId=205360118&linkCreation=true&spaceKey=CC&title=Introduction+to+masking+algorithms>

The lookup values for this algorithm are numbers ranging between 1-99, but with duplicated values between 20-55. This gives more probability of a lookup number value between 20-55.

For example:

- "55" → "24"
- "99" → "23"
- "123" → "45"
- "1" → "57"

8.1.8.19.6 BloodType SL

| Based > [Extensible Algorithm Framework](#)²⁹⁹

The BloodType SL algorithm is an instance of the [Secure Lookup Algorithm Framework](#) (see page 793).

This algorithm performs a lookup on the input value and returns a value from the provided lookup file. It is possible for this algorithm to produce the same output value for different input values. Inputs to this algorithm are case-sensitive so two inputs with the same value in different cases may mask to different values. Leading and trailing whitespaces are preserved by this algorithm.

The lookup values for this algorithm are blood types with one of the letters: A, B, AB, O followed by Rh-factor: '+' or '-'.

For example:

- "O positive" → "A-"
- "AB+" → "B-"
- "B-negative" → "AB+"

8.1.8.19.7 BusinessLegalEntityLookup

| Based > [Extensible Algorithm Framework](#)³⁰⁰

The BusinessLegalEntityLookup algorithm is an instance of the [Secure Lookup Algorithm Framework](#). (see page 793)

This algorithm performs a lookup on the input value and returns a value from the provided lookup file. It is possible for this algorithm to produce the same output value for different input values. Inputs to this algorithm are case-sensitive so two inputs with the same value in different cases may mask to different values. Leading and trailing whitespaces are preserved by this algorithm.

The lookup values for this algorithm are legal business names.

For example:

- "XYZ Corp." → "Boeing"
- "Alpha LLC" → "3M"
- "ABC Inc." → "Campbell Soup"

299 <https://delphixdocs.atlassian.net/wiki/spaces/CC/pages/9963078>

300 [https://delphixdocs.atlassian.net/wiki/pages/createpage.action?](https://delphixdocs.atlassian.net/wiki/pages/createpage.action?fromPageId=205361184&linkCreation=true&spaceKey=CC&title=Introduction+to+masking+algorithms)

[fromPageId=205361184&linkCreation=true&spaceKey=CC&title=Introduction+to+masking+algorithms](https://delphixdocs.atlassian.net/wiki/pages/createpage.action?fromPageId=205361184&linkCreation=true&spaceKey=CC&title=Introduction+to+masking+algorithms)

8.1.8.19.8 CommentLookup

| Based > [Extensible Algorithm Framework](#)³⁰¹

The CommentLookup algorithm is an instance of the [Secure Lookup Algorithm Framework](#) (see page 793).

This algorithm performs a lookup on the input value and returns a value from the provided lookup file. It is possible for this algorithm to produce the same output value for different input values. Inputs to this algorithm are case-sensitive so two inputs with the same value in different cases may mask to different values. Leading and trailing whitespaces are preserved by this algorithm. All non-empty and non-null inputs to this algorithm will mask to the same value.

The lookup value for this algorithm is a generic comment value.

For example:

- "6379315274824970" → "This data has been masked in all non-production environments as per Enterprise Information Security Policy(2013)."
- "ABCxyz123" → "This data has been masked in all non-production environments as per Enterprise Information Security Policy(2013)."
- "Si" → "This data has been masked in all non-production environments as per Enterprise Information Security Policy(2013)."
- "999-12-3456." → "This data has been masked in all non-production environments as per Enterprise Information Security Policy(2013)."
- "2000:a86f::1" → "This data has been masked in all non-production environments as per Enterprise Information Security Policy(2013)."

8.1.8.19.9 Department SL

| Based > [Extensible Algorithm Framework](#)³⁰²

The Department SL algorithm is an instance of the [Secure Lookup Algorithm Framework](#) (see page 793).

This algorithm performs a lookup on the input value and returns a value from the provided lookup file. It is possible for this algorithm to produce the same output value for different input values. Inputs to this algorithm are case-sensitive so two inputs with the same value in different cases may mask to different values. Leading and trailing whitespaces are preserved by this algorithm.

The lookup values for this algorithm are the common departments across all industries.

For example:

- "Product Management" → "Finance"
- "Loan Department" → "Marketing"
- "Outpatient Department" → "Sales"

301 <https://delphixdocs.atlassian.net/wiki/pages/createpage.action?fromPageId=205393748&linkCreation=true&spaceKey=CC&title=Introduction+to+masking+algorithms>

302 <https://delphixdocs.atlassian.net/wiki/spaces/CC/pages/9963078>

8.1.8.19.10 DrivingLicenseNoLookup

| **Based >Extensible Algorithm Framework**³⁰³

The DrivingLicenseNoLookup algorithm is an instance of the [Secure Lookup Algorithm Framework](#) (see page 793).

This algorithm performs a lookup on the input value and returns a value from the provided lookup file. It is possible for this algorithm to produce the same output value for different input values. Inputs to this algorithm are case-sensitive so two inputs with the same value in different cases may mask to different values. Leading and trailing whitespaces are preserved by this algorithm.

The lookup values for this algorithm are 9-digit driver's license IDs.

For example:

- "6379315274824970" → "865345234"
- "ABCxyz123" → "952731585"
- "US949382" → "164927562"

8.1.8.19.11 DummyHospitalNameLookup

| **Based >Extensible Algorithm Framework**³⁰⁴

The DummyHospitalNameLookup algorithm is an instance of the [Secure Lookup Algorithm Framework](#) (see page 793).

This algorithm performs a lookup on the input value and returns a value from the provided lookup file. It is possible for this algorithm to produce the same output value for different input values. Inputs to this algorithm are case-sensitive so two inputs with the same value in different cases may mask to different values. Leading and trailing whitespaces are preserved by this algorithm.

The lookup values for this algorithm are non-real hospital names.

For example:

- "Hospital 1" → "Community Hospital"
- "New York General Hospital" → "St. Patrick's Medical Center"
- "California Health Institute" → "Gotham City Mental Hospital"
- "Children's Hospital of Philadelphia" → "Hogwarts Medical Clinic"

8.1.8.19.12 EmailLookup

| **Based >Extensible Algorithm Framework**³⁰⁵

303 <https://delphixdocs.atlassian.net/wiki/pages/createpage.action?fromPageId=205393509&linkCreation=true&spaceKey=CC&title=Introduction+to+masking+algorithms>

304 <https://delphixdocs.atlassian.net/wiki/pages/createpage.action?fromPageId=205361053&linkCreation=true&spaceKey=CC&title=Introduction+to+masking+algorithms>

305 <https://delphixdocs.atlassian.net/wiki/pages/createpage.action?fromPageId=205427188&linkCreation=true&spaceKey=CC&title=Introduction+to+masking+algorithms>

The EmailLookup algorithm is an instance of the [Secure Lookup Algorithm Framework \(see page 793\)](#).
formation.

- A new email framework and two new email algorithm instances were introduced in version 6.0.9.0 and are the preferred methods for masking email values. See [Email \(see page 910\)](#), [Email SL \(see page 676\)](#), and [Email Unique \(see page 676\)](#) for more in

This algorithm performs a lookup on the input value and returns a value from the provided lookup file. It is possible for this algorithm to produce the same output value for different input values. Inputs to this algorithm are case-sensitive so two inputs with the same value in different cases may mask to different values. Leading and trailing whitespaces are preserved by this algorithm.

The lookup values for this algorithm are non-resolving email addresses.

For example:

- "bob@gmail.com" → "Andy.Samberg@nytimes.edu"
- "Albert_Einstein@nasa.gov" → "John.Smith@aol.gov"
- "abc123@delphix.com" → "Fred.James@yahoo.net"

8.1.8.19.13 Ethnicity SL

| **Based > Extensible Algorithm Framework**³⁰⁶

The Ethnicity SL algorithm is an instance of the [Secure Lookup Algorithm Framework \(see page 793\)](#).

This algorithm performs a lookup on the input value and returns a value from the provided lookup file. It is possible for this algorithm to produce the same output value for different input values. Inputs to this algorithm are case-sensitive so two inputs with the same value in different cases may mask to different values. Leading and trailing whitespaces are preserved by this algorithm.

The lookup values for this algorithm are widely used ethnicities within the USA.

For example:

- "Korean" → "Asian"
- "Filipino" → "Latino"
- "Asian" → "White"

8.1.8.19.14 FirstNameLookup

| **Based > Extensible Algorithm Framework** (see page 656)

The FirstNameLookup algorithm is an instance of the [Secure Lookup Algorithm Framework \(see page 793\)](#).

³⁰⁶ <https://delphixdocs.atlassian.net/wiki/spaces/CC/pages/9963078>

- A new name framework and a new first name algorithm instance was introduced in version 6.0.8.0 and are the preferred methods for masking name values. See [Name \(see page 773\)](#) and [FirstName \(see page 677\)](#) for more information.

This algorithm performs a lookup on the input value and returns a value from the provided lookup file. It is possible for this algorithm to produce the same output value for different input values. Inputs to this algorithm are case-sensitive so two inputs with the same value in different cases may mask to different values. Leading and trailing whitespaces are preserved by this algorithm.

The lookup values for this algorithm are first names.

For example:

- "lucas" → "Jacob"
- "Gabby Elizabeth" → "Dennis"
- "John Jacob Jingleheimer Schmidt" → "Ray"

8.1.8.19.15 FullNMLookup

Based > [Extensible Algorithm Framework](#)³⁰⁷

The FullNMLookup algorithm is an instance of the [Secure Lookup Algorithm Framework \(see page 793\)](#).

- A new full name framework and a new full name algorithm instance was introduced in version 6.0.8.0 and are the preferred methods for masking full name values. See [FullName \(see page 735\)\(framework\)](#) and [FullName \(see page 678\)\(instance\)](#) for more information.

This algorithm performs a lookup on the input value and returns a value from the provided lookup file. It is possible for this algorithm to produce the same output value for different input values. Inputs to this algorithm are case-sensitive so two inputs with the same value in different cases may mask to different values. Leading and trailing whitespaces are preserved by this algorithm.

The lookup values for this algorithm are full names (first & last name).

For example:

- "Harry Potter" → "John Wick"
- "joe" → "Carol Reed"
- "Robert Downey Jr." → "Aaron Burr"
- "Queen Elizabeth II" → "Ferris Bueller"

³⁰⁷ <https://delphixdocs.atlassian.net/wiki/pages/createpage.action?fromPageId=205329765&linkCreation=true&spaceKey=CC&title=Introduction+to+masking+algorithms>

8.1.8.19.16 Gender SL

| *Based >*[Extensible Algorithm Framework](#) (see page 1007)

The Gender SL algorithm is an instance of the [Secure Lookup Algorithm Framework](#) (see page 793) with the RANDOMIZE hash method.

This algorithm does not perform any lookup on the input value and returns a random value from the provided lookup file. It is possible for this algorithm to produce the same output value for different input values. Since the masked values are chosen at random from the lookup file, this algorithm does not maintain referential integrity. Leading and trailing whitespaces are preserved by this algorithm. The masked values maintain the statistical distribution (frequency of values) of the lookup file.

The lookup values for this algorithm are the most commonly used genders.

For example:

- "Female" → "Male"
- "Woman" → "Male"
- "Nonbinary" → "Female"
- "Male" → "Transgender"

8.1.8.19.17 JobTitle SL

| *Based >*[Extensible Algorithm Framework](#)³⁰⁸

The JobTitle SL algorithm is an instance of the [Secure Lookup Algorithm Framework](#) (see page 793).

This algorithm performs a lookup on the input value and returns a value from the provided lookup file. It is possible for this algorithm to produce the same output value for different input values. Inputs to this algorithm are case-sensitive so two inputs with the same value in different cases may mask to different values. Leading and trailing whitespaces are preserved by this algorithm.

The lookup values for this algorithm are the top 30 most commonly encountered job titles.

For example:

- "Zoologist" → "Teacher"
- "Recruiter" → "Sales Representative"
- "Gardener" → "Software Developer"

8.1.8.19.18 Language SL

| *Based >*[Extensible Algorithm Framework](#)³⁰⁹

The Language SL algorithm is an instance of the [Secure Lookup Algorithm Framework](#) (see page 793).

This algorithm performs a lookup on the input value and returns a value from the provided lookup file. It is possible for this algorithm to produce the same output value for different input values. Inputs to this

308 <https://delphixdocs.atlassian.net/wiki/spaces/CC/pages/9963078>

309 <https://delphixdocs.atlassian.net/wiki/spaces/CC/pages/9963078>

algorithm are case-sensitive so two inputs with the same value in different cases may mask to different values. Leading and trailing whitespaces are preserved by this algorithm.

The lookup values for this algorithm are the top 30 widely spoken languages across the world.

For example:

- "French" → "Chinese"
- "Turkish" → "Spanish"
- "Portuguese" → "Tamil"

8.1.8.19.19 LastCommaFirstLookup

| *Based >* [Extensible Algorithm Framework](#)³¹⁰

The LastCommaFirstLookup algorithm is an instance of the [Secure Lookup Algorithm Framework](#) (see page 793).



A new full name framework was introduced in version 6.0.8.0 and is the preferred methods for masking full name values. See [Full Name](#) (see page 735) for more information.

This algorithm performs a lookup on the input value and returns a value from the provided lookup file. It is possible for this algorithm to produce the same output value for different input values. Inputs to this algorithm are case-sensitive so two inputs with the same value in different cases may mask to different values. Leading and trailing whitespaces are preserved by this algorithm.

The lookup values for this algorithm are full names in the format Last Name, First Name.

For example:

- "Lincoln, Abe" → "Campbell, Allison"
- "George Washington" → "Douglas, Alfred"
- "teddy" → "Smith, Jack"

8.1.8.19.20 LastNameLookup

| *Based >* [Extensible Algorithm Framework](#)³¹¹

The LastNameLookup algorithm is an instance of the [Secure Lookup Algorithm Framework](#) (see page 793).

³¹⁰ <https://delphixdocs.atlassian.net/wiki/pages/createpage.action?fromPageId=205427851&linkCreation=true&spaceKey=CC&title=Introduction+to+masking+algorithms>

³¹¹ <https://delphixdocs.atlassian.net/wiki/pages/createpage.action?fromPageId=205394448&linkCreation=true&spaceKey=CC&title=Introduction+to+masking+algorithms>

- A new name framework and a new last name algorithm instance was introduced in version 6.0.8.0 and are the preferred methods for masking name values. See [Name](#) (see page 773) and [LastName](#) (see page 680) for more information.

This algorithm performs a lookup on the input value and returns a value from the provided lookup file. It is possible for this algorithm to produce the same output value for different input values. Inputs to this algorithm are case-sensitive so two inputs with the same value in different cases may mask to different values. Leading and trailing whitespaces are preserved by this algorithm.

The lookup values for this algorithm are last names.

For example:

- "Smith" → "Blair"
- "santa-cruz" → "Carney"
- "von Trapp" → "Washington"

8.1.8.19.21 MaritalStatus SL

| Based > [Extensible Algorithm Framework](#)³¹²

The MaritalStatus SL algorithm is an instance of the [Secure Lookup Algorithm Framework](#) (see page 793).

This algorithm performs a lookup on the input value and returns a value from the provided lookup file. It is possible for this algorithm to produce the same output value for different input values. Inputs to this algorithm are case-sensitive so two inputs with the same value in different cases may mask to different values. Leading and trailing whitespaces are preserved by this algorithm.

The lookup values for this algorithm are commonly used marital statuses.

For example:

- "Married" → "Single"
- "Single" → "Separated"
- "Partner" → "Divorced"

8.1.8.19.22 MedicalGenericDrug SL

| Based > [Extensible Algorithm Framework](#)³¹³

The MedicalGenericDrug SL algorithm is an instance of the [Secure Lookup Algorithm Framework](#) (see page 793).

This algorithm performs a lookup on the input value and returns a value from the provided lookup file. It is possible for this algorithm to produce the same output value for different input values. Inputs to this algorithm are case-sensitive so two inputs with the same value in different cases may mask to different values. Leading and trailing whitespaces are preserved by this algorithm.

312 <https://delphixdocs.atlassian.net/wiki/spaces/CC/pages/9963078>

313 <https://delphixdocs.atlassian.net/wiki/spaces/CC/pages/9963078>

The lookup values for this algorithm are the top 100 prescription drug - 'generic' names.

For example:

- "abilify" → "Ibuprofen"
- "Tylenol" → "Folic Acid"
- "oxycodone" → "Albuterol"

8.1.8.19.23 NationalOrigin SL

| *Based >* [Extensible Algorithm Framework](#)³¹⁴

The NationalOrigin SL algorithm is an instance of the [Secure Lookup Algorithm Framework](#) (see page 793).

This algorithm performs a lookup on the input value and returns a value from the provided lookup file. It is possible for this algorithm to produce the same output value for different input values. Inputs to this algorithm are case-sensitive so two inputs with the same value in different cases may mask to different values. Leading and trailing whitespaces are preserved by this algorithm.

The lookup values for this algorithm are the top 20 national origins by population worldwide.

For example:

- "French" → "Indian"
- "Egyptian" → "Thai"
- "German" → "Vietnamese"

8.1.8.19.24 PoliticalOrientation SL

| *Based >* [Extensible Algorithm Framework](#)³¹⁵

The PoliticalOrientation SL algorithm is an instance of the [Secure Lookup Algorithm Framework](#) (see page 793).

This algorithm performs a lookup on the input value and returns a value from the provided lookup file. It is possible for this algorithm to produce the same output value for different input values. Inputs to this algorithm are case-sensitive so two inputs with the same value in different cases may mask to different values. Leading and trailing whitespaces are preserved by this algorithm.

The lookup values for this algorithm are the top 10 political orientations worldwide.

For example:

- "Anarchist" → "Liberal"
- "Nationalist" → "Communist"
- "Right-wing populist" → "Progressive"

314 <https://delphixdocs.atlassian.net/wiki/spaces/CC/pages/9963078>

315 <https://delphixdocs.atlassian.net/wiki/spaces/CC/pages/9963078>

8.1.8.19.25 RandomValueLookup

| **Based >Extensible Algorithm Framework**³¹⁶

The RandomValueLookup algorithm is an instance of the [Secure Lookup Algorithm Framework](#) (see page 793).

This algorithm performs a lookup on the input value and returns a value from the provided lookup file. It is possible for this algorithm to produce the same output value for different input values. Inputs to this algorithm are case-sensitive so two inputs with the same value in different cases may mask to different values. Leading and trailing whitespaces are preserved by this algorithm.

The lookup values for this algorithm are 50-character random value strings.

For example:

- "6379315274824970" → "ufGcYiFgzC6RBmcBPC7Mvb0oOqEhjEVDUJZLHo6OYNWoi5PZKC"
- "ABCxyz123" → "Wk4iq8Y6Ngz8j84AhueDmHQo6uQgMjmnMLuGFxuPEPmDBLzzNf"
- "Si" → "8pz8lnVeQqNCe3jnQREPH2bfbQHkNRir6CHliwq1fMTY3sKFIY"
- "999-12-3456." → "37cq1Lve2rOi3kwrNjDE2p1CNPSeAUtnZYNRfUcDuGnikx6DE9"
- "2000:a86f::1" → "UUWWeetRXJXMR6puAk8414nrHWmN5nanrOxoWw7DesbHHZPLs1"

8.1.8.19.26 ReligiousOrientation SL

| **Based >Extensible Algorithm Framework**³¹⁷

The ReligiousOrientation SL algorithm is an instance of the [Secure Lookup Algorithm Framework](#) (see page 793).

This algorithm performs a lookup on the input value and returns a value from the provided lookup file. It is possible for this algorithm to produce the same output value for different input values. Inputs to this algorithm are case-sensitive so two inputs with the same value in different cases may mask to different values. Leading and trailing whitespaces are preserved by this algorithm.

The lookup values for this algorithm are the top 10 religious orientations worldwide.

For example:

- "Catholic" → "Buddhist"
- "Sikh" → "Jewish"
- "Jain" → "Muslim"

8.1.8.19.27 SexualOrientation SL

| **Based >Extensible Algorithm Framework**³¹⁸

The SexualOrientation SL algorithm is an instance of the [Secure Lookup Algorithm Framework](#) (see page 793) with the RANDOMIZE hash method.

316 <https://delphixdocs.atlassian.net/wiki/pages/createpage.action?fromPageId=205427657&linkCreation=true&spaceKey=CC&title=Introduction+to+masking+algorithms>

317 <https://delphixdocs.atlassian.net/wiki/spaces/CC/pages/9963078>

318 <https://delphixdocs.atlassian.net/wiki/spaces/CC/pages/9963078>

This algorithm does not perform any lookup on the input value and returns a random value from the provided lookup file. It is possible for this algorithm to produce the same output value for different input values. Since the masked values are chosen at random from the lookup file, this algorithm does not maintain referential integrity. Leading and trailing whitespaces are preserved by this algorithm. The masked values maintain the statistical distribution (frequency of values) of the lookup file.

The lookup values for this algorithm are commonly used sexual orientations.

For example:

- "Heterosexual" → "Bisexual"
- "Straight" → "Homosexual"
- "Gay" → "Heterosexual"

8.1.8.19.28 SchoolNameLookup

| *Based >* [Extensible Algorithm Framework](#)³¹⁹

The SchoolNameLookup algorithm is an instance of the [Secure Lookup Algorithm Framework](#) (see page 793).

This algorithm performs a lookup on the input value and returns a value from the provided lookup file. It is possible for this algorithm to produce the same output value for different input values. Inputs to this algorithm are case-sensitive so two inputs with the same value in different cases may mask to different values. Leading and trailing whitespaces are preserved by this algorithm.

The lookup values for this algorithm are schools and colleges in the format School Name, State, USA.

For example:

- "Columbia University" → "Smith College, Massachusetts, USA"
- "clown college" → "Ithaca College, New York, USA"
- "The Culinary Institute" → "Stanford University, California, USA"
- "UCLA" → "Rensselaer Polytechnic Institute, New York, USA"

8.1.8.19.29 SwiftCode SL

| *Based >* [Extensible Algorithm Framework](#)³²⁰

The SwiftCode SL algorithm is an instance of the [Secure Lookup Algorithm Framework](#) (see page 793).

This algorithm performs a lookup on the input value and returns a value from the provided lookup file. It is possible for this algorithm to produce the same output value for different input values. Inputs to this algorithm are case-sensitive, so two inputs with the same value in different cases may mask to different values. Leading and trailing whitespaces are preserved by this algorithm.

The lookup values for this algorithm are valid 8-character swift codes (bank identifier codes) of the banks across the world.

For example:

³¹⁹ <https://delphixdocs.atlassian.net/wiki/pages/createpage.action?fromPageId=205395034&linkCreation=true&spaceKey=CC&title=Introduction+to+masking+algorithms>

³²⁰ <https://delphixdocs.atlassian.net/wiki/pages/createpage.action?fromPageId=205395004&linkCreation=true&spaceKey=CC&title=Introduction+to+masking+algorithms>

- "ANZBAU3M" → "FOREEE2X"
- "CTBAAU2S" → "AEIBJPJX"
- "WPACAU2S" → "NBETETAA"
- "NATAAU33" → "CBETETAA"

8.1.8.19.30 USCitiesLookup

| *Based >* [Extensible Algorithm Framework](#)³²¹

The USCitiesLookup algorithm is an instance of the [Secure Lookup Algorithm Framework](#) (see page 793).

This algorithm performs a lookup on the input value and returns a value from the provided lookup file. It is possible for this algorithm to produce the same output value for different input values. Inputs to this algorithm are case-sensitive so two inputs with the same value in different cases may mask to different values. Leading and trailing whitespaces are preserved by this algorithm.

The lookup values for this algorithm are United States cities.

For example:

- "Los Angeles" → "Chicago"
- "New England" → "Oklahoma City"
- "cape town, south africa" → "Houston"
- "Princeton, New Jersey" → "Redwood City"

8.1.8.19.31 USCountiesLookup

| *Based >* [Extensible Algorithm Framework](#)³²²

The USCountiesLookup algorithm is an instance of the [Secure Lookup Algorithm Framework](#) (see page 793).

This algorithm performs a lookup on the input value and returns a value from the provided lookup file. It is possible for this algorithm to produce the same output value for different input values. Inputs to this algorithm are case-sensitive so two inputs with the same value in different cases may mask to different values. Leading and trailing whitespaces are preserved by this algorithm.

The lookup values for this algorithm are United States counties.

For example:

- "Schuyler County" → "Rock Island County"
- "orange co." → "Price County"
- "Yellowstone County, Montana" → "McKinley County"

321 <https://delphixdocs.atlassian.net/wiki/pages/createpage.action?fromPageId=205428323&linkCreation=true&spaceKey=CC&title=Introduction+to+masking+algorithms>

322 <https://delphixdocs.atlassian.net/wiki/pages/createpage.action?fromPageId=205428255&linkCreation=true&spaceKey=CC&title=Introduction+to+masking+algorithms>

8.1.8.19.32 USstatecodesLookup

| Based > [Extensible Algorithm Framework](#)³²³

The USstatecodesLookup algorithm is an instance of the [Secure Lookup Algorithm Framework](#) (see page 793).

This algorithm performs a lookup on the input value and returns a value from the provided lookup file. It is possible for this algorithm to produce the same output value for different input values. Inputs to this algorithm are case-sensitive so two inputs with the same value in different cases may mask to different values. Leading and trailing whitespaces are preserved by this algorithm.

The lookup values for this algorithm are 2-letter United States state and territory codes.

For example:

- "AL" → "CA"
- "Maine" → "UT"
- "west virginia" → "SD"
- "Italy" → "PR"

8.1.8.19.33 USstatesLookup

| Based > [Extensible Algorithm Framework](#)³²⁴

The USstatesLookup algorithm is an instance of the [Secure Lookup Algorithm Framework](#) (see page 793).

This algorithm performs a lookup on the input value and returns a value from the provided lookup file. It is possible for this algorithm to produce the same output value for different input values. Inputs to this algorithm are case-sensitive so two inputs with the same value in different cases may mask to different values. Leading and trailing whitespaces are preserved by this algorithm.

The lookup values for this algorithm are United States state and territory names.

For example:

- "Hawaii" → "Iowa"
- "KS" → "District of Columbia"
- "california" → "Northern Marianas Islands"

8.1.8.19.34 WebURLsLookup

| Based > [Extensible Algorithm Framework](#)³²⁵

The WebURLsLookup algorithm is an instance of the [Secure Lookup Algorithm Framework](#) (see page 793).

323 <https://delphixdocs.atlassian.net/wiki/pages/createpage.action?fromPageId=205423958&linkCreation=true&spaceKey=CC&title=Introduction+to+masking+algorithms>

324 <https://delphixdocs.atlassian.net/wiki/pages/createpage.action?fromPageId=205390236&linkCreation=true&spaceKey=CC&title=Introduction+to+masking+algorithms>

325 <https://delphixdocs.atlassian.net/wiki/pages/createpage.action?fromPageId=205390266&linkCreation=true&spaceKey=CC&title=Introduction+to+masking+algorithms>

This algorithm performs a lookup on the input value and returns a value from the provided lookup file. It is possible for this algorithm to produce the same output value for different input values. Inputs to this algorithm are case-sensitive so two inputs with the same value in different cases may mask to different values. Leading and trailing whitespaces are preserved by this algorithm.

The lookup values for this algorithm are website addresses.

For example:

- "www.google.com" → "http://www.blogspot.com"
- "delphix.com" → "http://www.gaurdian.co.uk"
- "https://en.wikipedia.org/wiki/Syslog#References" → "http://www.newegg.com"

8.1.8.20 dlp-core:TimeRange

| **Based > Extensible Algorithm Framework** (see page 656)

The TimeRange algorithm is an instance of the [Segment Mapping Algorithm Framework](#) (see page 797), masking the numeric values between ranges 0-59. This algorithm is appropriate for masking time units – usually minutes or seconds which range between 0 and 59.

For example:

- "0" → "46"
- "59" → "35"
- "1" → "13"
- "abc" → "abc" (and generates a non-conformant data event)



This algorithm may generate non-conformant data events.

8.1.8.21 dlp-core: IBAN

| **Extensible Algorithm Framework** (see page 656)

The IBAN algorithm allows you to mask [International Bank Account Numbers](#)³²⁶. It takes IBAN as an input and replaces each numeric character with another numeric character, and the same for alphabet characters. The first two characters of a valid IBAN are its country code. The country code is preserved, it is not masked. The second two characters of a valid IBAN are a checksum, generated from the other characters using the MOD97 method. This is regenerated by the algorithm after masking to ensure checksum validity. The replacement characters are sourced from a hash of the original input IBAN. This hash remains stable for an instance of the algorithm, so the output will always be the same for a given input. A small set of delimiters are preserved if present in the input IBAN: space, period, and hyphen. Delimiters will not be preserved in the first 4 characters of an input, as these are reserved for country code and checksum.

³²⁶ <https://www.iban.com/>

The algorithm attempts to mask all inputs even if they are not valid IBAN and will still replace numeric characters with numeric, and alphabet with alphabet. Unsupported delimiters or punctuation will be replaced with an alphanumeric character. If possible, a new checksum is calculated even if the input did not have a valid checksum. The algorithm cannot proceed with masking invalid data that has less than 4 characters. If this input is received, it will be padded to 4 characters and masking will reattempt.

8.1.8.21.1 Creating an Instance / Configuration Options

The default instance of the IBAN algorithm cannot be configured. Newly created instances of the IBAN algorithm have two options that can be configured. As with other algorithms, enter a name and an optional description. You can then choose to enable input validation and select how many characters to mask.

The number of characters to mask maximum is 64, but the longest country supported IBAN is currently Russia with 33 characters. Numbers higher than the length of an IBAN input will mask all characters possible. The minimum number is 6. Lower numbers will speed up the performance of the algorithm, but be conscious of reduction in data obfuscation as the number of changed characters goes down.

There is an option for IBAN input validation. This can be useful to identify corrupted or otherwise invalid input data. It does come at a moderate performance cost. The validation uses the [Apache Commons Validator](https://commons.apache.org/proper/commons-validator/)³²⁷. This tool validates that the country code is supported, the checksum is valid and that the structure of the IBAN is appropriate for the given country. It is updated with new countries as they adopt the IBAN standard. The current version used is 1.7. If invalid IBAN are found they are treated as non-conformant data.

Select Framework

- Secure Lookup
- Character Mapping
- Payment Card
- Date
- Dependent Date Shift
- Name
- Full Name
- Email
- Segment Mapping
- Mapping
- Binary Lookup
- Tokenization
- Min Max
- Data Cleansing
- Free Text Redaction
- Numeric Expression
- IBAN
- Extended

Create IBAN Algorithm [Learn More](#)

Algorithm Name

Description

Mask Type

STRING

Validate Input IBAN

Number of Characters to Mask (min: 6, max: 64)

Cancel Save

1. In the upper right-hand region of the **Algorithm** tab under **Settings**, click **Add Algorithm**.

³²⁷ <https://commons.apache.org/proper/commons-validator/>

2. Select **IBAN**. The "Create IBAN Algorithm" window will appear.
3. Enter an **Algorithm Name**.
Info: Name MUST be unique.
4. Enter a **Description** (optional).
5. Set **Number of Characters to Mask**. This value is the number of characters from the right that will be masked. Values for this field must be in the range [6-64].
6. When finished, click **Save**.

8.1.8.21.2 Examples

Valid IBAN inputs and masked outputs:

- NO8330001234567 → NO0631216940542
- GL8964710123456789 → GL3640635860760239


Invalid IBAN inputs and masked outputs:

- ???A???b???C???d???E??? → ??29S1076R7076M0100E0222J0177
- ABC123 → AB6447

Notice that it still respects a character's type and the first two as if they were a country code.

8.1.8.22 SecureShuffle

This algorithm masks by shuffling the values in a particular field or column to different lines or rows. For example, values for the FIRST_NAME column might be shuffled among a number database rows within a table. It guarantees that each value is moved to a different line or row, but will not prevent an input from masking to the same output in the case where the values shuffled are not unique.

 Because shuffling data does not redact or modify the individual data values in any way, careful consideration must be given to whether this form of obfuscation is sufficient to meet your security requirements.

The SecureShuffle algorithm may only be used with masking jobs that support batching, and will not be presented as an option in the inventory screen when it is not supported. The maximum number positions any particular value will be moved within the input is equal to the batch size.

Please refer to the Batch Masking section [here \(see page 0\)](#) for a full description of the Batch Masking mechanism, as well as details on batch size and which jobs support batching.

This algorithm will report non-conformant data whenever only one value is available to mask, meaning that no shuffling is possible.

8.1.9 Algorithm frameworks

This section covers the following topics:

- [Binary Lookup \(see page 705\)](#)
- [Character Mapping \(Algorithm frameworks\) \(see page 707\)](#)
- [Character Replacement \(Algorithm frameworks\) \(see page 712\)](#)
- [Data Cleansing \(Algorithm frameworks\) \(see page 714\)](#)
- [Date Replacement \(Algorithm frameworks\) \(see page 717\)](#)
- [Date Shift \(Algorithm frameworks\) \(see page 720\)](#)
- [Dependent Date Shift \(Algorithm frameworks\) \(see page 722\)](#)
- [Email \(Algorithm frameworks\) \(see page 726\)](#)
- [Free Text Redaction \(Algorithm frameworks\) \(see page 730\)](#)
- [Full Name \(Algorithm frameworks\) \(see page 735\)](#)
- [Mapping \(Algorithm frameworks\) \(see page 740\)](#)
- [Min Max Number \(Algorithm frameworks\) \(see page 754\)](#)
- [Min Max Date \(Algorithm frameworks\) \(see page 757\)](#)
- [Multi Column Address \(Algorithm frameworks\) \(see page 759\)](#)
- [Multi Column Condition \(Algorithm frameworks\) \(see page 767\)](#)
- [Name \(Algorithm frameworks\) \(see page 773\)](#)
- [Numeric Expression \(Algorithm frameworks\) \(see page 777\)](#)
- [Payment Card \(Algorithm frameworks\) \(see page 786\)](#)
- [Regex Decompose \(Algorithm frameworks\) \(see page 788\)](#)
- [Secure Lookup \(Algorithm frameworks\) \(see page 793\)](#)
- [Segment Mapping \(Algorithm frameworks\) \(see page 797\)](#)
- [String Algorithm Chain \(Algorithm Frameworks\) \(see page 804\)](#)
- [Tokenization \(Algorithm frameworks\) \(see page 806\)](#)

8.1.9.1 Binary Lookup

A Binary Lookup algorithm is much like the Secure Lookup algorithm but is used when entire files are stored in a specific column. This algorithm replaces objects that appear in object columns. For example, if a bank has an object column that stores images of checks, you can use a Binary Lookup algorithm to mask those images. The Delphix Engine cannot change data within images themselves, such as the names on X-rays or driver's licenses. However, you can replace all such images with a new, fictional image. This fictional image is provided by the owner of the original data.

8.1.9.1.1 Creating a Binary Lookup algorithm via UI

1. At the top right of the **Algorithms** page, click **+ Algorithm**.

Add Algorithm

×

- Details
- Configuration
- Summary

Details

Specify the algorithm details.

Name

Description

Framework Name

Mask Type
BYTE_BUFFER

Binary Lookup Framework: A Binary Lookup algorithm is much like the Secure Lookup algorithm but is used when entire files are stored in a specific column. This algorithm replaces objects that appear in object columns. For example, if a bank has an object column that stores images of checks, you can use a Binary Lookup algorithm to mask those images. The Delphix Engine cannot change data within images themselves, such as the names on X-rays or driver's licenses. However, you can replace all such images with a new, fictional image. This fictional image is provided by the owner of the original data.

Cancel
Back
Next
Save

2. Enter an **Algorithm Name**.
Info: This MUST be unique.
3. Enter a **Description**.
4. Select **Binary Lookup** as the Framework Name and click **Next**.

Add Algorithm

×

- Details
- Configuration
- Summary

Configuration

Configure the algorithm.

Binary Lookup Files ⓘ

[Select File](#) sql.txt
Successfully uploaded file.

temp.txt

sql.txt

Binary Lookup

Framework Description

Binary Lookup Framework: A Binary Lookup algorithm is much like the Secure Lookup algorithm but is used when entire files are stored in a specific column. This algorithm replaces objects that appear in object columns. For example, if a bank has an object column that stores images of checks, you can use a Binary Lookup algorithm to mask those images. The Delphix Engine cannot change data within images themselves, such as the names on X-rays or driver's licenses. However, you can replace all such images with a new, fictional image. This fictional image is provided by the owner of the original data.

Cancel
Back
Next
Save

5. Select a **Binary Lookup File** on your filesystem by clicking the **Select File** button.
6. Click **Next** to verify details on the Summary step.

Add Algorithm ×

- Details
- Configuration
- Summary

Summary

View the algorithm details below. Click the Back button to make changes or click the Save button to save the algorithm configuration.

Details

Name
test_binary_lookup

Description
Testing algorithm

Framework Name
Binary Lookup

Mask Type
BYTE_BUFFER

Configuration

Binary Lookup Files
temp.txt
sql.txt

Cancel Back Next Save

7. Click **Save**.



- A maximum of **50 Lookup Files** can be selected.
- To upload multiple files, use the **Select File** button once for each file.

For information on creating Binary Lookup algorithms through the API, see [API Calls for Creating Algorithms - Binary Lookup](#). (see page 900)

8.1.9.2 Character Mapping (Algorithm frameworks)

The Character Mapping framework maps text values, defined by a set of character groups, to other text values generated from the same character groups. Mappings are calculated algorithmically, so it is not necessary to provide the set of mapping values. The algorithm preserves any characters not assigned to a group. Any characters from the first Unicode plane can be mapped - this covers most characters used in modern languages. Other (supplementary) characters can only be preserved.

The particular set of permutations used is determined by the algorithm's key, so rekeying the algorithm will cause different outputs to be generated for each input.

The algorithm has the following properties:

- The masked value for each input is consistent unless the algorithm is rekeyed.
- No two text inputs produce the same text output. Collisions *are* possible for some data types, such as Numeric, where multiple text values, such as "001" and "1", are treated as the same value.
- As long as at least one maskable character is present in the input, the masked value will never match the input.
- Each masked position influences the mapping done at every other masked position.

For these reasons, this algorithm is useful for masking columns with uniqueness requirements, such as primary and foreign key columns.

This algorithm was introduced in version 6.0.5.0, and uses the algorithm extensibility framework, allowing it to be called from other algorithms using that framework.

To decide whether Character Mapping or Segment Mapping is the correct option for your use case, see [Choosing Between Character and Segment Mapping Frameworks](#) (see page 656).



The character mapping algorithm can be used for tokenization and reidentification jobs.

8.1.9.2.1 Creating a character mapping algorithm via UI

1. At the top right of the **Algorithms** page, click **+ Algorithm**.

Add Algorithm
✕

- Details
- Configuration
- Summary

Details

Specify the algorithm details.

Name

Description

Framework Name

Mask Type
STRING

Character Mapping Framework: It is a list of values defining the groups of characters to be masked. These may be expressed as simple strings - for example, "0123456789" to mask all digit characters, or as Java regex character ranges, such as "[0-9]". Presences of the "|" character in the first position determines the usage.

Multiple groups may be defined to control which characters can be interchanged during masking. For example, "[A-Z]" + "[0-9]" will replace digits only with digits, and letters only with letters. Any characters not covered by a configured characterGroup is preserved.

Any duplication of characters among character groups or within the same group will cause the configuration to be rejected by the framework's validation method.

Cancel Back Next Save

2. Enter an **Algorithm Name**.
Info: This MUST be unique.
3. Enter a **Description**.
4. Select **Character Mapping** as the Framework Name and click **Next**.

Add Algorithm



- Details
- Configuration
- Summary

Configuration

Configure the algorithm.

Character Groups Ⓢ ✎

There are no associated character groups.

Case Sensitive

Minimum Masked Positions

Data Preservation

None

Preserve Leading Zeros

Preserve Ranges

Character Mapping

Framework Description

Character Mapping Framework: It is a list of values defining the groups of characters to be masked.

These may be expressed as simple strings - for example, "0123456789" to mask all digit characters, or as Java regex character ranges, such as "[0-9]". Presences of the '[' character in the first position determines the usage.

Multiple groups may be defined to control which characters can be interchanged during masking. For example, "[A-Z]" + "[0-9]" will replace digits only with digits, and letters only with letters. Any characters not covered by a configured characterGroup is preserved.

Any duplication of characters among character groups or within the same group will cause the configuration to be rejected by the framework's validation method.

Cancel Back Next Save

- Define **Character Groups** for each group of characters among which you would like to map by clicking on ✎ using the List editor section. More information on the List Editor Section can be found [here \(see page 232\)](#).

Each group may defined either by specifying each literal character in the group, such as "0123456789", or using Java Regular Expression style character ranges, such as "[0-9]". The algorithm will freely map characters to other characters within the same group, so by defining groups "[0-9]" and "[A-Z]", numbers would be replaced by other numbers, and letters by other letters, but a number would never be replaced by a letter. Groups should not contain duplicate characters, and each character may belong to only one group. Any character that is not assigned to a group will be preserved (not masked) by the algorithm. The box below the entry area allows selection of character groups defined for other, preexisting Character Mapping algorithms.

Character Groups [?]




The screenshot shows a 'Character Groups' configuration window. It contains a list of two character groups:

- Group 1: [a-zA-Z] (with a plus sign to add and a trash icon to delete)
- Group 2: [0-9] (with a plus sign to add and a trash icon to delete)

At the bottom right of the window, there are two buttons: a grey 'X' button and a blue checkmark button.

6. Check the **Case Sensitive** box to cause the algorithm to treat upper and lower case characters as distinct characters for mapping.
7. Select a value for **Minimum Masked Position**, which sets the minimum number of characters that the algorithm must mask; fewer positions triggers non-conformant data handling. Null, empty, and all-whitespace values never trigger non-conformant data handling.
8. Check the **Preserve Leading Zeros** box to cause the algorithm to preserve any number of '0' characters at the beginning of each input. This is only useful if '0' has been assigned to a character group in step 5.

Warning: Masked results are not guaranteed to be unique if **Preserve Leading Zeros** is used, and the algorithm cannot be used for tokenization/re-identification jobs.

9. If desired, define ranges of the input value to ignore using the **Preserve Ranges** controls by clicking on  using the List editor section. More information on the List Editor Section can be found [here \(see page 232\)](#).

For Character Mapping algorithms, only characters that would otherwise be masked count when determining position for preserve ranges. Each preserve range is defined by:

- a. **Starting Position** - The position at which to start preserving, starting from 0.
- b. **Length** - The number of characters to preserve.
- c. **Direction**- The direction, either forward or reverse, determines whether to process from the beginning or end of input for this range.

Preserve Ranges

Preserve Ranges

Starting Position
1

Length
2

Direction
Forward

+
+ 🗑️

✕ ✓

10. Click **Next** to verify details on the Summary step.

Add Algorithm ✕

- Details
- Configuration
- Summary

Summary

View the algorithm details below. Click the Back button to make changes or click the Save button to save the algorithm configuration.

Details

Name
test_cm

Framework Name
Character Mapping

Mask Type
STRING

Configuration

Character Groups
[a-zA-Z]
[0-9]

Case Sensitive
false

Minimum Masked Positions
1

Preserve Leading Zeros
false

Preserve Ranges
Starting Position 1, Length 2, Direction Forward

Cancel Back Next Save

11. Click **Save**.

8.1.9.2.2 Examples

As an example, a Character Mapping algorithm could be defined with a single character group, "[0-9]". It might mask as follows:

- "(603) 867-5309" → "(463) 638-0193"
- "999-12-3456" → "453-71-6283"
- "Call Tom at 8:00PM" → "Call Tom at 2:75PM"

Securing sensitive data – 711

8.1.9.3 Character Replacement (Algorithm frameworks)

The character replacement framework allows creation of rules to replace specific characters in a string with other characters. The framework must wrap another string algorithm, which defaults to **dlpx-core:Alpha Numeric**. Character replacement is often used as part of a chained string algorithm, either before or after masking. Other features are removal of accented characters and ensuring that output of the algorithm is changed from the input. At this time there is no UI or default instance for the character replacement framework.

Replacement of characters is specified in a list of JSON rules. There is no enforced limit to the number of rules that can be created.

8.1.9.3.1 Creating a Character Replacement algorithm via API:

```
{
  "stringAlgorithm": {
    "name": "dlpx-core:CM Alpha-Numeric"
  },
  "replacementRules": [
    {
      "inputFilteredCharacters": null,
      "inputReplacementCharacter": null,
      "outputFilteredCharacters": ["1","2","3","4","5","6","7","8","9","0"],
      "outputReplacementCharacter": "*",
      "filteredCharacters": ["a","e","i","o","u"],
      "replacementCharacter": "+"
    }
  ],
  "requireInputChange": false,
  "filterAccents": "NONE"
}
```

i Filter pair: Two JSON keys which together describe the list of characters to filter, and the character they should be replaced with.

1. Define `stringAlgorithm` - **Object - Required**

- Define `name` - **String - Required**

- This must be a valid string algorithm in the Delphix engine. The string algorithm should always be chosen, but it defaults to **dlpx-core:CM Alpha Numeric**.

2. Define `requireInputChange` - **Boolean - Required**

- Whether to ensure the input to the algorithm is different than the output. Throws an exception if set to true and the input and output match.

3. Define `filterAccents` - Enum - Required

- The accents from characters like "é" or "ñ" can be stripped and decomposed to the root character like "e" or "n". This can be done before character replacement and chained algorithm masking, after, neither or both.
 - Accepted values: INPUT, OUTPUT, NONE, BOTH

4. Define `replacementRules` - List - Required

- Contains a list of JSON objects with filter pairs of characters to replace and their replacements. For each rule object you must define a filter pair for at least one of input, output or the unqualified `filteredCharacters` which apply to both:
 - `inputFilteredCharacters` - List - Optional (If another filter pair defined)
 - Characters to filter from initial input.
 - `inputReplacementCharacter` - String - Optional (If another filter pair defined)
 - Single character to replace those from `inputFilteredCharacters` with.
 - `outputFilteredCharacters` - List - Optional (If another filter pair defined)
 - Characters to filter from output of chained string algorithm.
 - `outputReplacementCharacter` - String - Optional (If another filter pair defined)
 - Single character to replace those from `outputFilteredCharacters` with.
 - `filteredCharacters` - List - Optional (If another filter pair defined)
 - Characters to filter from input and output.
 - `replacementCharacter` - String - Optional (If another filter pair defined)
 - Single character to replace those from `filteredCharacters` with.

8.1.9.3.2 Validation

Validation is done to ensure that:

- Each list of filtered characters has a corresponding replacement character (is a complete filter pair)
- Each replacement rule has at least one filter pair
- The replacement character is not present in the filtered characters
- At least one replacement rule is defined
- The `stringAlgorithm` key refers to a valid chained string algorithm

8.1.9.3.3 Examples

Examples of output vary widely because they depend on the chained string algorithm chosen. If we keep the default chained algorithm of **dlpx-core:Alpha Numeric**, it will never change punctuation marks and will allow us to create examples of punctuation filtering. Consider the following configuration:

```
{
  "stringAlgorithm": {
    "name": "dlpx-core:CM Alpha-Numeric"
  },
  "replacementRules": [
    {
      "inputFilteredCharacters": null,
      "inputReplacementCharacter": null,
      "outputFilteredCharacters": null,
      "outputReplacementCharacter": null,
      "filteredCharacters": ["@", "#", "$", "%", "&", "+", "?"],
      "replacementCharacter": "."
    }
  ],
  "requireInputChange": false,
  "filterAccents": "BOTH"
}
```

Potential masked results with the same keyed **dlpx-core:Alpha Numeric** instance as follows:

- "1+2+3+4+5+6+7+8+9+0" can become "0.0.0.4.8.3.7.7.7.4"
- "1+2&3?4+5&6?7+8&9?0" can become "0.0.0.4.8.3.7.7.7.4"

With filter accents, alphanumeric characters with accents can be masked to unaccented characters:

- "áéíóú" can become "bkxqw"
- "ÂÊÎÔÛ" can become "BKXQW"

In this example configuration altering the rule to filter on input, output or both makes no difference. Changing the filter point could produce different results with an algorithm that could potentially change, add or remove punctuation marks and accented characters.

8.1.9.4 Data Cleansing (Algorithm frameworks)

A data cleansing algorithm is used to standardize varied spellings, misspellings, and abbreviations for the same name. For example, "Ariz," "Az," and "Arizona" can all be cleansed to "AZ." Use this algorithm if the target data needs to be in a standard format prior to masking.

8.1.9.4.1 Creating a data cleansing algorithm via UI

1. At the top right of the **Algorithms** page, click **+ Algorithm**.

Add Algorithm



- Details
- Configuration
- Summary

Details

Specify the algorithm details.

Name
test_dc

Description

Framework Name
Data Cleansing

Mask Type
STRING

Data Cleansing Framework: This framework is used to standardize varied spellings, misspellings, and abbreviations for the same name. It uses a lookup file with mappings of value to replacement pairs. For example, "Ariz", "Az", and "Arizona" can all be cleansed to "AZ". Use this algorithm if the target data needs to be in a standard format prior to masking.

Framework Options:

Case Sensitive Lookup: Whether the case of the input string must match the values in the lookup file.

Trim Whitespace: Whether to trim leading and trailing whitespace from the input string. Note: This must be **checked** to cleanse fixed-width files and fixed-length database data types such as CHAR and NCHAR.

Lookup File: A file containing a newline separated list of value,replacement pairs separated by a delimiter.

Cancel Back **Next** Save

2. Enter an **Algorithm Name**.

Info: This **MUST** be unique.

3. Enter a **Description**.

4. Select **Data Cleansing** as the Framework Name and click **Next**.

Add Algorithm



- Details
- Configuration
- Summary

Configuration

Configure the algorithm.

Case Sensitive Lookup

Trim Whitespace

Lookup File ⓘ

Select File temp.txt x

Successfully uploaded file.

Lookup File Delimiter
=

Data Cleansing

Framework Description

Data Cleansing Framework: This framework is used to standardize varied spellings, misspellings, and abbreviations for the same name. It uses a lookup file with mappings of value to replacement pairs. For example, "Ariz", "Az", and "Arizona" can all be cleansed to "AZ". Use this algorithm if the target data needs to be in a standard format prior to masking.

Framework Options:

Case Sensitive Lookup: Whether the case of the input string must match the values in the lookup file.

Trim Whitespace: Whether to trim leading and trailing whitespace from the input string. Note: This must be **checked** to cleanse fixed-width files and fixed-length database data types such as CHAR and NCHAR.

Lookup File: A file containing a newline separated list of value,replacement pairs separated by a delimiter.

Lookup File Delimiter: The string delimiter that separates value,replacement pairs in the lookup file. The **default** is =, but it can be any string up to 50 characters long.

Cancel Back **Next** Save

5. Choose whether to use **Case Sensitive Lookup**. If this box is checked, the data to be cleansed must match the case of the value in the lookup file in order to be replaced.

For example, if the lookup file contains `Arizona=AZ` :

Original	Cleansed	Case Sensitive Lookup
Arizona	AZ	checked or not checked
arizona	AZ	not checked
arizona	arizona	checked

6. Choose whether to **Trim Whitespace**. If this box is checked, the leading and trailing whitespace of the data to be cleansed is removed prior to checking if the value is in the lookup file. This allows a single `value=replacement` in the lookup file to cleanse data containing extraneous leading and trailing whitespace.

Info

This must be checked to cleanse fixed-width files and fixed-length database data types such as CHAR and NCHAR.

7. Specify a **Lookup File**. You can either click the Select... button to choose a local file or enter the fileReferenceId value returned from the fileUpload API endpoint for uploading files to the Masking Engine. The file should contain a newline separated list of {value, replacement} pairs separated by the delimiter.
8. Specify a **Lookup File Delimiter** (value and replacement separator) up to 50 characters long. The default delimiter is `=`. You can change this to match the lookup file.
9. Click **Next** to verify details on the Summary step.

Add Algorithm ×

- Details
- Configuration
- Summary

Summary

View the algorithm details below. Click the Back button to make changes or click the Save button to save the algorithm configuration.

Details

Name
test_dc

Framework Name
Data Cleansing

Mask Type
STRING

Configuration

Case Sensitive Lookup
true

Trim Whitespace
true

Lookup File
temp.txt

Lookup File Delimiter
=

Cancel Back Next Save

10. Click **Save**.

Below is an example of a lookup file. It does not require a header. Make sure there are no spaces or returns at the end of the last line in the file. The following is sample file content:

```
NYC=NY
NY City=NY
New York=NY
Manhattan=NY
```

For information on creating Data Cleansing algorithms through the API, see [API Calls for Creating Algorithms - Data Cleansing](#). (see page 904)

8.1.9.5 Date Replacement (Algorithm frameworks)

The Date Replacement framework masks a date value based on specified beginning and end dates. Masked output values are calculated algorithmically using the algorithm's key, so rekeying the algorithm will cause a different output value to be generated for each input. It is possible for an input to be masked to itself.

8.1.9.5.1 Creating a Date Replacement Algorithm via UI

1. At the top right of the **Algorithms** page, click **+ Algorithm**.

Add Algorithm ✕

- Details
- Configuration
- Summary

Details

Specify the algorithm details.

Name

Description

Framework Name

Mask Type
LOCAL_DATE_TIME

Date Replacement Framework: Allow to define a range of dates that the input can be masked to. The algorithm take into account the exact minimum and maximum dates regardless of the input value. All outputs are formatted the same as the inputs, and all outputs are deterministically changed to a new value.

Cancel Back Next Save

- 2. Enter an **Algorithm Name**.
Info: This MUST be unique.
- 3. Enter a **Description**.
- 4. Select **Date Replacement** as the Framework Name and click **Next**.

Add Algorithm ✕

- Details
- Configuration
- Summary

Configuration

Configure the algorithm.

Minimum Date Time

Maximum Date Time

Unit

Date Replacement

Framework Description

Date Replacement Framework: Allow to define a range of dates that the input can be masked to. The algorithm take into account the exact minimum and maximum dates regardless of the input value. All outputs are formatted the same as the inputs, and all outputs are deterministically changed to a new value.

Cancel Back Next Save

- 5. Enter **Min Date** and **Max Date**. These define the range from which the algorithm will select output values. The range is inclusive of both values. All units of time less than the specified unit must be set

to 0. For example, a configuration with the unit set to **Days** must have the time portion set to 00:00:00.

- Choose the **Unit** of time from the drop-down: **Days, Hours, Minutes, or Seconds**. This represents the unit of time the range is expressed in. Any unit smaller than the specified unit will be set to 0 in the masked output. For example, with a unit of **Days**, all masked time values will be 00:00:00. For a more detailed explanation, see the [Examples \(see page 719\)](#) section.
- Click **Next** to verify details on the Summary step.

Add Algorithm ✕

- Details
- Configuration
- Summary

Summary

View the algorithm details below. Click the Back button to make changes or click the Save button to save the algorithm configuration.

Details

Name
test_dr

Framework Name
Date Replacement

Mask Type
LOCAL_DATE_TIME

Configuration

Minimum Date Time
2024-02-01 10:00:00

Maximum Date Time
2024-05-06 10:00:00

Unit
Seconds

Cancel
Back
Next
Save

- Click **Save**.

For information on creating Date Replacement algorithms through the API, see [API Calls for Creating Algorithms - Date Replacement](#). (see page 905)

8.1.9.5.2 Examples

As an example, a Date Replacement algorithm with a minimum range of "2020-01-01 00:00:00" and a maximum range of "2020-01-05 00:00:00" with the unit set to **Days** will replace the input value with a date in the specified range. Dates may mask as follows:

- "1995-03-05 13:25:00" → "2020-01-02 00:00:00"
- "2021-10-13 01:59:59" → "2020-01-04 00:00:00"
- "1856-07-31 00:00:00" → "2020-01-01 00:00:00"

Another example with a minimum range of "2020-01-01 01:00:00" and a maximum range of "2020-01-01 03:00:00" with the unit set to **Hours** provides 3 possible mask values:

- "2020-01-01 01:00:00"
- "2020-01-01 02:00:00"
- "2020-01-01 03:00:00"

Using the same range of "2020-01-01 01:00:00" to "2020-01-01 03:00:00" but with the unit set to **Minutes**, there are 121 possible output values as the unit is the granularity at which time is subdivided. Note that the range is inclusive of both range values. Possible masked values may be as follows:

- "2020-01-01 01:00:00"
- "2020-01-01 01:14:00"
- "2020-01-01 01:59:00"
- "2020-01-01 02:23:00"
- "2020-01-01 03:00:00"

All inputs with the same value masked with the same algorithm configuration will result in the same output values.

8.1.9.6 Date Shift (Algorithm frameworks)

The Date Shift framework masks date values to different dates based on a specified range around the input value. Masked values are calculated algorithmically using the algorithm's key, so rekeying the algorithm will cause different outputs to be generated for each input. All valid input values will be masked to a new value, and the new value will never match the input.

8.1.9.6.1 Creating a date shift algorithm via UI

1. At the top right of the **Algorithms** page, click **+ Algorithm**.

Add Algorithm

×

- Details
- Configuration
- Summary

Details

Specify the algorithm details.

Name

Description

Framework Name

Mask Type
LOCAL_DATE_TIME

Date Shift Framework: Allow users to configure a range in reference to the input that will output masked dates within the specified range. All outputs are formatted the same as the inputs, and all outputs are deterministically changed to a new value.

Cancel
Back
Next
Save

2. Enter an **Algorithm Name**.
Info: This MUST be unique.
3. Enter a **Description**.

- Select **Date Shift** as the Framework Name and click **Next**.

Add Algorithm

✕

○ Details

● **Configuration**

○ Summary

Configuration

Configure the algorithm.

Minimum Range

Maximum Range

Roll

Unit

Date Shift

Framework Description

Date Shift Framework: Allow users to configure a range in reference to the input that will output masked dates within the specified range. All outputs are formatted the same as the inputs, and all outputs are deterministically changed to a new value.

Cancel
Back
Next
Save

- Enter **Min Value** and **Max Value**. These values provide a range in which the masked value will differ from the input given a specified unit of time. The range is inclusive of both values where negative values represent units of time in the past and positive values represent units of time in the future. 0 may be included in the range or as one of the range values, but the input will not mask to the same value. A minimum value and maximum value that are equal will result in a fixed shift of that amount of time. For example, entering 3 as a min value and 3 as a max value with a unit of **Days** will mask all input values to 3 days in the future.
- Check the **Roll** box to preserve all units of time larger and smaller than the specified unit. Only the value of the specified unit will change. This option is supported for units months, days, hours, minutes, and seconds.
- Choose the **Unit** of time from the drop-down: **Years, Months, Days, Hours, Minutes, or Seconds**. This represents the unit of time the range is expressed in.
- Click **Next** to verify details on the Summary step.

Add Algorithm ×

- Details
- Configuration
- Summary

Summary

View the algorithm details below. Click the Back button to make changes or click the Save button to save the algorithm configuration.

Details

Name
test_ds

Framework Name
Date Shift

Mask Type
LOCAL_DATE_TIME

Configuration

Minimum Range
1

Maximum Range
4

Roll
false

Unit
Days

Cancel Back Next Save

9. When you are finished, click **Save**.

For information on creating Date Shift algorithms through the API, see [API Calls for Creating Algorithms - Date Shift](#). (see page 907)

8.1.9.6.2 Examples

As an example, a Date Shift algorithm with a minimum value of 3 and a maximum value of 5 with the unit set to **Days** will shift the input value from 3 to 5 days into the future. Dates may mask as follows:

- "2021-02-03 12:30:00" → "2021-02-06 12:30:00"
- "1905-12-10 00:00:00" → "1905-12-15 00:00:00"
- "2001-07-31 23:45:30" → "2001-08-04 23:45:30"

With roll enabled and the same configuration, a date at the end of a month will wrap around to the beginning of the month. Dates may mask as follows:

- "2021-02-25 10:00:00" → "2021-02-01 10:00:00"
- "1932-05-03 01:15:15" → "1932-05-08 01:15:15"
- "1999-08-31 18:30:00" → "1999-08-03 18:30:00"

All inputs with the same value masked with the same algorithm configuration will result in the same output values.

8.1.9.7 Dependent Date Shift (Algorithm frameworks)

The Dependent Date Shift algorithm masks two dates while maintaining a dependency between them. Examples of such dependent date pairs include:

- date of admission and date of discharge

- date of birth and date of death.

If dependent date pairs are masked independently, two problems can occur. First, the chronological order of the dates might be reversed. For example, the masked date of discharge might be chronologically earlier than the masked date of admission. Second, the interval between the masked dates might be too small or too large. By interval, we mean the difference in time between two dates. For example, suppose a given patient's record has an 80 year interval between date of birth and date of death. Independently masking the date of birth and date of death might result in a 5 month interval. This would turn an 80 year old patient into a 5 month old patient which might make the patient ineligible for certain procedures, benefits, etc.

The Dependent Date Shift algorithm addresses these problems by masking dependent date pairs while:

- maintaining the chronological relationship between the dates (i.e., the later date always stays later)
- maintaining a configurable interval between the dates

The Dependent Date Shift algorithm takes as input:

- `date1` : a primary date that will be masked with the [Date Shift \(see page 720\)](#) algorithm
- `date2` : a secondary (dependent) date
- `minRange` , `maxRange` , `unit` , `roll` : parameters to the Date Shift algorithm
- `intervalRange` :
 - Negative values are not allowed.
 - A value of zero (0) indicates that the interval between the `date1` and `date2` will also be the interval between `date1_masked` and `date2_masked` . For example if the unmasked dates were 1970-01-01 and 1970-01-15, `unit` was Days, and the `intervalRange` was 0, then the interval between the unmasked dates is 14 days and that would also be the interval between the masked dates.
 - Values greater than zero (0) generate an inclusive set of possible interval adjustment values. For example, a value of 3 would generate the following set of possible interval adjustment values: `[-3, -2, -1, 0, 1, 2, 3]` . The generated set is automatically adjusted to omit any values that would change the chronological order of the dates. For example, if the interval difference between `date1` and `date2` is 2 and the specified `intervalRange` is 3, then the set of possible interval adjustment values would be `[-1, 0, 1, 2, 3]` .

At a high level, the masking process is as follows:

1. If `date2` is not provided (`null`), mask `date1` with Date Shift and return
2. If `date1` is not provided (`null`), mask `date2` with Date Shift and return
3. Calculate `date1_masked` by applying the Date Shift algorithm to `date1`
4. Calculate the set of possible interval adjustment values using the `intervalRange`
5. Using the value of `date2` , select an interval adjustment value from the set of possible interval adjustment values
6. Calculate the new interval using the original interval and the selected interval adjustment value
7. Calculate `date2_masked` using `date1_masked` and the new interval

The masked results are deterministic for the same algorithm key and inputs. The algorithm's output will never be equal to the input.

8.1.9.7.1 Creating a dependent date shift algorithm via UI

1. At the top right of the **Algorithms** page, click **+ Algorithm**.

Add Algorithm
✕

- Details
- Configuration
- Summary

Details

Specify the algorithm details.

Name

Description

Framework Name

Mask Type
GENERIC_DATA_ROW

Dependent Date Shift Framework: This is the framework to cover the scenarios where it is necessary to mask a date based on another date. For example Birth Date/Death date or Date of admittance/date of discharge

Framework Options:

Minimum Range: Smallest number of time units that will be added to date1 when masking. The range is inclusive of this value. Negative values represent units of time in the past and positive values represent units of time in the future. If date1 is not provided, this is applied to date2.

Maximum Range: Largest number of time units that will be added to date1 when masking. The range is inclusive of this value. Negative values represent units of time in the past and positive values represent units of time in the future. If date1 is not provided, this is applied to date2.

Interval Range: A number representing the +/- range value to shift the interval inclusive of the range value. A value of 0 will not change the interval between date1 and date2. This number may not be less than 0. If the specified unit difference between date1 and date2 is within the bounds of the Interval Range, the value...

Cancel Back Next Save

2. Enter an **Algorithm Name**.
Info: This MUST be unique.
3. Enter a **Description**.
4. Select **Dependent Date Shift** as the Framework Name and click **Next**.

Add Algorithm



- Details
- Configuration
- Summary

Configuration

Configure the algorithm.

Minimum Range

Maximum Range

Interval Range

Roll

Unit

Dependent Date Shift

Framework Description

Dependent Date Shift Framework: This is the framework to cover the scenarios where it is necessary to mask a date based on another date. For example Birth Date/Death date or Date of admittance/date of discharge

Framework Options:

Minimum Range: Smallest number of time units that will be added to date1 when masking. The range is inclusive of this value. Negative values represent units of time in the past and positive values represent units of time in the future. If date1 is not provided, this is applied to date2.

Maximum Range: Largest number of time units that will be added to date1 when masking. The range is inclusive of this value. Negative values represent units of time in the past and positive values represent units of time in the future. If date1 is not provided, this is applied to date2.

Interval Range: A number representing the +/- range value to shift the interval inclusive of the range value. A value of 0 will not change the interval between dates. This number may not be less than 0. If the specified unit difference between date1 and date2 is within the bounds of the interval Range, only values will be provided such that the sign of the difference is

Cancel Back Next Save

5. Enter a **Minimum Range**.
6. Enter a **Maximum Range**.
7. Enter an **Interval Range**.
8. Configure the **Roll**.
9. Choose the **Unit** of time from the drop-down: **Years, Months, Days, Hours, Minutes, or Seconds**.
10. Click **Next** to verify details on the Summary step.

Add Algorithm



- Details
- Configuration
- Summary

Summary

View the algorithm details below. Click the Back button to make changes or click the Save button to save the algorithm configuration.

Details

Name
test_dds

Framework Name
Dependent Date Shift

Mask Type
GENERIC_DATA_ROW

Configuration

Minimum Range
2

Maximum Range
6

Interval Range
3

Roll
false

Unit
DAYS

Cancel Back Next Save

11. When you are finished, click **Save**.

For information on creating Dependent Date Shift algorithms through the API, see [API Calls for Creating Algorithms - Dependent Date Shift](#). (see page 908)

8.1.9.7.2 Examples

As an example, a Dependent Date Shift algorithm with a Minimum Range value of 3, a Maximum Range value of 5, and an Interval Range of 5 with the unit set to **Days** will shift the `date1` input value by 3 to 5 days into the future. It will then change the interval by a range of +/-5 days from the original interval to mask `date2`. Dates may mask as follows:

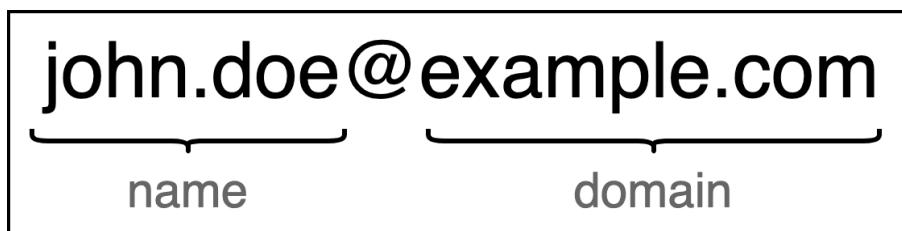
- 1905-12-10 00:00:00, 1907-08-01 10:14:00 → 1905-12-13 00:00:00, 1907-08-06 00:00:00
- 2001-07-31 23:45:30, 2005-04-12 07:13:00 → 2001-08-03 23:45:30, 2005-04-12 23:45:30
- 2021-02-03 12:30:00, 2021-02-07 12:34:00 → 2021-02-06 12:30:00, 2021-02-14 12:30:00

With roll enabled and the same configuration, a date at the end of a month will wrap around to the beginning of the month. Dates may mask as follows:

- 1905-12-10 00:00:00, 1907-08-01 10:14:00 → 1905-12-13 00:00:00, 1907-08-04 00:00:00
- 2001-07-31 23:45:30, 2005-04-12 07:13:00 → 2001-07-03 23:45:30, 2005-03-18 23:45:30
- 2021-02-03 12:30:00, 2021-02-07 12:34:00 → 2021-02-06 12:30:00, 2021-02-14 12:30:00

8.1.9.8 Email (Algorithm frameworks)

The **Email** framework masks string values by splitting the input on the @ symbol, independently masking the name and domain portions of the email address. Masked values are calculated algorithmically using the algorithm's key, so rekeying the algorithm will cause different outputs to be generated for each input. All inputs to this framework are valid and the framework will not generate non-conformant data events. Note that it is possible for chained algorithms specified for the *Algorithm* option to generate non-conformant data events.



8.1.9.8.1 Malformed input handling

- **Inputs without an @ symbol:** apply the name action to the entire input
- **Inputs with no name portion:** apply the domain action to the entire input
- **Inputs with no domain portion:** apply the name action to the entire input
- **Inputs with no name portion and no domain portion:** return an @ symbol
- **Empty or null input:** return input value

8.1.9.8.2 Creating an email algorithm via UI

1. At the top right of the **Algorithms** page, click **+ Algorithm**.

Add Algorithm ×

- Details
- Configuration
- Summary

Details

Specify the algorithm details.

Name
test_email

Description

Framework Name
Email

Mask Type
STRING

Email Framework: This is the framework to cover the scenarios where it is required to mask email with deterministic or unique masking results

Framework Options:
Mask Name With:

- **Unique Value** - Apply SHA-256 hash of the entire input then base 32 encodes the hash value which becomes the new **Name** part of the input.
- **Lookup value** - A file reference with list of values for masking **Name** part of the input
- **Algorithm** - String type Extensible Algorithm instance to be used to mask **Name** part of the input
- **Multi Algorithm** - Two string type Extensible Algorithm instances to be used to mask **First Name** and **Last Name** part of the input independently

Lookup File for Name (Conditional): Select button or text field to provide lookup file for email **Name** Part masking.

Cancel Back Next Save

2. Enter an **Algorithm Name**.
Info: This MUST be unique.
3. Enter a **Description**.
4. Select **Email** as the Framework Name and click **Next**.

Add Algorithm



○ Details

● Configuration

○ Summary

Configuration

Configure the algorithm.

Mask Name With
Unique Value

Mask Domain With
Replacement text

Domain Replacement
abc.com

Error Handling Action
Throw Exception (Default)

Filter Character Accent Marks

Email

Framework Description

Email Framework: This is the framework to cover the scenarios where it is required to mask email with deterministic or unique masking results

Framework Options:

Mask Name With:

- **Unique Value** - Apply SHA-256 hash of the entire input then base 32 encodes the hash value which becomes the new **Name** part of the input.
- **Lookup value** - A file reference with list of values for masking **Name** part of the input
- **Algorithm** - String type Extensible Algorithm instance to be used to mask **Name** part of the input
- **Multi Algorithm** - Two string type Extensible Algorithm instances to be used to mask **First Name** and **Last Name** part of the input independently

Lookup File for Name (Conditional): Select button or text field to provide lookup file for email **Name** Part masking.

Select Algorithm for Name (Conditional): DropDown to select Algorithm for email **Name** Part masking.

Cancel Back **Next** Save

5. From the dropdown **Mask Name With**, choose one of the following options:
 - **Unique Value:** applies a SHA-256 hash of the entire input then Base32 encodes the hash value.
 - **Lookup Value:** applies a secure lookup using the values provided in the uploaded file or file reference.
 - **Algorithm:** applies the specified string type extensible algorithm.
 - The **Unique Value** option may produce masked name portions with lengths of up to 52 characters.
 - **Multi Algorithm:** applies specified string algorithms to first name and last name segments. Segments must precede the @ symbol and be delimited by email-allowed punctuation.
6. If applicable, complete the configuration for masking the name portion as follows:
 - **Lookup Value:** upload a lookup file with new line-separated values or provide a file reference.
 - **Algorithm:** select a string-type extensible algorithm to be used to mask the name portion of the input.
7. From the dropdown **Mask Domain With**, choose one of the following options:
 - **Replacement Text:** replaces the domain portion with a fixed value.
 - **Algorithm:** applies the specified extensible algorithm instance.
 - **Preserve Value:** does not mask domain portion.
8. Complete the configuration for masking the domain portion as follows:
 - **Replacement Text:** enter a value to replace the entire domain portion.
 - **Algorithm:** applies the specified extensible algorithm instance.
9. Select an error handling action for exceptions thrown by chained algorithms:

- **Exception:** default action allowing exceptions from chained algorithms to be handled by global settings.
 - **Character Mapping:** masks the name or domain segment causing the exception with the **CM Alphanumeric** algorithm.
 - **Preserve Value:** returns the exception causing segment unmasked.
10. Choose whether to filter accents:
- **Filter Accents** (checkbox): selecting true will remove the accent from characters in the email address and replace them with the base Latin character prior to masking, such as converting `é` to `e`.
11. Click **Next** to verify details on the Summary step.

Add Algorithm ✕

- Details
- Configuration
- Summary

Summary

View the algorithm details below. Click the Back button to make changes or click the Save button to save the algorithm configuration.

Details

Name
test_email

Framework Name
Email

Mask Type
STRING

Configuration

Mask Name With
Unique Value

Mask Domain With
Replacement text

Domain Replacement
abc.com

Error Handling Action
Throw Exception (Default)

Filter Character Accent Marks
false

Cancel Back Next Save

12. Click **Save**.

For information on creating Email algorithms through the API, visit the [API Calls for Creating Algorithms - Email](#) (see page 910) page.

8.1.9.8.3 Examples

As an example, an Email algorithm that uses *Lookup Value* to mask the name portion and *Replacement Text* to mask the domain portion with the following configuration:

Lookup File:

Amy
Bob
Jake
Katherine

Replacement text: example.com

May mask as follows:

- "albert@delphix.com" → "Bob@example.com"
- "albert@gmail.com" → "Bob@example.com"
- "andrew_smith_123@delphix.com" → "Katherine@example.com"

Another example that uses the *Algorithm* option for both the name and domain portion with the following configuration:

Name algorithm: [dlpx-core:FirstName](#) (see page 677)

Domain algorithm: [dlpx-core:CM Alpha-Numeric](#) (see page 672)

May mask as follows:

- "bob@gmail.com" → "alton@dqpnx.fsy"
- "bob@hotmail.com" → "alton@poatzdw.bya"
- "alex@gmail.com" → "jameel@dqpnx.fsy"
- "joe_123@yahoo.com" → "miryam@wbpaq.kts"



The Email framework will not generate non-conformant data events, but the chained algorithm may generate such events.

All inputs with the same value masked with the same algorithm configuration will result in the same output values.

8.1.9.9 Free Text Redaction (Algorithm frameworks)

The Free Text Redaction Algorithm Framework is designed to effectively remove sensitive information from unstructured text fields, such as "Notes" columns in databases. Utilizing this framework involves configuring the algorithm to recognize and mask sensitive data embedded within text, which requires some expertise.

To identify the sensitive information, the algorithm relies on a list of predetermined lookup words associated with the type of data to be masked. For instance, to redact addresses, users might configure the algorithm to search for key indicators like "St," "Cir," or "Blvd." Additionally, the framework supports pattern matching to flag potentially sensitive data patterns. A common example is the pattern "123-45-6789" used to detect Social Security Numbers.

Both lookup words and regular expressions employed by the algorithm focus on matching entire words within the text. Consequently, even if a regular expression is set to match a part of a word, the algorithm is programmed to redact the entire word, ensuring that partial data elements are not inadvertently exposed. This method guarantees that the sensitive data is fully obscured, enhancing the security of the masked text.

8.1.9.9.1 Example

An example is provided below to help clarify how the information above works.

In this example, the redacted (masked) value is field_2 in a delimited file.

```
field_1,field_2,field_3
1,000123000,last
2,000123,000123,last
3,123000,last
4,000 123 000,last
5,000 123,last
6,123 000,last
7,123,last
```

Making it easier to read - I have only shown field 2 below.

Example1:

- RegEx: '123'
- Redacted with: 'xxx'

orig	redact	Comment
0000123000	0000123000	< Not redacted - 123 is part of the string.
0000123	0000123	< Not redacted - 123 is part of the string.123000
123000		< Not redacted - 123 is part of the string.
000 123	000 xxx	< Redacts the 'word' 123 to 'xxx'
000 123 000	000 xxx 000	< Redacts the 'word' 123 to 'xxx'
123	xxx	< Redacts the 'word' 123 to 'xxx'
000 123.	000 xxx.	< Redacts the 'word' 123 to 'xxx'

Example2:

- RegEx: '123.*'
- Redacted with: 'xxx'

orig	redact	Comment
0000123000	0000123000	< Not redacted - 123 is part of the string.
0000123	0000123	< Not redacted - 123 is part of the string.
123000	123000	< Redacts the 'word' 123000 to 'xxx'
000 123	000 xxx	< Redacts the 'word' 123 to 'xxx'
000 123 000	000 xxx 000	< Redacts the 'word' 123 to 'xxx'
123	xxx	< Redacts the 'word' 123 to 'xxx'
000 123.	000 xxx.	< Redacts the 'word' 123 to 'xxx'

Example3:

- RegEx: '.*123'
- Redacted with: 'xxx'

orig	redact	Comment
0000123000	0000123000	< Not redacted - 123 is part of the string.
0000123	0000123	< Redacts the 'word' 000123 to 'xxx'
123000	123000	< Not redacted - 123 is part of the string.
000 123	000 xxx	< Redacts the 'word' 123 to 'xxx'
000 123 000	000 xxx 000	< Redacts the 'word' 123 to 'xxx'
123	xxx	< Redacts the 'word' 123 to 'xxx'
000 123.	000 xxx.	< Redacts the 'word' 123 to 'xxx'

As can be seen - if we match a value ('word') the complete 'word' is redacted. The Free Text Redaction can't redact a segment of characters in a word.

Note:

In order to mask/redact a segment - one need to use a PlugIn (from TS).

You can use a Free Text Redaction Algorithm Framework to show or hide information by displaying either a "DenyList" or an "AllowList."

DenyList – Designated material will be redacted (removed). For example, you can set a deny list to hide patient names and addresses. The deny list feature will match the data in the lookup file to the input.

AllowList – ONLY designated material will be visible. For example, if a drug company wants to assess how often a particular drug is being prescribed, you can use an allow list so that only the name of the drug will appear in the notes.

8.1.9.9.2 Creating a free text redaction algorithm via UI

1. At the top right of the **Algorithms** page, click **+ Algorithm**.

Add Algorithm
✕

- Details
- Configuration
- Summary

Details

Specify the algorithm details.

Name

Description

Framework Name

Free Text Redaction

Mask Type
STRING

Free Text Redaction Framework: A Free Text Redaction Algorithm Framework helps you remove sensitive data that appears in free-text columns such as "Notes." This type of algorithm requires some expertise to use because you must set it to recognize sensitive data within a block of text.

The algorithm uses a list of lookup words to determine what information it needs to mask. You can decide which words the algorithm uses to search for material such as addresses. For example, you can set the algorithm to look for "St," "Cir," "Blvd," and other words that suggest an address. You can also use pattern matching to identify potentially sensitive information. For example, a number that takes the form 123-45-6789 is likely to be a Social Security Number. Lookup words and regular expressions will match individual words within the input text, rather than phrases.

You can use a Free Text Redaction Algorithm Framework to show or hide information by displaying either a "DenyList" or an "AllowList."

DenyList Designated material will be redacted (removed). For example, you can set a deny list to hide patient names and addresses. The deny list feature will match the data in the lookup file to the input.

Cancel
Back
Next
Save

2. Enter an **Algorithm Name**.

Info: This MUST be unique.

3. Enter a **Description**.
4. Select **Free Text Redaction** as the Framework Name and click **Next**.

Add Algorithm
✕

- Details
- Configuration**
- Summary

Configuration

Configure the algorithm.

Redact Type

Allow List

Lookup File Ⓢ

Select File temp.txt ✕

Successfully uploaded file.

Lookup File Redact Value

XXX

Regular Expressions Ⓢ ✎

There are no associated regular expressions.

Regular Expression Redact Value

XXX

Free Text Redaction

Framework Description

Free Text Redaction Framework: A Free Text Redaction Algorithm Framework helps you remove sensitive data that appears in free-text columns such as "Notes." This type of algorithm requires some expertise to use because you must set it to recognize sensitive data within a block of text.

The algorithm uses a list of lookup words to determine what information it needs to mask. You can decide which words the algorithm uses to search for material such as addresses. For example, you can set the algorithm to look for "St," "Cir," "Blvd," and other words that suggest an address. You can also use pattern matching to identify potentially sensitive information. For example, a number that takes the form 123-45-6789 is likely to be a Social Security Number. Lookup words and regular expressions will match individual words within the input text, rather than phrases.

You can use a Free Text Redaction Algorithm Framework to show or hide information by displaying either a "DenyList" or an "AllowList."

DenyList Designated material will be redacted (removed). For example, you can set a deny

Cancel
Back
Next
Save

5. Select a **Redact Type:** the **Deny List** or **Allow List**.
6. Select a **Lookup File** and enter a **Redaction Value** OR/AND
7. Enter **Regular Expressions** by clicking on ✎ using the List editor section. More information on the List Editor section can be found [here \(see page 232\)](#).
8. Enter a **Redaction Value** for Regular Expression.
9. Click **Next** to verify details on the Summary step.

Add Algorithm ×

- Details
- Configuration
- Summary

Summary

View the algorithm details below. Click the Back button to make changes or click the Save button to save the algorithm configuration.

Details

Name
test_ftr

Framework Name
Free Text Redaction

Mask Type
STRING

Configuration

Redact Type
Allow List

Lookup File
temp.txt

Lookup File Redact Value
XXX

Regular Expressions

Regular Expression Redact Value
XXX

Cancel Back Next Save

10. Click **Save**.

8.1.9.9.2.1 Existing limitations:

1. The maximum number of supported **Regular Expressions** is **50**. Exceeding this number will lead to the Component Configuration exception.
2. The maximum number of supported words in the **Lookup File** is **1000**. Exceeding this number may affect the algorithm performance.
3. The **Lookup File** format must be **txt**.
4. Every entry in the **Lookup File** must be a new line separated. Phrases are not supported. Case sensitive.
5. The maximum length of an input text to mask is **32768**. Exceeding this number will lead to the Non-Conformant data exception.

For information on creating Free Text Redaction algorithms through the API, see [API Calls for Creating Algorithms - Free Text Redaction](#). (see page 912)

8.1.9.9.3 Examples

Input:

The customer Bob Jones is satisfied with the terms of the sales agreement. Please call to confirm at 718-223-7896.

8.1.9.9.3.1 Algorithm configuration:

1. The Redact Type is **DenyList**
2. Lookup File entries:

```
Bob  
Jones  
agreement
```

3. The Lookup File Redaction Value is **XXXX**
4. Regular Expressions entry:

```
[0-9]{3}-[0-9]{3}-[0-9]{4}
```

- a. The Regular Expression Redaction Value is **YYYY**

8.1.9.9.3.2 Masking result:

```
The customer XXXX XXXX is satisfied with the terms  
of the sales XXXX. Please call to confirm at YYYY.
```

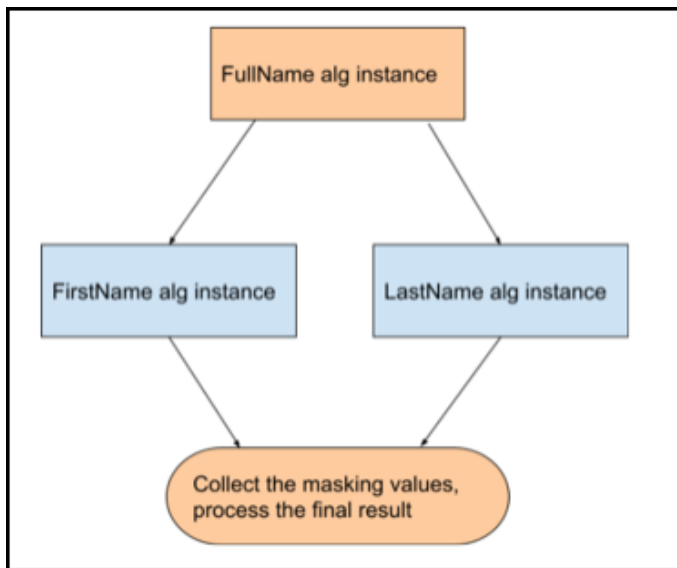
"Bob", "Jones", "agreement" and the phone number are redacted.

8.1.9.10 Full Name (Algorithm frameworks)

The Full Name algorithm (introduced in Masking Engine version 6.0.8.0) has logic to separate the input into two parts: First and Last names. It can also limit the number of masked names (removing the rest) and “smart trim” the result (masked) output to the required length.

If Name framework algorithm instances are used, the Full Name algorithm can consider particles. The Full Name algorithm uses the Last Name algorithm’s particles when determining which part of the input is the last name. The remainder of the input is considered the first name and the First Name algorithm’s particles apply to that.

After distinguishing the parts of the input string, the Full Name algorithm feeds each word from the first name part (which also includes middle names, treated the same as first names) individually to the First Name algorithm instance, and the whole last name part to the Last Name algorithm instance. Then it combines the masking results, according the embedded logic and the configuration.



If the input string contains only a single word - this word is considered as a first name or last name (depending on the **Consider Single Word Input as Last Name** flag) and forwarded for masking to the corresponding chained algorithm instance. A single word input is always masked, even if it is a configured particle.

Main features of the Full Name Framework:

- Deterministic output: The masked result for each input is consistent when using the same algorithm key, same configuration and same chained algorithm instances.
- Not unique: The masked result might be the same for different inputs.
- Garbage in garbage out: the algorithm returns the unmasked input / null / empty string if input is one of the following: null, empty string "", white spaces only " ".
- Single word input: considered either as a Last Name (default) or as a First Name, even if configured in one of the particles files.
- When particle is configured in both particles files: the remove action takes precedence.
- Multiple first names: masks only first N names (1-4, as configured, default = 2), the rest are removed. (Note: only one name can be considered the last name; the rest are masked as first names.)
- Full Name Convention: if a configured last name separator is detected, or the configured convention is "last-first-middle", then the input is interpreted as last-first-middle. Otherwise, it is first-middle-last (default). Leading/trailing and duplicate white spaces are not preserved.
- Smart trim: if trimming of the masked value is required, it's done in a way to keep the full name as long and realistic looking as possible. For instance, first we trim the leading/trailing preserved particles. If that's not enough, we abbreviate the masked first/middle names (one by one, starting with the last one). If that's still no enough, we remove the particles prior to the last name, etc.

Below is an example of smart trim. Let's suppose our masked result (prior to checking of the maxLength) is:

"President George Herbert Walker Van Bush Jr."

maxLengthOfMaskedFullName value	action	result
55	Nothing. The string is shorter.	President George Herbert Walker Van Bush Jr.
42	Cut particle at the end (if known)	President George Herbert Walker Van Bush
30	Cut particle at the beginning (if known)	George Herbert Walker Van Bush
26	Abbreviate FNs starting last	George Herbert W. Van Bush
22	Continue abbreviate FNs	George H. W. Van Bush
17	Continue abbreviate FNs	G. H. W. Van Bush
14	Cut FN abbreviation(s) starting last	G. H. Van Bush
11	Continue cutting abbreviations	G. Van Bush
8	Leave LN if possible	Van Bush
4	Leave LN if possible	Bush
2	Cut the LN from the end	Bu

The chained instances for First Name and Last name masking can be any existing extensible algorithm instance that masks the String type. However, it is recommended to use the instances based on the Name framework. Particles can only be considered for Name framework algorithms, and it's recommended that the First Name and Last Name algorithm instances have the same particle files.

8.1.9.10.1 Creating a Full Name algorithm via UI

1. At the top right of the **Algorithms** page, click **+ Algorithm**.

Add Algorithm



- Details
- Configuration
- Summary

Details

Specify the algorithm details.

Name
test_fn

Description

Framework Name
Full Name

Mask Type

STRING

Full Name Framework: This is the framework to cover the scenarios where it is required to mask a full name input string with deterministic and not unique masking results. The goal of a Full name masking framework is to mask the first, middle and last names consistently when they appear in the same field using algorithms that can be applied to the component parts (e.g. First Name alone).

Framework Options:

First Name Algorithm (Required): String type Extensible Algorithm instance to be used to mask first and middle name of the full name input.

Last Name Algorithm (Required): String type Extensible Algorithm instance to be used to mask last name of the full name input.

Maximum First Names: Total number of first/middle names to be masked, the rest would be ignored. Minimum value is 1, maximum is 4 and **default** is 2;

Maximum Masked Full Name Length: Max number of characters or length of masked output string. **default** is 0, which means unlimited.

Cancel Back **Next** Save

2. Enter an **Algorithm Name**.

Info: This MUST be unique.

3. Enter a **Description**.

4. Select **Full Name** as the Framework Name and click **Next**.

Add Algorithm



- Details
- Configuration
- Summary

Configuration

Configure the algorithm.

First Name Algorithm
dlpx-core:FirstName

Last Name Algorithm
dlpx-core:LastName

Maximum First Names
2

Maximum Masked Full Name Length
0

Full Name Convention
First-Middle-Last

Consider Single Word Input as Last Name

Last Name Separators

• ,

FullName

Framework Description

Full Name Framework: This is the framework to cover the scenarios where it is required to mask a full name input string with deterministic and not unique masking results. The goal of a Full name masking framework is to mask the first, middle and last names consistently when they appear in the same field using algorithms that can be applied to the component parts (e.g. First Name alone).

Framework Options:

First Name Algorithm (Required): String type Extensible Algorithm instance to be used to mask first and middle name of the full name input.

Last Name Algorithm (Required): String type Extensible Algorithm instance to be used to mask last name of the full name input.

Maximum First Names: Total number of first/middle names to be masked, the rest would be ignored. Minimum value is 1, maximum is 4 and **default** is 2;

Maximum Masked Full Name Length: Max number of characters or length of masked output string. **default** is 0, which means unlimited.

Full Name Convention: A flag to configure the

Cancel Back **Next** Save

5. Choose the **First Name Algorithm**. (Required) In the dropdown menu, you will be suggested to choose from the existing extensible algorithms of String type.

6. Choose the **Last Name Algorithm**. (Required) In the dropdown menu, you will be suggested to choose from the existing extensible algorithms of String type.
7. Choose the **Maximum First Names** configuration. (Optional. Integer. min value = 1, max value = 4, default = 2) Total number of first/middle names to be masked. The rest would be ignored.
8. Choose the **Maximum Masked Full Name Length**. (Optional. Integer. Default is 0) This number should be ≥ 0 (i.e. not negative). This is the maximum number of characters for the masked output string. The masked result is "smart trimmed" to fit that length (see explanation and example table above for details on smart trimming). Value 0 means length is unlimited.


Info:

We also try to detect the length of the destination field. Some Data Sources provide that value, while others don't. For example: if Data Source provides value **10** for the destination column length and current configuration field is set to **0** or any value longer than 10 - the shortest value wins, i.e. in this example masked result would be trimmed to 10 characters.

9. Specify a **Full Name Convention**. (Optional. Enum. Default: "First-Middle-Last") The dropdown menu provides a choice of 2 values:

First-Middle-Last
Last-First-Middle

This configuration helps the Full Name algorithm distinguish between first name(s) and last name if the **Last Name Separator(s)** are not configured or not detected in the input string.

10. Choose the **Consider Single Word Input as Last Name**. (Optional. Boolean. Default is true) If chosen (default case) - consider the single word input as a last name. Otherwise as a first name.
11. Configure **Last Name Separators** (Optional. List. Default: contains comma ',') by clicking on  using the List editor section. More information on List Editor section can be found [here \(see page 232\)](#).

Here you can add a single punctuation mark (but hyphen '-' and dot '.', which are reserved for another logic) which will serve for identifying the last name in the input. The first identified separator makes that distinguishing, the rest are ignored.

By default, a comma is included as a separator.

Last Name Separators 

,

+
+ 

×
✓

Here is an example of how the last name separator works:

Let's suppose our configured separators are comma ',' and colon ':':

Input: "dela Cruz, Maria Cristina: Manansala"

The first detected separator (framework reads the input left to right) is after the word "Cruz".

So "dela Cruz" will be detected as a last name part, and "Maria Cristina: Manansala" as a first name.

The masking result would be in the same order with the same separator, for example: "Maritnas, Antonio Stephan".

- Click **Next** to verify details on the Summary step.

Add Algorithm

×

- Details
- Configuration
- Summary

Summary

View the algorithm details below. Click the Back button to make changes or click the Save button to save the algorithm configuration.

Details

Name
test_fn

Framework Name
FullName

Mask Type
STRING

Configuration

First Name Algorithm
dlpx-core:FirstName

Last Name Algorithm
dlpx-core:LastName

Maximum First Names
2

Maximum Masked Full Name Length
0

Full Name Convention
First-Middle-Last

Consider Single Word Input as Last Name
True

Last Name Separators
,

Cancel
Back
Next
Save

- Click **Save**.

For information on creating Full Name algorithms through the API, see [API Calls for Creating Algorithms - Full Name](#). (see page 914)

8.1.9.11 Mapping (Algorithm frameworks)

A Mapping algorithm allows you to state what fictitious values will replace the original data. It maps original data values to masked values pre-populated in a lookup table through the Continuous Compliance Engine user interface. There will be no collisions in the masked data because it always matches the same input to the same output. For example, *David* will always become *Raj* and *Melissa* will always become *Jasmine*. The algorithm checks whether an input has already been mapped; if so, the algorithm changes the data to its designated output.

The Mapping algorithm can be used on any set of values, of any length, but you must know how many values are being masked. To that end, provide **at least** the same amount of values as there are unique values being masking; though, more is acceptable. For example, if there are 10,000 unique values in the column being masked, you must give the Mapping algorithm **at least** 10,000 values.

The Mapping algorithm can be configured for mappings managed locally on the Continuous Compliance Engine or remotely on a user-managed PostgreSQL database. Remote mapping should be used for those who want to manage the storage allocated for mappings or share the same mappings from this algorithm across multiple Continuous Compliance Engines. More information about remote mapping can be found in the [Remote mapping](#)³²⁸ page.

- i** **Continuous Compliance Engine 6.0.9.0 and earlier:** When you use a Mapping algorithm, you cannot mask more than one table at a time. You must mask tables serially.
- Continuous Compliance Engine 6.0.10.0 and later:** A single Mapping algorithm can have multiple jobs running concurrently.

8.1.9.11.1 Tokenization/Reidentification

Mapping algorithms can be used with Tokenization and Reidentification jobs. However, if `ignoreCharacters` are configured for the algorithm, Tokenization/Reidentification cannot be used.

8.1.9.11.2 Sync

Mapping algorithms can be synced in one of two ways:

1. **Syncing a locally managed Mapping algorithm:** This can be done to effectively *make a copy* of an algorithm from one Continuous Compliance Engine to another. In addition to syncing the algorithm, the mappings must be manually exported from the source engine and imported into the target engine. Once this is complete, the two algorithms (on the source and target) will have the same names and initial set of mappings (at the time of sync) but will function as two separate algorithms. That is to say, adding new mappings on the source *will not* have any impact on the algorithm on the target.
2. **Syncing a remotely managed Mapping algorithm:** This can be done to *share* the same Mapping algorithm across Continuous Compliance Engines. In this case, once synced, the algorithm on the source and target(s) would point to the SAME remote mapping database. This would mean that adding/removing/manipulating the mappings would affect the algorithm on all engines that use it.

More information on Sync can be found on the [Introduction \(Managing multiple engines for masking\)](#)³²⁹ page.

³²⁸ <https://masking.delphix.com/docs/latest/remote-mapping>

³²⁹ <https://masking.delphix.com/docs/latest/introduction-managing-multiple-engines-for-masking>

8.1.9.11.3 Creating a mapping algorithm via the UI

i If there are insufficient mappings available when running a masking job, the job will fail and produce an error, similar to the one shown below:

```
[JOB_ID_1797_292] WARNING-UNMASKED-DATA: RED_Mapping_Name failed to mask data at tcedCUST.CUST_NAME: Failed to map value. No free mappings are available. (4000000+ occurrences)
```

To resolve this issue, add more mappings to the algorithm framework before rerunning the job.

1. At the top right of the **Algorithms** page, click **+ Algorithm**.

Add Algorithm

×

- Details
- Configuration
- Summary

Details

Specify the algorithm details.

Name

Description

Framework Name

Mapping
▾

Mask Type
STRING

Mapping Framework: A Mapping algorithm allows you to state what fictitious values will replace original data. It maps original data values to masked values pre-populated in a lookup table through the Continuous Compliance Engine user interface. There will be no collisions in the masked data, because it always matches the same input to the same output. For example, David will always become Raj and Melissa will always become Jasmine. The algorithm checks whether an input has already been mapped; if so, the algorithm changes the data to its designated output.

The Mapping algorithm can be used on any set of values, of any length, but you must know how many values are being masked. To that end, provide at least the same amount of values as there are unique values being masking; though, more is acceptable. For example, if there are 10,000 unique values in the column being masked, you must give the Mapping algorithm at least 10,000 values.

The Mapping algorithm can be configured for mappings managed locally on the Continuous Compliance Engine or remotely on a user-managed PostgreSQL database. Remote mapping should be used for those who want to manage the storage allocated for mappings or share the same mappings

Cancel
Back
Next
Save

2. Enter an **Algorithm Name**.
Info: This MUST be unique.
3. Enter a **Description**.
4. Select **Mapping** as the Framework Name and click **Next**.

Add Algorithm



- Details
- Configuration
- Summary

Configuration

Configure the algorithm.

Local Mapping Store

Ignore Characters (separated by comma)

Ignore Commas (,)

Mapping

Framework Description

Mapping Framework: A Mapping algorithm allows you to state what fictitious values will replace original data. It maps original data values to masked values pre-populated in a lookup table through the Continuous Compliance Engine user interface. There will be no collisions in the masked data, because it always matches the same input to the same output. For example, David will always become Raj and Melissa will always become Jasmine. The algorithm checks whether an input has already been mapped; if so, the algorithm changes the data to its designated output.

The Mapping algorithm can be used on any set of values, of any length, but you must know how many values are being masked. To that end, provide at least the same amount of values as there are unique values being masking; though, more is acceptable. For example, if there are 10,000 unique values in the column being masked, you must give the Mapping algorithm at least 10,000 values.

Cancel Back Next Save

5. Select whether or not the mappings will live locally or remotely by toggling the **Local Mapping Store** checkbox appropriately. If using a local mapping store, proceed to step 8. For more information about remote mapping stores, visit the [Remote mapping](#)³³⁰ page.
6. Specify **Hostname/IP**, **Port**, **Mapping Database**, and **Schema** of the remote database.

Add Algorithm



- Details
- Configuration
- Summary

Configuration

Configure the algorithm.

Local Mapping Store

Hostname/IP

Port

Database

Schema

Mapping Connection Properties ⓘ

[Select File](#) File not selected

Ignore Characters (separated by comma)

Ignore Commas (,)

Mapping

Framework Description

Mapping Framework: A Mapping algorithm allows you to state what fictitious values will replace original data. It maps original data values to masked values pre-populated in a lookup table through the Continuous Compliance Engine user interface. There will be no collisions in the masked data, because it always matches the same input to the same output. For example, David will always become Raj and Melissa will always become Jasmine. The algorithm checks whether an input has already been mapped; if so, the algorithm changes the data to its designated output.

The Mapping algorithm can be used on any set of values, of any length, but you must know how many values are being masked. To that end, provide at least the same amount of values as there are unique values being masking; though, more is acceptable. For example, if there are 10,000 unique values in the column being masked, you must give the Mapping algorithm at least 10,000 values.

The Mapping algorithm can be configured for mappings managed locally on the Continuous Compliance Engine or remotely on a user-managed PostgreSQL database. Remote mapping should be used for those who want to manage the storage allocated for mappings or share the same mappings from this algorithm across multiple Continuous Compliance Engine. More

Cancel Back Next Save

330 <https://masking.delphix.com/docs/latest/remote-mapping>

7. Enter any remaining connection parameters in a properties file specified by the **Mapping Connection Properties** field.
8. To ignore specific characters, enter one or more characters in the **Ignore Character** box. Separate values with a comma.
9. To ignore the comma character (,), select the **Ignore comma (,)** checkbox.
10. Click **Next** to verify details on the Summary step.

Add Algorithm ×

- Details
- Configuration
- Summary

Summary

View the algorithm details below. Click the Back button to make changes or click the Save button to save the algorithm configuration.

Details	Configuration
<p>Name test_map</p> <p>Framework Name Mapping</p> <p>Mask Type STRING</p>	<p>Ignored Characters @, #</p> <p>Ignore Commas false</p> <p>Local Mapping Store true</p>

Cancel Back Next Save

11. When you are finished, click **Save**.

Before you can use the algorithm by specifying it in a profiling job, you must add it to a domain. If you are not using the Continuous Compliance Engine Profiler to create your inventory, you do not need to associate the algorithm with a domain.

For information on creating Mapping algorithms through the API, visit the [Mapping \(API client\)](#)³³¹ page.

8.1.9.11.4 Managing mappings via UI

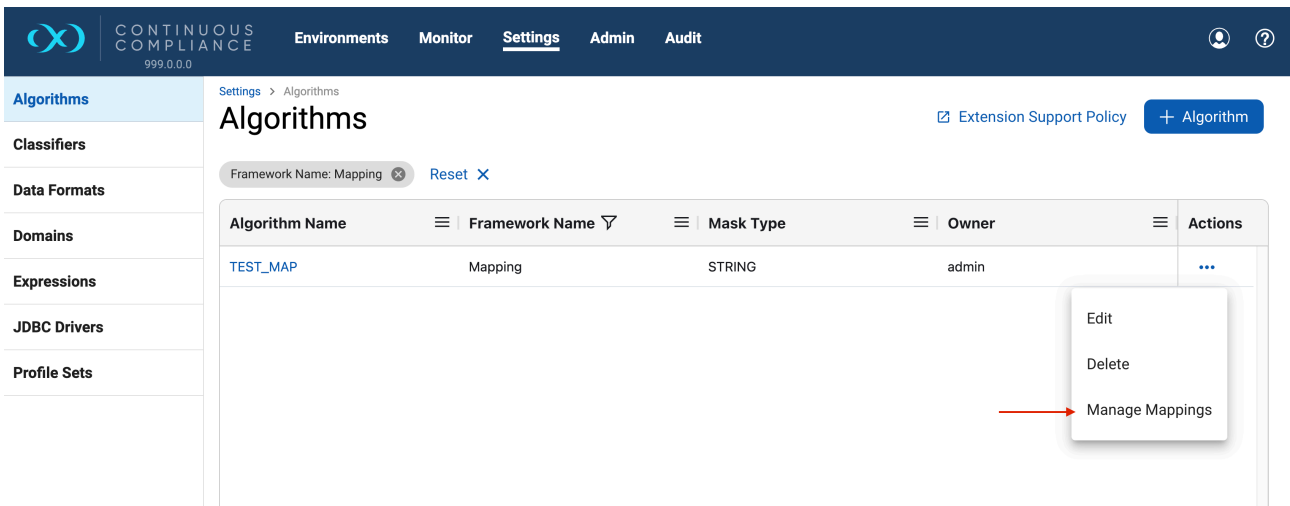
Regardless of where the mappings reside (local or remote), the management process is the same. Use the UI to perform options like import/export, delete, or reset mappings.

- i** These tasks can only be performed by a user with sufficient privileges per each task, as follows:
- **Export mappings**
 - `admin` privileges required.

³³¹ <https://masking.delphix.com/docs/latest/mapping>

- A passphrase is required, meaning exports will be encrypted.
- Due to the encryption, it will not be possible to see the allocated mappings.
- **Import mappings**
 - `algorithm: update` privileges required.
- **Delete mappings**
 - `algorithm: update` privileges required.
- **Reset mappings**
 - `algorithm: update` privileges required.

To Manage Mappings, click the (...) button to the right of the corresponding Mapping algorithm row under the **Actions** column and select the **Manage Mappings** option.



At the top, there are two statistics provided for the mappings:

1. **Total Mappings** is the number of mapping outputs that exist for this algorithm.
2. **Available Mappings** is the number of mappings that have not yet been assigned to an input value.

Import Mappings ✕

Total Mappings : 0

Available Mappings : 0

Action
Import Mappings

File Type
CSV

Import Mapping/Outputs Upload Mapping File ⓘ

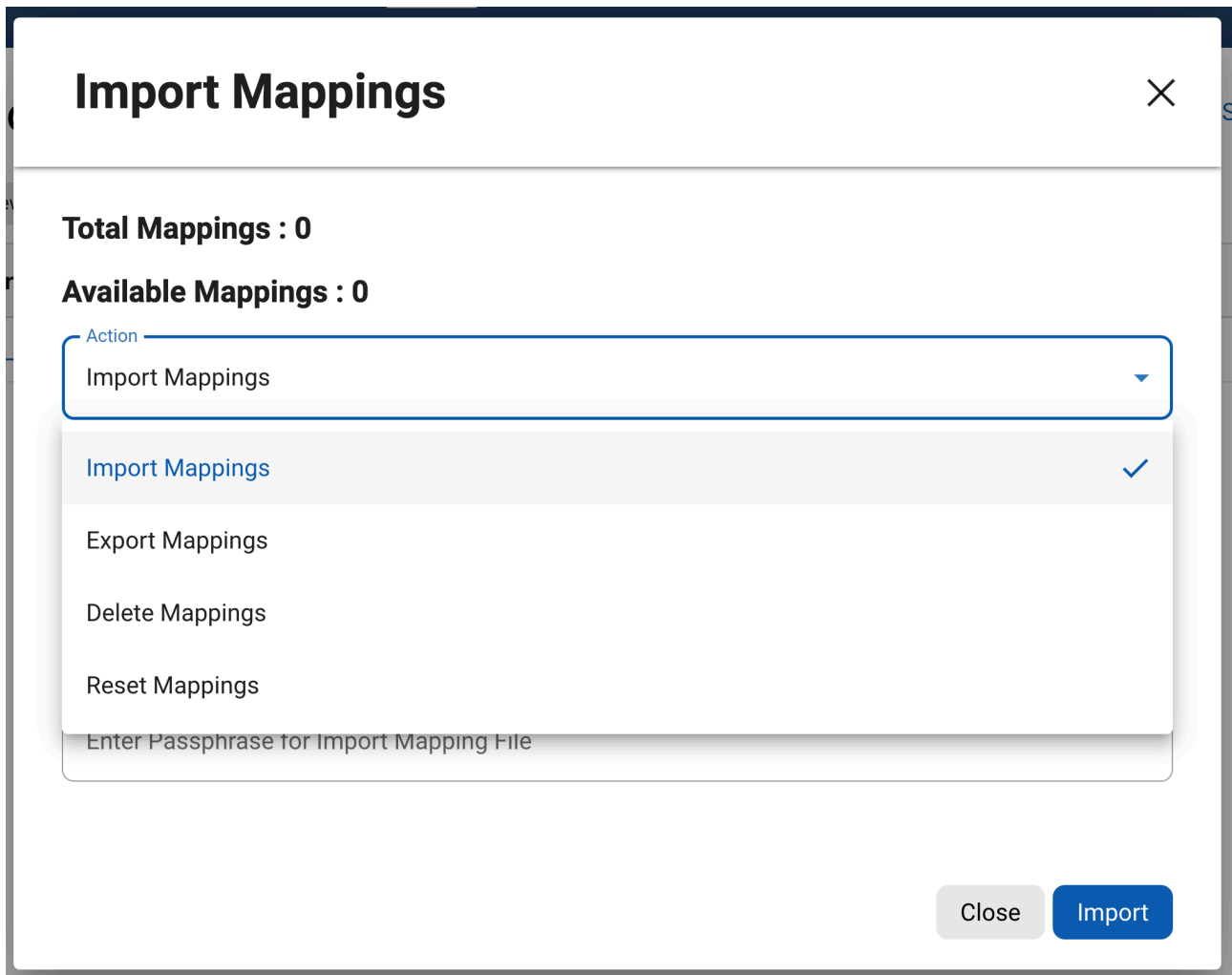
[Select File](#) File not selected

Enter Passphrase for Import Mapping File

Close Import

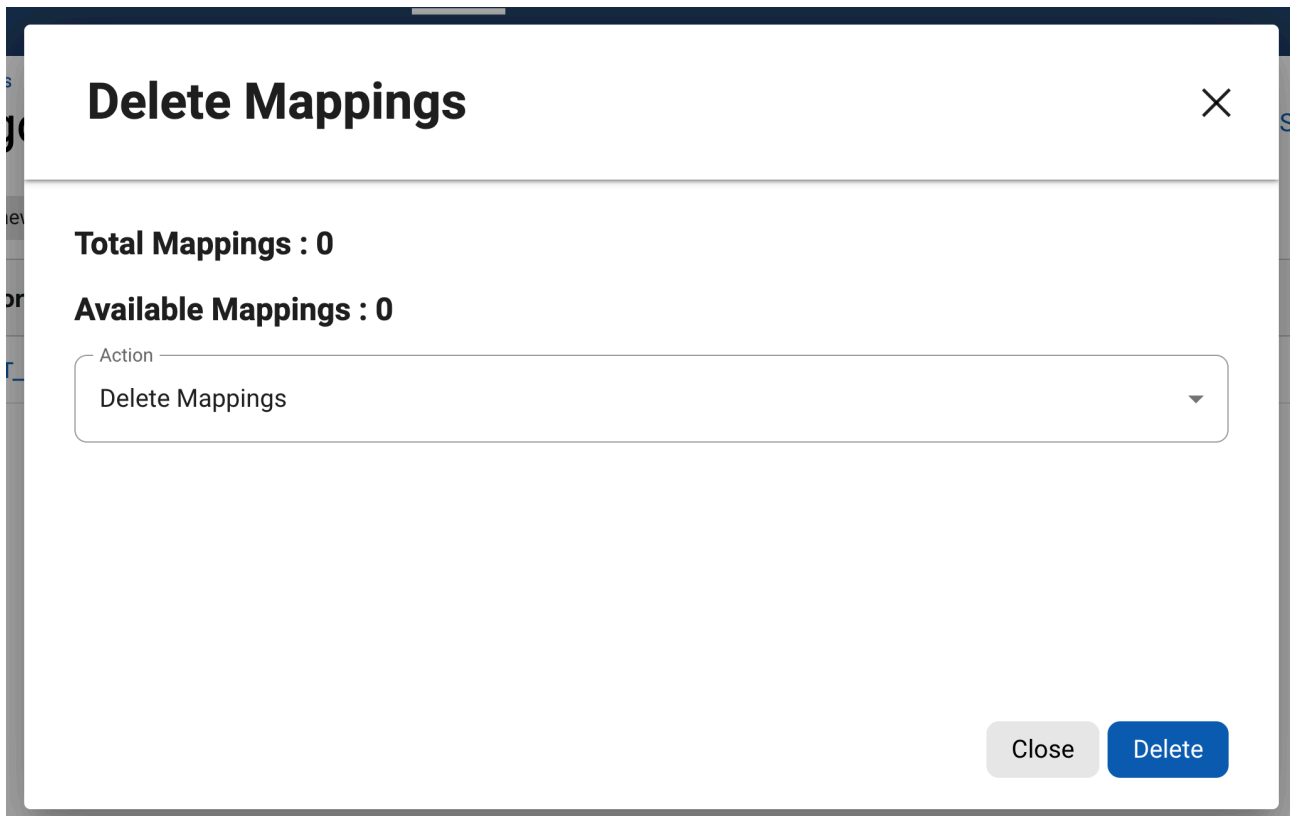
i When a job using the Mapping algorithm runs, the mappings are loaded into memory. This means that enough memory must be provided to the job to load the mappings. A Mapping algorithm with 2GB worth of mappings will require a job with a larger configured XMX than what is needed for a Mapping algorithm with 2MB worth of mappings.

In addition to mapping statistics, the import/export, delete, or reset mappings can be performed by selecting Action.



8.1.9.11.4.1 Delete mappings

This action will delete all input/output combinations and effectively start this algorithm fresh. For this option to take effect, select the **Manage Mappings** option, then select Action **Delete Mappings**.



8.1.9.11.4.2 Export mappings

This action will export all mappings into a file that can then be used to seed another mapping algorithm or exist as a backup list of established mappings. For security purposes, a passphrase is required to encrypt the file on export.

To export mappings, select the **Export Mappings** action and provide a **passphrase**, then click **Export**.

i If you wish to decrypt the exported file from the command line, run the following command:

```
openssl enc -aes-128-cbc -a -d -pass stdin -pbkdf2 -iter 100000 -md  
SHA256 -in PATH_TO_EXPORT_FILE
```

Export Mappings ✕

Total Mappings : 0

Available Mappings : 0

Action

Export Mappings ▾

Enter Passphrase for Export Mapping File

Close Export

Once the export is clicked, it will start the Async Task for exporting mappings. File download option will be available on the 'Monitor > Async Task' Page.

Export Mappings ✕

i Started Async task with ID "25". You can check the export status by navigating to 'Monitor > Async Tasks'.

Total Mappings : 0

Available Mappings : 0

Action

Export Mappings ▾

Enter Passphrase for Export Mapping File

avc

Close Export

8.1.9.11.4.3 Import mappings

This action will add mappings to the mapping algorithm. Mappings can be provided in two different formats; **PLAINTEXT** and **CSV**.

Import Mappings ✕

Total Mappings : 0

Available Mappings : 0

Action

Import Mappings
▾

File Type

CSV
▾

Import Mapping/Outputs Upload Mapping File ?

[Select File](#) File not selected

Enter Passphrase for Import Mapping File

Close
Import

PLAINTEXT

A PLAINTEXT mapping file can ONLY provide mapping outputs (i.e.: values you want to mask to). The file must have NO header. Make sure there are no spaces or returns at the end of the last line in the file.

The following is a sample PLAINTEXT mapping file. Notice that there is no header and only a list of values.

```


Smallville
Clarkville
Farmville
Townville
Cityname
Citytown
Towneaster
    
```

CSV


A CSV mapping file can provide both mapping inputs and outputs. That is, you can determine beforehand what you want your mappings to be. The CSV file must have only two columns – input and output. The first line of the file must be the header `input,output`. Make sure there are no spaces or returns at the end of the last line in the file.

The following is a sample CSV mapping file.

```
input,output
New York,Smallville
Boston,Clarkville
San Francisco,Townville
"",Cityname
"",Citytown
"",Towneast
```

 An input value does not have to be specified, but an output value must be specified for a line to be considered valid. Invalid lines are silently ignored.

Once a **File Type** is selected, choose the mapping file in the **Import Mappings/Outputs** field.

 If providing a previously exported mapping file that has been encrypted with a passphrase, select the **CSV** file type, provide the *unaltered* encrypted file, and provide a **passphrase**.

When the appropriate selections have been made, click **Import**.

 Any duplicate values provided will be silently ignored.

8.1.9.11.4.4 Reset mappings

This action will delete all inputs for provided mappings, giving you a mapping algorithm with as many outputs as you had before, but with all of them available for assignment the next time the mapping algorithm is used.

Reset Mappings ×

Total Mappings : 0

Available Mappings : 0

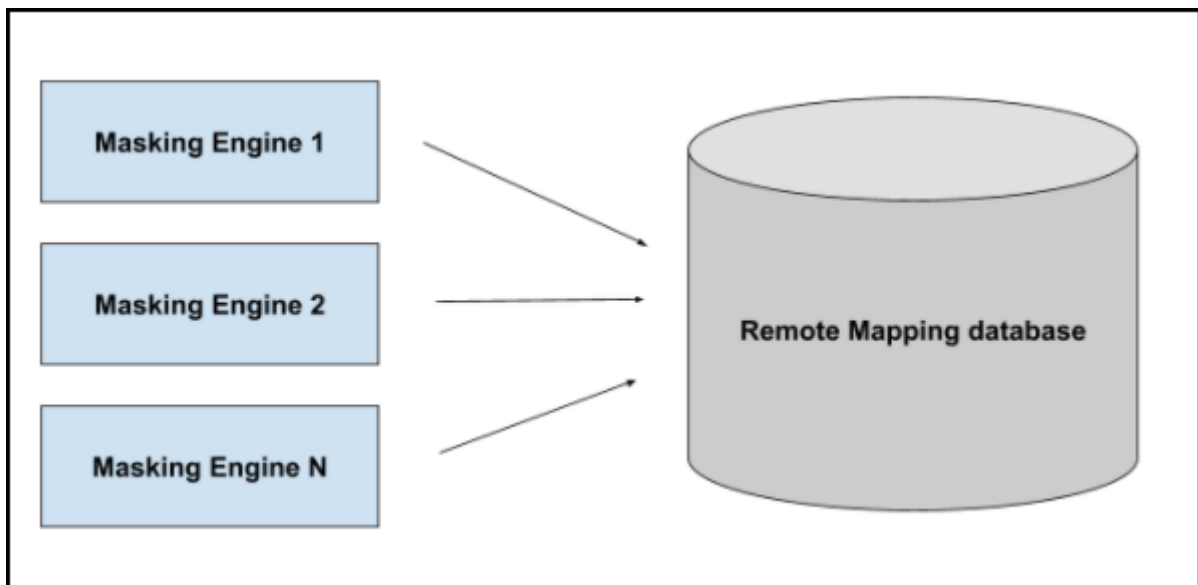
Action

Reset Mappings ▼

Close Reset

8.1.9.11.5 Remote Mapping

With the release of version 6.0.10.0 of the Masking Engine, the Mapping Algorithm now provides support for storing all mappings on a user-supplied database. This enables users to share mappings for the same Mapping Algorithm across engines. The mapping database connection info can be provided when a Mapping Algorithm is added or edited.



In order to serve as a mapping database, the following requirements must be met:

- The database must be a PostgreSQL database version 9.5 or newer.
- The database must be reachable by the Masking Engine

All necessary tables and functions to successfully run the Mapping Algorithm will be created by the Masking Engine upon connection to the remote mapping database.

Remote mappings are managed in the same way as local mappings via the Masking Engine GUI or APIs.

i It is completely fine to use the same remote database for multiple Mapping Algorithms on the same Masking Engine or across many Masking Engines.

i Given that the Masking Engine will need to query the remote database, network latency will have an effect on how fast a job running a Mapping Algorithm will run, especially on the "initial" run of a Mapping Algorithm when the majority of new mappings are established.

8.1.9.11.5.1 Expectations

By opting to manage their own mappings, the user agrees to be responsible for:

- Database uptime
- Database security
- Network connectivity
- Database storage

8.1.9.11.5.2 Configuring the connection

The user may opt to configure their PostgreSQL database however they wish. With the exception of host, port, database and schema, all other connection properties may be provided via a properties file, per the [PostgreSQL JDBC Driver documentation](#)³³².

For databases with SSL/TLS connections, the correct properties should be supplied via the properties file.

8.1.9.12 Min Max Number (Algorithm frameworks)

The Continuous Compliance Engine provides a "MinMax Number" to normalize data within a range. Values that are extremely high or low in certain categories allow viewers to infer someone's identity, even if their name has been masked. For example, a record with the highest salary might be for a company's CEO. You can use a MinMax Number algorithm to move all values of this kind into the midrange. This algorithm allows you to make sure that all the values in the database are within a specified range. The algorithm framework applies to numeric data types.

The **Replacement Value for Nonconforming Data** value is used when the underlying data to be masked is of type String and conversion to a number is required.

³³² <https://jdbc.postgresql.org/documentation/datasource/>

8.1.9.12.1 Creating a MinMax Number Algorithm via UI

1. At the top right of the **Algorithms** page, click **+ Algorithm**.

Add Algorithm ×

- Details
- Configuration
- Summary

Details

Specify the algorithm details.

Name
test_mmn

Description

Framework Name
MinMax Number

Mask Type
BIG_DECIMAL

MinMax Number Framework: The Continuous Compliance Engine provides Min Max algorithm frameworks normalize data within a range. Values that are extremely high or low in certain categories allow viewers to infer someone's identity, even if their name has been masked. For example, a salary of \$1 suggests a company's CEO. You can use a Min Max algorithm to move all values of this kind into the midrange. This algorithm allows you to make sure that all the values in the database are within a specified range. This algorithm frameworks is applicable to date data type.

Cancel Back **Next** Save

2. Enter an **Algorithm Name**.
Info: This MUST be unique.
3. Enter a **Description**.
4. Select **MinMax Number** as the Framework Name and click **Next**.
5. Enter the **Minimum Number** and the **Maximum Number**.

Add Algorithm



- Details
- Configuration**
- Summary

Configuration

Configure the algorithm.

Minimum Number

Maximum Number

Replacement Value for Nonconforming Data

MinMax Number

Framework Description

MinMax Number Framework: The Continuous Compliance Engine provides Min Max algorithm frameworks normalize data within a range. Values that are extremely high or low in certain categories allow viewers to infer someone's identity, even if their name has been masked. For example, a salary of \$1 suggests a company's CEO. You can use a Min Max algorithm to move all values of this kind into the midrange. This algorithm allows you to make sure that all the values in the database are within a specified range. This algorithm frameworks is applicable to date data type.

The **Replacement Value for Nonconforming Data** value is used when the underlying data to be masked is of type String and conversion to a date is required.

Cancel Back Next Save

6. Enter the **Replacement Value for Nonconforming Data** if needed.
7. Click **Next** to verify details on the Summary step.

Add Algorithm



- Details
- Configuration
- Summary**

Summary

View the algorithm details below. Click the Back button to make changes or click the Save button to save the algorithm configuration.

Details

Name
test_mmn

Framework Name
MinMax Number

Mask Type
BIG_DECIMAL

Configuration

Minimum Number
3

Maximum Number
7

Replacement Value for Nonconforming Data
1

Cancel Back Next Save

8. Click **Save**.

For information on creating Min Max algorithms through the API, see [API Calls for Creating Algorithms - Min Max](#). (see page 918)

8.1.9.12.2 Examples

Example: Age less than 18 years - enter Min Number 0 and Max Number 18.

8.1.9.13 Min Max Date (Algorithm frameworks)

The Continuous Compliance Engine provides a "MinMax Date" to normalize dates within a range. Values that are extremely high or low in certain categories allow viewers to infer someone's identity, even if their name has been masked. For example, some age ranges suggest higher insurance risk. You can use a MinMax Date algorithm to move all values of this kind into the midrange. This algorithm allows you to make sure that all the values in the database are within a specified range. The algorithm framework applies to date data types.

The **Replacement Value for Nonconforming Data** value is used when the underlying data to be masked is of type String and conversion to a date is required.

8.1.9.13.1 Creating a MinMax Date Algorithm via UI

1. At the top right of the **Algorithms** page, click **+ Algorithm**.

Add Algorithm
✕

- Details
- Configuration
- Summary

Details

Specify the algorithm details.

Name

Description

Framework Name

MinMax Date
▼

Mask Type
LOCAL_DATE_TIME

MinMax Date Framework: The Continuous Compliance Engine provides Min Max algorithm frameworks normalize data within a range. Values that are extremely high or low in certain categories allow viewers to infer someone's identity, even if their name has been masked. For example, some age ranges suggest higher insurance risk. You can use a Min Max algorithm to move all values of this kind into the midrange. This algorithm allows you to make sure that all the values in the database are within a specified range. This algorithm frameworks is applicable to date data type.

Cancel
Back
Next
Save

2. Enter an **Algorithm Name**.
Info: This MUST be unique.
3. Enter a **Description**.
4. Select **MinMax Date** as the Framework Name and click **Next**.
5. Enter the **Minimum Date** and the **Maximum Date**.

Add Algorithm



- Details
- Configuration
- Summary

Configuration

Configure the algorithm.

Minimum Date

1/1/2013, 12:00:00 AM 📅

Maximum Date

1/1/2016, 12:00:00 AM 📅

Replacement Value for Nonconforming Data

MinMax Date

Framework Description

MinMax Date Framework: The Continuous Compliance Engine provides Min Max algorithm frameworks normalize data within a range. Values that are extremely high or low in certain categories allow viewers to infer someone's identity, even if their name has been masked. For example, some age ranges suggest higher insurance risk. You can use a Min Max algorithm to move all values of this kind into the midrange. This algorithm allows you to make sure that all the values in the database are within a specified range. This algorithm frameworks is applicable to date data type.

The **Replacement Value for Nonconforming Data** value is used when the underlying data to be masked is of type String and conversion to a date is required.

Cancel Back Next Save

6. Enter the **Replacement Value for Nonconforming Data** if needed.
7. Click **Next** to verify details on the Summary step.

Add Algorithm



- Details
- Configuration
- Summary

Summary

View the algorithm details below. Click the Back button to make changes or click the Save button to save the algorithm configuration.

Details

Name
test_mmd

Framework Name
MinMax Date

Mask Type
LOCAL_DATE_TIME

Configuration

Minimum Date
2013-01-01 00:00:00

Maximum Date
2016-01-01 00:00:00

Replacement Value for Nonconforming Data

Cancel Back Next Save

8. Click **Save**.

For information on creating Min Max algorithms through the API, see [API Calls for Creating Algorithms - Min Max](#) (see page 918).

8.1.9.14 Multi Column Address (Algorithm frameworks)

8.1.9.14.1 Overview

The Multi Column Address framework allows multi-column masking of address information for a physical location. The purpose of the framework is to take data across many columns, with inconsistent formatting or completion of individual columns make sure that input referring to a single physical address is masked to an output that corresponds to a single real or contrived physical address.

This means that a full address in a single column would mask to the same output as the same address spread across several columns of address components like street number and city.

The data in these two rows can mask to the same thing using this framework, because the location they refer to is the same even though the information is stored differently:

CITY	STREET	ZIP	HOUSE_NUMBER	COUNTRY	REGION	ADDRESS
Lorem City	Ipsum Ave	99999	1111	Dolor	Sit Amet	N/A
N/A	N/A	N/A	N/A	N/A	N/A	1111 Ipsum Ave, Lorem City, Sit Amet, Dolor, 99999

The framework offers 17 logical columns that can be assigned to columns in a database. All columns are optional. The algorithm concatenates the input column values to a hash and this hash is used to look up a row from user provided CSV data using a [SecureLookup \(see page 793\)](#) approach.

There are several advanced features:

- **Grouping:** The user can assign a single column that determines groups or “buckets” to be masked to within the lookup file. If assigned to the zip code column, this could ensure that data with a certain zip code was only masked to values in the lookup file with that same zip code.
- **US Zip Matching:** The United States can use either 5 or 9 digit zip codes to refer to the same place. This feature will trim the 9 digit codes to the 5 digit format and use that value to calculate the hash used for the lookup file, and to determine the masked output zip code.
- **Filter City:** If a city value is followed by the state code (ex: "San Diego CA") and the state code ("CA") is one of the values in the **filterCityValueList** the value is removed for hashing and after masking the state code can be re-applied. This feature can mask to the same address result for city values with and without a state code.
- **Option Columns:** These 3 logical columns can be used to apply an action to data columns in the source that fall outside of the standard columns provided. The behaviors available are:
 - **LOOKUP:** Use the column value from the CSV file.
 - **NULL:** Always return null.
 - **BLANK:** Always return a blank space/empty string.

- **ALGO_IN:** Chain the input value to a configured algorithm.
- **ALGO_OUT:** Chain the output from the CSV lookup value to a configured algorithm.



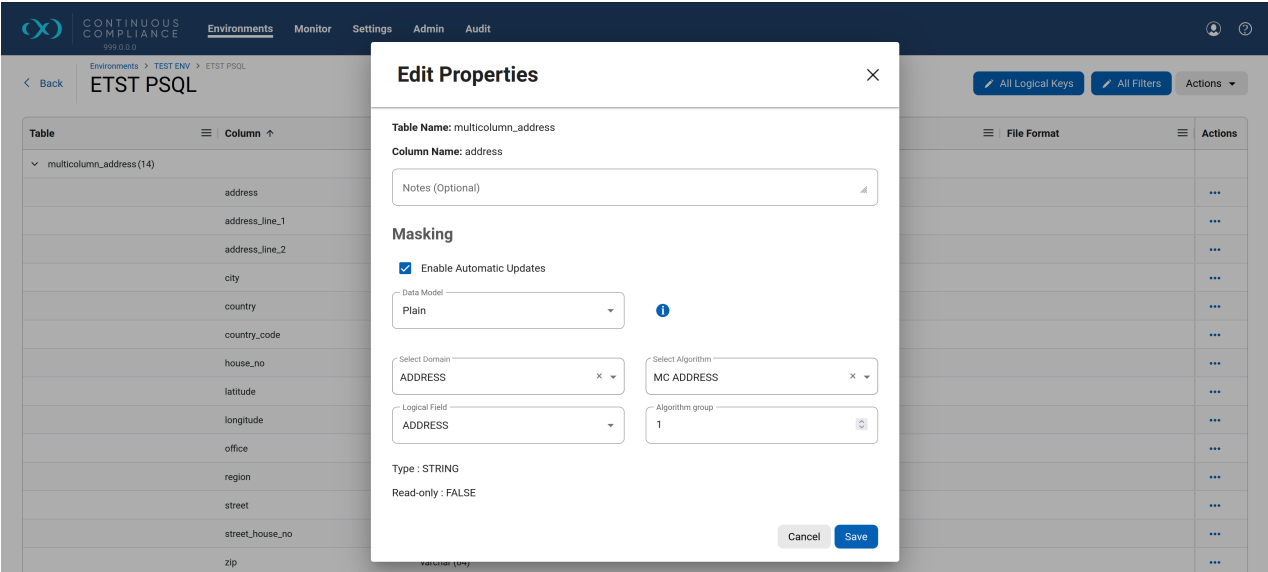
The descriptions for all individual configuration keys are found at the end of this documentation.

8.1.9.14.2 Column assignment

The following 17 logical columns are available in the algorithm for assignment:

- CITY
- STREET
- ZIP
- HOUSE_NO
- STREET_NO
- REGION
- COUNTRY
- COUNTRY_CODE
- LONGITUDE
- LATITUDE
- OFFICE
- ADDRESS
- ADDRESS_LINE1
- ADDRESS_LINE2
- OPTION1
- OPTION2
- OPTION3

Logical columns are assigned to columns in a masking ruleset. Multi-column algorithms are assigned to a ruleset column, and then a logical column from the algorithm is assigned:



8.1.9.14.2.1 Creating a Lookup File

The lookup file used for masking must be created by the user. Each row **must** contain 17 columns for the 17 different logical columns in the algorithm in the **same order**. These do not all need to contain data, but they should exist at minimum as a column defined by the configured delimiter, defaulting to "|". There should not be a row defining column names.

Row format and required sequence:

```
CITY|STREET|ZIP|HOUSE_NO|STREET_NO|REGION|COUNTRY|COUNTRY_CODE|LONGITUDE|LATITUDE|
OFFICE|ADDRESS|ADDRESS_LINE1|ADDRESS_LINE2|OPTION1|OPTION2|OPTION3
```

The following example:

```
Lorem City|Ipsum Ave|99999|99|1111|Sit Amet|Dolor|DOL|99.0|99.0|99|1111 Ipsum Ave,
Lorem City, Sit Amet, Dolor, 99999|1111 Ipsum Ave|Lorem City, Set Amet, Dolor, 99999|
NULL|NULL|NULL
```

And this example:

```
Lorem City|Ipsum Ave|99999|99|1111|Sit Amet|Dolor| | | | | | | | | | | | | | | | |
```

Are both valid rows for the lookup file.

Remember that all columns do not need to be filled out. Just the ones that create the masked output format desired.

8.1.9.14.3 Basic algorithm configuration

The most basic configuration of the Multi Column Address algorithm takes values from the data source using the assigned logical columns, concatenates the values for each row, hashes them and uses the hash to retrieve a row from the user provided lookup file. The column values for this row are inserted in the output row columns as the masked values.

An example config could look like this:

```
{
  "option1": "LOOKUP",
  "option2": "LOOKUP",
  "option3": "LOOKUP",
  "filterCity": false,
  "lookupFile": {
    "uri": "delphix-file://upload/f_9999988bde64469eb08f514c74a3244c/
YOUR_LOOKUP.csv"
  },
  "regexFields": null,
  "csvDelimiter": "\\|",
  "addressGroups": [],
  "fallbackGroup": null,
  "usZipMatching": false,
  "enforceNotNull": true,
  "matchingColumn": "",
  "option1Algorithm": null,
  "option2Algorithm": null,
  "option3Algorithm": null,
  "filterCityReapply": false,
  "inputRemoveSpaces": true,
  "inputCaseSensitive": false,
  "externalSecurityKey": 0,
  "filterCityValueList": "",
  "preserveEmptyFields": true,
  "columnsUsedToBuildHash": "",
  "enforceNotNullReplacement": "_"
}
```

Notice that **regexFields** and **addressGroups**, two of the more advanced features, are not used for simple configurations.

8.1.9.14.4 Advanced algorithm configuration

Because of the large number of options in this algorithm it is impossible to provide an example of all possible permutations. The following takes advantage of the advanced features not used in the basic configuration.

```
{
  "option1": "LOOKUP",
```



```

"option2": "LOOKUP",
"option3": "LOOKUP",
"filterCity": false,
"lookupFile": {
  "uri": "delphix-file://upload/f_9999988bde64469eb08f514c74a3244c/
YOUR_LOOKUP.csv"
},
"regexFields": {
  "regexZIP": [],
  "regexCITY": [],
  "regexHASH": [],
  "regexOFFICE": [],
  "regexREGION": [],
  "regexSTREET": [],
  "regexADDRESS": [],
  "regexCOUNTRY": [],
  "regexOPTION1": [],
  "regexOPTION2": [],
  "regexOPTION3": [],
  "regexHOUSE_NO": [],
  "regexLATITUDE": [],
  "regexLONGITUDE": [],
  "regexCOUNTRY_CODE": [],
  "regexADDRESS_LINE1": [],
  "regexADDRESS_LINE2": [],
  "regexMatchingColumn": [
    "G[0-9]",
    "G[0-9]([0-9])"
  ],
  "regexSTREET_HOUSE_NO": []
},
"csvDelimiter": "\\|",
"addressGroups": [
  {
    "group": "A",
    "matchList": [
      "G00",
      "G01"
    ]
  },
  {
    "group": "B",
    "matchList": [
      "G02",
      "G03"
    ]
  },
  {
    "group": "X",
    "matchList": [
      "G04"
    ]
  }
]

```

```

],
"fallbackGroup": "X",
"usZipMatching": false,
"enforceNotNull": true,
"matchingColumn": "ZIP",
"option1algorithm": null,
"option2algorithm": null,
"option3algorithm": null,
"filterCityReapply": false,
"inputRemoveSpaces": true,
"inputCaseSensitive": false,
"externalSecurityKey": 0,
"filterCityValueList": "",
"preserveEmptyFields": true,
"columnsUsedToBuildHash": "ZIP,CITY,STREET",
"enforceNotNullReplacement": "_"
}

```

Note that:

- The **matchingColumn** is set to "ZIP". To determine a group for an input row the ZIP column is used.
- The **regexMatchingColumn** has two regex filters. The input from the ZIP column will pass through these filters and the matching values will be used to determine group.
- There are 3 groups defined in **addressGroups**. The regex filtered matching column value from ZIP will be assigned to the group where it is contained in the **matchList**. This means that value G001-AAA would be regex filtered to G00 and assigned to group A. The input row for that value could only be masked to a row in the lookup file matching group A.
- The **columnsUsedToBuildHash** mentions a subset of the available logical columns. Only these columns will be considered when building a hash to retrieve a masking row from the lookup file.
- There is a regex list for each logical column, seen like this: **regexStreet**. Any value from the logical column with this name will be filtered through these regex before being used to build the lookup hash.

8.1.9.14.4.1 Algorithm configuration options

- **matchingColumn**: [String]

This mandatory setting contains the name of the column to be used as a matching value for grouping. When loading the CSV each data row is mapped into a specific group based on the **matchingColumn** value.

Example: The **matchingColumn** ZIP has input value 34564 but because the **regexMatchingColumn** is enabled with **[0-9]{1}** the ZIP column input value is filtered to the first numeric character '3'. The value 3 belongs to group A because 3 is an entry in the **matchList** of group A.

```
"matchingColumn": "ZIP"
```



This key is only used when groups are defined. It is ignored in simple configurations.

- **addressGroups: [List of JSON Arrays]**

If using a **matchingColumn** this setting must contain at least one group. Each group has a **matchList** containing possible input from the **matchingColumn**. Each **matchList** value should exist only once in the entire definition (otherwise first found is applied). An input row matched into a group can be assigned a deterministic value from and row in the lookup matching values in its **matchList**.

```
"addressGroups" : [
  {"group": "A","matchList": ["1","2","3"]},
  {"group": "B","matchList": ["37","38","39"]}
]
```

- **columnsUsedToBuildHash: [String]**


Optional property containing a comma separated list of columns used for matching. The column values are concatenated for hashing. If not defined all columns are used to compute the hash.

```
"columnsUsedToBuildHash": "ZIP,CITY,STREET_HOUSE_NO"
```

- **fallbackGroup: [String]**

This setting is mandatory for configurations using **addressGroups**. If a record from a **matchingColumn** does not match a value in the match list for any group, this group is used for masking the row's values.

```
"fallbackGroup": "X"
```

 This key is only used when groups are defined. It is ignored in simple configurations.

- **lookupFile: [FileReference]**

Mandatory URI to a file reference for the CSV data file containing masking output address material. The URI can be obtained through the masking engine API for any file uploaded.

```
"lookupFile": {
  "uri": "delphix-file://upload/f_9999988bde64469eb08f514c74a3244c/
YOUR_LOOKUP.csv"
}
```

- **inputCaseSensitive: [boolean]**

Determines if the input ignores case. Important for determining lookup hashes. Defaults to **false** and this will satisfy most use cases.

- **inputRemoveSpaces: [boolean]**

When true any space in the input string is filtered out. Important for determining lookup hashes. This is helpful to achieve the same result for differently structured inputs like column 'STREET' and column

'HOUSE_NO' versus a single column containing street and house number. The single column would contain a space between the values where two concatenated columns may not. Defaults to **true** and this will satisfy most use cases.

- **filterCity: [boolean]**

If **filterCity** is enabled the city value will be checked to see if it's end contains a value from the **filterCityValueList**. If it does, the matching ending is removed for hashing. Uncommonly used. Defaults to **false** and this will satisfy most use cases.

- **filterCityValueList: [String]**

Comma separated list of String values that are to be removed from the end of the city column input value if **filterCity** is true. Uncommonly used.

- **filterCityReapply: [boolean]**

If enabled and the **filterCity** feature has detected a match and **filterCityReapply** is enabled the masked output **COUNTRY_CODE** value will be added to the city column. Example:

San Diego CA > CA is filtered > masked value Boston > we detect that **filterCity** occurred and MA is appended > Final result: Boston MA.

Uncommonly used.

- **preserveEmptyFields: [boolean]**

If set to **true** any null or empty string from input will be preserved as an empty string ("").

- **csvDelimiter: [String - Single Character]**

Required. Character to be used as the delimiter in lookup CSV file. Defaults to "|".

- **enforceNotNull: [boolean]**

Required. This setting controls whether a masked column output can be null or an empty string. The value is configured under property **enforceNotNullValue**.

- **enforceNotNullReplacement: [String]**

Required. This setting controls what value should be used as replacement for null or empty strings in masked output. Useful to avoid breaking database table not null constraints. Defaults to an underscore "_".



This value will be seen in masked output for configurations not using the group feature. If groups are utilized, the replacement value will be masked as a member of the **fallbackGroup**. The replacement can also be set as a value that matches a different group- effectively creating a different fallback for null values.

- **option1/option2/option3: [String]**

Used to assign behavior to the wildcard option columns. Options are:

- **LOOKUP**: This option will use a value from CSV lookup.
- **NULL**: This option will always return NULL as the masked value.
- **EMPTY**: This option will always return an empty string ("") as the masked value.
- **BLANK**: This option will always return a blank space (" ") as the masked value.


- **ALGO_IN**: This option will send the input value to a configured algorithm and return its result.
- **ALGO_OUT**: This option will send the value returned from the CSV lookup to a configured algorithm and return its result.
- **option1Algorithm/option2Algorithm/option3Algorithm**: [String]

Assigns a chained algorithm for **option** fields configured with **ALGO_IN** or **ALGO_OUT**.

```
"option1Algorithm": {"name": "dlpx-core:CM Alpha-Numeric"}
```

- **regexFields** [JSON]

The input value from any logical field can be filtered through regex before being concatenated into the value to be hashed for lookup. The keys available for this object correspond to each logical field. Each logical field can have multiple regex filters. The output of each filter match is concatenated into a end result. This can allow multiple filters to match various parts of a large input and achieve a normalized value to use in hashing.

 When multiple regex filters are used on a single field, the matching values for each are concatenated in order. The filters list is an **AND** implementation, not an **OR**.

```
"regexADDRESS": [
  "[a-zA-Z]{1,3}",
  "[0-9]{1,2}"
]
```

Example input:

```
Samplestreet 123
```

Filtered and concatenated output:

```
1Sam
```

8.1.9.15 Multi Column Condition (Algorithm frameworks)

The Multi Column Condition framework will mask columns with different algorithms based on the value in a **key** column. The JSON used to configure the algorithm supports a list of various possible values found in the key column, or **conditions**. Each condition determines how the other columns referenced by the algorithm are masked.

An instance of the algorithm can reference:

- One **key** column to determine the condition used.

- Ten string columns.
- Three date columns.
- Three numeric columns.
- Three binary columns.
- One `concat` column that receives the concatenated value of other columns.

Each of the typed columns declared must have a value that is a valid algorithm. This can be user-created algorithm instances or built-in algorithms. If an invalid algorithm name is given, an error will be shown.

This is an example configuration that contains all possible example values and is not a valid algorithm configuration. Descriptions of all keys are below. In the Addition examples section below, you can find examples of valid algorithm configurations.

```
{
  "conditions": [
    {
      "key": [
        "FN"
      ],
      "string1": {
        "name": "dlpx-core:CM Alpha-Numeric"
      }
      "date1": {
        "name": "DateShiftDiscrete"
      }
      "numeric1": {
        "name": "dlpx-core:CM Numeric"
      },
      "binary1": {
        "name": "NULL SL"
      },
      "ignoreColumns": [
        "string2"
      ]
    },
    {
      "key": [
        "LN"
      ],
      "string1": {
        "name": "Custom String 1 (Example)"
      },
      "string2": {
        "name": "dlpx-core:LastName"
      },
      "date1": {
        "name": "Custom Date 1 (Example)"
      }
      "numeric1": {
        "name": "Custom Numeric 1 (Example)"
      },
      "binary1": {
```

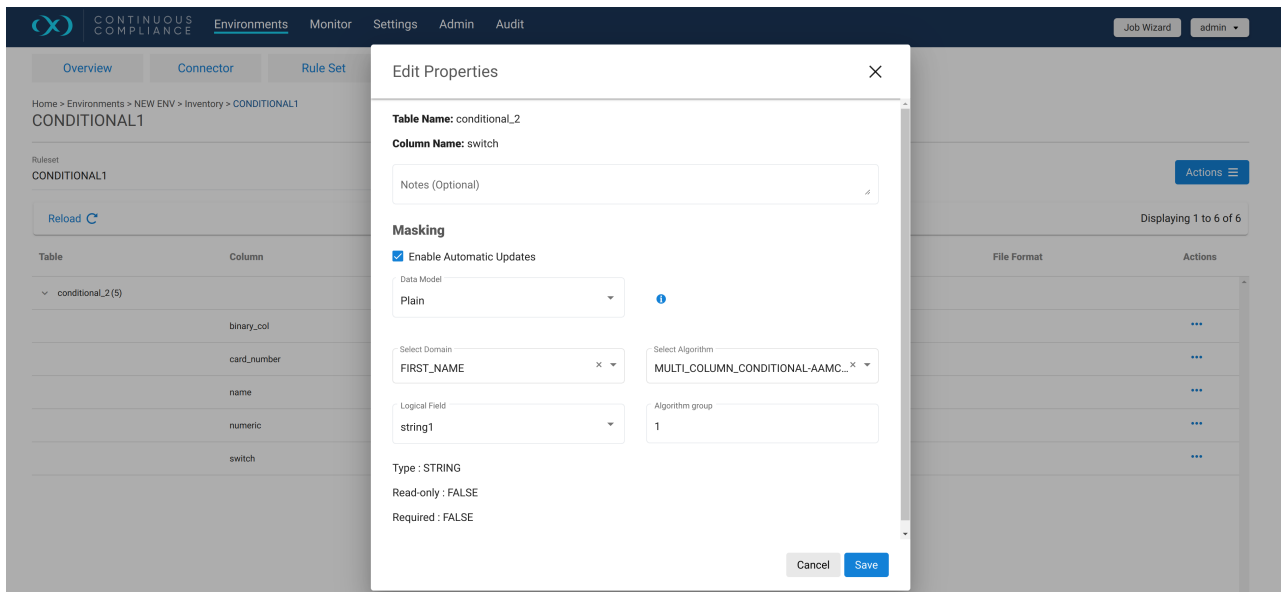
```

        "name": "Custom Binary 1 (Example)"
    },
    "concat1": {
        "dateFormat": "ddMMyyyy",
        "concatPattern": [
            "<string1>",
            " custom string ",
            "<string2>"
        ],
        "concatBeforeMasking": false,
        "preserveEmptyValues": true,
        "algoAfterConcatenating": {
            "name": "dlpx-core:CM Alpha-Numeric"
        }
    }
},
"fallbackKey": "FN",
"fallbackAlgo": {
    "dateAlgo": null,
    "binaryAlgo": null,
    "stringAlgo": {
        "name": "dlpx-core:CM Alpha-Numeric"
    },
    "numericAlgo": null
},
"filterLength": 0,
"filterDirection": "FIRST",
"keyCaseSensitive": false
}

```

8.1.9.15.1 Assigning column keys

The conditional key “key” and each of the typed keys are assigned to a database column in the UI inventory screen as shown below.



- **Conditions [list]**

The `conditions` key is a list that contains any number of JSON objects. Each object must reference at least one "key" and determine a custom set of algorithms and actions to take for masking columns.

- **Key [list]**

Required for each condition. The list contains any number of strings referencing a possible value in the database column assigned to `key` for the multi-column condition algorithm. If one of these values are found, the algorithms for this condition will be used.

- **String 1-10 [key/value]**

Each of these keys, expressed as `string1` through `string10`, specify the algorithm that will be applied to a string type column in the database.

- **Numeric 1-3 [key/value]**

Each of these keys, expressed as `numeric1` through `numeric3`, specify the algorithm that will be applied to a numeric type column in the database.

- **Date 1-3 [key/value]**

Each of these keys, expressed as `date1` through `date3`, specify the algorithm that will be applied to a numeric type column in the database.

- **Binary 1-3 [key/value]**

Each of these keys, expressed as `binary1` through `binary3`, specify the algorithm that will be applied to a binary type column in the database.

- **Concat 1 [key/value]**

There is a single optional concatenation column called `concat1` available. The row it is assigned to had data inserted formed from the concatenation of other columns in the inventory and string literals. The `concatPattern` key defines the structure of the output. In the example configuration above, the value of the assigned `string1` column will be concatenated with "custom string" and then the value of `string2`.

- Other concat parameters:

- `concatBeforeMasking` – Whether the source column data should be masked with its respective algorithms before insertion to the concat column.
- `preserveEmptyValues` – Whether an empty source column value should be preserved prior to concatenation.
- `algoAfterConcatenation` – The algorithm to be applied to the complete concatenated data
- `dateFormat` – If a source column is a date, this is the format to use for conversion to string.
- **IgnoreColumns [list]**
This list of columns will be **excluded from masking for the parent condition**. This takes precedence over all fallback algorithms. Output data will contain the unmasked values for the specified columns with these keys. This cannot be used for a fallback key. Keys that where it is needed implement an ignoreColumns list must be explicitly defined.

8.1.9.15.2 Fallbacks

There are three levels of possible fallback algorithms to ensure that data is still masked in unexpected situations. These fallback algorithms are defined outside of the conditions list. They are listed and described below.

- **Fallback Key**
If masking finds a value in the `key` column that cannot be found in the key list for any defined conditions, this key will be used. The fallback key defined must be found in the key list of a condition. All algorithms for that condition will be applied normally to this row.
- **User-defined Fallbacks**
Outside the list of conditions is the `fallbackAlgo` key. It is a JSON object with four possible key/value pairs. The keys `stringAlgo`, `numericAlgo`, `dateAlgo`, and `binaryAlgo` may correspond with an algorithm value of the specified data type. If no fallback key is defined, or the algorithms for the fallback key do not describe a given column, these algorithms will be used.
- **Global Fallback**
These are internal to the algorithm. If no fallback key applies and no user-defined fallback algorithm for the input type is defined, this algorithm will mask the data. This is the last tier of fallback. If everything else fails these algorithms will be used.
 - The global fallback algorithms by type are:
 - String – `d\px-core:CM Alpha-Numeric`
 - Numeric – `d\px-core:CM Numeric`
 - Date – `DateShiftDiscrete`
 - Binary – `NullValueLookup`

8.1.9.15.3 Key filter

Sometimes only a portion of the value in the `key` column is needed to determine a condition. The filter options allow a subsection of the value to be used.

8.1.9.15.3.1 Filter parameters:

- `filterLength` – The number of characters from the `key` column value to use. If set to zero, the entire value is used.
- `filterDirection` – Allowed values "FIRST" and "LAST". Whether the first or last characters of the `key` column string are preserved according to the `filterLength`.
- `keyCaseSensitive` – Whether character case is considered when matching keys for a condition.

8.1.9.15.4 Additional examples

8.1.9.15.4.1 Example 1

This algorithm masks as first name or last name based on the key.

```
{
  "conditions": [
    {
      "key": [ "FN" ],
      "string1": { "name": "dlpx-core:FirstName" }
    },
    {
      "key": [ "LN" ],
      "string1": { "name": "dlpx-core:LastName" }
    }
  ],
  "fallbackAlgo": { "stringAlgo": { "name": "dlpx-core:CM Alpha-Numeric" } },
  "filterLength": 0,
  "filterDirection": "FIRST",
  "keyCaseSensitive": false
}
```

8.1.9.15.4.2 Example 2

This algorithm masks multiple columns based on (key) being 'PRS' (Person) or 'ORG' (Organization). The example also uses Concatenation.

```
{
  "conditions": [
```

```

{
  "key": [ "PRS" ],
  "string1": { "name": "dlpx-core:FirstName" },
  "string2": { "name": "dlpx-core:LastName" },
  "string3": { "name": "dlpx-core:CM Digits" },
  "date1": { "name": "DateShiftFixed"},
  "concat1": { "concatPattern": [ "<string1>", " ", "<string2>" ],
    "concatBeforeMasking": false,
    "preserveEmptyValues": true
  }
},
{
  "key": [ "ORG" ],
  "string1": { "name": "BusinessLegalEntityLookup" },
  "date1": { "name": "DateShiftDiscrete" },
  "string3": { "name": "dlpx-core:CM Alpha-Numeric" }
}
],
"fallbackAlgo": { "stringAlgo": { "name": "dlpx-core:CM Alpha-Numeric" }},
"filterLength": 3,
"filterDirection": "FIRST",
"keyCaseSensitive": false
}

```

8.1.9.16 Name (Algorithm frameworks)

Starting in version 6.0.8.0, Delphix has introduced a builtin Extensible *Name* Algorithm Framework, co-existing with the legacy *FIRST NAME SL* and *LAST NAME SL* ones. Name Framework provides masking functionality for String type input. It's based on Secure Lookup mechanism, and includes additional configuration flags making it more flexible and robust.

Similar to Secure Lookup it creates masking results which are deterministic (i.e. the same algorithm with the same configuration and security key will provide the same result for the same input) and not unique. If you are looking for a framework whose algorithm(s) will provide unique masking results, you should consider using other frameworks (for example Character Mapping).

The new framework uses SHA256 hashing method and allows case configurations for input and output (i.e. masked) values. It also allows filtering accents and configuring the maximum length of the masked value and the maximum number of masked names returned. A multi-word input name may contain particles, such as prefixes, suffixes, titles, etc. The new framework allows configuring which particles are to be preserved and removed.

8.1.9.16.1 Creating a Name Algorithm via UI

1. At the top right of the **Algorithms** page, click **+ Algorithm**.

Add Algorithm



- Details
- Configuration
- Summary

Details

Specify the algorithm details.

Name

Description

Framework Name

Mask Type
STRING

Name Framework: This is the framework to cover the scenarios where it is required to mask string with deterministic and not unique masking results

Framework Options:

Output (Masked) Case:

- **Preserve Lookup File Case** - keep masked value as found in Lookup File
- **Preserve Input Case (Default)**- check the input case, which can be one of following three:
 - All uppercase - in that case force whole masked value to uppercase
 - All lowercase - in that case force whole masked value to lowercase
 - Mixed (if at least 1 character case is different from others) - in that case keep masked value as found in Lookup File
- **Force all Uppercase** - forces whole masked value to uppercase

Cancel Back **Next** Save

2. Enter an **Algorithm Name**.

Info: This MUST be unique.

3. Enter a **Description**.

4. Select **Name** as the Framework Name and click **Next**.

Add Algorithm



- Details
- Configuration
- Summary

Configuration

Configure the algorithm.

Case Sensitive Lookup

Filter Accent

Output (Masked) Case

Maximum Number of Names

Maximum Masked Name Length

Lookup File ⓘ

Select File

Successfully uploaded file.

Particles to Preserve ⓘ

Select File

Successfully uploaded file.

Particles to Remove ⓘ

Select File

Successfully uploaded file.

Name

Framework Description

Name Framework: This is the framework to cover the scenarios where it is required to mask string with deterministic and not unique masking results

Framework Options:

Output (Masked) Case:

- **Preserve Lookup File Case** - keep masked value as found in Lookup File
- **Preserve Input Case (Default)**- check the input case, which can be one of following three:
 - All uppercase - in that case force whole masked value to uppercase
 - All lowercase - in that case force whole masked value to lowercase
 - Mixed (if at least 1 character case is different from others) - in that case keep masked value as found in Lookup File
- **Force all Uppercase** - forces whole masked value to uppercase
- **Force all Lowercase** - forces whole masked value to lowercase

Maximum Masked Name Length: Max number of characters or length of masked output string. **default** is 0, which means unlimited.

Cancel Back **Next** Save

5. Choose the **Case Sensitive Lookup** configuration. The default is *false (unchecked)*, which means the lookup is not case-sensitive, so the same input of different cases will be masked to the same value. For example:

```
Peter -> John
peter -> john
```

If the **Case Sensitive Lookup** box is checked, then the same input of different cases will be masked to the different values, for example:

```
Peter -> John
peter -> andrew
```

6. Choose the **Filter Accent** configuration. The default is *true (checked)*. If the **Filter Accent** box is checked, then similar input with and without accented symbols will be masked to the same values. For example:

```
Adrián -> John
Adrian -> John
```

If the **Filter Accent** box is unchecked, it will be masked to the different values, for example:

```
Adrián -> John
Adrian -> Peter
```

7. Choose the **Output (Masked) Case** configuration. (Default is *Preserve Input Case*)

Output (Masked) case options:

- **Preserve Lookup File Case** - keep the masked value as found in the Lookup File.
 - **Preserve Input Case (Default)** - check the input case, which can be one of the following three:
 - All uppercase - in that case, force the whole masked value to uppercase
 - All lowercase - in that case, force the whole masked value to lowercase
 - Mixed (if at least 1 character case is different from others) - in that case keep the masked value as found in the Lookup File
 - **Force all Uppercase** - forces the whole masked value to uppercase
 - **Force all lowercase** - forces the whole masked value to lowercase
8. Choose the **Maximum Number of Names**. (Range 1-4, default 2). This is the maximum number of names to be masked and returned. The rest are dropped.
9. Choose the **Maximum Masked Name Length**. (A number greater than or equal to 0, default is 0). This is the maximum number of characters or length of the masked output string. The masked result is trimmed to fit that length. Value 0 means length is unlimited.

Info:

We also try to detect the length of the destination field. Some Data Sources provide that value, while others don't. For example: if Data Source provides value **10** for the destination column length and the current configuration field is set to **0** or any value longer than 10 - the shortest value wins, i.e. in this example masked result would be trimmed to 10 characters.

Warning:

Some UTF-8 characters might take 2 bytes. If the lookup file contains those characters - the trimmed result might be not as expected since we trim by the number of characters and not the number of bytes. There is a bug open for that mismatch.

10. Specify a **Lookup File**. (Required. A locally chosen file or a FileReference)

This file is a single list of values. It does not require a header. Every line of the Lookup File might be used as a masked value. The Lookup File must be ASCII or UTF-8 encoding compatible. The following is sample file content:

```
Ann
Marie
Tomas
Ann-Marie
Basil
Mark
```

11. Specify a **Particles to Preserve File**. (Optional. A locally chosen file or a FileReference) Contains a list of particles to be preserved. These particles are not masked. For example, if the file contains the particle "von":

```
von Froum -> von Smith
```

12. Specify a **Particles to Remove File**. (Optional. A locally chosen file or a FileReference) Contains a list particles to be removed. These particles are removed before masking and do not affect the masking result. For example, if the file contains the particle "von":

```
von Froum -> Smith
Froum -> Smith
```

Info:

If the particle is found in both the "Preserve" and "Remove" files - it will be removed.
If the input contains only particles, a particle will be masked as if it were a name.

13. Click **Next** to verify details on the Summary step.

Add Algorithm



- Details
- Configuration
- Summary

Summary

View the algorithm details below. Click the Back button to make changes or click the Save button to save the algorithm configuration.

Details

Name
test_name

Framework Name
Name

Mask Type
STRING

Configuration

Case Sensitive Lookup
false

Filter Accent
true

Output (Masked) Case
Preserve Input Case

Maximum Number of Names
2

Maximum Masked Name Length
0

Lookup File
temp.txt

Particles to Preserve
boolean_lookup.txt

Particles to Remove
boolean_look_true.txt

Cancel Back Next Save

14. Click **Save**.

For information on creating Name algorithms through the API, see [API Calls for Creating Algorithms - Name](#). (see page 921)

8.1.9.17 Numeric Expression (Algorithm frameworks)

Numeric Expression algorithms mask numeric input by evaluating it within a one-line, mathematical expression written by the user in the Java programming language. The expression can reference the current unmasked value via an implicit variable called `input`.

For example, to mask a numeric column by always multiplying the input by 50%, the following expression could be used:

```
input * 0.5
```

In addition to `input`, the expression can reference user-defined constant variables whose values are determined at the beginning of a masking job and remain fixed for the life of the masking job.

See below for examples of expressions and constants.

8.1.9.17.1 Creating a numeric expression algorithm via UI

1. At the top right of the **Algorithms** page, click **+ Algorithm**.

Add Algorithm ✕

- Details
- Configuration
- Summary

Details

Specify the algorithm details.

Name

Description

Framework Name

Mask Type
BIG_DECIMAL

Numeric Expression Framework: Numeric Expression algorithms mask numeric input by evaluating it within a one-line, mathematical expression written by the user in the Java programming language. The expression can reference the current unmasked value via an implicit variable called **input**. For example, to mask a numeric column by always multiplying the input by 50%, the following expression could be used: **input * 0.5**

In addition to **input**, the expression can reference user-defined constant variables whose values are determined at the beginning of a masking job and remain fixed for the life of the masking job.

Cancel Back Next Save

2. Enter an **Algorithm Name**.
Info: This MUST be unique.
3. Enter a **Description**.
4. Select **Numeric Expression** as the Framework Name and click **Next**.

Add Algorithm ✕

- Details
- Configuration
- Summary

Configuration

Configure the algorithm.

Expression

Input Type

Replacement Value for Nonconforming Data

Constants ✎

There are no constants.

Numeric Expression Framework Description

Numeric Expression masks input by evaluating it within a one-line, user-defined expression, which must be a valid Java expression that returns a java.math.BigDecimal object or another object that can be converted into a BigDecimal. The expression can reference user-defined constant variables that remain fixed for the life of the algorithm, as well as a special long integer "seed" constant whose value is based on the algorithm key.

Cancel Back Next Save

5. Enter an **Expression**. This must be a one-line, mathematical expression written in the Java programming language that references `input` (the current unmasked value), e.g. `input * 0.5` or `input + Math.random()`. See below for more examples of expressions.

6. Choose the **Input Type**. This is the data type that `input` conforms to within the expression. The default *double* option causes `input` to be treated as a double-precision floating-point variable in expressions such as:

```
input * 0.5
```

or


```
input + Math.random()
```

Input Type can also be set to *long*, which causes `input` to be treated as a long integer variable in expressions such as:

```
Long.sum(input, 50L)
```

The final **Input Type** option is *BigDecimal*, which causes `input` to be treated as a `java.math.BigDecimal` variable in expressions such as:

```
input.scaleByPowerOfTen(3)
```

7. Enter an optional **Replacement Value for Nonconforming Data** if necessary. This is the default masked value to be used if the unmasked input is not a numeric data type and can't automatically be converted to one.
8. Optional: define any constants used by the expression by clicking on  by using the List editor section. More information on the List Editor section can be found [here \(see page 232\)](#).

Add Algorithm ×

○ Details

● **Configuration**

○ Summary

Configuration

Configure the algorithm.

Constants

+

Learn about Constants ⓘ

Name:

Value: + 🗑️

Name:

Value: + 🗑️

✕ ✓

Numeric Expression

Framework Description

Numeric Expression masks input by evaluating it within a one-line, user-defined expression, which must be a valid Java expression that returns a `java.math.BigDecimal` object or another object that can be converted into a `BigDecimal`. The expression can reference user-defined constant variables that remain fixed for the life of the algorithm, as well as a special long integer "seed" constant whose value is based on the algorithm key.

Cancel Back Next Save

Constants are variables that the expression can reference by name and whose values remain fixed for the life of a masking job. For example, to mask every column value in a masking job by multiplying them all by the same random number, you could use an expression such as:

```
input * theSameRandomNumber
```

but `theSameRandomNumber` would need to be defined as a constant whose **Name** is `theSameRandomNumber` and whose **Value** is something like `new java.util.Random().nextDouble()`. Find more examples of constants [here \(see page 783\)](#).

Add Algorithm ✕

- Details
- Configuration
- Summary

Configuration

Configure the algorithm.

Expression

Input Type

Replacement Value for Nonconforming Data

Constants ✎

Name	var1
Value	2
Name	var2
Value	5

Numeric Expression

Framework Description

Numeric Expression masks input by evaluating it within a one-line, user-defined expression, which must be a valid Java expression that returns a java.math.BigDecimal object or another object that can be converted into a BigDecimal. The expression can reference user-defined constant variables that remain fixed for the life of the algorithm, as well as a special long integer "seed" constant whose value is based on the algorithm key.

Cancel Back Next Save

9. Click **Next** to verify details on the Summary step.

Add Algorithm ✕

- Details
- Configuration
- Summary

Summary

View the algorithm details below. Click the Back button to make changes or click the Save button to save the algorithm configuration.

Details

Name
test_numeric

Description
Numeric Expression

Framework Name
Numeric Expression

Mask Type
BIG_DECIMAL

Configuration

Expression
input * 0.5

Input Type
DOUBLE

Replacement Value for Nonconforming Data
XXX

Constants
Name: var1, Value: 2
Name: var2, Value: 5

Cancel Back Next Save

10. Click **Save**.

For information on creating Numeric Expression algorithms through the API, see [API Calls for Creating Algorithms - Numeric Expression](#). (see page 923)

8.1.9.17.2 Writing good expressions & constants

Expressions and the Java programming language are powerful. Care must be taken to avoid writing bad expressions, which will manifest in the form of failed masking jobs. It is highly recommended to stage

complex expressions with a Java IDE such as [Eclipse](#)³³³ or [IntelliJ IDEA](#)³³⁴ before using them in a masking job.

The requirement that expressions must be written in Java might be intimidating to non-programmers, but simple mathematical equations in Java look similar to simple mathematical equations in general. The four most common operators are supported: addition (+), subtraction (-), multiplication (*), and division (/). For operators not supported by Java, use methods from the [java.lang.Math](#)³³⁵ library. For example, one might expect `input ^ 5` to mean "take input to the fifth power," but `^` is not a power operator in Java. Instead, use `Math.pow(input, 5.0)`.

To isolate parts of the expression for clarity or to enforce order of operations, use open and closed parentheses () only. Do not use square braces [] or curly braces { } .

8.1.9.17.3 Expression do's and don'ts

Do use an **Input Type** (explained above) that corresponds to the data type of the column being masked. For columns whose values are floating-point numbers (i.e. numbers that have digits to the right of the decimal point) set **Input Type** to *double* (the default) or *BigDecimal* if the expression needs to treat the input as a [java.math.BigDecimal](#)³³⁶ object in order to perform more complex math. For columns whose values are integers (whole numbers), set **Input Type** to *long*.

Don't write expressions that do mathematically impossible things (e.g. divide by zero) or will result in numeric overflow or values that are too large or too small to fit in the database column being masked.

Don't use line breaks or other whitespace to force an expression to be longer than one line.

Don't attempt to assign an expression to a variable. For example, this won't work:

```
output = input * 0.5
```

but this will:

```
input * 0.5
```

The result of the expression will be automatically assigned as the masked value. It's not necessary or allowed to assign it to anything else.

Don't use the `return` keyword or end the expression with a semicolon.

Don't write expressions that return a non-numeric value, e.g.

```
java.util.Arrays.asList(input)
```

333 <https://www.eclipse.org/downloads/packages/>

334 <https://www.jetbrains.com/idea/>

335 <https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html>

336 <https://docs.oracle.com/javase/8/docs/api/java/math/BigDecimal.html>

The above expression would return a `List` object, which can't be converted into a numeric value.

Expressions must return a value whose type is numeric: an `int`, `short`, `long`, `float`, or `double` Java primitive type (or their object wrappers) as well as `java.math.BigDecimal` and `java.math.BigInteger`. Returning `String` and `char[]` (character array) values is also acceptable as long as they can be converted into a numeric value.

Do fully-qualify any Java class the expression references that isn't in the `java.lang` package, e.g.

```
input * new java.util.Random().nextDouble()
```

This won't work:

```
input * new Random().nextDouble()
```

because Java's `Random` class is in the `java.util` package rather than `java.lang`.

Don't use the `import` keyword in an attempt to import non-`java.lang` classes that are referenced frequently by the expression and/or constants. Fully-qualify such Java classes every time they're referenced.

8.1.9.17.4 Constants

Constants are variables that the expression can reference by name and whose values remain fixed for the life of a masking job. Constant names must be [valid Java variable names](#)³³⁷. No two constants can have the same name, nor can "input" or "seed" be used as a constant name.

Constant values are very much like the expression: one-line Java expressions that must return a value. However, unlike the algorithm's main expression, constant values aren't required to be numeric.

Constants can reference by name other constants defined before them.

8.1.9.17.4.1 seed

There is a built-in constant named `seed`. Its value is a long integer that's based on the algorithm key, so the value of `seed` is guaranteed to remain the same across multiple masking jobs as long as the algorithm key remains the same. A common use case for `seed` is to seed a random number generator to produce the same (i.e. predictable) "random" number(s) among different masking jobs.

³³⁷ <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/variables.html#naming>

8.1.9.17.5 Numeric Expression Examples

8.1.9.17.5.1 Example 1

A numeric column must be masked by multiplying all of its values by the same random percentage. The random percentage must remain the same across every masking job.

Solution:

A single constant is required for the random percentage:

Name	Value
randomPercentage	<code>new java.util.Random(seed).nextDouble()</code>

Note that the built-in `seed` constant is being used to seed the random number generator, an instance of `java.util.Random`, which is used to produce a single random number.

The expression can then reference `randomPercentage` like this:

```
input * randomPercentage
```

8.1.9.17.5.2 Example 2

A numeric column must be masked by taking the square root of each value, then rounding it to a certain number of decimal places. Initially, it will be rounded to two decimal places, but the number of decimal places will be changed frequently, so it should be easily adjustable by the user.

Solution:

We'll define two constants this time:

Name	Value
decimalPlaces	2
multiplier	<code>Math.pow(10.0, decimalPlaces)</code>

then use this expression:

```
Math.floor(Math.sqrt(input) * multiplier + 0.5) / multiplier
```

The heavy lifting is being done by the main expression, which uses the `multiplier` constant. Note that `multiplier` references `decimalPlaces`, whose value could be easily changed by someone who is not inclined mathematically and doesn't understand how the expression is rounding numbers.

8.1.9.17.5.3 Example 3

We must mask a numeric column that represents the day of the current month, e.g. 1-31 (or 1-28, 1-29, 1-30). This column will be masked by adding to it a random number of days, which can be between 1 and the highest day in the current month, inclusive. If the masked value exceeds the highest day in the current month, it will simply be set to the highest day in the current month.

Solution:

First, since the day of the current month is an integer (whole number), set the algorithm's **Input Type** to *long* (integer) instead of the default *double* (floating point).

Then define three constants:

Name	Value
<code>calendar</code>	<code>java.util.Calendar.getInstance()</code>
<code>lastDayOfMonth</code>	<code>calendar.getActualMaximum(java.util.Calendar.DAY_OF_MONTH)</code>
<code>randomDays</code>	<code>new java.util.Random().ints(1, lastDayOfMonth + 1).iterator().nextInt()</code>

`calendar` is a new instance of `java.util.Calendar` set to the current date and time.

`lastDayOfMonth` uses `calendar` to determine the last day of the current month.

`randomDays` uses `lastDayOfMonth` to generate a random number between 1 and `lastDayOfMonth` (inclusive).

The expression will then look like this:

```
(input + randomDays > lastDayOfMonth) ? lastDayOfMonth.longValue() : input + randomDays
```

This expression leverages Java's ternary operator to mask conditionally. If the unmasked input plus `randomDays` exceeds `lastDayOfMonth`, then the masked value will simply be `lastDayOfMonth`. Otherwise, the masked value will be the unmasked input plus `randomDays`.

8.1.9.18 Payment Card (Algorithm frameworks)

The Payment Card framework masks payment card numbers based on the starting digits to be preserved and the minimum number of positions to be masked. This framework is built on top of the [Character Mapping Algorithm Framework](#) (see page 707) with a character set of [0-9]. All characters outside of this character group remain unmasked. Masked values are calculated algorithmically using the algorithm's key, so rekeying the algorithm will cause different outputs to be generated for each input. The last digit may remain the same if the calculated check digit is equivalent to the last digit of the input. Any inputs with more than one digit will never mask to the original value.



Any inputs with a single digit will remain unmasked.

This framework preserves the validity of the payment card number using the Luhn check. All input values with valid Luhn checks will be masked to values with valid Luhn checks. All invalid values with invalid Luhn checks will be masked to values with invalid Luhn checks.

8.1.9.18.1 Creating a payment algorithm via UI

1. At the top right of the **Algorithms** page, click **+ Algorithm**.

Add Algorithm
×

- Details
- Configuration
- Summary

Details

Specify the algorithm details.

Name

Description

Framework Name

Mask Type
STRING

Payment Card Framework: The Payment Card framework masks payment card numbers based on the starting digits to be preserved and the minimum number of positions to be masked. This framework is built on top of the **Character Mapping Algorithm Framework** with a character set of [0-9]. All characters outside of this character group remain unmasked. Masked values are calculated algorithmically using the algorithm's key, so rekeying the algorithm will cause different outputs to be generated for each input. The last digit may remain the same if the calculated check digit is equivalent to the last digit of the

Cancel Back Next Save

2. Enter an **Algorithm Name**.
Info: This MUST be unique.
3. Enter a **Description**.

- Select **Payment Card** as the Framework Name and click **Next**.

Add Algorithm

✕

- Details
- Configuration**
- Summary

Configuration

Configure the algorithm.

Minimum Masked Positions

Preserve Starting Digits

Payment Card

Framework Description

Payment Card Framework: The Payment Card framework masks payment card numbers based on the starting digits to be preserved and the minimum number of positions to be masked. This framework is built on top of the **Character Mapping Algorithm Framework** with a character set of [0-9]. All characters outside of this character group remain unmasked. Masked values are calculated algorithmically using the algorithm's key, so rekeying the algorithm will cause different outputs to be generated for each input. The last digit may remain the same if the calculated check digit is equivalent to the last digit of the input. Any inputs with more than one digit will never mask to the original value.

Any inputs with a single digit will remain unmasked.

This framework preserves the validity of the payment card number using the Luhn check. All input values with valid Luhn checks will be masked to values with valid Luhn checks. All invalid values with invalid Luhn checks will be

Cancel
Back
Next
Save

- Set **Minimum Masked Positions**. This value is the minimum number of positions that must be replaced for masking to be considered successful. If fewer positions are masked, a non-conforming data handling error is triggered. Values for this field must be in the range [0-32].
- Set **Preserve Starting Digits**. This value specifies how many maskable characters should be preserved from the beginning of the input. Only maskable characters are included in this count. Values for this field must be in the range [0-32].
- Click **Next** to verify details on the Summary step.

Add Algorithm



- Details
- Configuration
- Summary

Summary

View the algorithm details below. Click the Back button to make changes or click the Save button to save the algorithm configuration.

Details

Name
test_pc

Framework Name
Payment Card

Mask Type
STRING

Configuration

Minimum Masked Positions
1

Preserve Starting Digits
1

Cancel Back Next Save

8. When you are finished, click **Save**.

For information on creating Payment Card algorithms through the API, see [API Calls for Creating Algorithms - Payment Card](#). (see page 925)

8.1.9.18.2 Examples

As an example, a Payment Card algorithm with a *minMaskedPositions* value of 6 and a *preserve* value of 6 may mask as follows:

- "5419033646326699" → "5419036803270758"
- "5419-0336-4632-6699" → "5419-0368-0327-0758"
- "5319abc0339def4632ghi6599!" → "5319abc0364def1507ghi4137!"

All inputs with the same sequence of digits masked with the same algorithm configuration will result in the same output values.

8.1.9.19 Regex Decompose (Algorithm frameworks)

The Regex Decompose framework masks values that match specified [Java 8 regular expressions](#)³³⁸. The algorithm attempts to match the algorithm input against each regular expression, and once a match is found, the associated action is applied to transform either the entire input, or each capturing group (parts of the input) defined by the expression. A fallback action may be provided for use when none of the defined regular expressions match the input. If no fallback action is defined and an input fails to match any of the defined regular expressions, the algorithm may be configured to generate a non-conformant data exception.

Capturing groups are used in regular expressions to create subgroups. These can be expressed in regular expressions using parentheses to group characters together. This algorithm allows for different capturing

³³⁸ <https://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html>

groups to be assigned different mask actions. Nested capturing groups are unsupported and may lead to unpredictable behavior. If no capturing groups are defined, the first action is applied to the entire match. In this case, the action list should contain only one action.

8.1.9.19.1 Creating a regex decompose algorithm via UI

1. At the top right of the **Algorithms** page, click **+ Algorithm**.

Add Algorithm

×

- Details
- Configuration
- Summary

Details

Specify the algorithm details.

Name

Description

Framework Name

Regex Decompose
▼

Mask Type
STRING

Regex Decompose framework: The Regex Decompose framework masks values that match specified Java 8 regular expressions. The algorithm attempts to match the algorithm input against each regular expression, and once a match is found, the associated action is applied to transform either the entire input, or each capturing group (parts of the input) defined by the expression. A fallback action may be provided for use when none of the defined regular expressions match the input. If no fallback action is defined and an input fails to match any of the defined regular expressions, the algorithm may be

Cancel
Back
Next
Save

2. Enter an **Algorithm Name**.
Info: This MUST be unique.
3. Enter a **Description**.
4. Select **Regex Decompose** as the Framework Name and click **Next**.

Add Algorithm



- Details
- Configuration
- Summary

Configuration

Configure the algorithm.

Regex Statements
No regex statements. Add Regex Statement

Fallback Action

Trim Input

Required Mask

Max Input Length

Regex Decompose

Framework Description

Regex Decompose framework: The Regex Decompose framework masks values that match specified Java 8 regular expressions. The algorithm attempts to match the algorithm input against each regular expression, and once a match is found, the associated action is applied to transform either the entire input, or each capturing group (parts of the input) defined by the expression. A fallback action may be provided for use when none of the defined regular expressions match the input. If no fallback action is defined and an input fails to match any of the defined regular expressions, the algorithm may be configured to generate a non-conformant data exception.

Capturing groups are used in regular expressions to create subgroups. These can be expressed in regular expressions using parentheses to group characters together. This algorithm allows for different capturing groups to be assigned different mask actions. Nested capturing groups are unsupported and

Cancel Back Next Save

5. Enter Regex statements by clicking on the **Add Regex Statement**.

Add Regex Statement

Regex

```
^(?:[+-]?([1-8]?[0-9])[NEWS]?$
```

Actions
A list of actions to apply to the input. If the regex defines capturing groups, actions are applied in the listed order to each group. If the regex doesn't define capturing groups, the first action is applied to the entire match. The algorithm will fail with an error if the number of actions is insufficient to mask all matched capturing groups. Nested capturing groups are unsupported and may lead to unpredictable behavior.

Add Action

Action 1

Type

Preserve ▼

+ 🗑️

Cancel Add

Regex statements can be modified by using the corresponding edit and delete icon of the regex.

Add Algorithm



- Details
- Configuration
- Summary

Configuration

Configure the algorithm.

Regex Statements

Add Regex Statement

```
^(?-[+]?([1-8]?[0-9])NEWS)?$
```

Fallback Action

Type: Apply Algorithm

Algorithm: ALG_AL3M3KGP

Trim Input

Regex Decompose

Framework Description

Regex Decompose framework: The Regex Decompose framework masks values that match specified Java 8 regular expressions. The algorithm attempts to match the algorithm input against each regular expression, and once a match is found, the associated action is applied to transform either the entire input, or each capturing group (parts of the input) defined by the expression. A fallback action may be provided for use when none of the defined regular expressions match the input. If no fallback action is defined and an input fails to match any of the defined regular expressions, the algorithm may be configured to generate a non-conformant data exception.

Capturing groups are used in regular expressions to create subgroups. These can be expressed in regular expressions using parentheses to group characters together. This algorithm allows for different capturing groups to be assigned different mask actions. Nested capturing groups are unsupported and may lead to unpredictable behavior. If no capturing groups are defined, the first action is applied to the entire match. In this case, the action list should contain only one action.

Cancel Back **Next** Save

6. Select **Fallback Action**.
7. Choose **Trim Input** and **Required Mask** options.
8. Enter **Max Input Length**.
9. Click **Next** to verify details on the Summary step.

Add Algorithm



- Details
- Configuration
- Summary

Summary

View the algorithm details below. Click the Back button to make changes or click the Save button to save the algorithm configuration.

Details

Name
test_rd

Framework Name
Regex Decompose

Mask Type
STRING

Configuration

Regex Statement
^(?-[+]?([1-8]?[0-9])NEWS)?\$

Capturing Group Actions
1

Capturing Group 1 Action
Preserve

Fallback Action
Apply Algorithm > ALG_AL3M3KGP

Trim Input
True

Required Mask
False

Max Input Length
50

Cancel Back Next **Save**

10. Click **Save**.

For information on creating Regex Decompose algorithms through the API, see [API Calls for Creating Algorithms - Regex Decompose](#). (see page 926)

8.1.9.19.2 Examples

As an example, a Regex Decompose algorithm with the following configuration:

```
Mask Pattern:
  Regular Expression: "[0-9]*"
  Action: Redact
  Redact String: "redacted"
  Require Mask: false
  Trim Input: true
  Maximum Input Length: 10
```

Will produce masked results as follows:

- "12345" → "redacted"
- " 6789 " → " redacted "
- "12345678901" → non-conformant data
 - exceeds maximum input length
- "abc123" → "abc123"
 - remains unmasked since it does not match the regex pattern

The provided regular expression matches any inputs with 0 or more digits in the range [0-9] and any inputs that match will be replaced with the string "redacted". Any inputs that contain characters outside of the range [0-9] will not be masked. If require mask was set to true, the last example "abc123" would trigger a non-conformant data event as the value would not be masked by the algorithm.

Another example that includes capturing groups with the following configuration:

```
Mask Pattern:
  Regular Expression: "([1-9]*)-([a-z]*)"
  Action 1: Redact
  Redact Character: 'X'
  Action 2: Preserve
  Require Mask: true
  Trim Input: true
  Maximum Input Length: 10
  Fallback Action: Redact
  Redact String: "redacted"
```

Will produce masked results as follows:

- "12345-abc" → "XXXXX-abc"
- "abc-123" → "redacted"
 - does not match the pattern so the fallback action is applied
- "1-a" → "X-a"
- "-" → "redacted"

- does match the pattern but the masked output would be "-" which breaks the requirement that the output must be different from the input so the fallback action is applied
- "redacted" → non-conformant data
 - does not match the pattern so the fallback action is applied but the fallback action does not change the value so it fails the requirement that the input must be masked

The provided regular expression matches any inputs with 0 or more digits in the range [1-9], a dash, and 0 or more characters in the range [a-z]. Any inputs that do not match that pattern will be masked by the fallback action. If the fallback action fails to change the input, a non-conformant data event will occur.

All inputs with the same input value masked with the same algorithm configuration will result in the same output values.

8.1.9.20 Secure Lookup (Algorithm frameworks)

Secure Lookup is the most commonly used type of algorithm. It is easy to generate and works with different languages. When this algorithm replaces real, sensitive data with fictional data, it is possible that it will create repeating data patterns, known as "collisions." For example, the names "Tom" and "Peter" could both be masked as "Matt". Because names and addresses naturally recur in real data, this mimics an actual data set. However, if you want the Masking Engine to mask all data into unique outputs, you should use Character Mapping.

Starting in version 6.0.4.0, we introduced a built in Extensible Secure Lookup Algorithm Framework. The new framework uses SHA256 hashing method and allows case configurations for input and output (i.e. masked) values.

8.1.9.20.1 Creating a secure lookup algorithm via UI

1. At the top right of the **Algorithms** page, click **+ Algorithm**.

Add Algorithm ✕

- Details
- Configuration
- Summary

Details

Specify the algorithm details.

Name

Description

Framework Name

Mask Type
STRING

Secure Lookup Framework: Secure Lookup is the most commonly used type of algorithm. It is easy to generate and works with different languages. When this algorithm replaces real, sensitive data with fictional data, it is possible that it will create repeating data patterns, known as "collisions." For example, the names "Tom" and "Peter" could both be masked as "Matt". Because names and addresses naturally recur in real data, this mimics an actual data set. However, if you want the Masking Engine to mask all data into unique outputs, you should use Character Mapping.

Starting in version 6.0.4.0, we introduced a built in Extensible Secure Lookup Algorithm Framework. The new framework uses SHA256 hashing method and allows case configurations for input and output (i.e. masked) values.

Cancel Back Next Save

2. Enter an **Algorithm Name**.
Info: This MUST be unique.
3. Enter a **Description**.
4. Select **Secure Lookup** as the Framework Name and click **Next**.

Add Algorithm ✕

- Details
- Configuration
- Summary

Configuration

Configure the algorithm.

Hash Method

Lookup File for Name ⓘ

Select File temp.txt ✕

Successfully uploaded file.

Case sensitive lookup

Trim whitespace in lookup file

Trim whitespace from inputs

Output (Masked) Case

Secure Lookup

Framework Description

Secure Lookup Framework: Secure Lookup is the most commonly used type of algorithm. It is easy to generate and works with different languages. When this algorithm replaces real, sensitive data with fictional data, it is possible that it will create repeating data patterns, known as "collisions." For example, the names "Tom" and "Peter" could both be masked as "Matt". Because names and addresses naturally recur in real data, this mimics an actual data set. However, if you want the Masking Engine to mask all data into unique outputs, you should use Character Mapping.

Starting in version 6.0.4.0, we introduced a built in Extensible Secure Lookup Algorithm Framework. The new framework uses SHA256 hashing method and allows case configurations for input and output (i.e. masked) values.

Cancel Back Next Save

5. Choose the **Hash Method** configuration for lookup determination.

- **SHA256** - This hash method is the default hash method for extensible secure lookup algorithms.
 - **LEGACY** - This hash method is used to mimic the legacy secure lookup behavior in the extensibility framework.
 - **RANDOMIZE** - This method replaces the input value with a random value from the lookup file. It internally uses a PRNG (pseudorandom number generator) to generate a random index with a uniform distribution (equal probability) which is subsequently used to fetch the value from the lookup file.
6. Specify a **Lookup File**. This file is a single list of values that does not require a header, every line of the Lookup File might be used as a masked value. The Lookup File must be ASCII or UTF-8 encoding compatible. The lookup file can be referenced locally or with a specified/uploaded URI. The following is sample file content:

```
Smallville  
Clarkville  
Farmville  
Townville  
Cityname  
Citytown  
Towneaster
```

7. Choose the **Case Sensitive Lookup** configuration. If the **Case Sensitive Lookup** box is marked then the same input of different cases will be masked to the different values. For example:

```
Peter -> John  
peter -> Andrew
```

If that setting is not marked (which is a default option), then the lookup would be case insensitive, for example:

```
Peter -> John  
peter -> John
```

8. Choose the **Trim Whitespace** selections. The handling of whitespace should be carefully considered. By default, leading and trailing whitespace is preserved and will be passed back into the data around the new masked string. The following options are available:
- **Trim Whitespace in Lookup File:** This assumes that any whitespace leading or trailing strings in your lookup file is erroneous and should be removed. If such whitespace is part of your valid data format, this option should not be selected. If not selected, whitespace will be output as part of the masked data.
 - **Trim Whitespace from Inputs:** This assumes that any whitespace in your unmasked source data is erroneous and should be removed. If such whitespace is part of your valid data format

or you want leave erroneous whitespace in place to simulate realistic dirty data, this option can be left unselected.



Please note that the options selected could cause the following problems if not carefully considered:

1. Job fails to mask table or file with the database or connector reporting a string length is longer than the length supported by the column or field.
 - a. If *Trim Whitespace from Input* is **not** selected and the field is completely space padded to the maximum length, and the masked string is longer than the source unmasked string, the result could be longer than the maximum length allowed. This is more likely to occur when dealing with non-ASCII data, as field length reporting in relation to length in bytes versus characters is inconsistent among all databases and connectors.
2. The job succeeds but referential integrity is broken at the application level. In rare cases, the definition of two fields that should have referential integrity, may have different maximum lengths. If whitespace is part of the valid data format, output whitespace could be trimmed to fit the shorter of the fields but fit untrimmed in another field.
 - a. If such a possibility exists, users should select *Trim Whitespace from Input*, deselect *Trim Whitespace in Lookup File*, and provide any required whitespace via the lookup file up to, but not exceeding, the maximum length of the shortest field masked by the same algorithm.

9. Choose the **Output (Masked) Case** configuration.
 - a. **Preseve Lookup File Case** - keep the masked value as found in the Lookup File.
 - b. **Preserve Input Case** - check the input case, which can be one of the following three:
 - i. All uppercase - in that case, force the whole masked value to uppercase.
 - ii. All lowercase - in that case, force the whole marked value to lowercase.
 - iii. Mixed (if at least 1 character case is different from others) - in that case keep the masked value as found in the Lookup File
 - c. **Force all Uppercase** - forces the whole masked value to uppercase
 - d. **Force all Lowercase** - forces the whole masked value to lowercase
10. Click **Next** to verify details on the Summary step.

Add Algorithm



- Details
- Configuration
- Summary

Summary

View the algorithm details below. Click the Back button to make changes or click the Save button to save the algorithm configuration.

Details

Name
test_sl

Framework Name
Secure Lookup

Mask Type
STRING

Configuration

Hash Method
SHA256

Lookup File for Name
temp.txt

Case sensitive lookup
false

Output (Masked) Case
Preserve Input Case

Trim whitespace from inputs
false

Trim whitespace in lookup file
true

Cancel Back Next Save

11. When you are finished, click **Save**.



- Since the **RANDOMIZE** hash method chooses a random replacement value every time, this **does not** maintain **referential integrity** and should never be used in cases where referential integrity is required. This is however useful in certain cases where masked values need to reflect a statistical distribution (ex: genders, model a population distribution, etc.)
- Before using the algorithm in a profiling job, you must add it to a domain.

For information on creating Secure Lookup algorithms through the API, see [API Calls for Creating Algorithms - Secure Lookup](#). (see page 929)

8.1.9.21 Segment Mapping (Algorithm frameworks)

Segment Mapping algorithms produce no overlaps or repetitions in the masked data. They let you create unique masked values by dividing a target value into separate segments and masking each segment individually.

You might use this method if you need columns with unique values, such as Social Security Numbers, primary key columns, or foreign key columns. When using segment mapping algorithms for primary and foreign keys, in order to make sure they match, you must use the same Segment Mapping algorithm for each. You can set the algorithm to produce alphanumeric results (letters and numbers) or only numbers.

With Segment Mapping, you can set the algorithm to ignore specific characters. For example, you can choose to ignore dashes [-] so that the same Social Security Number will be identified no matter how it is formatted. You can also preserve certain values. For example, to increase the randomness of masked values,

you can preserve a single number such as 5 wherever it occurs. Or if you want to leave some information unmasked, such as the last four digits of Social Security numbers, you can preserve that information.

This algorithm can be used for tokenization and re-identification jobs if the following conditions are met:

- All alpha-numeric and numeric segments have Value Ranges with "Mask values with: The same ranges"
- There are no segments with "Segment Treatment: Mask with a constant value"
- If a numeric segment is defined, "Short Numeric Segment Handling: Report nonconforming data" is selected

To decide whether Character Mapping or Segment Mapping is the correct option for your use case, see [Choosing Between Character and Segment Mapping Frameworks](#). (see page 656)

8.1.9.21.1 Creating a segment mapping algorithm via UI

1. At the top right of the **Algorithms** page, click **+ Algorithm**.

Add Algorithm
✕

- Details
- Configuration
- Summary

Details

Specify the algorithm details.

Name

Description

Framework Name

Mask Type
STRING

Segment Mapping Framework: The Segment Mapping Algorithm allows for masking of a string on a per-segment basis. A string can be subdivided into many segments, which can then define their masking behavior: mask alpha-numeric, mask numeric, mask with a constant value, and preserve (do not mask). The input and masked value ranges are customizable. All masking values for a specific instance of this algorithm are deterministic.

See [Segment Mapping documentation](#) for more information.

You can specify value ranges for each segment based on the **Segment Treatment**, provided as either individual values, ranges, or a combination thereof.

Mask alpha-numeric

You can specify an original value range and a mask value range. If either of these fields is left blank, it will use the default value range, which is 0-9,A-Z. Use the value range fields to specify individual values and ranges, for example 'A-F,P,R,1-5,7,9'. The masking will only look to mask these values and will preserve

Cancel Back Next Save

2. Enter an **Algorithm Name**.
Info: This MUST be unique.
3. Enter a **Description**.
4. Select **Segment Mapping** as the Framework Name and click **Next**.

Add Algorithm



○ Details

● **Configuration**

○ Summary

Configuration

Configure the algorithm.

Segments ✎

There are no segments.

Short Numeric Segment Handling

Report nonconforming data

Ignore Characters

Ignore specific characters

Specific Characters (separated by comma)

Ignore Commas

Add Control Characters by clicking on 'CTRL' button CTRL

Process Preserve Segments Before Ignore Characters

Segment Mapping

Framework Description

Segment Mapping Framework: The Segment Mapping Algorithm allows for masking of a string on a per-segment basis. A string can be subdivided into many segments, which can then define their masking behavior: mask alpha-numeric, mask numeric, mask with a constant value, and preserve (do not mask). The input and masked value ranges are customizable. All masking values for a specific instance of this algorithm are deterministic.

See [Segment Mapping documentation](#) for more information.

You can specify value ranges for each segment based on the **Segment Treatment**, provided as either individual values, ranges, or a combination thereof.

Mask alpha-numeric

You can specify an original value range and a mask value range. If either of these fields is left blank, it will use the default value range, which is 0-9,A-Z. Use the value range fields to specify individual values and ranges, for example 'A-F,PR,1-5,7,9'. The masking will only look to mask these values and will preserve any other values. Letters are masked to letters and digits to digits.

Cancel
Back
Next
Save

5. Add **Segments** by clicking on using List Editor section. A minimum of 1 and a Maximum of 10 segments are allowed. More information on the List Editor section can be found [here](#) (see page 232).

6. For each segment, select its:

- **Length** (number of characters). The maximum is 6.
- **Segment Treatment:** Mask alpha-numeric, Mask numeric, Preserve, or Mask with a constant value.
- **Value Ranges.** Optional for alpha-numeric and numeric, required for constant. See [Specifying Value Ranges](#). (see page 0)

Numeric segments are masked as whole segments. **Alpha-numeric** segments are masked by individual characters.

Value Ranges 

Length

Segment Treatment

If original values are

Mask values with

Replace values with



Note: If the original and replacement values and ranges are not the same, the algorithm is not reversible and cannot be used for tokenization/re-identification.

- If you would like to allow the masking of short numeric segments, change the **Short Numeric Segment Handling** drop-down to select **Mask partial segments**. This option allows masking to proceed if an input string is truncated midsegment. For example, you define a numeric segment of length 4, but the input string ends midsegment so you have a 2-digit number instead of 4. This only applies to **Mask numeric** segments. Other segment treatments always apply to partial segments.

If **Mask partial segments** are selected AND a **Mask numeric** segment is defined, the algorithm is not reversible and cannot be used for tokenization/re-identification.

By default, the segment mapping algorithm will **Report nonconforming data** for short numeric segments and the **Job Execution** page will display a warning that can be used to report the non-conformant data events. This will result in the non-conformant data not being masked.

Example:

Your content goes here

Segment 1: length 2, mask alpha-numeric. **Segment 2:** length 4, mask numeric.

Input	Output	Short Numeric Segment Handling
AB1234	DL9148	Either

Input	Output	Short Numeric Segment Handling
AB12	AB12	Report nonconforming data (reported)
AB0012	DL3619	Report nonconforming data (not reported)
AB12	DL3619	Mask partial segments

- Select the appropriate **Ignore Characters** handling. Ignored characters are removed from the input value before masking and restored to their original positions after masking. When **Automatically ignore special characters** is selected, all non-maskable characters are ignored. When **Ignore specific characters** option is selected, only specified characters are ignored.

Enter the characters you wish to ignore in the **Specific Characters** box, separated by a comma.

To ignore the comma character (,), check the **Ignore Commas** checkbox.

To ignore control characters, **Add Control Characters** by clicking on the 'CTRL' button and add the desired characters to ignore.

Ignore Characters

Ignore specific characters ▼

Specific Characters (separated by comma)

@,#

Ignore Commas

Add Control Characters by clicking on 'CTRL' button

^@ [NUL],^A [SOH]

Process Preserve Segments Before Ignore Characters

^@ [NUL]	^A [SOH]	^B [STX]	^C [ETX]
^D [EOT]	^E [ENQ]	^F [ACK]	^G [BEL]
^H [BS]	^I [TAB]	^J [LF]	^K [VT]
^L [FF]	^M [CR]	^N [SO]	^O [SI]
^P [DLE]	^Q [DC1]	^R [DC2]	^S [DC3]
^T [DC4]	^U [NAK]	^V [SYN]	^W [ETB]
^X [SUB]	^Y [ESC]	^Z [CAN]	^_ [GS]
^^ [RS]	^[[EM]	^/ [US]	^/ [FS]

- Lastly, the checkbox for **Process Preserve Segments Before Ignore Character** selects whether to process segments with "Segment Treatment: Preserve" first, before removing ignore characters, so ignore characters count as length when finding preserve segments in the input, and then process the remaining segments.

The default is for this to be unchecked, so ignore characters are removed first, and then the segments are processed in order.

Note: This option exists to support backward compatibility with the legacy Segment Mapping algorithm configuration and is **not recommended** for newly created algorithms, as it may cause some segments to be processed out of order.

- Click **Next** to verify details on the Summary step.

Add Algorithm
✕

- Details
- Configuration
- **Summary**

Summary

View the algorithm details below. Click the Back button to make changes or click the Save button to save the algorithm configuration.

Details	Configuration
<p>Name test_sm</p> <p>Framework Name Segment Mapping</p> <p>Mask Type STRING</p>	<p>Segments</p> <p>Segment 1 Segment Treatment : MASK_ALPHANUMERIC Length : 1 If original values are : A-E Mask values with : The same ranges</p> <p>Segment 2 Segment Treatment : MASK_ALPHANUMERIC Length : 1 If original values are : 0-Z Mask values with : Different ranges Replace values with : A-E</p> <p>Short Numeric Segment Handling Report nonconforming data</p> <p>Ignore Characters Ignore specific characters</p> <p>Specific Characters (separated by comma) @, #</p> <p>Ignore Commas false</p> <p>Control Characters ^@ [NUL]*A [SOH]</p> <p>Process Preserve Segments Before Ignore Characters false</p>

Cancel Back Next Save

- When you are finished, click **Save**.

Before you can use the algorithm in a profiling job, you must add it to a domain. If you are not using the Masking Engine Profiler to create your inventory, you do not need to associate the algorithm with a domain.

8.1.9.21.1.1 Specifying value ranges


You can specify values ranges for each segment based on the **Segment Treatment**.

For **Mask alpha-numeric**, you can specify an original value range and a mask value range. If either of these fields is left blank, it will use the default value range, which is 0-9,A-Z. Use the value range fields to specify individual values and ranges, for example 'A-F,P,R,1-5,7,9'.

i The masking will only look to mask these values and will preserve any other values. Letters are masked to letters and digits to digits.


i If the original and replacement values and ranges are not the same, the algorithm is not reversible and cannot be used for tokenization/re-identification.

For **Mask numeric**, you can specify an original value range and a mask value range. If either of these fields is left blank it will use the default value range, which is 0 to the max integer that can fit into the segment length (ex: 000-999 for a segment of length 3). Use the value range fields to specify integer values and ranges, for example '10,30,50-875'.

 The masking will only look to mask these values and will preserve any other values.

For **Preserve**, you cannot specify any value ranges as whatever is encountered in this segment will be preserved.

For **Mask with a constant value**, you cannot specify an original value range, and your replace value must be a single value the same length as the segment (ex: if the segment length is 3, 'ABC' would be a valid replacement).

 The Segment Mapping pattern and sub-patterns need to match the data in order for it to be masked. If the data is longer than the defined pattern it will be passed through unmasked. To avoid this unwanted behavior - patterns (segments) and Ignore Characters should be set to match the data.

For information on creating Segment Mapping algorithms through the API, see [API Calls for Creating Algorithms - Segment Mapping](#). (see page 932)

8.1.9.21.2 Examples

Perhaps you have an account number for which you need to create a segment-mapping algorithm. You can separate the account number into segments, preserving the first two-character segment, replacing a segment with a specific value, and preserving a hyphen. The following is a sample value for this account number:

NM831026-04

Where:

- **NM** is a plan code number that you want to preserve, always a two-character alphanumeric code.
- **831026** is the uniquely identifiable account number. To ensure that you do not inadvertently create actual account numbers, you can replace the first two digits with a sequence that never appears in your account numbers in that location. (For example, you can replace the first two digits with 98 because 98 is never used as the first two digits of an account number.) To do that, you want to split these six digits into two segments. The first of these segments would be a 2-character constant segment mapping to 98. The second of these 2 could be a 4-character numeric segment.
- **-04** is a location code. You want to preserve the hyphen and you can replace the two digits with a number within a range (in this case, a range of 1 to 77).

8.1.9.22 String Algorithm Chain (Algorithm Frameworks)

The **String Algorithm Chain** framework is a wrapper for multiple string algorithms to be executed consecutively as one. This can allow core algorithms or novel ones created with the **Delphix Algorithm SDK** to modify the input before or after a primary masking algorithm, or simply perform a few small modular masking transformations. For example, some inputs could have a very low occurrence of certain values (think a department in a company with very few members or an uncommon political affiliation), and with deterministic masking the occurrence frequency in the masked data could be used to infer the original value. An initial custom SDK algorithm or regex framework could remove or alter the occurrence frequency of these inputs and ensure that the data is masked adequately.

Error handling and fallback behaviors are left to the chained algorithms to ensure that the chain does not override configured options. Exceptions thrown by the chained algorithms will be preserved and handled by platform settings.

i The String Algorithm Chain framework will not generate non-conformant data events, but the chained algorithms may generate such events.

Up to eight algorithms can be chained, and a minimum of two. As with other chaining algorithms, an algorithm cannot be chained to itself and circular dependency is prevented at any depth.

8.1.9.22.1 Creating a String Algorithm Chain via UI

1. At the top right of the **Algorithms** page, click **+ Algorithm**.

Add Algorithm
×

- Details
- Configuration
- Summary

Details

Specify the algorithm details.

Name

Description

Framework Name

Mask Type
STRING

String Algorithm Chain Framework: This framework can chain multiple algorithms together to mask a string. The output of one algorithm is the input of the next. Algorithms may be built in instances or plugins.

Framework Options:
In the UI, select the number of algorithms and pick each from the dropdowns. Dropdowns will contain all string algorithms provided by the engine.

Cancel
Back
Next
Save

2. Enter an **Algorithm Name**.

Info: This MUST be unique.

3. Enter a **Description**.
4. Select **String Algorithm Chain** as the Framework Name and click **Next**.

×

Add Algorithm

- Details
- Configuration**
- Summary

Configuration

Configure the algorithm.

Input Type

2

Chained Algorithm 1

Chained Algorithm
AccNoLookup

Chained Algorithm 2

Chained Algorithm
ALG_AL3M3KGP

String Algorithm Chain

Framework Description

String Algorithm Chain Framework: This framework can chain multiple algorithms together to mask a string. The output of one algorithm is the input of the next. Algorithms may be built in instances or plugins.

Framework Options:
In the UI, select the number of algorithms and pick each from the dropdowns. Dropdowns will contain all string algorithms provided by the engine.

Cancel Back Next Save

5. Select the Number of Algorithms to Chain in the **Input Type** and select **Chained Algorithms** from the dropdowns. Dropdowns will contain all string algorithms provided by the engine.
6. Click **Next** to verify details on the Summary step.

×

Add Algorithm

- Details
- Configuration
- Summary**

Summary

View the algorithm details below. Click the Back button to make changes or click the Save button to save the algorithm configuration.

Details

Name
test_sac

Framework Name
String Algorithm Chain

Mask Type
STRING

Configuration

Chained Algorithm Name 1
AccNoLookup

Chained Algorithm Name 2
ALG_AL3M3KGP

Cancel Back Next Save

7. Click **Save**

An instance of String Algorithm Chain can be created via the API by providing a list of algorithms in the order they should be applied.

```
{
  "algorithms":[
    {
      "name":"algorithm name 1"
    },
    {
      "name":"algorithm name 2"
    }
  ]
}
```

Validation is done to ensure the existence of the string algorithms specified, and that there are between two and eight algorithms in the list. The algorithms will be executed on the input value in the order they occur in the list.

8.1.9.23 Tokenization (Algorithm frameworks)

The Tokenization framework allows you to mask data and reverse its masking. For example, you can use a Tokenization algorithm to mask data before you send it to an external vendor for analysis. The vendor can then identify accounts that need attention without having any access to the original, sensitive data. Once you have the vendor's feedback, you can reverse the masking and take action on the appropriate accounts.

The Tokenization algorithm is designed to be used in Tokenization/Re-Identification jobs, though it can also be used in Masking.

The algorithm tokenizes values using AES-128 encryption in CBC-CTS mode, with an optional initialization vector (IV), and Base64 encoding. The results are alpha-numeric strings that are longer than the original values. If the result is too long to fit in the field, the algorithm can be configured to either (a) fallback to a reversible masking algorithm, which produces a result that is the same length as the original value, or (b) fail the job.

The algorithm has the following properties:

- The masked value for each input is consistent when using the same algorithm **and** the initialization vector length is 0. Changing the key for the algorithm or using an initialization vector length greater than 0 will result in different masked values.
- As long as at least one maskable character is present in the input, the masked value will never match the input.
- The algorithm used to mask a value can change depending on the length of the input.
- The algorithm only works on string data types. Numbers can be masked if the column data type is a String type, such as VARCHAR or TEXT.

This new algorithm framework was introduced in version 6.0.13.0 to replace the existing Tokenization algorithm and adds the ability to select a fallback algorithm.

8.1.9.23.1 Creating a tokenization algorithm via UI

1. At the top right of the **Algorithms** page, click **+ Algorithm**.

Add Algorithm ×

- Details
- Configuration
- Summary

Details

Specify the algorithm details.

Name
test_tok

Description

Framework Name
Tokenization

Mask Type
STRING

Tokenization framework: The Tokenization framework allows you to mask data and reverse its masking. For example, you can use a Tokenization algorithm to mask data before you send it to an external vendor for analysis. The vendor can then identify accounts that need attention without having any access to the original, sensitive data. Once you have the vendor's feedback, you can reverse the masking and take action on the appropriate accounts.

The Tokenization algorithm is designed to be used in Tokenization/Re-identification jobs, though it can also be used in Masking.

The algorithm tokenizes values using AES-128 encryption in CBC-CTS mode, with an optional initialization vector (IV), and Base64 encoding. The results are alpha-numeric strings that are longer than the original values. If the result is too long to fit in the field, the algorithm can be configured to either (a) fallback to a reversible masking algorithm, which produces a result that is the same length as the original value, or (b) fail the job.

The algorithm has the following properties:

Cancel Back Next Save

2. Enter an **Algorithm Name**.
Info: This **MUST** be unique.
3. Enter a **Description**.
4. Select **Tokenization** as the Framework Name and click **Next**.

Add Algorithm



Details

Configuration

Summary

Configuration

Configure the algorithm.

Initialization Vector Length
16

Fallback
None

Tokenization

Framework Description

Tokenization framework: The Tokenization framework allows you to mask data and reverse its masking. For example, you can use a Tokenization algorithm to mask data before you send it to an external vendor for analysis. The vendor can then identify accounts that need attention without having any access to the original, sensitive data. Once you have the vendor's feedback, you can reverse the masking and take action on the appropriate accounts.

The Tokenization algorithm is designed to be used in Tokenization/Re-Identification jobs, though it can also be used in Masking.

The algorithm tokenizes values using AES-128 encryption in CBC-CTS mode, with an optional initialization vector (IV), and Base64 encoding. The results are alpha-numeric strings that are longer than the original values. If the result is too long to fit in the field, the algorithm can be configured to either (a) fallback to a reversible masking algorithm, which produces a result that is the same length as the original value, or

Cancel
Back
Next
Save

5. Select an **Initialization vector length**. The default length is 16, which offers the most security. The tradeoff is that this increases the length of the masked result. Selecting a lower IV length decreases the length of the masked result. It is recommended that you only select an IV length of 0 if you require the masked value for each input to be consistent between jobs and for the same input to only mask to one output.
6. Select a **Fallback** algorithm. An AES-encrypted result is always longer than the original value. If an AES encrypted result is too long to fit into the field, the job will fail if Fallback is "None". When Fallback is "Character Mapping", the Character Mapping algorithm is used to tokenize the value, which produces a result that is the same length as the input.
7. Click **Next** to verify details on the Summary step.

Add Algorithm ×

- Details
- Configuration
- **Summary**

Summary

View the algorithm details below. Click the Back button to make changes or click the Save button to save the algorithm configuration.

Details

Name
test_tok

Framework Name
Tokenization

Mask Type
STRING

Configuration

Initialization Vector Length
16

Fallback
None

Cancel Back Next Save

8. Click **Save**.

If **Character Mapping** is selected as the Fallback, and a Character Mapping algorithm is created, which will be used to tokenize values that cannot be tokenized with AES encryption because the encrypted result is too long for the field. When selected, two additional configuration options will appear: **Minimum Masked Positions** and **Character Groups**. Unlike standalone Character Mapping algorithms, the Character Mapping algorithm used for Tokenization fallback does not support **Preserve Ranges** and **Preserve Leading Zeroes**, and **Case Sensitive** is permanently set to **true**.

Add Algorithm



- Details
- Configuration
- Summary

Configuration

Configure the algorithm.

Initialization Vector Length

Fallback

Minimum Masked Positions

Character Groups


There are no associated character groups.

Tokenization**Framework Description**

Tokenization framework: The Tokenization framework allows you to mask data and reverse its masking. For example, you can use a Tokenization algorithm to mask data before you send it to an external vendor for analysis. The vendor can then identify accounts that need attention without having any access to the original, sensitive data. Once you have the vendor's feedback, you can reverse the masking and take action on the appropriate accounts.

The Tokenization algorithm is designed to be used in Tokenization/Re-identification jobs, though it can also be used in Masking.

The algorithm tokenizes values using AES-128 encryption in CBC-CTS mode, with an optional initialization vector (IV), and Base64 encoding. The results are alpha-numeric strings that are longer than the original values. If the result is too long to fit in the field, the algorithm can be configured to either (a) fallback to a reversible masking algorithm, which produces a result that is the same length as the original value, or

1. Enter a value for **Minimum Masked Positions**, which sets the minimum number of characters that the algorithm must mask; fewer positions trigger non-conformant data handling. Null, empty, and all-whitespace values never trigger non-conformant data handling.
2. Define **Character Groups** for each group of characters among which you would like to map by clicking on  using the List editor section. More information on the List Editor section can be found [here \(see page 232\)](#).

Each group may be defined either by specifying each literal character in the group, such as "0123456789", or using Java Regular Expression style character ranges, such as "[0-9]". The algorithm will freely map characters to other characters within the same group, so by defining groups "[0-9]" and "[A-Z]", numbers would be replaced by other numbers, and letters by other letters, but a number would never be replaced by a letter. Groups should not contain duplicate characters, and each character may belong to only one group. Any character that is not assigned to a group will be preserved (not masked) by the algorithm. It is recommended that all characters are in one group so there is more randomization and the values are more obfuscated. The default is the Base64 character set ["[A-Za-z0-9+/]"], which contains the same characters that appear in an AES encrypted result.

Once you have created an algorithm, you may associate it with a domain.

1. In the upper right-hand region of the **Domains** tab under **Settings**, click **+ Domain**.
2. Enter a **Domain Name**.
3. Select algorithms from both the **Algorithm Name** and **Tokenization Algorithm Name** drop-down menus.

Add Domain ✕

- Domain Name
- Algorithm
- Tokenized Algorithm
- Summary**

Summary

View the domain details below. Click the Back button to make changes or click the Save button to save the domain.

Details

Domain Name
TK

Tokenized Algorithm

Name
ACCOUNT_TK

Algorithm Framework
Tokenization

Masking Algorithm

Name
ACCOUNT_TK

Algorithm Framework
Tokenization

Cancel Back Next Save

Next, create a Tokenization Environment:

1. In **Environments**, click the **+ Environment** button.

Add Environment ✕

Select Application

App1
✕ ▼

Environment Name

Test_tok

Purpose

Tokenization/Re-Identification
▼

Enable Approval Workflow (Database RuleSets only)

Cancel Save Save And View

2. For **Purpose**, select **Tokenize/Re-Identification**. This environment will also be used to re-identify your data.

3. Set up a Tokenization job using the Tokenization Method. Execute the job.

Create Tokenization Job

✕

- Details
- Configuration
- Summary

Details

Enter the Job Name, Select Tokenization Method and Rule Set. Click the Next button to continue.

Target: Test_tok

Multi Tenant

Select Tokenization Method

Select Rule Set

Non-conforming Data

Stop job on first occurrence

This field applies only when the user chooses 'Mark job as Failed' for Non-Conforming Data behavior on Settings > Algorithms page or when Application Setting 'DefaultNonConformantDataHandling' is set to 'FAIL'.

Cancel
Back
Next
Save

8.1.9.23.2 Examples

Here is example data before and after Tokenization:

Before Tokenization

```

1,Erasmus,245 Park Ave,123-45-6789
2,Salathiel,245 park ave,123-45-6789
3,Salathiel,1003 Stant Drive,111-11-1111
```

After Tokenization

```

ID, fname, address, ssn
1, FQL71CmqK/pkd8B2vVP90304+ /
krT91dscS0rKQRACQ=, XFLst0IcSb0a2U1E0m1ACPkca0EVczZsEdxl225kF1M=, x6tJ4eyL4it4ji84h8Pzo
CW4QBZphEqD0y3hEj4h1jE=
2, 4bGZoCLpbV2zAMsTkcc51MTBKksvOP+t fAWucq+BnKM=, 0A9dJ5HN5oRx18ZY01f5Y8DofvhFoRo98cuQHZ
7YeEo=, Evj+LnETt7ABbX1TDPyNvvJe8WJnrhEWeS0lqtqrr4U=
3, L14T49FrCBYRiB0AKOY4vbnswb0n1RpqBU97EGg4RvA=, f6AR0T+HBoTW7+l0e8ok9rImj872PUnYYNYMDY
Sy4dw=, wYMvEhktV371kqH607afJHZloT+4DYNJxehWIcPZJzI=
```

8.1.10 General UI for extended algorithms

8.1.10.1 Overview

An algorithm plugin can be configured through the graphical user interface by entering the plugin's required configuration in JSON format. The following section describes how to use this feature.

8.1.10.2 GUI steps

1. At the top right of the **Algorithms** page, click **+ Algorithm**.

Add Algorithm ×

- Details
- Configuration
- Summary

Details

Specify the algorithm details.

Name

Description

Framework Name

2. Enter an **Algorithm Name**.
Info: This MUST be unique.
3. Enter a **Description**.
4. Select **Extended** as the Framework Name and click **Next**.

Add Algorithm

X

Configuration

Configure the algorithm.

Framework Name
IBAN (Plugin: dlpx-core)

Algorithm Extension

```
{
  "validateInput": false,
  "numCharsToMask": 36
}
```

Format JSON Validate Configuration

```
{
  "_typename": "AlgorithmFramework",
  "description": "This IBAN algorithm preserves country code, masks account and bank information and generates a new check number. A configurable amount of digits between 6 and 36 can be masked. Validation of the input and masking produced IBAN can be configured for custom instances of the algorithm.",
  "extensionSchema": {
    "id": "urn:jsonschema:algorithm:plugin:iban:IBAN",
    "properties": {
      "validateInput": {
        "type": "boolean",
        "required": true,
        "description": "Validation of IBAN numbers from data source."
      },
      "numCharsToMask": {
        "type": "integer",
        "required": true,
        "description": "Maximum number of chars from right of input to be masked, if present."
      }
    }
  },
  "frameworkId": 7,
  "frameworkName": "IBAN",
  "frameworkType": "STRING",
  "plugin": {
    "typename": "PluginBase",

```

Cancel Back Next Save

- In the second step, Use the **Framework Name** drop-down to create an instance corresponding with the selection.
- Based on the Framework Name option, the **Algorithm Extension** will be populated with default values in the corresponding text area. Provide a valid extension of the corresponding framework in JSON format in **Algorithm Extension** (Required).
- Selected framework details and configuration schema will be displayed on the right side of the **Algorithm Extension**.
- The **Format JSON** button is used to format the text from the **Algorithm Extension** text area into JSON format.
- The **Validate Configuration** button will validate the **Algorithm Extension** format and validate against the selected framework configuration schema.

For a plugin with a specific GUI like Character Mapping or Secure Lookup, their respective GUI will be shown when editing.

For other plugin instances, the user can only modify the description and extension of the algorithm instance from the plugin GUI. The select framework and algorithm name fields will be read-only.

The default selected framework populates the corresponding **Algorithm Extension** in the text area. When the framework changes, the **Algorithm Extension** and Schema details will be populated automatically.

Add Algorithm



- Details
- Configuration
- Summary

Configuration

Configure the algorithm.

Framework Name
 Dependent Date Shift (Plugin: dlp-core)

Algorithm Extension

```
{
  "minRange": 0,
  "maxRange": 0,
  "unit": "DAYS",
  "roll": false,
  "intervalRange": 0
}
```

```
{
  "__typename": "AlgorithmFramework",
  "description": "This Dependent Date Shift algorithm takes in 2 dates (designated date1 and date2). It masks date1 based on the provided values for minRange, maxRange, unit and roll. It then modifies the original interval based on intervalRange and unit to calculate date2. If the dates differ but the returned interval is zero (i.e.: the difference between the dates is smaller than the interval value), we assume the interval value to be 1 if date2 is later than date1 and -1 if date1 is later than date2. The masked results are deterministic for each pair of inputs with the same algorithm key and date and interval ranges. The algorithm does not allow for zero mask so all masked values will never be equal to the input. If date1 is not provided, date2 will be masked based on the provided values for minRange, maxRange, unit and roll.",
  "extensionSchema": {
    "id": "urn:jsonschema:algorithm:plugin:dateAlgorithms:DependentDateSh",
    "properties": {
      "minRange": {
        "type": "integer",
        "required": true,
        "description": "This number represents the smallest number of time units that will be added to date1 when masking. The range is inclusive of this value. Negative values represent units of time in the past and positive values represent units of time in the future. If date1 is not provided, this is applied to date2."
      }
    }
  }
}
```

Format JSON Validate Configuration

Cancel Back Next Save

If there are any issues with the **Algorithm Extension** on clicking **Validate Configuration**, an **Error message** will appear.



Algorithm Extension

```
{  
  "validateInput": false,  
}
```



SyntaxError: Expected double-quoted property name in JSON at position 30 (line 3 column 1)

If the **Algorithm Extension** is valid, a **Success message** will appear.



Algorithm Extension

```
{  
  "validateInput": false,  
  "numCharsToMask": 36  
}
```



Validation Successful

A **Utility** icon on the right top of the **Algorithm Extension** text area is available to upload and copy the local file reference and to pick an algorithm reference from existing algorithm instances.



Algorithm Extension

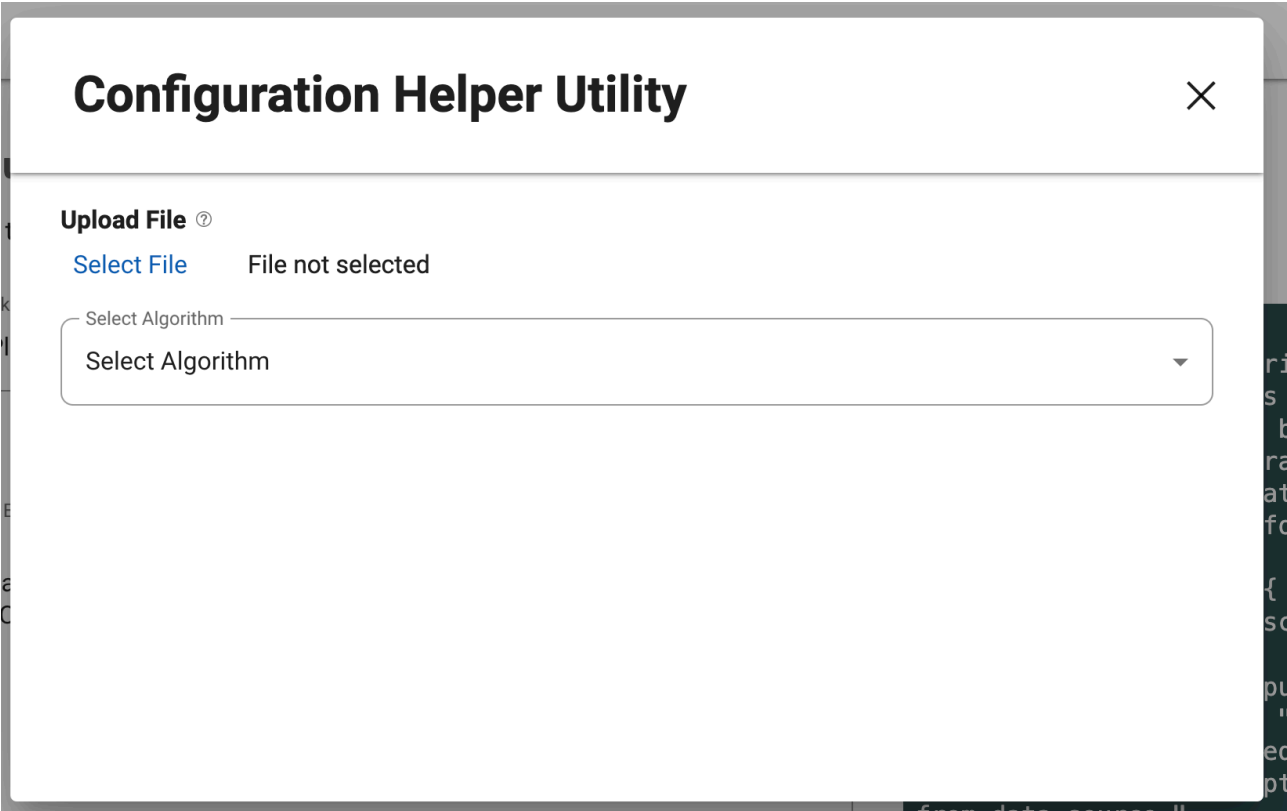
```
{  
  "validateInput": false,  
  "numCharsToMask": 36  
}
```

//

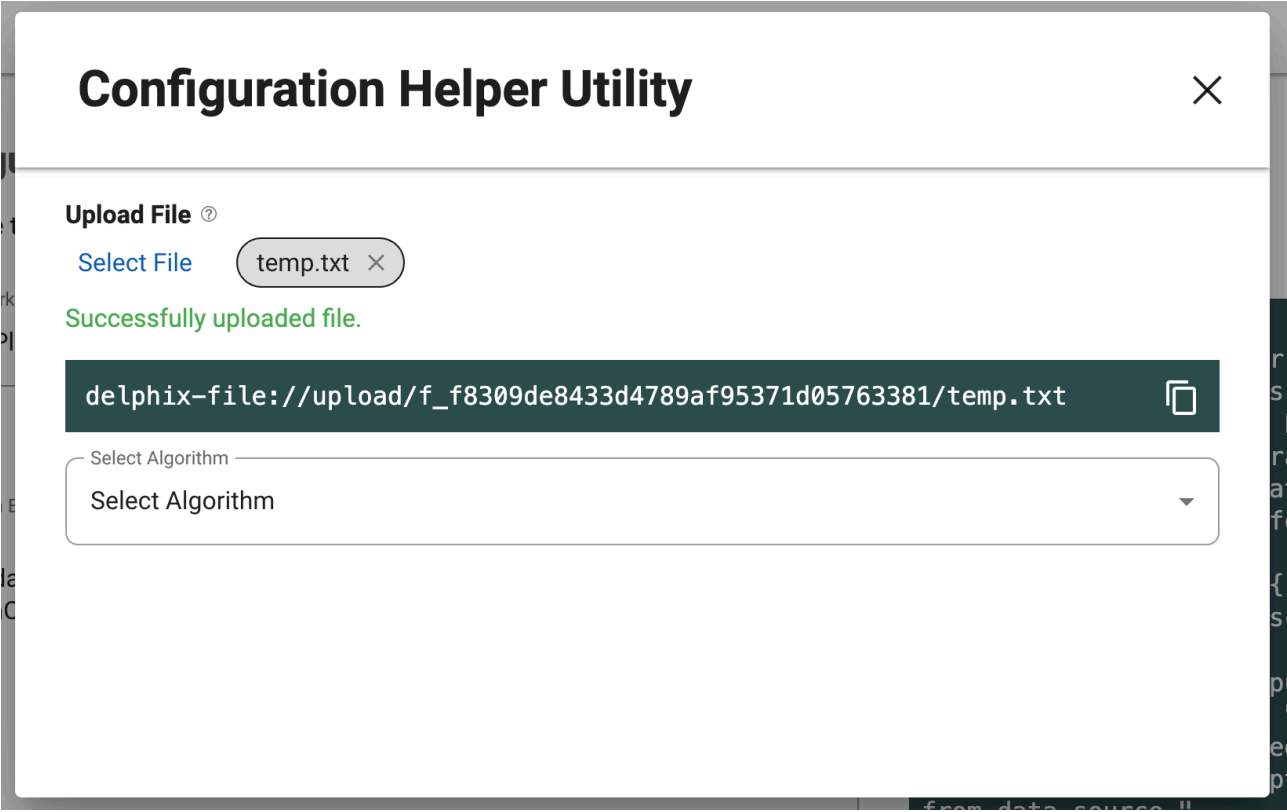
Format JSON

Validate Configuration

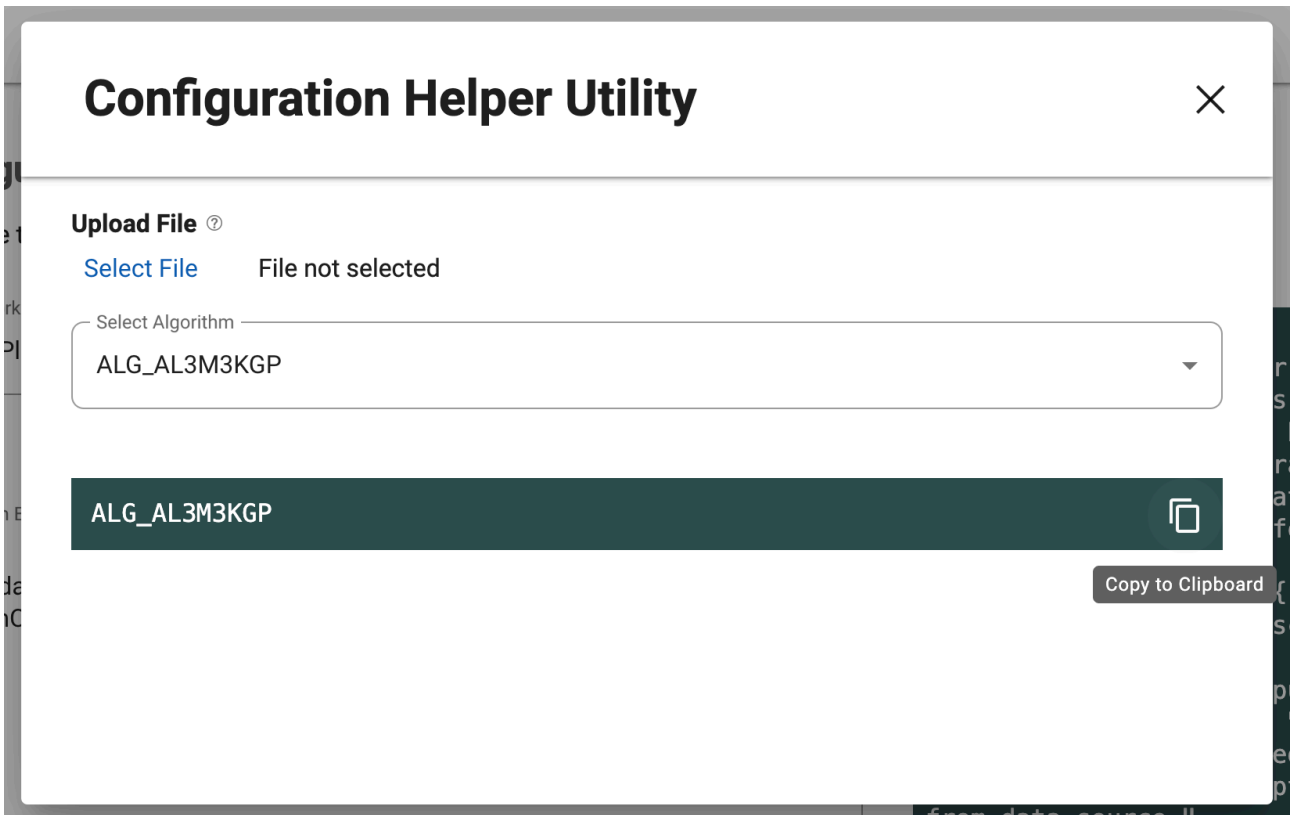
The **Configuration Helper Utility** offers a way to upload a file and receive a reference ID for algorithm extension, or to select an algorithm for copying algorithm name.



In the **Configuration Helper Utility**, click **Select File**. Select a file to upload, it uploads a file and renders a copyable value.



The **Configuration Helper Utility** also has a **Select Algorithm** option which renders a list of available algorithms to select.



8.2 Builtin Driver Supports

8.2.1 Introduction

In 6.0.11.0, Delphix introduced the first built-in driver support plugin for the Oracle database platform.

The native connector types with a built-in driver support plugin are:

Native Connector Type	Release
Oracle	6.0.11.0
MSSQL	6.0.12.0
PostgresSQL	11.0.0.0
Db2 LUW	14.0.0.0

Native Connector Type	Release
Db2 z/OS	16.0.0.0
Db2 iSeries	16.0.0.0

The built-in driver support plugins replace and improves upon the native connector database masking options of Disable Constraints, Drop Indexes, and Disable Triggers that have long had issues with functionality and negatively affecting job performance. Delphix has implemented the built-in driver support plugin for native connectors with Disable Constraints, Drop Indexes, and Disable Triggers tasks using the [Driver support plugin framework \(see page 996\)](#) released in 6.0.9.0. These optimizations apply to masking, reidentification, and tokenization jobs where these tasks are enabled.

For details on how to enable/disable these tasks on supported native connector jobs using the new Driver Support Plugin Framework, see [API Calls for managing masking job driver support tasks. \(see page 958\)](#)

To retrieve information about job failures due to driver support task failures, an execution event will be raised and is accessible via the `GET /execution-events` endpoint: 1. `eventType` -

`DRIVER_SUPPORT_TASK_FAILURE` 2. `exceptionDetail` - Error message about the task failure that will typically include the error code that is specific to the database platform

8.2.2 Oracle

For details on usage and known limitations of the Oracle Disable Constraints, Drop Indexes, and Disable Triggers driver support tasks, see [Oracle Built-in driver support plugin. \(see page 822\)](#)

8.2.3 MSSQL

For details on usage and known limitations of the MSSQL Disable Constraints, Drop Indexes, and Disable Triggers driver support tasks, see [MSSQL Built-in driver support plugin. \(see page 823\)](#)

8.2.4 PostgreSQL

For details on usage and known limitations of the PostgreSQL Drop Constraints, Drop Indexes, and Disable Triggers driver support tasks, see [PostgreSQL Built-in Driver Support Plugin. \(see page 826\)](#)

8.2.5 Db2 LUW

For details on usage and known limitations of the Db2 LUW Drop Constraints, Drop Indexes, and Drop Triggers driver support tasks, see [Db2 LUW Built-in Driver Support Plugin \(see page 828\)](#).

8.2.6 Db2 z/OS

For details on usage and known limitations of the Db2 z/OS Drop Constraints and Drop Triggers driver support tasks, see [Db2 z/OS Built-in Driver Support Plugin \(see page 829\)](#).


8.2.7 Db2 iSeries

For details on usage and known limitations of the Db2 iSeries Drop Constraints, Drop Indexes, and Disable Triggers driver support tasks, see [Db2 iSeries Built-in Driver Support Plugin](#) (see page 831).

8.2.8 Built-in Oracle driver support plugin

For instructions on how to enable/disable Disable Constraints, Drop Indexes and Disable Triggers on Oracle jobs, see [API Calls for managing masking job driver support tasks](#). (see page 958)

8.2.8.1 Optimizations


 Disable Constraints disables and re-enables constraints *while keeping the index associated with the constraint*. In order to drop and re-create the index associated with the constraint, enable Drop Indexes along with Disable Constraints.

For **in-place** jobs:

1. **Disable Constraints** disables and re-enables constraints on only masked columns.
2. **Drop Indexes** drops and re-creates indexes on only masked columns.
3. **Disable Triggers** disables and re-enables triggers on only tables with masked columns.

For **on-the-fly** jobs, the tasks will execute on all columns/tables in the ruleset.

8.2.8.2 Task execution order

 The order of task execution for built-in driver support plugins is fixed/unmodifiable.

The order of the tasks is as follows:

Pre-job:

1. Disable Constraints
2. Drop Indexes
3. Disable Triggers

Post-job (mirrored order):

1. Enable Triggers

2. Create Indexes
3. Enable Constraints

8.2.8.3 Enabling tasks on a job

For instructions on how to enable driver support tasks on jobs, see [API calls for managing masking job driver support tasks](#). (see page 958)

8.2.8.3.1 Important considerations

1. If masking primary key fields:
 - a. Use the same deterministic algorithms on primary key fields that reference each other, so that referential integrity is maintained when the masking transformation completes and all constraints are re-enabled.
 - b. Enable both Disable Constraints and Drop Indexes.
2. If dropping indexes on masked fields with constraints is desired, enable both Disable Constraints and Drop Indexes. The implementation of the optimizations has been modified, such that Disable Constraints only disables constraints and keeps indexes automatically created and Drop Indexes handles dropping/recreating indexes. The change in task order and separation of concerns with the functionality of the tasks resolves issues around missing indexes present with the legacy database masking option of Disable Constraints.

8.2.8.4 Known limitations

1. If masking a primary key field, if only Disable Constraints is enabled, the job will fail during the transformation. It is recommended to enable both Disable Constraints and Drop Indexes on any applicable job per the usage instructions above. In order to not have Drop Indexes enabled, adding a prescript that disables the desired constraints will also work, but note that this workaround may result in missing indexes.
2. Delphix supports the below indexes:
 - Normal indexes
 - Functional indexes (6.0.12.0 and later)
 - Local, global, and partial partition indexes (6.0.15.0 and later)

8.2.9 Built-in MSSQL driver support plugin

This page provides guidance on how the MSSQL driver support plugin interacts with various database objects, such as Primary Keys (PK), Foreign Keys (FK), Unique Constraints, and Unique Indexes during masking jobs.

8.2.9.1 Handling Primary Keys (PK)

In SQL Server, Primary Keys are managed as Constraints but are reported as Indexes in both the SQL Server Management Studio (SSMS) and the Continuous Compliance Engine UI.

To drop Primary Keys during a masking job, use the **Disable Constraints** option on the compliance engine UI.

 Selecting **Drop Index** will not affect Primary Keys, even though the UI may imply otherwise.

8.2.9.1.1 Example

To drop a Primary Key, the following SQL command is executed:

```
DROP CONSTRAINT [PK_tbl_maskPK]
```

8.2.9.1.2 Create table example


```
CREATE TABLE [dbo].[tbl_maskPK] (  
    NOT NULL,  
    CONSTRAINT [PK_tbl_maskPK] PRIMARY KEY CLUSTERED ([mask_pk] ASC)  
);
```

8.2.9.2 Handling Foreign Keys (FK)

Foreign Keys must be managed carefully during masking operations, as any table with a Foreign Key linking to a Primary Key must be included in the Rule Set. It is recommended to select both **Drop Indexes** and **Disable Constraints** in the compliance engine UI for these.

8.2.9.3 Handling Unique Constraints

For Unique Constraints, select **Disable Constraints** in the compliance engine. Similar to Primary Keys, Unique Constraints are managed as Constraints, even though they may appear as Indexes.

 Attempting to drop them via **Drop Index** will result in an error because SQL Server does not allow dropping Unique Constraints via the DROP INDEX command.

8.2.9.3.1 Example

To drop a Unique Constraint, the following SQL command is executed:


```
DROP CONSTRAINT [UCmask]
```

8.2.9.3.2 Create table example

```
CREATE TABLE [dbo].[tbl_maskUC] (  
    NOT NULL,  
    CONSTRAINT UCmask UNIQUE(mask)  
);
```

8.2.9.4 Handling Unique Indexes

For Unique Indexes, select **Drop Index** in the compliance engine.

 The **Disable Constraints** option does not affect Unique Indexes.

8.2.9.4.1 Example

To drop a Unique Index, the following SQL command is executed:

```
DROP INDEX [UIxmask] ON [dbo].[tbl_maskUIx]
```

8.2.9.4.2 Create table example

```
CREATE TABLE [dbo].[tbl_maskUIx] (  
    NOT NULL  
);  
CREATE UNIQUE INDEX UIxmask ON [dbo].[tbl_maskUIx] (mask);
```

8.2.9.5 Handling Check Constraints (untested)

For Check Constraints, SQL Server allows the use of the following SQL command to disable (not drop) the constraint:

```
ALTER TABLE [table]
NOCHECK CONSTRAINT [constraint];
```



It is important to verify whether the compliance engine **Disable Constraints** option correctly handles Check Constraints.

8.2.9.5.1 Create table example

```
CREATE TABLE [dbo].[tbl_maskCC] (
    [mask] [int] CHECK (mask >= 10)
);
```

8.2.10 Built-in PostgreSQL driver support plugin

For instructions on how to enable/disable Drop Constraints, Drop Indexes, and Disable Triggers on PostgreSQL jobs, see [API Calls for Managing Masking Job Driver Support Tasks](#) (see page 958).



Unlike other database platforms such as Oracle and MSSQL, PostgreSQL doesn't support the disabling of constraints. Therefore, this plugin has a Drop Constraints task instead of a Disable Constraints task.

8.2.10.1 Tasks

For **in-place** jobs:

1. **Drop Constraints** drops and re-creates constraints on masked columns only.
2. **Drop Indexes** drops and re-creates indexes on masked columns only (except for indexes automatically generated by PostgreSQL to support a primary key or unique constraint - those are dropped by Drop Constraints).

3. **Disable Triggers** disables and re-enables triggers on tables with masked columns only.

For **on-the-fly** jobs, the tasks will execute on all columns and tables in the ruleset.

8.2.10.2 Task Execution Order



The order of task execution for built-in driver support plugins is fixed/unmodifiable.

The order of the tasks is as follows:

Pre-job:

1. Drop Constraints
2. Drop Indexes
3. Disable Triggers

Post-job (mirrored order):

1. Enable Triggers
2. Create Indexes
3. Enable Constraints

8.2.10.3 Important Considerations

1. If masking primary key fields, use the same deterministic algorithms on primary key fields that reference each other so that referential integrity is maintained when the masking transformation completes and all constraints are re-created.
2. If dropping indexes on masked fields with constraints is desired, enable both Drop Constraints and Drop Indexes. PostgreSQL automatically creates unique indexes to support primary key and unique constraints. Those indexes cannot be dropped by the Drop Indexes task, but enabling Drop Constraints will allow it to drop those indexes when the constraints they support are dropped.

8.2.10.4 Known Limitations

1. If both a foreign key column and the primary key or unique column it references are not masked with the same deterministic algorithm (guaranteeing referential integrity), the job will fail due to an integrity constraint violation.
2. If masking a field that is a primary key or has a unique constraint, and Drop Indexes is the only task enabled, the job will fail. Drop Constraints must also be enabled so that it can drop the indexes PostgreSQL automatically creates to support constraints.

8.2.11 Built-in DB2 LUW driver support plugin

For instructions on how to enable/disable Drop Constraints, Drop Indexes, and Drop Triggers on DB2 LUW jobs, see [API Calls for Managing Masking Job Driver Support Tasks](#). (see page 958)



Unlike other database platforms, DB2 does not support disabling constraints, indexes, or triggers. Therefore, this plugin has Drop Constraints, Drop Indexes, and Drop Triggers tasks.

8.2.11.1 Tasks

For **in-place** jobs:

1. **Drop Constraints** drop and re-create constraints on masked columns only.
2. **Drop Indexes** drop and re-create indexes on masked columns only (except for indexes automatically generated by DB2 LUW to support a primary key or unique constraint, those are dropped by Drop Constraints).
3. **Drop Triggers** drop and re-create triggers on tables with masked columns only.

For **on-the-fly** jobs, the tasks will execute on all columns and tables in the ruleset.

8.2.11.2 Task Execution Order



The order of task execution for built-in driver support plugins is fixed/unmodifiable.

The order of the tasks is as follows:

Pre-job:

1. Drop Constraints
2. Drop Indexes
3. Drop Triggers

Post-job (mirrored order):

1. Create Triggers
2. Create Indexes
3. Create Constraints

8.2.11.3 Important Considerations

1. If masking primary key fields, use the same deterministic algorithms on primary key fields that reference each other so that referential integrity is maintained when the masking transformation completes and all constraints are re-created.
2. If dropping indexes on masked fields with constraints is desired, enable both Drop Constraints and Drop Indexes. DB2 LUW automatically creates indexes to support primary key and unique constraints. Those indexes cannot be dropped by the Drop Indexes task, but enabling Drop Constraints will allow it to drop those indexes when the constraints they support are dropped.

8.2.11.4 Known Limitations

1. If both a foreign key column and the primary key or unique column it references are not masked with the same deterministic algorithm (guaranteeing referential integrity), the job will fail due to an integrity constraint violation.
2. If masking a field that is a primary key or has a unique constraint, and Drop Indexes is the only task enabled, the job will fail. Drop Constraints must also be enabled so that it can drop the indexes DB2 LUW automatically creates to support constraints.

8.2.12 Built-in DB2 z/OS driver support plugin

For instructions on how to enable/disable Drop Constraints and Drop Triggers on DB2 z/OS jobs, see [API Calls for Managing Masking Job Driver Support Tasks](#).³³⁹



Unlike other database platforms, DB2 does not support disabling constraints or triggers. Therefore, this plugin has Drop Constraints and Drop Triggers tasks.

8.2.12.1 Tasks

For **in-place** jobs:

1. **Drop Constraints** drops and re-creates constraints on masked columns only.
2. **Drop Triggers** drops and re-creates triggers on tables with masked columns only.

For **on-the-fly** jobs, the tasks will execute on all columns and tables in the ruleset.

³³⁹ <https://delphixdocs.atlassian.net/wiki/spaces/CC/pages/60622355/17.0.0.0+API+calls+for+managing+masking+job+driver+support+tasks>

8.2.12.2 Task execution order



The order of task execution for built-in driver support plugins is fixed/unmodifiable.

The order of the tasks is as follows:

Pre-job:

1. Drop Constraints
2. Drop Triggers

Post-job (mirrored order):

1. Create Triggers
2. Create Constraints

8.2.12.3 Important considerations

1. There is no Drop Indexes task available.
2. If masking primary key fields, use the same deterministic algorithms on primary key fields that reference each other so that referential integrity is maintained when the masking transformation completes and all constraints are re-created.

8.2.12.4 Known limitations

1. Drop Constraints is unable to detect or drop foreign key constraints with `PERIOD BUSINESS_TIME` in their keys.
2. Foreign key constraints dropped and recreated by Drop Constraints will not have the `ENABLE QUERY OPTIMIZATION` clause if originally specified.
3. If both a foreign key column and the primary key or unique column it references are not masked with the same deterministic algorithm (guaranteeing referential integrity), the job will fail due to an integrity constraint violation.

8.2.13 Built-in DB2 iSeries driver support plugin

For information about how to enable/disable Drop Constraints, Drop Indexes, and Disable Triggers on DB2 iSeries jobs, refer to [API Calls for Managing Masking Job Driver Support Tasks](https://portal.document360.io/continuous-compliance-10-0-0-0/docs/api-calls-for-managing-masking-job-driver-support-tasks).³⁴⁰



Unlike other database platforms, DB2 does not support disabling constraints or indexes. Therefore, this plugin has Drop Constraints, Drop Indexes, and Disable Triggers tasks.

8.2.13.1 Tasks

For **in-place** jobs:

1. **Drop Constraints** drops and re-creates constraints on masked columns only.
2. **Drop Indexes** drops and re-creates indexes on masked columns only (except for indexes automatically generated by DB2 iSeries to support a primary key or unique constraint, those are dropped by Drop Constraints).
3. **Disable Triggers** disable and re-enable triggers on tables with masked columns only.

For **on-the-fly** jobs, the tasks will execute on all columns and tables in the ruleset.

8.2.13.2 Task Execution Order



The order of task execution for built-in driver support plugins is fixed/unmodifiable.

The order of the tasks is as follows:

Pre-job:

1. Drop Constraints
2. Drop Indexes
3. Disable Triggers

Post-job (mirrored order):

1. Enable Triggers
2. Create Indexes

³⁴⁰ <https://portal.document360.io/continuous-compliance-10-0-0-0/docs/api-calls-for-managing-masking-job-driver-support-tasks>

3. Create Constraints

8.2.13.3 Important Considerations

1. If masking primary key fields, use the same deterministic algorithms on primary key fields that reference each other so that referential integrity is maintained when the masking transformation completes and all constraints are re-created.

8.2.13.4 Known Limitations

1. If both a foreign key column and the primary key or unique column it references are not masked with the same deterministic algorithm (guaranteeing referential integrity), the job will fail due to an integrity constraint violation.
2. Disable Triggers are only effective for Db2 iSeries version 7.2 or later, as disabling triggers weren't supported in versions 7.1 and earlier. If run against those prior versions, Disable Triggers will operate silently without dropping any triggers.
3. Drop Indexes cannot detect indexes on masked columns if those column names are used in key expressions.
4. Indexes dropped and recreated by Drop Indexes will not have the following if originally specified:
 - a. `FOR SYSTEM NAME system-object-identifier` clause
 - b. `RCDFMT format-name` clause
 - c. `RENAME table-system-column-name TO` clause
5. Requires journaling to be enabled on the tables, the job will fail with ErrorCode(-7008).

9 Masked provisioning

This section contains the following topics:

- [Configuring virtualization service for masked provisioning \(see page 833\)](#)
- [Provision masked VDBs \(see page 834\)](#)

9.1 Configuring virtualization service for masked provisioning

9.1.1 Introduction

During the VDB provisioning process, the Virtualization Engine can optionally run a masking job from a Continuous Compliance engine on the VDB. Use these instructions to customize the host address, port number, and/or login credentials that the Virtualization Engine will use to contact the Masking Engine.



Important validation notices

When configuring masked provisioning, ensure that the versions of the Virtualization Engine and Masking Engine are compatible. See the [compatibility matrix](#). (see page 0)

Old versions of the serviceconfig or any information associated with them are not tracked. In particular, if you have been using the local masking service or a remote service and then change to a new remote service Delphix will start throwing out any old job information on the next masking job/fetch or GUI reload. Users should not rely on that information being preserved through serviceconfig updates.

Delphix does not validate network availability between the two engines or any other hosts that both engines might want to communicate with. The state or availability of either host is not checked, if either host becomes unduly slow, congested, or unresponsive Delphix will not be able to issue compelling warnings regarding those issues.

9.1.2 Instructions

Use these instructions to customize the host address, port number, and/or login credentials that the Virtualization Engine will use to contact the Masking Engine.



This does not alter the Continuous Compliance Engine UI port. It is specific to coordinating communication between the Virtualization Engine and a Masking Engine about available masking jobs and job results.

To change the Virtualization Engine's connection details for its Masking Engine:

1. Using a shell, login to the **CLI** using:
 - On 5.2 and earlier releases: **delphix_admin**.
 - On 5.3 and later releases: **admin**.
2. At the **CLI** root prompt, type **maskingjob**.
3. At the **maskingjob** prompt, type **serviceconfig**.
4. To list service configurations, type **ls**.
5. At the serviceconfig, type **select `MASKING_SERVICE_CONFIG-1**.
6. To view the configurations, type **ls**.
7. With this service config selected, enter **update**.
8. In the update mode, use the **set** command to modify the configuration. For example, type **set port=[YOUR DESIRED PORT NUMBER]** to change the port number.
9. Commit the change by typing **commit**.
10. Type **ls** to confirm the configurations.
11. Type **exit** to exit the CLI.

9.2 Provision masked VDBs

Masked virtual databases (VDBs) function just like normal VDBs. The only distinction is that the data they contain has been masked by a masking job. Masked VDBs can be replicated to a separate Delphix Engine (in non-prod) without sending the original data that was obfuscated during masking using a process called Selective Data Distribution (SDD). This topic describes how to work with masked VDBs.

9.2.1 Prerequisites

Before attempting to create a Masked VDB, you should be familiar with both Delphix Virtualization and Delphix Masking concepts and workflows.

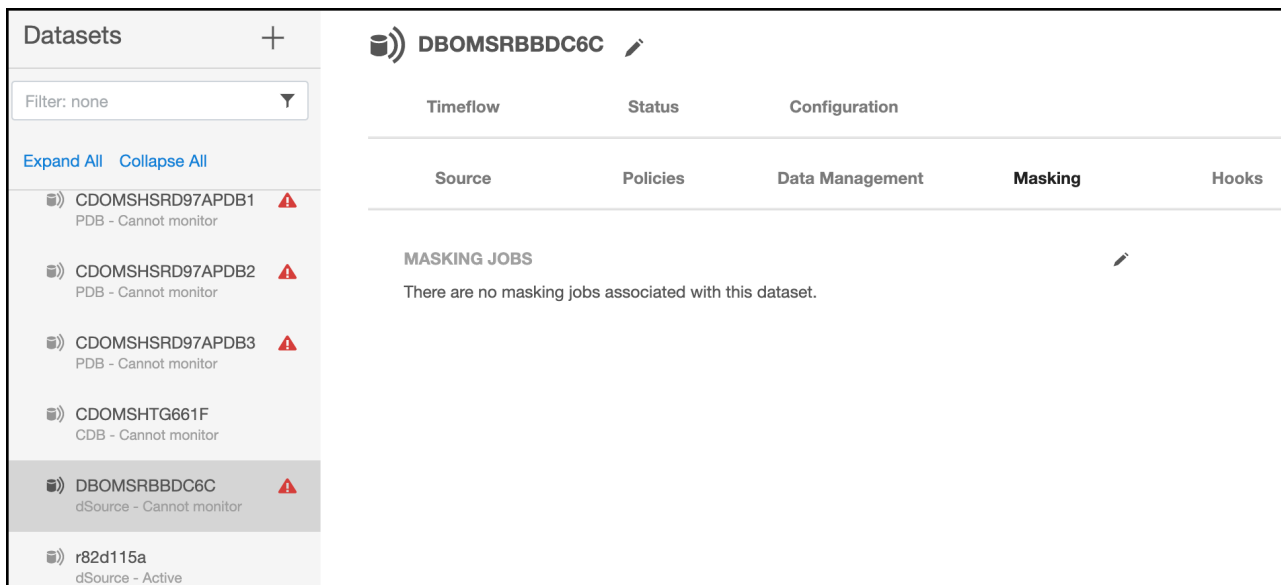
9.2.2 Restrictions

- A single masking job cannot be assigned to multiple VDBs simultaneously. If you are using the same masking ruleset on multiple VDBs, be sure to create a unique job for each VDB to avoid any issues with provisioning or refreshing.
- Provisioning or refreshing masked VDBs is only supported for Oracle, MS SQL Server, and Sybase. Provisioning or refreshing other types of masked VDBs such as DB2 are not supported.
- You cannot apply additional masking jobs to a masked VDB or its children.
- If a masking job has been applied to a VDB, you cannot create an unmasked snapshot of that VDB.

- Masking must take place during the process of provisioning a VDB. If an existing VDB has not had a masking job applied to it, then you cannot mask that particular VDB at any point in the future. All the data within the VDB and its parents will be accessible if it is replicated using SDD.
- When selecting a connector to use for Masked Provisioning, a "basic" connector must be used **unless** you are masking an Oracle Pluggable Database (PDB), in which case an "advanced" connector must be used.
- Only in-place masking jobs can be selected.
- Masked Provisioning is supported on Oracle RAC only when used with "script-based masking" and not when a masking job is used for SDD.

9.2.3 Identifying and navigating to masked VDBs

Masked VDBs appear in the Virtualization Engine's Datasets pane, just like regular VDBs. They are most obviously identified by the different icons used to represent them. In addition, a masked VDBs Configuration tab will contain information about the masking job that you applied to it. Generally, anything you can do with an unmasked VDB is also possible with a masked VDB.



The screenshot displays the 'Datasets' interface. On the left, a list of datasets is shown with their respective icons and status indicators. The main panel shows the configuration for the selected dataset 'DBOMSRBDC6C'. The 'Masking' tab is active, indicating that no masking jobs are currently associated with this dataset.

9.2.4 Provisioning masked VDBs

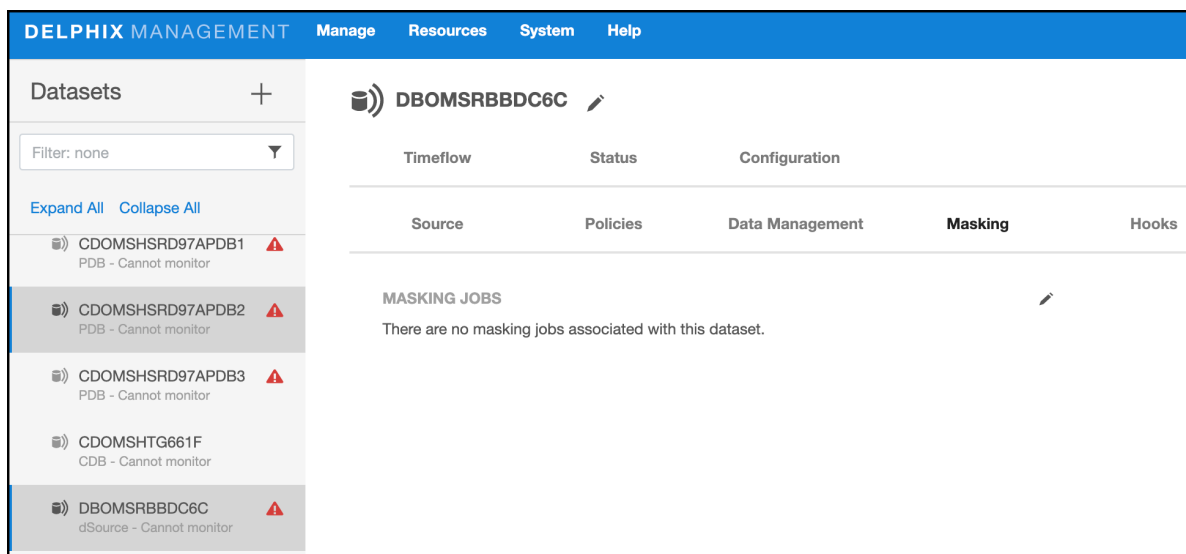
- In the Virtualization Engine, associate a masking job with a dSource.
- Use the dSource provision wizard to provision a VDB with a masking job.

9.2.4.1 Associating a masking job with the dSource

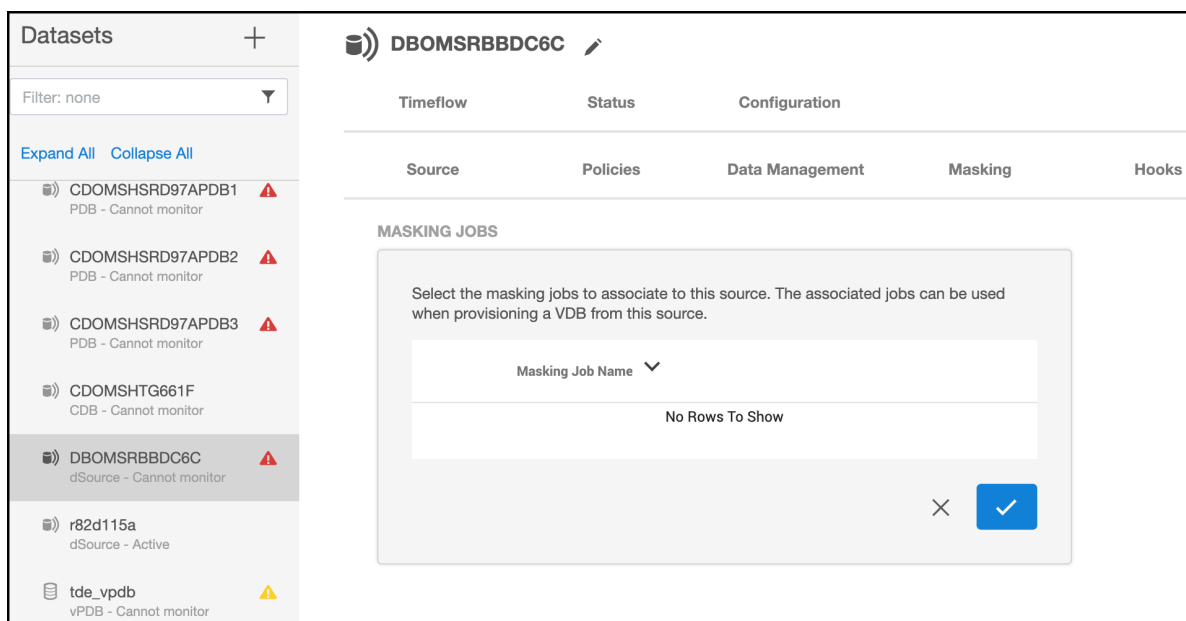
To provision a masked VDB, you must first indicate that the masking job you are using is complete and applicable to a particular database. You do this by associating the masking job with a dSource.

1. In the **Datasets** panel on the left-hand side of the screen, click the dSource to which the masking job is applicable and with which it will be associated.

2. Click the **Configuration** tab.
3. Click the **Masking** tab.




4. Click the **pencil** icon to edit. All masking jobs on this Delphix Engine that have not been associated with another dSource will be listed on the right-hand side.
5. Select the **job** you want to associate with this dSource.
6. Click the tickmark symbol to confirm.



7. Repeat for any other jobs that you want to associate with this dSource at this time.

The Delphix Engine now considers this masking job to be applicable to this dSource and ready for use. When provisioning from snapshots of this dSource, this masking job will now be available.

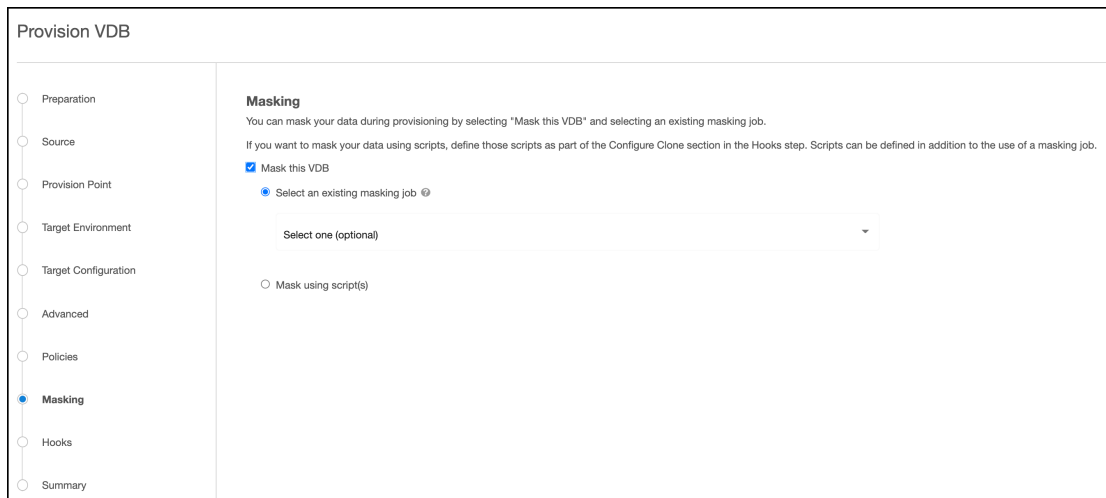
 Masking jobs can also be associated with virtual sources in addition to dSources.

A masking job must be Multi-Tenant for creating a masked VDB. The Multi-Tenant option allows existing rulesets to be reused to mask identical schemas via different connectors. The connector can be selected at job execution time.

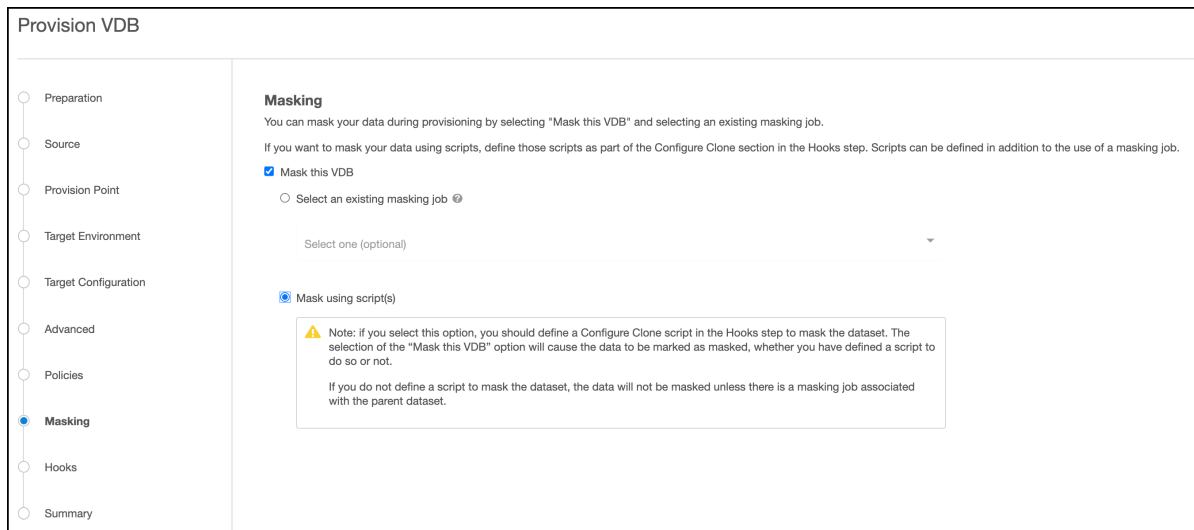
9.2.4.2 Provisioning a masked VDB using the dSource provisioning wizard

The steps required to provision a masked VDB are almost identical to the steps required to provision an unmasked VDB. Once you have created a masked VDB, you cannot unmask it, nor can you alter which masking job it uses. All snapshots in the VDBs TimeFlow will always be masked using the masking method that you selected when you provisioned the masked VDB.

1. In the **Datasets** panel on the left-hand side of the screen, select the dSource.
2. Click the **TimeFlow** tab.
3. Click the **Provision VDB** icon.
4. Review the information for Installation Home, Database Unique Name, SID, and Database Name. Edit as necessary.
5. Review the Mount Base and Environment User. Edit as necessary.
 - If you want to use login credentials on the target environment that are different from the login credentials associated with the Environment User, select Specify Privileged Credentials.
6. Click **Next**.
7. If necessary, edit the **Target Group** for the VDB.
8. Select the **None** option for the Snapshot Policy for the VDB.
 - **Snapshot Policy Selection:** For almost all use cases involving Masked VDBs, a Snapshot Policy of None is appropriate. Using a Snapshot Policy in conjunction with SDD can result in the leak of sensitive data.
9. Click **Next**.
10. Click **Mask this VDB**. You will be presented with two options to mask this VDB:
 - **Select an existing masking job:** Choose this option if you want to mask using a preconfigured Masking Job. Only masking jobs that have been associated with the parent dSource will be available.
 - **Selecting Unique Masking Jobs:** If you are using the same masking ruleset on multiple VDBs, be sure to create a unique job for each VDB to avoid any issues when provisioning or refreshing.



- **Masking using script(s):** Alternatively, you may define some Configure Clone scripts in the Hooks step to perform masking.
- **Defining Configure Clone Hooks to Mask VDB:** If you choose to mask using script(s), you must define the Configure Clone Hooks to run masking jobs yourself. If you don't define any Configure Clone hooks in the Hooks step, the data will be marked as masked, but it will not be masked.



11. Click **Next**.
12. Specify any **Pre or Post Scripts** that should be used during the provisioning process. If the VDB was configured before running the masking job using scripts that impact either user access or the database schema, those same scripts should also be used here. Be sure to define the **Configure Clone** hooks to run the masking job if you choose to mask using a script(s) in the Masking step.

Provision VDB

- Preparation
- Source
- Provision Point
- Target Environment
- Target Configuration
- Advanced
- Policies
- Masking
- Hooks**
- Summary

Hooks

Scripts can be defined to run at multiple hook points in the process. Select a Hook Point and then click + to add a script to run at that point.

Hook Points +

- ▼ **Configure Clone**
No operations
- ▼ **Pre Refresh**
No operations
- ▼ **Post Refresh**
No operations
- ▼ **Pre Rollback**
No operations
- ▼ **Post Rollback**
No operations
- ▼ **Pre Snapshot**
No operations
- ▼ **Post Snapshot**
No operations
- ▼ **Pre Start**
No operations
- ▼ **Post Start**
No operations
- ▼ **Pre Stop**
No operations
- ▼ **Post Stop**
No operations

Configure Clone

DESCRIPTION
Operations performed on initial provision and after a refresh.

OPERATION ORDERING

No operations for this hook point.

13. Click **Next**.

14. Click **Submit**.

If you click Actions in the upper right-hand corner, the Actions sidebar will appear and list an action indicating that masking is running. You can verify this and monitor progress by going to the Masking Engine page and navigating to the **Monitor > Job Executions** page.

CONTINUOUS COMPLIANCE 22.0.0.0
Environments **Monitor** Settings Admin Audit
?

Async Tasks

Job Executions


Monitor > Job Executions

0 Jobs Queued
 1/7 Jobs Running
 1024/6028 Memory Usage(MB)

Environment	Job Name	Status	Progress	Submit Time	Start Time	Type	Actions
Env1	sample_mask	◻ RUNNING	◻ 5 of 10 - Starting	4 Apr 2024 13:30 IST	4 Apr 2024 13:30 IST	MASK	⋮
Env1	sample_mask	✔ SUCCEEDED	✔ 10 of 10 - Job Completed	4 Apr 2024 13:28 IST	4 Apr 2024 13:28 IST	MASK	⋮
Env1	sample_profile	✔ SUCCEEDED	✔ 5 of 5 - Profiling Complete	4 Apr 2024 13:27 IST	4 Apr 2024 13:27 IST	PROFILE	⋮
env_0H0S0FSL	tokenize_OC5J00DU	✔ SUCCEEDED	✔ 10 of 10 - Job Completed	3 Apr 2024 19:47 IST	3 Apr 2024 19:47 IST	TOKENIZE	⋮
env_05M20WUQ	mask_BUHJAZT9	▲ FAILED	▲ 10 of 10 - Job Completed	3 Apr 2024 19:47 IST	3 Apr 2024 19:47 IST	MASK	⋮
env_05M20WUQ	mask_BUHJAZT9	✔ SUCCEEDED	✔ 10 of 10 - Job Completed	3 Apr 2024 19:46 IST	3 Apr 2024 19:46 IST	MASK	⋮
env_KSSLZXSV	mask_M2WJE59T	▲ FAILED	▲ 10 of 10 - Job Completed	3 Apr 2024 19:46 IST	3 Apr 2024 19:46 IST	MASK	⋮
env_05M20WUQ	mask_BUHJAZT9	✔ SUCCEEDED	✔ 10 of 10 - Job Completed	3 Apr 2024 19:44 IST	3 Apr 2024 19:44 IST	MASK	⋮
env_B2MM3PQ1	prof_OLTBALS1	✔ SUCCEEDED	✔ 5 of 5 - Profiling Complete	3 Apr 2024 19:42 IST	3 Apr 2024 19:43 IST	PROFILE	⋮
env_0H0S0FSL	tokenize_OC5J00DU	✔ SUCCEEDED	✔ 10 of 10 - Job Completed	3 Apr 2024 19:41 IST	3 Apr 2024 19:41 IST	TOKENIZE	⋮
env_07PPBN7P	mask_BMGJOMST	✔ SUCCEEDED	✔ 10 of 10 - Job Completed	3 Apr 2024 19:37 IST	3 Apr 2024 19:37 IST	MASK	⋮

Displaying 1 to 11 of 95

Masked provisioning – 839

- 
 Once you have created a masked VDB, you can provision its masked data to create additional VDBs, in the same way, that you can provision normal VDBs. Since the parent masked VDB contains masked data, child VDBs will only have masked data. This is a great way to distribute multiple independent copies of masked data that is both time and space-efficient.

9.2.5 Refresh a masked VDB

You refresh a masked VDB in exactly the same way as you refresh a normal VDB. As with provisioning a masked VDB, the masking job will be run during the refresh process.

1. Login to the Delphix Management application.
2. Click Manage.
3. Select Datasets.
4. Select the VDB you want to refresh.
5. Click the Refresh VDB button (2 circular arrows).
6. Select More Accurate and Next.
7. Select the desired refresh point snapshot or click the eye icon to choose the latest available range, A point in time, or An SCN to refresh from.
8. Click Next.
9. Click Submit to confirm.
10. Click the Actions link to watch the progress of the refresh job.
11. To see when the VDB was last refreshed/provisioned, check the Time Point on the Status page.

9.2.6 Disassociating a masking operation on a dSource

If a masking job is found to be unsuitable or should be retired, you can disassociate it through the same database card that you used to associate it.

1. Deselect the job.
2. Click the green arrow to confirm. Note that this will only prevent the creation of new masked VDBs with this job. It will not alter existing masked VDBs in any way. When disassociating a job, review the existing masked VDBs and consider whether you need to delete or disable any of them.

9.2.7 Masked VDB data operations

The following data operations are available to masked VDBs:

- Rewind: Alter the database to contain masked data from a previous point in time.

- Refresh: Get new data from the parent dSource and mask it.
- Disable: Turn off the database and remove it from the host system.
- Enable: Turn on the database and make it available on the host system.

9.2.8 Continuous Data and Continuous Compliance Engine compatibility matrix

Virtualization Engine Version	Masking Engine Version
5.0 releases	5.0 releases (minor versions do not need to match)
5.1 releases	5.1 releases (minor versions do not need to match)
5.2 releases	5.2 releases (minor versions do not need to match)
5.2.5.0 (or later 5.2 minor release)	5.2.5.0 (or later 5.2 minor release)
5.3.0.0 and later, including later major releases (e.g. 6.0)	5.3.0.0 and later, including later major releases (e.g. 6.0) and minor versions do not need to match

10 Managing multiple engines for masking

This section contains the following topic:

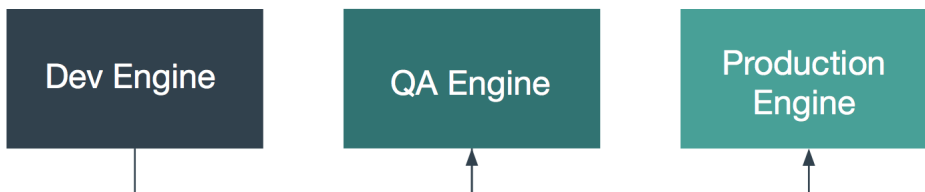
- [Introduction \(Managing multiple engines for masking\)](#) (see page 842)
- [Sync concepts](#) (see page 844)
- [Sync endpoints](#) (see page 854)
- [Algorithm syncability](#) (see page 859)
- [User workflow examples](#) (see page 860)
- [Change log](#) (see page 875)

10.1 Introduction (Managing multiple engines for masking)

Your organization may have more than one masking engine, and in certain circumstances, it may want to coordinate the operation of those engines. In particular, there are two specific scenarios in which an organization could benefit from some level of interaction and orchestration between multiple masking engines.

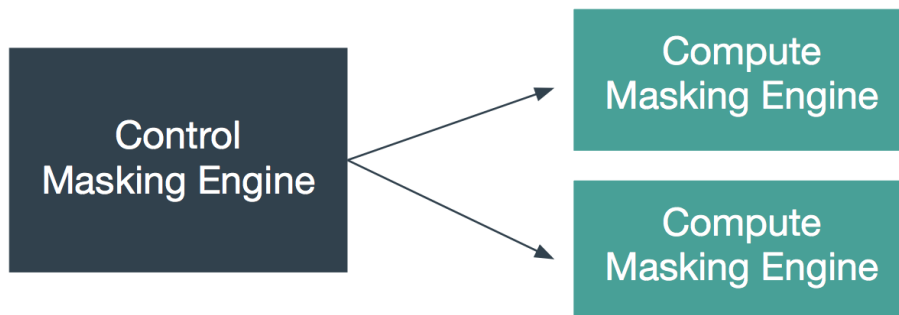
10.1.1 Software Development Life Cycle (SDLC)

Using an SDLC process often requires setting up multiple masking engines, each for a different part of the cycle (Development, QA, Production).



10.1.2 Horizontal scale

For many organizations, the size of the profiling and masking workloads requires more than one production masking engine. These masking engines can be identical in configuration or be partially equivalent depending on the organization's needs.



10.1.3 Best practice guide and example architectures for synchronizing

Both of these use cases require various objects to be moved between masking engines, such as Connectors, Rule Sets, and more. Engine synchronization provides a general and flexible way to move the objects necessary to run an identical job on another engine. The following sections describe how to use the Masking APIs to accomplish this.

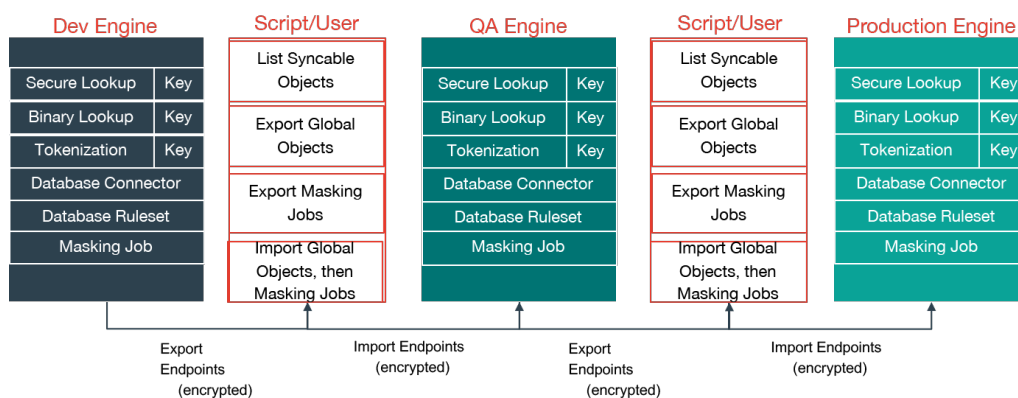
It is recommended that the syncable objects move in **only one direction**. That is, objects should be exported from one engine and imported into others but should not go in the other direction. This recommendation is primarily to simplify management of which objects exist on which engine.

For each of the scenarios above, an example architecture is described below. Note that the two architectures could be combined by having multiple production engines instead of a single one.

10.1.3.1 SDLC

The first architecture addresses the desire to author algorithms on one engine, to test and certify them on another, and finally to deploy them to a production engine. Here, algorithms are authored on the first engine, labeled “Dev Engine” in the diagram below. When the developer is satisfied, the algorithms are exported from the Dev Engine and imported to the QA Engine where they can be tested and certified. Finally, they are exported from the QA engine and imported to the production engine.

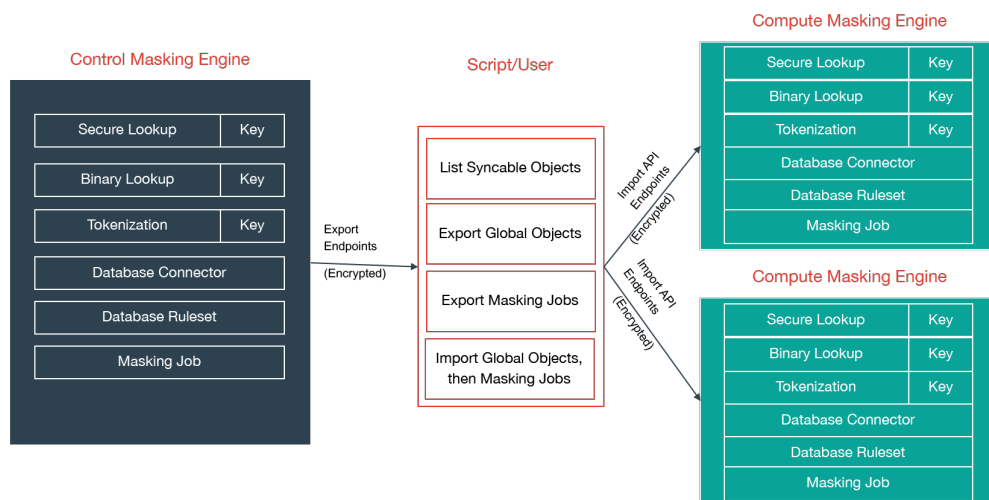
SDLC (Algorithm) Use Case



10.1.3.2 Horizontal Scale

The second architecture aims to address the problem of horizontal scale – that is, achieving consistent masking across a large data estate by deploying multiple masking engines. In this architecture, syncable objects are authored on one engine, labeled “Control Masking Engine” in the diagram below. Those objects are then distributed to “Compute Masking Engines” using the engine synchronization APIs. The synchronized algorithms and masking jobs will produce the same masked output on all of the engines, thus enabling large data estates to be masked consistently.

Horizontal Scale Use Case



10.2 Sync concepts

10.2.1 Syncable object

Syncable objects are external representations of masking engine objects that can be exported from one engine and imported into another.



Sync does not currently support the following objects:

- Application
- Certain algorithms (see [Algorithm Syncability](#) (see page 859) for details)



Forward compatibility is not supported by sync, meaning that sync bundles from newer version engines may not import successfully into older version engines. If attempted, this could potentially result in an unexpected state or an error on the older version engine. However, backwards compatibility is supported and sync bundles from older version engines will import as expected into newer version engines, unless the sync bundle contains objects for a deprecated feature that no longer exists on the newer version engine.

10.2.2 Object identifiers and types

Sync uses object identifiers to name unique objects within the engine. The `/syncable-objects` endpoint provides a list of all object identifiers for a particular object type.

The following object types are currently supported:

- ALGORITHM_PLUGIN
- APPLICATION_SETTINGS
- CLASSIFIER
- CREDENTIAL_PATH
- DATABASE_CONNECTOR
- DATABASE_RULESET
- DATASET_CONNECTOR
- DATASET_FORMAT
- DATASET_RULESET
- DOMAIN
- DRIVER_SUPPORT_PLUGIN
- ENVIRONMENT
- FILE_CONNECTOR
- FILE_FORMAT
- FILE_RULESET
- GLOBAL_OBJECT
- JDBC_DRIVER
- KEY
- MASKING_JOB
- MOUNT_INFORMATION
- PASSWORD_VAULT
- PROFILE_EXPRESSION
- PROFILE_TYPE_EXPRESSION
- PROFILE_JOB
- PROFILE_SET
- REIDENTIFICATION_JOB
- TOKENIZATION_JOB
- USER_ALGORITHM

The following lists the object types that are simply for the purpose of referencing a particular state of the exported object. These are not meant to be exported by request. The functions of these are further explained in the latter sections.

- ALGORITHM_REFERENCE
- DOMAIN_REFERENCE
- PROFILE_EXPRESSION_REFERENCE
- PROFILE_SET_REFERENCE
- SOURCE_DATABASE_CONNECTOR
- SOURCE_DATASET_CONNECTOR
- SOURCE_FILE_CONNECTOR

10.2.3 Dependencies



When exporting masking objects, a single export cannot contain multiple objects with the same name (e.g., two connectors with the same name).

Most objects within the Masking Engine are compositional. Properly capturing the behavior of a syncable object requires both the object and its dependencies. Fortunately, all the necessary dependencies are exported along with the object you request; thus, the dependencies are not something you need to keep track of and worry about.

10.2.3.1 Syncable Object dependencies relationship

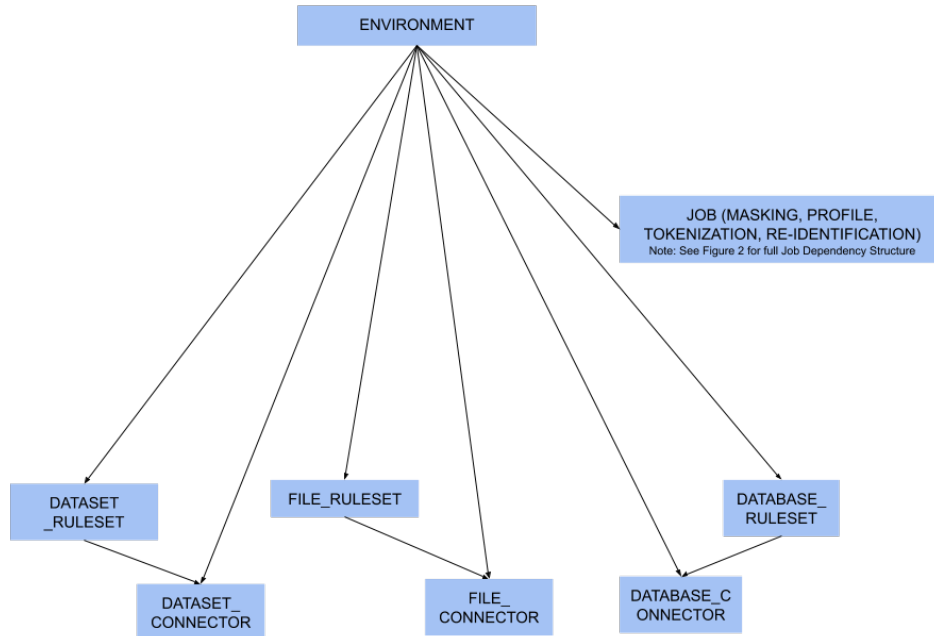


Figure 1 - environment dependencies

- While rulesets and connectors are dependencies for Jobs (see Figure 2), you may also have rulesets and connectors that are not assigned to a job. In this case, they are considered to be direct dependencies for an environment.

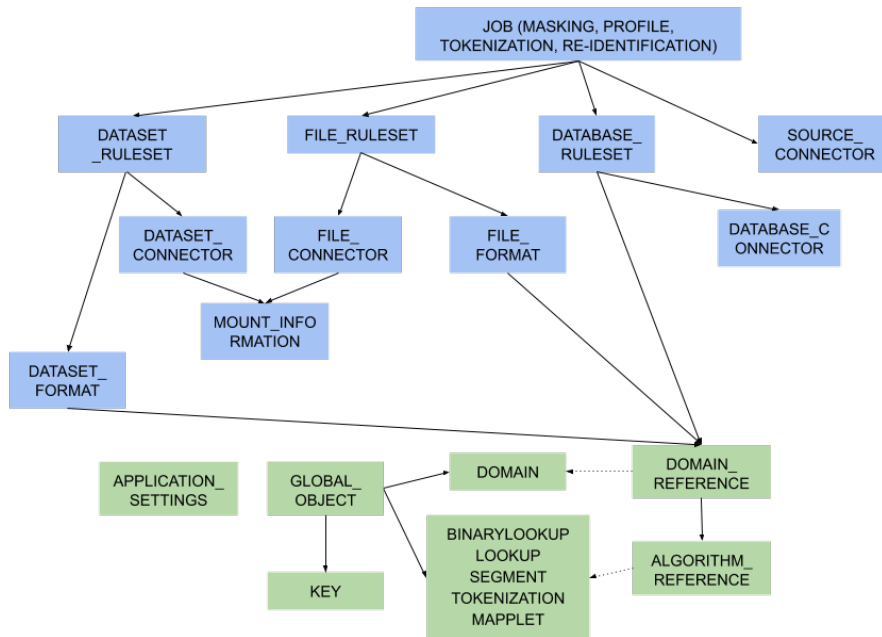


Figure 2 - object dependencies

= Green represents global objects (objects that are central to the entire engine), and blue represents objects that need to be a part of an environment

10.2.4 Object revision tracking

The revision hash is used to help you determine whether the behavior of a syncable object is the same between engines. Because objects within the Masking Engine are compositional, the behavior of an object is influenced by all of its dependencies. When a syncable object is listed or exported, the Masking Engine computes a revision_hash, which uniquely identifies the object’s behavior.

The revision_hash is a SHA1 hash that represents that object’s state, as well as the state of all objects it depends on. If two objects have the same revision hash, it is safe to assume that the behavior of the objects is the same. However, it is possible for two objects to have the same behavior but have divergent revision hashes. For example, you could have two lookup algorithms with the same name, lookup file, and key, and they do not necessarily guarantee to have the same revision hash.

- The `revision_hash` does not change when the password or the ssh key for the `FILE_CONNECTOR`, `DATASET_CONNECTOR` or `DATABASE_CONNECTOR` is updated. This is intentionally done because we do not export the password or the ssh key for security purposes. This allows users to update the password after import without changing the `revision_hash`. If a user is **overriding** a connector that already has a password set, the import **does not** reset the password and will leave the current, pre-import value.

- The `revision_hash` may change from version to version, and the hash comparison should be done only if both the source engine and the target engine are on the same version of the product. It is also not guaranteed to be the same between two engines at the same version if they are synced from an engine at some other version. E.g. There are three engines as follows:

- Engine A - version 5.3.2.0
- Engine B - version 5.3.3.0
- Engine C - version 5.3.3.0

If B and C are synced from A, then the `revision_hash` is not guaranteed to be the same between B and C.

Best Practice: A -> B -> C.

10.2.5 Export document

You can export one or more syncable objects that are listed in the `/syncable-objects` endpoint. The export document will include the set of objects that you requested for export and all of their dependencies that are required to properly import those objects into another engine.

The export document is exported as an opaque blob. Do not edit it outside of the Masking Engine.

10.2.6 Security

In most cases, an export document contains all the state necessary to re-create each of its objects (see [this note](#) (see page 849) about connector objects for one exception). In some instances, users might consider an object to be sensitive. For example, an algorithm object contains all of the information needed to produce identical algorithm results on a different engine (the algorithm's secret key, etc.). If the algorithm is being used in a production environment, then users may consider the algorithm definition and any export document containing the algorithm to be sensitive information. Therefore, export document access control, transmission, and storage should all be considered with care.

10.2.6.1 Access control

The Masking Engine only allows administrative users to make Sync API calls. When creating an administrative user account, keep in mind that the account owner will be able to access the Sync APIs to export and import objects. For this and other reasons, administrative accounts should only be created for trusted individuals.

Non-administrative accounts are not allowed to use the Sync APIs.

10.2.6.2 Transmission security

An export document containing a sensitive object should only be transmitted over a secure channel. This applies to situations where the Masking Engine is one of the transmission endpoints and when it is not. For example, when uploading (downloading) an export document to (from) the Masking Engine, the Sync API calls, like all Masking API calls, should be performed over HTTPS. Similarly, if an export document is transferred from a user's laptop to a server, the export document should be transmitted securely.

10.2.6.3 Storage security

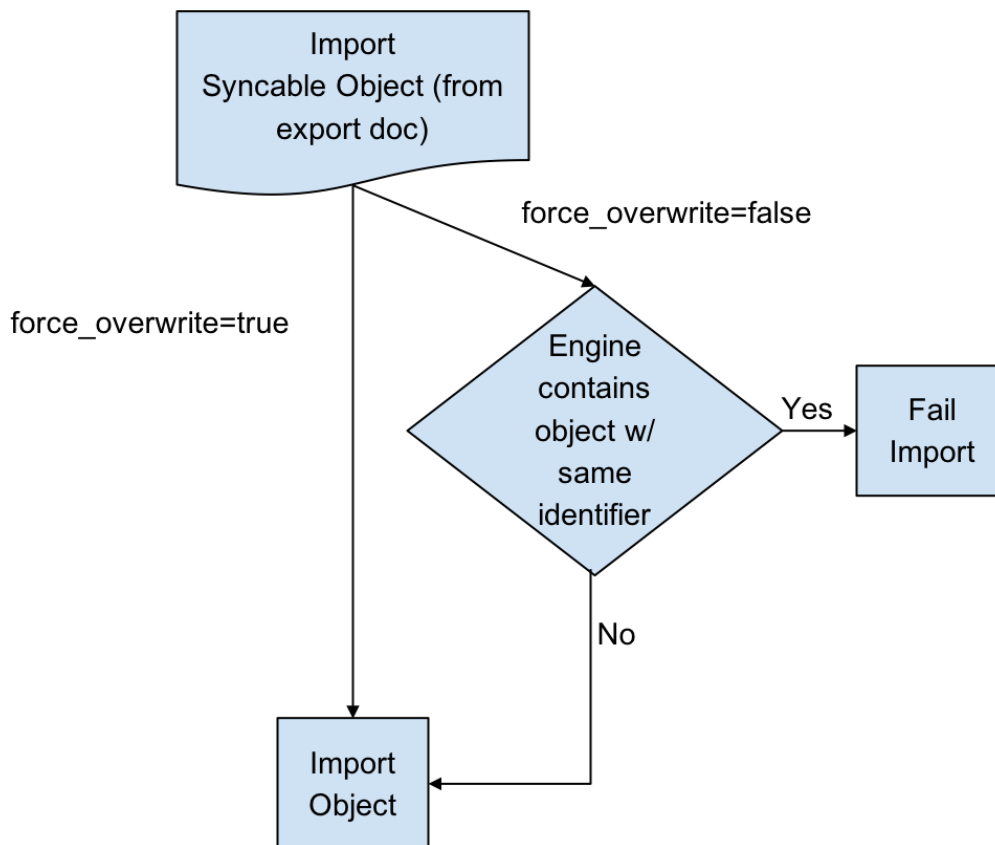
An export document containing a sensitive object should be encrypted before it is stored persistently. Users are free to apply an encryption mechanism of their choosing to an export document. As a convenience, you can request that the export document be encrypted by the Masking Engine using a passphrase. The Masking Engine will encrypt the export document with 3DES using SHA1 (PBESWithSHA1AndDESede). Once the document is encrypted with the passphrase, the engine forgets the passphrase. You will need to provide the same passphrase during import to decrypt the document.

10.2.7 Digital signature

In order to detect accidental or malicious modification of the export document, each document is digitally signed. If the export document does not match its expected digital signature, a Masking Engine will not import the document.

10.2.8 Overwrite

When an object to be imported has the same name as a currently existing object, importing it will cause the other object to be changed. Since this might not be intended, we offer a flag called `force_overwrite`. If `force_overwrite` is set to false and doing the import will change an existing object on the masking engine, we fail the import. This workflow is shown below.



10.2.8.1 Attempting to import identical objects

The Masking Engine checks for the existence of the same object contents during the import of an object. If it is determined that the engine and the document being imported contain the same content, a result of SUCCESS will be returned without repeating the work of a full import. For example, importing an entire ruleset with hundreds of thousands of tables can be quite time consuming, and this should not be repeated if the same object already exists. If the object content matches and we skip the full import we note this in the application log.

Below is an example of a log statement when an identical database connector was imported:

```

2017-07-19 10:17:06,075 [http-nio-exec-4] INFO
c.d.s.marshalls.SyncableMarshaller - Skipping import process for
{
  "objectType": "DATABASE_CONNECTOR",
  "id": {
    "@type": "type.googleapis.com/IntegerIdentifier",
    "id": 1
  }
}, due to no discrepancy between the existing and importing object
  
```

Depending on the object type, some define an object by a String (name) and some by an Integer (object id). Objects that can have the same name in multiple environments, such as connectors, rulesets, and masking jobs, are exported based on a unique id associated with them. Global objects, which do not have overlapping

names, are exported and identified based on their names. Something to note here is that objects exported based on their ids will overwrite the object with the *same name* rather than the same id. This means that for all importing objects, we define the identity of an object to be based on the name in the same environment.

For example, if I export a database connector named `testConnector` with the following export object metadata:

```
{
  "objectIdentifier": {
    "id": 5
  },
  "objectType": "DATABASE_CONNECTOR",
  "revisionHash": "68eaffef400e426520a5fcbb683419db3be53317"
}
```

And then I import this object into some engine's environment with the following list of connectors:

id	connector name	more information
1	testConnector	...
5	otherConnector	...

testConnector of id 1 will be overwritten, instead of *otherConnector*.

10.2.8.2 Overwrite of the encryption key

The global encryption key is somewhat special in that it always exists. Specifying *force_overwrite=false* will always fail to import the encryption key unless the encryption key has been previously synchronized using *force_overwrite=true*.

Specifying *force_overwrite=true* will always overwrite the engine's encryption key with the contents of the encryption key in the export document.


10.2.9 Error handling

Export documents often have multiple objects to be imported at once. For example, when exporting a database ruleset, you will export both the database ruleset and the database connector since a ruleset depends on a connector.

The engine will import one object at a time, where the dependencies are imported first. If there is an error importing an object, the import process will abort and all objects that have successfully been imported during this request will get rolled back. For example, say you are importing objects A, B, and C. Import successfully imports A. During the import of B, the engine encounters an error. The import of A will roll back, and the import of C will never execute. This will leave the engine in a state identical to the one it was in prior to the failed import.

10.2.10 Concurrent sync operations

To prevent race conditions with concurrent imports and jobs running, we currently do not allow concurrent import operations. We also do not allow imports while masking jobs are running. It is best to do imports when a machine is not running jobs in order to guarantee that the final state of each of those operations is as expected. If they are done at the same time, the operations will fail with relevant error messages.

 Beginning in version 20.0.0.0, sync export operations are allowed to run concurrently with sync import operations. Beware that in this situation, the sync export document may contain a partial set of modifications made by the running import process. It is best to avoid running import and export operations simultaneously that cover the same configuration objects.

10.2.11 Global objects

GLOBAL_OBJECT is a syncable object type that is a collection of all syncable algorithms, ALGORITHM_PLUGIN(s), DOMAIN(s), JDBC_DRIVER(s), PROFILE_SET(s), PROFILE_EXPRESSION(s) and the (Global) KEY. This represents objects in the Masking Engine that are available across all environments, and are not a part of any specific environment. When a user requests to export GLOBAL_OBJECT, every syncable algorithm, profile set, profile expression and domain on the engine will be exported as the bundle. If a DOMAIN, PROFILE_SET, or PROFILE_EXPRESSION has a dependency on a non-syncable algorithm, such as Mapping, it will not be exported.

This separation was added because global objects 1) containing large lookup files are projected to be time-consuming and 2) are expected to be synchronized much less frequently than any masking job-related metadata. Examples on how to use the GLOBAL_OBJECT are available in the [Example User Workflow section](#). (see page 860)

10.2.11.1 Global KEY

Before the 6.0.15.0 release, some algorithms used the Global KEY as part of their configuration. These algorithms only produced the same results on different engines if the global KEY was synchronized. Since the 6.0.15.0 release, no algorithms include the Global KEY as part of their configuration.

10.2.12 Reference objects

As mentioned in the *Global Objects* section, we expect the users to synchronize global objects and masking jobs at different frequencies. To avoid any unnecessary export of large algorithms, any objects (MASKING_JOB, PROFILE_JOB, DATABASE_RULESET, FILE_FORMAT and FILE_RULESET) that have dependencies on algorithms will export just the references to the objects by default. This way we check whether the necessary dependency exists on the importing engine by comparing the references; if not, we fail the import execution with an appropriate message. Domains, profile sets, and profile expressions are the exception to this. Exporting any of these objects will also export the full algorithm.

10.2.13 On-the-fly masking jobs

By definition, On-The-Fly (OTF) masking jobs work with a source environment/connector and a target environment/connector, masking the data from the source connector into that of the target connector. With masking jobs, a target *environment_id* is always required to specify which environment to import the job and its target connector. In addition to the target *environment_id*, OTF masking jobs require the specification of a *source_environment_id* into which to import the source connector. The source connector is copied into the specified source environment (*source_environment_id*), and is represented by the SOURCE_DATABASE_CONNECTOR or

SOURCE_FILE_CONNECTOR for database and file masking jobs respectively in the export document. These source connectors are virtually identical to their DATABASE_CONNECTOR and FILE_CONNECTOR counterparts, but are represented differently in the OTF jobs to distinguish them from the target connector (i.e., DATABASE_CONNECTOR or FILE_CONNECTOR).

10.2.14 Circular dependencies

It is possible to have a set of objects that end up depending on each other. This would be the case if a PROFILE_SET depended on a PROFILE_EXPRESSION that depended on a DOMAIN that depended on a REDACTION algorithm that depended on the original PROFILE_SET. The masking application will detect such scenarios on export and refuse to export such configurations.

This can be worked around by creating a second PROFILE_SET that contains PROFILE_EXPRESSIONS that do not depend on a DOMAIN that depends on a REDACTION algorithm. Simply ensure that the regular expressions are the same in the newly created PROFILE_EXPRESSIONS and assign the REDACTION algorithm to the new PROFILE_SET instead. The REDACTION algorithm will function the same but the dependency loop will have been broken.

10.3 Sync endpoints

10.3.1 Export endpoints



When exporting masking objects, a single export cannot contain multiple objects with the same name (e.g., two connectors with the same name).

10.3.1.1 GET /syncable-objects

```
GET /syncable-objects[?object_type=<type>]
```

This endpoint lists all objects in an engine that are syncable and can be exported. Any object which can be exported can be imported into another engine. The endpoint takes an optional parameter to filter by a specific object type. Each object is listed with its revision_hash. Note that if a syncable object depends on a non-syncable object (e.g. DOMAIN using a mapping algorithm), it will say so in the “revisionHash” attribute, and will not be exportable.

Example CURL command using the object_type parameter to only retrieve the list of LOOKUP algorithm objects:

```
curl -X GET
--header 'Accept: application/json'
--header 'Authorization: 21c45f0e-82f4-4b04-9072-b49072986231'
'http://masking-engine.com/masking/api/syncable-objects?object_type=LOOKUP'
```

10.3.1.2 POST /export

This endpoint allows you to export one or more objects in batch fashion. The result of the export is an export document and a set of metadata that describes what was exported. You are expected to specify which objects to export by copying their object identifiers from the /syncable-objects endpoint.



- The Sync POST /export API will timeout after 3 minutes.
- To upload objects that takes more than 3 minutes of time in uploading, use the export-async API.

The endpoint has a single optional header, *passphrase*. If you provide the passphrase, the export document will be encrypted using it.

Example CURL command using the optional passphrase header:

```
curl -X POST
--header 'Content-Type: application/json'
--header 'Accept: application/json'
--header 'Authorization: 21c45f0e-82f4-4b04-9072-b49072986231'
--header 'passphrase: my example passphrase'
-d '[
{
"objectIdentifier": {"id": 1},
"objectType": "MASKING_JOB",
"revisionHash": "asdfjkl12jijfdsaklfj21ojsdk"
}
]'
```

10.3.1.3 POST /export-async

This endpoint does exactly the same thing as /export, but the execution is done asynchronously. The response returns an async task in the form of this:

```
{
  "asyncTaskId": 66,
  "operation": "EXPORT",
  "reference": "EXPORT-ZXhwb3J0X2RvY3VtZW50XzJjcm1EV09yLmpzb24=",
  "status": "RUNNING",
  "startTime": "2018-04-13T17:49:55.354+0000",
  "cancellable": false
}
```

Example CURL command:

```
curl -s -X POST
--header 'Content-Type: application/json'
--header 'Accept: application/json'
--header 'Authorization: 21c45f0e-82f4-4b04-9072-b49072986231'
-d "[
{
  "objectIdentifier": {"id": 1},
  "objectType": "MASKING_JOB",
  "revisionHash": "asdfjkl12jijfdsaklfj21ojasdk"
}
]"
"http://masking-engine.com/masking/api/export-async"
```

The *reference* is used to retrieve the export document of completed async export tasks from the /file-downloads endpoint. The downloaded file from this reference should look exactly the same as the response from /export.

Example CURL command:

```
curl -s -X GET
--header 'Accept: application/octet-stream'
--header 'Authorization: 21c45f0e-82f4-4b04-9072-b49072986231'
-o "<OUTPUT_FILE_PATH>" "http://masking-engine.com/masking/api/file-downloads/EXPORT-ZXhwb3J0X2RvY3VtZW50XzJjcm1EV09yLmpzb24="
```

10.3.1.3.1 Error handling

If an error occurs while exporting one or more elements in the export document, the entire export will abort.

10.3.2 Import endpoints



The **POST /import** and **POST /import-async** APIs support a document up to 512 MB in size.

10.3.2.1 POST /import

```
POST /import?force_overwrite=<true|false>[&environment_id=<id>]
[&source_environment_id=<id>]
```

This endpoint allows you to import a document exported from another engine. The response returns a list of objects that were imported and whether the import was successful.

The endpoint has one required parameter, *force_overwrite*, two optional parameters *environment_id* and *source_environment_id*, and an optional HTTP header, *passphrase*, which if provided, will cause the engine to attempt to decrypt the document using the specified passphrase. The required *force_overwrite* parameter dictates how to deal with conflicting objects. *environment_id* is necessary for all non-global objects that need to belong in an environment. *source_environment_id* is used for On-The-Fly masking jobs.

The endpoint has a single optional header, *passphrase*. If you provide the passphrase, the import document will be decrypted using it.



Containerized Masking does not support some objects which might be exported from a Virtual Machine Masking Engine. Containerized Masking will generate an error if an import bundle contains one of these objects. To successfully import environments that contain these objects on a containerized engine, the problem objects will have to be removed on the export source engine and re-exported.

Unsupported objects include:

- Connectors using the FTP connection method
- Connectors using Kerberos credentials for DB authentication
- Connectors using IBM's custom DB2 JDBC driver
- Connectors using OAUTH2 authentication
- Engine Setup based NFS/CIFS mounts

Example CURL command using the optional passphrase header:

```
curl -X POST
--header 'Content-Type: application/json'
--header 'Accept: application/json'
```

```

--header 'Authorization: 21c45f0e-82f4-4b04-9072-b49072986231'
--header 'passphrase: my example passphrase'
-d '{
  "exportResponseMetadata": {
    "exportHost": "masking-engine.com",
    "exportDate": "Mon Aug 13 16:29:30 UTC 2018",
    "exportedObjectList": [
      {
        "objectIdentifier": {
          "algorithmName": "lookup_alg"
        },
        "objectType": "LOOKUP",
        "revisionHash": "cf84d82c21f0e9d4105d37ae7979c0848486d861"
      },
      {
        "objectIdentifier": {
          "keyId": "global"
        },
        "objectType": "KEY",
        "revisionHash": "1d8e9bc552d3ca1dcd218f9e197ea3955ccc29be"
      }
    ]
  },
  "blob": "<OMITTED>",
  "signature": "<OMITTED>", \
  "publicKey": "<OMITTED>" \
}'
'http://masking-engine.com/masking/api/import?force_overwrite=true'

```

10.3.2.2 POST /import-async

```

POST /import-async?force_overwrite=<true|false>[&environment_id=<id>]
[&source_environment_id=<id>]

```

This endpoint does exactly the same thing as /import, but the execution is done asynchronously and the body is taken in as a file. The response returns an async task in the form of this:

```

{
  "asyncTaskId": 67,
  "operation": "IMPORT",
  "reference": "IMPORT-ZXhwb3J0X2RvY3VtZW50XzJjcm1EV09yLmpzb24=",
  "status": "RUNNING",
  "startTime": "2018-04-13T17:49:55.354+0000",
  "cancellable": false
}

```


The *reference* is used to retrieve the import status of completed async import tasks from the `/file-downloads` endpoint. The downloaded file from this reference should look exactly the same as the response from `/import`.

Example CURL command:

```
curl -s -X POST
--header 'Content-Type: multipart/form-data'
--header 'Accept: application/json'
--header 'Authorization: 21c45f0e-82f4-4b04-9072-b49072986231'
-F "file=@<DOWNLOADED_FILE_PATH>"
"http://masking-engine.com/masking/api/import-async?force_overwrite=true"
```

10.4 Algorithm syncability

10.4.1 Overview

Algorithms are fully syncable between Masking Engines. To obtain a list of syncable algorithms, use the `GET /syncable-objects` API with `object_type` set to `USER_ALGORITHM` :

```
GET https://host.example.com/masking/api/syncable-objects?object_type=USER_ALGORITHM
```



The semantics of syncing an instance of the Mapping Algorithm Framework depend on its configuration. See [Mapping Algorithm Sync](#)³⁴¹ for more information.



Syncing out of the box algorithm instances updates their individual keys but does not replace their existing associated files.

10.4.2 Non-deterministic Algorithms

Some algorithms are non-deterministic. While exporting these algorithms is supported, they will not produce identical masking results on different engines. The following table lists the built-in non-deterministic algorithms:

³⁴¹ <https://masking.delphix.com/docs/latest/mapping-algorithm-frameworks>

Algorithm API Name	Algorithm UI Name
DateShiftVariable	DATE SHIFT(VARIABLE)
SecureShuffle	SECURE SHUFFLE

10.4.3 Fixed Algorithms

Some algorithms produce constant (fixed) results. Although these algorithms can be exported, they do not need to be synchronized since they always produce the same results on different engines. The following table lists the built-in fixed algorithms:

Algorithm API Name	Algorithm UI Name
DateShiftFixed	DATE SHIFT(FIXED)
RepeatFirstDigit	ZIP+4

10.5 User workflow examples

This page provides some examples of some typical user workflows. More information on exactly how each endpoint works is available on the [Sync endpoints \(see page 854\)](#) section.

10.5.1 Syncing all global objects

The following steps can be used to sync all global objects from Masking Engine A to Masking Engine B. This will sync all algorithms and domains and should be done prior to syncing jobs or rulesets which might depend on them. For more information on the global object, see the [Sync concepts \(see page 844\)](#) section.

10.5.1.1 Source masking engine steps

10.5.1.1.1 1. Login

Login on the source Masking Engine to obtain an Authorization token value.

```
POST https://a.example.com/masking/api/login
HEADER
```

```
Content-Type: application/json
Accept: application/json
```

```
BODY
{
  "username": "user123",
  "password": "pw123"
}
```

CURL example:

```
curl -X POST --cacert /path/to/cert --header 'Content-Type: application/json' --
header 'Accept: application/json' -d '{ "username": "user123", "password": "pw123" }'
'https://a.example.com/masking/api/login'
```

Expected Result:

```
{
  "Authorization": "dc2cff8b-e20d-4e28-8b7e-5d7c4aad0e2a"
}
```

10.5.1.1.2 2. Get the identifier

Call `GET /syncable-objects` to obtain the GLOBAL_OBJECT's information.

```
GET https://a.example.com/masking/api/syncable-objects?object_type=GLOBAL_OBJECT
```

```
HEADER
Authorization: dc2cff8b-e20d-4e28-8b7e-5d7c4aad0e2a (value from the /login response)
Accept: application/json
```

CURL example:

```
curl -X GET --cacert /path/to/cert --header 'Accept: application/json' --header
'Authorization: dc2cff8b-e20d-4e28-8b7e-5d7c4aad0e2a' 'https://a.example.com/masking/
api/syncable-objects?object_type=GLOBAL_OBJECT'
```

Expected Result:

```
{
  "_pageInfo": {
    "numberOnPage": 1,
    "total": 1
  },
  "responseList": [
    {
```


```

    "objectIdentifier": {
      "id": "global"
    },
    "objectType": "GLOBAL_OBJECT",
    "revisionHash": "8d5236bb029c2176aa568b930786b63253e4f9e4"
  }
]
}

```

10.5.1.1.3 3. Export the object

Call `POST /export-async` to asynchronously export the GLOBAL_OBJECT and use the passphrase header to encrypt the export.

 The optional passphrase header cannot be specified using the interactive [API Client tool](#)³⁴². An example of how to specify this header using a cURL command is shown below.

```
POST https://a.example.com/masking/api/export-async
```

HEADER

```
Authorization: dc2cff8b-e20d-4e28-8b7e-5d7c4aad0e2a
Content-Type: application/json
Accept: application/json
passphrase: my example passphrase
```

BODY

```
[
  {
    "objectIdentifier": {
      "id": "global"
    },
    "objectType": "GLOBAL_OBJECT"
  }
]
```

```
curl -X POST --cacert /path/to/cert --header 'Content-Type: application/json' --
header 'Accept: application/json' --header 'Authorization: dc2cff8b-
e20d-4e28-8b7e-5d7c4aad0e2a' --header 'passphrase: my example passphrase' -d
' [{"objectIdentifier":{"id":"global"},"objectType":"GLOBAL_OBJECT"}]' 'https://
a.example.com/masking/api/export-async'
```

Expected Result:

³⁴² <https://delphixdocs.atlassian.net/continuous-compliance-10-0-0-0/docs/masking-api-client#api-client>

```
{
  "asyncTaskId": 2,
  "operation": "EXPORT",
  "reference": "EXPORT-ZXhwb3J0X2RvY3VtZW50Xzk0Wjlva3JDLmpzb24=",
  "status": "RUNNING",
  "startTime": "2018-06-15T20:36:35.483+0000",
  "cancellable": false
}
```

10.5.1.1.4 4. Download the export document

Use the reference above to download the export document via the /file-download endpoint.

```
GET https://a.example.com/masking/api/file-downloads/EXPORT-
ZXhwb3J0X2RvY3VtZW50Xzk0Wjlva3JDLmpzb24=
```

HEADER

```
Authorization: dc2cff8b-e20d-4e28-8b7e-5d7c4aad0e2a
Accept: application/octet-stream
```

CURL example:

```
curl -X GET --cacert /path/to/cert --header 'Accept: application/json' --header
'Authorization: dc2cff8b-e20d-4e28-8b7e-5d7c4aad0e2a' 'https://a.example.com/masking/
api/file-downloads/EXPORT-ZXhwb3J0X2RvY3VtZW50Xzk0Wjlva3JDLmpzb24='
```

Expected Result: An export document that will look like this.

```
{
  "exportResponseMetadata": {
    "exportHost": "a.example.com",
    "exportDate": "Fri Jun 15 20:16:20 UTC 2018",
    "requestedObjectList": [
      {
        "objectIdentifier": {
          "id": "global"
        },
        "objectType": "GLOBAL_OBJECT",
        "revisionHash": "579850b1c88baf74cee6bad61d81e2aa3dcc206c"
      }
    ],
    "exportedObjectList": [
      {
        "objectIdentifier": {
          "id": "DRIVING_LC"
        },
        "objectType": "DOMAIN",

```

```

    "revisionHash": "9ee90782488d14d369f9595dad7f593c961e785f"
  },
  {
    "objectIdentifier": {
      "algorithmName": "DrivingLicenseNoLookup"
    },
    "objectType": "LOOKUP",
    "revisionHash": "e08ac9bfd4ed9f64d486cb47cdc07deb30ccc20f"
  },
  ...
]
},
"blob":
"RAAAAaokZmZhNWIxNjktODMwMC00N2FLLWJjZmMtNjVhNDUzYWI3OTBjEhgyMDE4LTA2LTE1VDIwOjE2OjIw
LjY2MFogBSgBFwIAAAokZmZhNWIxNjktODMwMC00N2FLLWJjZmMtNjVhNDUzYWI3OTBjEu4DCi8IFBIrCiV0e
XBllmdvb2dsZWFWaXMuY29tL0ludGVnZXJJZGVudGlmawVvYegIIARivCA4SKwo1dHlwZS5nb29nbGVhcGlzLm
...",
"signature": "MCwCFAWgf/97wb+oYuSQizj8U12n7jpQAhQKGCa0J4U8XyDA0EhMUWkzZXHrpw==",
"publicKey": "MIHxMIGoBgcqhkJ00AQBMIgCAkEA/
KaCzo4Syrom78z3EQ5SbbB4sF7ey80etKII864WF64B81uRpH5t9jQTxeEu0ImbzRMqzVDZkVG9xD7nN1kuFw
IVAjYu3cw2nLq0uyY05rahJtk0bjjFAkBNhHGyepz0TukaScUUfbGpq.."
}

```

10.5.1.1.5 5. Cleanup

When the export document is no longer needed, use the /export-async endpoint to cleanup the exported documents.

```
DELETE https://a.example.com/masking/api/export-async
```

HEADER

```
Authorization: dc2cff8b-e20d-4e28-8b7e-5d7c4aad0e2a
```

```
Accept: application/json
```

CURL example:

```
curl -X DELETE --header 'Accept: application/json' --header 'Authorization: dc2cff8b-
e20d-4e28-8b7e-5d7c4aad0e2a' 'https://a.example.com/masking/api/export-async'
```

Expected Result: no content

10.5.1.2 Destination Masking Engine steps

10.5.1.2.1 1. Login

Login on the destination Masking Engine to obtain an Authorization token value (see example above).

10.5.1.2.2 2. Import the object

On Masking Engine B, use the import-async endpoint to import the document downloaded from engine A.

```
POST https://b.example.com/masking/api/import-async?force_overwrite=true
```

HEADER

```
Authorization: dc2cff8b-e20d-4e28-8b7e-5d7c4aad0e2a
Content-Type: multipart/form-data
Accept: application/json
passphrase: my example passphrase
```

CURL example:

```
curl -X POST --cacert /path/to/cert --header 'Content-Type: multipart/form-data' --header 'Accept: application/json' --header 'Authorization: dc2cff8b-e20d-4e28-8b7e-5d7c4aad0e2a' --header 'passphrase: my example passphrase' -F file=@export.json 'https://b.example.com/masking/api/import-async?force_overwrite=true'
```

Expected Result:

```
{
  "asyncTaskId": 1,
  "operation": "IMPORT",
  "reference": "IMPORT-aW1wb3J0X2RvY3VtZW50X2lZQVFKWEFsLmpzb24=",
  "status": "WAITING",
  "cancellable": false
}
```

10.5.1.2.3 3. Verify status

On Masking Engine B, call the /file-downloads endpoint using the reference from the returned Async Task response to retrieve the completed import status.

```
GET https://b.example.com/masking/api/file-downloads/IMPORT-aW1wb3J0X2RvY3VtZW50X2lZQVFKWEFsLmpzb24=
```

HEADER

```
Authorization: dc2cff8b-e20d-4e28-8b7e-5d7c4aad0e2a
Accept: application/octet-stream
```

CURL example:

```
curl -X GET --cacert /path/to/cert --header 'Accept: application/octet-stream' --
header 'Authorization: dc2cff8b-e20d-4e28-8b7e-5d7c4aad0e2a' 'https://b.example.com/
masking/api/file-downloads/IMPORT-aW1wb3J0X2RvY3VtZW50X2lZQVFKWEFsLmpzb24='
```

Expected Result:

An import status document that reports the success or failure of each object imported.

```
[
  {
    "objectIdentifier": {
      "id": 7
    },
    "importedObjectIdentifier": {
      "id": 7
    },
    "objectType": "PROFILE_EXPRESSION",
    "importStatus": "SUCCESS"
  },
  {
    "objectIdentifier": {
      "id": "CERTIFICATE_NO"
    },
    "importedObjectIdentifier": {
      "id": "CERTIFICATE_NO"
    },
    "objectType": "DOMAIN",
    "importStatus": "SUCCESS"
  },
  ...
]
```

10.5.1.2.4 4. Cleanup

Once the status is no longer needed, use the /import-async endpoint to cleanup the exported documents.

```
DELETE https://b.example.com/masking/api/import-async
```

HEADER

```
Authorization: dc2cff8b-e20d-4e28-8b7e-5d7c4aad0e2a
```

```
Accept: application/json
```

CURL example:

```
curl -X DELETE --cacert /path/to/cert --header 'Accept: application/json' --header
'Authorization: dc2cff8b-e20d-4e28-8b7e-5d7c4aad0e2a' 'https://b.example.com/masking/
api/import-async'
```



Expected Result: no content

10.5.2 Syncing a masking job

The following steps provide an example of how to export a Masking Job from Masking Engine A to Masking Engine B using the synchronous endpoints of /export and /import. This presumes that all of the global objects such as algorithms and domains that the masking job relies on have already been synced. This can also be done via the asynchronous endpoint with the same workflow as above.

10.5.2.1 1. Export the job

Before this step, the /login and /syncable-objects endpoints should have been called to obtain the authorization token and job identifier respectively. Then use the /export endpoint to obtain an export document with the desired MASKING_JOB. In this example, the optional passphrase is used to encrypt the export document.

 To sync a profile job, swap out the objectType for "PROFILE_JOB" and provide the id of the profile job to sync. Profile jobs are syncable starting in version 5.3.2.0.

```
POST http://a.example.com/masking/api/export

HEADER
Authorization: dc2cff8b-e20d-4e28-8b7e-5d7c4aad0e2a
Content-Type: application/json
Accept: application/json
passphrase: password to encrypt the export document

BODY
[
  {
    "objectIdentifier": {
      "id": 4
    },
    "objectType": "MASKING_JOB"
  }
]
```

CURL example:

```
curl -X POST --cacert /path/to/cert --header 'Content-Type: application/json' --header 'Accept: application/json' --header 'Authorization: dc2cff8b-e20d-4e28-8b7e-5d7c4aad0e2a' --header 'passphrase: my example passphrase' -d '[ { "objectIdentifier": { "id": 4 }, "objectType": "MASKING_JOB" } ]' 'https://a.example.com/masking/api/export'
```

Expected Result:

```

{
  "exportResponseMetadata": {
    "exportHost": "a.example.com",
    "exportDate": "Fri Jun 15 20:16:20 UTC 2018",
    "requestedObjectList": [
      {
        "objectIdentifier": {
          "id": 1
        },
        "objectType": "MASKING_JOB",
        "revisionHash": "579850b1c88baf74cee6bad61d81e2aa3dcc206c"
      }
    ],
    "exportedObjectList": [
      {
        "objectIdentifier": {
          "id": 1
        },
        "objectType": "DATABASE_RULESET",
        "revisionHash": "bf63b401129cbc84f90eeb708281e98121f5a829"
      },
      {
        "objectIdentifier": {
          "id": "FIRST_NAME"
        },
        "objectType": "DOMAIN_REFERENCE",
        "revisionHash": "e6a52079843afd2625f20237fd50f56254c7e630"
      },
      {
        "objectIdentifier": {
          "id": 1
        },
        "objectType": "MASKING_JOB",
        "revisionHash": "579850b1c88baf74cee6bad61d81e2aa3dcc206c"
      },
      {
        "objectIdentifier": {
          "id": 1
        },
        "objectType": "DATABASE_CONNECTOR",
        "revisionHash": "6455f39dfa354a54bdf4ef69d6511a6c2bb19db3"
      },
      {
        "objectIdentifier": {
          "algorithmName": "FirstNameLookup"
        },
        "objectType": "ALGORITHM_REFERENCE",
        "revisionHash": "13b0a51a7e3904f52526c442419c54b39033dca3"
      }
    ]
  }
}

```

```

},
"blob":
"RAAAAaokZmZhNWIxNjktODMwMC00N2FLLWJjZmMtNjVhNDUzYWl3OTBjEhgyMDE4LTA2LTE1VDIwOjE2OjIw
LjY2MFogBSgBFwIAAAokZmZhNWIxNjktODMwMC00N2FLLWJjZmMtNjVhNDUzYWl3OTBjEu4DCi8IFBIrCiV0e
XBllmdvb2dsZWFWaXMuY29tL0ludGVnZXJJZGVudGhmaWVyEgIIARivCA4SKwoIdHlwZS5nb29nbGVhcGlzLm
...",
"signature": "MCwCFAWGf/97wb+oYuSQizj8U12n7jpQAhQKGCa0J4U8XyDA0EhMUWkzZXHrpw==",
"publicKey": "MIHxMIGoBgcqhkJ00AQBMIgCAkEA/
KaCzo4Syrom78z3EQ5SbbB4sF7ey80etKII864WF64B81uRpH5t9jQTxeEu0ImbzRMqzVDZkVG9xD7nN1kuFw
IVAJYu3cw2nLqOuyY05rahJtk0bjjFAkBNhHGyepz0TukaScUUfbGpq.."
}

```



The requestedObjectList returns the list of objects you've requested in the export, and the exportedObjectList returns a list of all objects that were exported. This will include both the requested ones and their dependencies.

10.5.2.2 2. Import the job

On Masking Engine B, import the masking job. You will need to provide an environment for it to import into.

```
POST http://b.example.com/masking/api/import?force_overwrite=false&environment_id=1
```

HEADER

```

Authorization: dc2cff8b-e20d-4e28-8b7e-5d7c4aad0e2a
Content-Type: application/json
Accept: application/json
passphrase: password to encrypt the export document

```

PARAMETER

force_overwrite and environment_id. See the details in the Masking API Call Concepts section [for](#) more details .

BODY

(Whatever gets returned from export)

CURL example:

```

curl -X POST --cacert /path/to/cert --header 'Content-Type: application/json' --
header 'Accept: application/json' --header 'Authorization: dc2cff8b-
e20d-4e28-8b7e-5d7c4aad0e2a' --header 'passphrase: my example passphrase' -d @/path/
to/export.json 'https://b.example.com/masking/api/import?
force_overwrite=false&environment_id=1'

```

Expected Result:

```
[
  {
    "objectIdentifier": {
      "id": 3033
    },
    "importedObjectIdentifier": {
      "id": 1
    },
    "objectType": "DATABASE_CONNECTOR",
    "importStatus": "SUCCESS"
  },
  {
    "objectIdentifier": {
      "id": 5421
    },
    "importedObjectIdentifier": {
      "id": 1
    },
    "objectType": "DATABASE_RULESET",
    "importStatus": "SUCCESS"
  }
  ...
]
```

10.5.3 Syncing an environment

Syncing an environment differs from syncing other objects in that we don't sync any of the environment's metadata, only its dependencies (jobs, connectors and rulesets). You can think of syncing an environment as an easy way to sync a large group of objects in the environment, without having to sync them one at a time. As such, the environment's revisionHash is not important.

10.5.3.1 1. Export the environment

Before this step, the `/login` and `/syncable-objects` endpoints should have been called to obtain the authorization token and environment identifier respectively. Then use the `/export` endpoint to obtain an export document with the desired ENVIRONMENT. In this example, the optional passphrase is used to encrypt the export document.

```
POST http://a.example.com/masking/api/export
```

```
HEADER
```

```
Authorization: dc2cff8b-e20d-4e28-8b7e-5d7c4aad0e2a
```

```
Content-Type: application/json
```

```
Accept: application/json
```

```
passphrase: password to encrypt the export document
```

```
BODY
```

```
[
```

```
{
  "objectIdentifier": {
    "id": 3
  },
  "objectType": "ENVIRONMENT"
}
```

CURL example:

```
curl -X POST --cacert /path/to/cert --header 'Content-Type: application/json' --
header 'Accept: application/json' --header 'Authorization: dc2cff8b-
e20d-4e28-8b7e-5d7c4aad0e2a' --header 'passphrase: my example passphrase' -d
'[{ "objectIdentifier": { "id": 3 }, "objectType": "ENVIRONMENT" } ]' 'https://
a.example.com/masking/api/export'
```

Expected Result:


```
{
  "exportResponseMetadata": {
    "exportHost": "a.example.com",
    "exportDate": "Tue Apr 21 21:57:32 UTC 2020",
    "requestedObjectList": [
      {
        "objectIdentifier": {
          "id": 3
        },
        "objectType": "ENVIRONMENT",
        "revisionHash": "c2f2f4bd8a043c32d0977cff8f915d64f1aaf518"
      }
    ],
    "exportedObjectList": [
      {
        "objectIdentifier": {
          "id": 4
        },
        "objectType": "DATASET_CONNECTOR",
        "revisionHash": "db7bc78d098f3df47199fc00c2ba83dee5a52a34"
      },
      {
        "objectIdentifier": {
          "id": 3
        },
        "objectType": "ENVIRONMENT",
        "revisionHash": "c2f2f4bd8a043c32d0977cff8f915d64f1aaf518"
      },
      {
        "objectIdentifier": {
          "id": 4
        },
        "objectType": "MASKING_JOB",

```

```


    "revisionHash": "2497260ee897303fc317b9268486c5e36663dad0"
  },
  {
    "objectIdentifier": {
      "id": 4
    },
    "objectType": "DATASET_RULESET",
    "revisionHash": "cb864b0f3f208c4ea5273389055d335d8d57028c"
  },
  {
    "objectIdentifier": {
      "id": 1
    },
    "objectType": "DATASET_FORMAT",
    "revisionHash": "0513a494c736d7f8993dee4720f200c0aa3bd749"
  }
]
},
"blob": "RAAAAAokZDg5Zjg5NWQtYzJjMi00ZjkyLWlXNjEtMTA0NDRjZDk5YWlxEhgyMDI...",
"signature": "MCwCFF9wqsdqMG/x7q+knwd4LLhwc4h+AhR9YF5rQZyp5YLQf8e7rI39kjkyUQ==",
"publicKey": "MIHwMIGoBgcqhkJ00AQBMIgCAkEA/KaCzo4Syrom78z3EQ5SbbB4sF7ey8..."
}

```

-  The requestedObjectList returns the list of objects you've requested in the export, and the exportedObjectList returns a list of all objects that were exported. This will include both the requested ones and their dependencies.

10.5.3.2 2. Create a new environment on the target engine

Since we do not import the environment metadata (such as name or type) we must first create an environment on the target which we wish to import our data into. At this step we would also need to create a source environment if we are importing any On-The-Fly jobs.

-  All source connectors will end up being imported into the since source environment that we specify. If you wish for these to be in separate environments, they must then be manually managed after import.

- The source engine's global object must be synced to the target engine before syncing the environment. If the global object is out of sync between the engines, an error will occur. For example, suppose the exported environment contains an inventory that uses a domain that is not in sync with the target engine, an error will be generated.

10.5.3.3 3. Import the environment into the newly created environment

On Masking Engine B, import the environment. You will need to provide an environment for it to import into.

```
POST http://b.example.com/masking/api/import?force_overwrite=false&environment_id=1
```

HEADER

```
Authorization: dc2cff8b-e20d-4e28-8b7e-5d7c4aad0e2a
Content-Type: application/json
Accept: application/json
passphrase: password to encrypt the export document
```

PARAMETER

force_overwrite and environment_id. See the details in the Masking API Call Concepts section [for](#) more details .

BODY

(Whatever gets returned from export)

CURL example:

```
curl -X POST --cacert /path/to/cert --header 'Content-Type: application/json' --header 'Accept: application/json' --header 'Authorization: dc2cff8b-e20d-4e28-8b7e-5d7c4aad0e2a' --header 'passphrase: my example passphrase' -d @/path/to/export.json 'https://b.example.com/masking/api/import?force_overwrite=false&environment_id=1'
```

Expected Result:

```
[
  {
    "objectIdentifier": {
      "id": 4
    },
    "importedObjectIdentifier": {
      "id": 5
    },
    "objectType": "DATASET_CONNECTOR",
    "importStatus": "SUCCESS"
  }
]
```

```
},
{
  "objectIdentifier": {
    "id": 3
  },
  "importedObjectIdentifier": {
    "id": 1
  },
  "objectType": "ENVIRONMENT",
  "importStatus": "SUCCESS"
},
{
  "objectIdentifier": {
    "id": 1
  },
  "importedObjectIdentifier": {
    "id": 1
  },
  "objectType": "DATASET_FORMAT",
  "importStatus": "SUCCESS"
},
{
  "objectIdentifier": {
    "id": 4
  },
  "importedObjectIdentifier": {
    "id": 5
  },
  "objectType": "DATASET_RULESET",
  "importStatus": "SUCCESS"
},
{
  "objectIdentifier": {
    "id": 4
  },
  "importedObjectIdentifier": {
    "id": 5
  },
  "objectType": "MASKING_JOB",
  "importStatus": "SUCCESS"
}
]
```


10.6 Change log

10.6.1 Changes in 6.0

10.6.1.1 New syncable objects

We added the following new syncable objects in 6.0. Refer to the main documentation for more information on what they are, and how to use them.

- 6.0.0.0 Release
 - MOUNT_INFORMATION
- 6.0.1.0 Release
 - JDBC_DRIVER
 - REIDENTIFICATION_JOB
 - TOKENIZATION_JOB
- 6.0.2.0 Release
 - DATASET_CONNECTOR
 - DATASET_FORMAT
 - DATASET_RULESET
 - ENVIRONMENT
- 6.0.3.0 Release
 - ALGORITHM_PLUGIN
 - USER_ALGORITHM

10.6.2 Changes in 5.3

10.6.2.1 New syncable objects

We added the following new syncable objects in 5.3. Refer to the main documentation for more information on what they are, and how to use them.

- 5.3.0.0 Release
 - DATABASE_RULESET
 - DATE_SHIFT
 - DOMAIN
 - FILE_CONNECTOR
 - FILE_FORMAT
 - FILE_RULESET

- GLOBAL_OBJECT
- MASKING_JOB
- 5.3.3.0 Release
 - PROFILE_EXPRESSION
 - PROFILE_JOB
 - PROFILE_SET

We also added the following new syncable algorithms in 5.3.

- 5.3.2.0 Release
 - CLEANSING
 - MIN_MAX
- 5.3.3.0 Release
 - REDACTION

10.6.2.2 Key per algorithm

In pre-5.3, a global key for the engine was used by all algorithms that required a seed to determine the outcome of masked values. This included algorithms such as Lookup and Binary Lookup. Thus, in 5.2, exporting a Lookup Algorithm would automatically export the global encryption key as a dependency. In this release, we allow each algorithm to have its own independent key, exported as a part of the algorithm. Refer to the [Key Management](#)³⁴³ section for more detail.

10.6.2.3 Changed model of import status reporting

In 5.2, the import status looked like this: some browsers enable drag-n-drop only when dataTransfer has data

```
{
  "objectIdentifier": {
    "keyId": "global"
  },
  "objectType": "KEY",
  "importStatus": "SUCCESS"
}
```

Starting in 5.3.0, the import status of an object has extended to include the id or name it has imported into to reduce any confusion introduced with IntegerIdentifiers. For more information on the reason for this change, refer to Logic Behind Overwrite of IntegerIdentifier and StringIdentifier. For examples on what it now looks like, refer to the [Example User Workflow](#) (see page 860) section.

³⁴³ <https://delphixdocs.atlassian.net/wiki/spaces/CC/pages/9930099/Key+management>

10.6.2.4 Changed granularity of transactions for import

Starting in 5.3, an import of however many objects is performed as an atomic execution rather than using per-object atomicity. This means that the execution will either succeed at importing all objects or fail and import none at all. Refer to the Error Handling of Import logic flow diagram for more information.

10.6.2.5 Filter for /syncable-objects

Now that we have a large list of syncable objects, we have added a new feature for filtering based on the object type. Refer to the [Endpoints \(see page 854\)](#) page and the [Example User Workflow \(see page 860\)](#) section for more information.

10.6.2.6 Async endpoints

Exporting a large MASKING_JOB with many dependencies can potentially take a long time. So we have decided to provide a new endpoint that exports and imports the objects asynchronously. Refer to the [Endpoints \(see page 854\)](#) section in the main documentation and the [Example User Workflow \(see page 860\)](#) page for more information.

11 Delphix masking APIs

This section covers the following topics:

- [API client \(see page 878\)](#)
- [API workflows \(see page 892\)](#)
- [API examples \(see page 978\)](#)

11.1 API client

This section covers the following topics:

- [Continuous Compliance API client \(see page 878\)](#)
- [Backwards compatibility API usage \(see page 890\)](#)
- [API response escaping \(see page 892\)](#)

11.1.1 Continuous Compliance API client

11.1.1.1 Introduction

The launch of API v5 on the Delphix Continuous Compliance Engine marks a significant advancement, enabling scripting and automation capabilities. This development offers exciting new possibilities for Continuous Compliance users. This page provides a high-level overview of API v5's features and includes some useful links to help you get started.

11.1.1.1.1 REST

API v5 is a RESTful API. REST stands for REpresentational State Transfer. A REST API will allow you to access and manipulate a textual representation of objects and resources using a predefined set of operations to accomplish various tasks.

11.1.1.1.2 JSON

API v5 uses JSON (JavaScript Object Notation) to ingest and return representations of the various objects used throughout various operations. JSON is a standard format and, as such, has many tools available to help with creating and parsing the request and response payloads, respectively.

Here are some UNIX tools that can be used to parse JSON – [Parsing JSON with UNIX tools](#)³⁴⁴. That being said, this is only the tip of the iceberg when it comes to JSON parsing and the reader is encouraged to use their method of choice.

³⁴⁴ <https://stackoverflow.com/questions/1955505/parsing-json-with-unix-tools>

11.1.1.1.3 API client

The various operations and objects used to interact with API v5 are defined in a specification document. This allows us to utilize various tooling to ingest that specification to generate documentation and an API Client, which can be used to generate cURL commands for all operations. To see how to log into the API client and for some starter recipes, please check out API Cookbook document.

To access the API client on your Continuous Compliance Engine, go to `http://myEngine.myDomain.com/masking/api-client`.

To access the API client documentation without an engine, please refer to the static HTML representations here:

[Masking API 5.1.10 Documentation \(Version 6.0.10.0\).html](#)³⁴⁵

[Masking API 5.1.11 Documentation \(Version 6.0.11.0\).html](#)³⁴⁶

[Masking API 5.1.12 Documentation \(Version 6.0.12.0\).html](#)³⁴⁷

[Masking API 5.1.13 Documentation \(Version 6.0.13.0\).html](#)³⁴⁸

[Masking API 5.1.14 Documentation \(Version 6.0.14.0\).html](#)³⁴⁹

[Masking API 5.1.15 Documentation \(Version 6.0.15.0\).html](#)³⁵⁰

[Masking API 5.1.16 Documentation \(Version 6.0.16.0\).html](#)³⁵¹

[Masking API 5.1.17 Documentation \(Version 6.0.17.0\).html](#)³⁵²

[Masking API 5.1.18 Documentation \(Version 7.0.0.0\).html](#)³⁵³

³⁴⁵ [https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.10%20Documentation%20\(Versions%206.0.10.0\).html?api=v2&cacheVersion=1&modificationDate=1724058614467&version=1](https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.10%20Documentation%20(Versions%206.0.10.0).html?api=v2&cacheVersion=1&modificationDate=1724058614467&version=1)

³⁴⁶ [https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.11%20Documentation%20\(Versions%206.0.11.0\).html?api=v2&cacheVersion=1&modificationDate=1724058614939&version=1](https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.11%20Documentation%20(Versions%206.0.11.0).html?api=v2&cacheVersion=1&modificationDate=1724058614939&version=1)

³⁴⁷ [https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.12%20Documentation%20\(Versions%206.0.12.0\).html?api=v2&cacheVersion=1&modificationDate=1724058615313&version=1](https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.12%20Documentation%20(Versions%206.0.12.0).html?api=v2&cacheVersion=1&modificationDate=1724058615313&version=1)

³⁴⁸ [https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.13%20Documentation%20\(Versions%206.0.13.0\).html?api=v2&cacheVersion=1&modificationDate=1724058615832&version=1](https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.13%20Documentation%20(Versions%206.0.13.0).html?api=v2&cacheVersion=1&modificationDate=1724058615832&version=1)

³⁴⁹ [https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.14%20Documentation%20\(Versions%206.0.14.0\).html?api=v2&cacheVersion=1&modificationDate=1724058616350&version=1](https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.14%20Documentation%20(Versions%206.0.14.0).html?api=v2&cacheVersion=1&modificationDate=1724058616350&version=1)

³⁵⁰ [https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.15%20Documentation%20\(Versions%206.0.15.0\).html?api=v2&cacheVersion=1&modificationDate=1724058616877&version=1](https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.15%20Documentation%20(Versions%206.0.15.0).html?api=v2&cacheVersion=1&modificationDate=1724058616877&version=1)

³⁵¹ [https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.16%20Documentation%20\(Versions%206.0.16.0\).html?api=v2&cacheVersion=1&modificationDate=1724058617381&version=1](https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.16%20Documentation%20(Versions%206.0.16.0).html?api=v2&cacheVersion=1&modificationDate=1724058617381&version=1)

³⁵² [https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.17%20Documentation%20\(Versions%206.0.17.0\).html?api=v2&cacheVersion=1&modificationDate=1724058617766&version=1](https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.17%20Documentation%20(Versions%206.0.17.0).html?api=v2&cacheVersion=1&modificationDate=1724058617766&version=1)

³⁵³ [https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.18%20Documentation%20\(Versions%207.0.0.0\).html?api=v2&cacheVersion=1&modificationDate=1724058618181&version=1](https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.18%20Documentation%20(Versions%207.0.0.0).html?api=v2&cacheVersion=1&modificationDate=1724058618181&version=1)

- [Masking API 5.1.19 Documentation \(Version 8.0.0.0\).html](#)³⁵⁴
- [Masking API 5.1.20 Documentation \(Version 9.0.0.0\).html](#)³⁵⁵
- [Masking API 5.1.21 Documentation \(Version 10.0.0.0\).html](#)³⁵⁶
- [Masking API 5.1.22 Documentation \(Version 11.0.0.0\).html](#)³⁵⁷
- [Masking API 5.1.23 Documentation \(Version 12.0.0.0\).html](#)³⁵⁸
- [Masking API 5.1.24 Documentation \(Version 13.0.0.0\).html](#)³⁵⁹
- [Masking API 5.1.25 Documentation \(Version 14.0.0.0\).html](#)³⁶⁰
- [Masking API 5.1.26 Documentation \(Version 15.0.0.0\).html](#)³⁶¹
- [Masking API 5.1.27 Documentation \(Version 16.0.0.0\).html](#)³⁶²
- [Masking API 5.1.27 Documentation \(Version 16.0.0.0\).html](#)³⁶³
- [Masking API 5.1.28 Documentation \(Version 17.0.0.0\).html](#)³⁶⁴
- [Masking API 5.1.29 Documentation \(Version 18.0.0.0\).html](#)³⁶⁵

-
- 354 [https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.19%20Documentation%20\(Versions%208.0.0.0\).html?api=v2&cacheVersion=1&modificationDate=1724058618581&version=1](https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.19%20Documentation%20(Versions%208.0.0.0).html?api=v2&cacheVersion=1&modificationDate=1724058618581&version=1)
- 355 [https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.20%20Documentation%20\(Versions%209.0.0.0\).html?api=v2&cacheVersion=1&modificationDate=1724058619000&version=1](https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.20%20Documentation%20(Versions%209.0.0.0).html?api=v2&cacheVersion=1&modificationDate=1724058619000&version=1)
- 356 [https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.21%20Documentation%20\(Versions%2010.0.0.0\).html?api=v2&cacheVersion=1&modificationDate=1724058619586&version=1](https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.21%20Documentation%20(Versions%2010.0.0.0).html?api=v2&cacheVersion=1&modificationDate=1724058619586&version=1)
- 357 [https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.22%20Documentation%20\(Versions%2011.0.0.0\).html?api=v2&cacheVersion=1&modificationDate=1724058620079&version=1](https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.22%20Documentation%20(Versions%2011.0.0.0).html?api=v2&cacheVersion=1&modificationDate=1724058620079&version=1)
- 358 [https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.23%20Documentation%20\(Versions%2012.0.0.0\).html?api=v2&cacheVersion=1&modificationDate=1724058620414&version=1](https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.23%20Documentation%20(Versions%2012.0.0.0).html?api=v2&cacheVersion=1&modificationDate=1724058620414&version=1)
- 359 [https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.24%20Documentation%20\(Versions%2013.0.0.0\).html?api=v2&cacheVersion=1&modificationDate=1724058620839&version=1](https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.24%20Documentation%20(Versions%2013.0.0.0).html?api=v2&cacheVersion=1&modificationDate=1724058620839&version=1)
- 360 [https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.25%20Documentation%20\(Versions%2014.0.0.0\).html?api=v2&cacheVersion=1&modificationDate=1724058614007&version=1](https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.25%20Documentation%20(Versions%2014.0.0.0).html?api=v2&cacheVersion=1&modificationDate=1724058614007&version=1)
- 361 [https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.26%20Documentation%20\(Versions%2015.0.0.0\).html?api=v2&cacheVersion=1&modificationDate=1724058613524&version=1](https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.26%20Documentation%20(Versions%2015.0.0.0).html?api=v2&cacheVersion=1&modificationDate=1724058613524&version=1)
- 362 [https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.27%20Documentation%20\(Versions%2016.0.0.0\).html?api=v2&cacheVersion=1&modificationDate=1724058613147&version=1](https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.27%20Documentation%20(Versions%2016.0.0.0).html?api=v2&cacheVersion=1&modificationDate=1724058613147&version=1)
- 363 [https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.27%20Documentation%20\(Versions%2016.0.0.0\).html?api=v2&cacheVersion=1&modificationDate=1724058613147&version=1](https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.27%20Documentation%20(Versions%2016.0.0.0).html?api=v2&cacheVersion=1&modificationDate=1724058613147&version=1)
- 364 [https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.28%20Documentation%20\(Versions%2017.0.0.0\).html?api=v2&cacheVersion=1&modificationDate=1724058612045&version=1](https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.28%20Documentation%20(Versions%2017.0.0.0).html?api=v2&cacheVersion=1&modificationDate=1724058612045&version=1)
- 365 [https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.29%20Documentation%20\(Versions%2018.0.0.0\).html?api=v2&cacheVersion=1&modificationDate=1724058610733&version=1](https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.29%20Documentation%20(Versions%2018.0.0.0).html?api=v2&cacheVersion=1&modificationDate=1724058610733&version=1)

- [Masking API 5.1.30 Documentation \(Version 19.0.0.0\).html](#)³⁶⁶
- [Masking API 5.1.31 Documentation \(Version 20.0.0.0\).html](#)³⁶⁷
- [Masking API 5.1.32 Documentation \(Version 21.0.0.0\).html](#)³⁶⁸
- [Masking API 5.1.33 Documentation \(Version 22.0.0.0\).html](#)³⁶⁹
- [Masking API 5.1.34 Documentation \(Version 23.0.0.0\).html](#)³⁷⁰
- [Masking API 5.1.35 Documentation \(Version 24.0.0.0\).html](#)³⁷¹
- [Masking API 5.1.36 Documentation \(Version 25.0.0.0\).html](#)³⁷²
- [Masking API 5.1.37 Documentation \(Version 26.0.0.0\).html](#)³⁷³

11.1.1.1.4 Supported features

API v5 is in active development but does not currently support all features that are accessible in the GUI. The list of supported features will expand over the course of subsequent releases.

For a full list of supported APIs, the best place to look is the API client on your Continuous Compliance Engine.


-
- ³⁶⁶ [https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.30%20Documentation%20\(Versions%2019.0.0.0\).html?api=v2&cacheVersion=1&modificationDate=1724058610347&version=1](https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.30%20Documentation%20(Versions%2019.0.0.0).html?api=v2&cacheVersion=1&modificationDate=1724058610347&version=1)
 - ³⁶⁷ [https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.31%20Documentation%20\(Versions%2020.0.0.0\).html?api=v2&cacheVersion=1&modificationDate=1724058609991&version=1](https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.31%20Documentation%20(Versions%2020.0.0.0).html?api=v2&cacheVersion=1&modificationDate=1724058609991&version=1)
 - ³⁶⁸ [https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.32%20Documentation%20\(Versions%2021.0.0.0\).html?api=v2&cacheVersion=1&modificationDate=1724058609411&version=1](https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.32%20Documentation%20(Versions%2021.0.0.0).html?api=v2&cacheVersion=1&modificationDate=1724058609411&version=1)
 - ³⁶⁹ [https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.33%20Documentation%20\(Versions%2022.0.0.0\).html?api=v2&cacheVersion=1&modificationDate=1724058609037&version=1](https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.33%20Documentation%20(Versions%2022.0.0.0).html?api=v2&cacheVersion=1&modificationDate=1724058609037&version=1)
 - ³⁷⁰ [https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.34%20Documentation%20\(Versions%2023.0.0.0\).html?api=v2&cacheVersion=1&modificationDate=1724058608601&version=1](https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.34%20Documentation%20(Versions%2023.0.0.0).html?api=v2&cacheVersion=1&modificationDate=1724058608601&version=1)
 - ³⁷¹ [https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.35%20Documentation%20\(Versions%2024.0.0.0\).html?api=v2&cacheVersion=1&modificationDate=1724058608389&version=1](https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.35%20Documentation%20(Versions%2024.0.0.0).html?api=v2&cacheVersion=1&modificationDate=1724058608389&version=1)
 - ³⁷² [https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.36%20Documentation%20\(Versions%2025.0.0.0\).html?api=v2&cacheVersion=1&modificationDate=1724058608164&version=1](https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.36%20Documentation%20(Versions%2025.0.0.0).html?api=v2&cacheVersion=1&modificationDate=1724058608164&version=1)
 - ³⁷³ [https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.37%20Documentation%20\(Versions%2026.0.0.0\).html?api=v2&cacheVersion=1&modificationDate=1724087769579&version=1](https://delphixdocs.atlassian.net/wiki/download/attachments/205423750/Masking%20API%205.1.37%20Documentation%20(Versions%2026.0.0.0).html?api=v2&cacheVersion=1&modificationDate=1724087769579&version=1)

11.1.1.2 Application settings APIs

11.1.1.2.1 General group settings

Setting Group	Setting Name	Type	Description	Default Value
general	EnableMonitorRowCount	Boolean	Controls whether a job displays the total number of rows that are being masked. Setting this to false reduces the startup time of all jobs.	true
	PasswordTimeSpan	Integer [0, ∞)	The number of hours a user is locked out for before they can attempt to log in again.	23
	PasswordCount	Integer [0, ∞)	The number of incorrect password attempts before a user is locked out.	3
	AllowPasswordResetRequest	Boolean	When true, users can request a password reset link be sent to the email associated with their account.	true
	PasswordResetLinkDuration	Integer [1, ∞)	Controls how many minutes the password reset link is valid for.	5
	NumSimulJobsAllowed	Integer [0, ∞)	Max number of jobs allowed to run simultaneously. Setting this number to zero will lead to a dynamically calculated limit (see page 485) based on the number of available CPU cores.	7
	DefaultApiVersion	String	Used to set default API Version. If the version is omitted from the base path of the request's URL, but wishes to be treated using a specific masking API version that is not the latest version, set the DefaultApiVersion application setting. If the DefaultApiVersion is not set and the version is omitted from the URL, the latest version of the API on that engine will be used. Sample API Version format is like v5.1.5 etc.	Blank

Setting Group	Setting Name	Type	Description	Default Value
	DataRetentionInterval	Integer [-1, ∞)	The length of time that specific historical data is retained. This setting value is in integer days. Certain log files and internal processing data are retained in case problem diagnosis is needed. Since we cannot keep this data indefinitely, this setting is the length of time that old data is retained. Data older than this will be purged on a periodic basis. Special Values -1 : disable this pruning method 0 : each cleanup removes all files	60
	DataRetentionMaxDirectorySize	Integer [-1, 100]	The percentage of disk space allowed for all logfiles located in specific directories. This setting value is in integer percent. For log files written to disk, the DataRetentionInterval setting (above) ensures that we keep these job log files for only a specific period of time. This setting avoids problems where significant activity in a short time might overwhelm available disk space. This setting is a backstop to the DataRetentionInterval setting which is intended to be the primary driver for managing retention. Special Values -1 : disable this pruning method 0 : each cleanup removes all files	10
	LoginScreenMessage	String	To configure a security/branding banner message on the Engine. All users will see the banner message on the Login screen.	
	UserSessionTimeoutMinutes	Integer [5, 1440]	This is used to configure the session's idle timeout in minutes. The user will be logged out after the idle timeout.	60

 NumSimulJobsAllowed setting should be set based on engine configuration. It is risky to run many jobs at once in an environment where you have scheduled more jobs than the system has memory for. When the system runs out of memory all jobs will fail.

11.1.1.2.2 Algorithm group settings

Setting Group	Setting Name	Type	Description	Default Value
algorithm	DefaultNonConformantDataHandling	String {DONT_MASK, FAIL}	Default algorithm behavior for Handling of Non-conformant Data patterns.	DONT_MASK

11.1.1.2.3 Database group settings

Setting Group	Setting Name	Type	Description	Default Value
database	DB2zDateFormat	String	Default Date String format to use for DB2 zOS if the database is not using one of the pre-defined IBM DB2 zOS Date String formats ³⁷⁴ . Default is ISO Date String format.	yyyy-MM-dd

11.1.1.2.4 LDAP group settings

Setting Group	Setting Name	Type	Description	Default Value
ldap	Enable	Boolean	Used to enable and disable LDAP authentication	false
	LdapHost	String	Host of LDAP server	10.10.10.31
	LdapPort	Integer [0, ∞)	Port of LDAP server	389
	LdapBasedn	String	Base DN of LDAP server	DC=tbspune,DC=com

³⁷⁴ https://www.ibm.com/support/knowledgecenter/en/SSEPEK_11.0.0/sqlref/src/tpc/db2z_datetimestringrepresentation.html

Setting Group	Setting Name	Type	Description	Default Value
	LdapFilter	String	Filter for LDAP authentication	(&(objectClass=person) (sAMAccountName=?))
	MsadDomain	String	MSAD Domain for LDAP authentication	AD
	LdapTlsEnable	Boolean	Enable and disable the use of TLS for LDAP connections.	false



In the LDAP group, once the "Enable" setting is set to "true", all users logging in will be authenticated via the LDAP server. Local authentication will no longer work. Before setting this to true set all other LDAP settings correctly and create the necessary LDAP users on the Continuous Compliance Engine.

11.1.1.2.5 Mask group settings

Setting Group	Setting Name	Type	Description	Default Value
mask	DatabaseCommitSize	Integer [1, ∞)	Controls how many rows are updated (Batch Update) to the database before the transaction is committed.	10000
	DefaultStreams	Integer [1, ∞)	Default number of streams for a masking job.	1
	DefaultUpdateThreads	Integer [1, ∞)	Default number of database update threads for a masking job.	1
	DefaultMaxMemory	Integer [1024, ∞)	Default maximum memory for masking jobs (in megabytes).	1024

Setting Group	Setting Name	Type	Description	Default Value
	DefaultMin Memory	Integer [1024, ∞)	Default minimum memory for masking jobs (in megabytes).	1024
	CharStreamingBuffer LimitRate	Integer [1, 50]	Used for calculating maximum allowed buffer size for Character streaming parsers to buffer data. Only used in JSON file and Document store type masking.	25

11.1.1.2.6 Profile group settings

These settings apply only to the legacy profiler, not the ASDD profiler, unless specifically noted in the setting description.

Setting Group	Setting Name	Type	Description	Default Value
profile	EnableDataLevelCount	Boolean	<p>When enabled (true), the Continuous Compliance Engine counts the number of rows in the profiled table. If the number of rows are less than or equal to DataLevelRows, then it uses the number of rows as the sample size. Otherwise, it uses DataLevelRows.</p> <p>When disabled (false), the Continuous Compliance Engine uses DataLevelRows.</p>	false
	DataLevelRows	Integer [1, ∞)	<p>The number of rows a data level profiling job samples when profiling a column. The DataLevelRows will only take into account if</p> <ul style="list-style-type: none"> • EnableDataLevelCount is false. • EnableDataLevelCount is true and number of rows is greater than DataLevelRows. 	100
	DataLevelPercentage	Double (0, ∞)	Percentage of rows that must match the data level regex to consider this column a match, and thus sensitive.	80.0

Setting Group	Setting Name	Type	Description	Default Value
	IgnoreDatatype	String	Datatypes that a profiling job should ignore. Columns of these types will not be assigned a domain/algorithm pair.	BIT,BOOLEAN,CHAR,CHAR#,VARCHAR,VARCHAR#,NCCHAR,NCCHAR#,NVARCHAR,NVARCHAR#,NVARCHAR2,NCLOB,NCLOB,NCLOB,NCLOB,NCLOB,BFILE,RAW,ENUM,BFILE
	DefaultStreams	Integer [1, ∞)	Default number of streams for a profiling job.	1
	DefaultMaxMemory	Integer [1024, ∞)	Default maximum memory for profiling jobs (in megabytes).	1024
	DefaultMinMemory	Integer [1024, ∞)	Default minimum memory for profiling jobs (in megabytes).	1024
	Optimization Level	Integer [0, 9)	Optimization level for the profiling job which is defined as below, 0: No optimizations are performed. 1: JavaScript runs in interpreted mode. 9: Performs the most optimization with faster script execution, but compiles slower. 1-9: All optimizations are performed.	-1
	DefaultMultipleProfilerExpressionAlgorithm	String	Default Multiple PHI masking algorithm which will be used when the Multiple Profiler Expression will be true for profile job. This value is used by both the legacy and ASDD profilers.	NullValue Lookup

11.1.1.2.7 ASDD group settings

Setting Group	Setting Name	Type	Description	Default Value
ASDD	DefaultMaximumColumnSize	Integer [1, 65536]	The maximum length of the column value to profile. Any value beyond that is truncated to this length for ASDD profiling and the remaining part is ignored.	1024
	DefaultTableSampleRows	Integer [1, ∞)	The number of database rows for the ASDD profiler to sample for each table.	1000
	DefaultAssignmentThreshold	Integer [1, 100]	The confidence threshold that must be met or exceeded for the ASDD profiler to make a domain and algorithm assignment. Assignment threshold can otherwise be set on a profile set level; see Configuring profile sets (see page 0) .	1
	DefaultJobExecutionStreams	Integer [1, ∞)	The number of streams to use by default for new ASDD profiler jobs	1

Setting Group	Setting Name	Type	Description	Default Value
	DefaultNullFilterThreshold	Integer [0, 100]	The percentage of column values that must be null or empty to trigger an additional query to retrieve more column values.	75
	DefaultProfileSetName	String	The name of the default ASDD profile set that should be selected automatically while creating a Profile Job.	ASDD Standard

11.1.1.2.8 Job group settings

Setting Group	Setting Name	Type	Description	Default Value
job	JobLoggingLevel	String {Basic, Detailed}	Controls the amount of information being logged from a job's output. Warning: the Detailed setting may log sensitive information when errors occur. Although this information can be very valuable when debugging a problem, it should be used with care.	Basic

11.1.1.2.9 SMTP group settings

Setting Group	Setting Name	Type	Description	Default Value
smtp	EmailEnabled	Boolean	Used to enable and disable SMTP Email authentication	TRUE

Setting Group	Setting Name	Type	Description	Default Value
	Host	String	The address of the SMTP server used for sending emails	
	Password	String	The password required to authenticate with the SMTP server	
	EmailAddressFrom	String	The email address that will appear in the "From" field of sent emails	
	EmailSubject	String	The subject line of the email being sent	
	UserName	String	The username required to authenticate with the SMTP server	
	Port	Integer	The port number used by the SMTP server, typically 25, 465, or 587	25
	TlsEnabled	Boolean	Whether to use SSL/TLS encryption for secure email sending	FALSE
	AuthEnabled	Boolean	Indicates if authentication is needed to connect to the SMTP server	FALSE

11.1.2 Backwards compatibility API usage



In all examples, replace **<myMaskingEngine>** with the hostname or IP address of your virtual machine.

In all examples, replace **<myMaskingEngine>** with the hostname or IP address of your virtual machine.

11.1.2.1 API versioning context

The Masking API is versioned in accordance with the Semantic Versioning format: <http://semver.org/>. When the Masking API is updated, a new API version will be released. Scripts must reference an explicit API version or else there are no guarantees that the scripts will work on future releases of the Masking API.

11.1.2.2 Pinning down a version number to guarantee backwards-compatibility

'**http://<myMaskingEngine>/masking/api/v5.0.0/environments**'

This is the format for specifying a version in the URL of an API request targeting the **environments** endpoints.

Specifying the version for endpoint guarantees that the requester receives a response containing all of the fields that were present in that version of the API. This is intended to allow scripts that specify a masking API version in the URL to continue working upon future upgrades of the Masking product—even if a newer version of the API is available in the future Masking product.

For example, consider the scenario where a script is being developed today with a pinned down version **v5.0.0** in the URL of the API requests. Upon upgrade to a future release of the Masking product that has the API **v5.1.0** available, the same, untouched script that was developed with the pinned down version **v5.0.0** in the URL of the API requests are expected to continue working. That said, in order to leverage any new features of the API **v5.1.0**, the original script will need to be updated to specify the new API version in the URL, and the requests may need to be updated to conform to the new API specification.

While specifying a version for endpoint guarantees that all fields present in that version will be contained in the API response, it does **not** mean that new fields that have since been added to that endpoint in subsequent versions will be excluded. We, therefore, recommend that API users write their scripts to parse the JSON response objects by key name, rather than by key index, to prevent these additional fields from breaking any scripts.

11.1.2.3 Omitted version numbers

'**http://<myMaskingEngine>/masking/api/environments**'

This is the format for not specifying a version in the URL of an API request targeting the **environments** endpoints. When the API version number is omitted, the latest API version is taken as a default. In the first 5.2 release, an API request with an omitted version number will be interpreted as a request against the **v5.0.0** version of the API. In a future release that hypothetically has the API **v5.3.0** available, an API request with an omitted version number will be interpreted as a request against the **v5.3.0** version of the API.

Scripts that omit the version of the Masking API in the URL are not guaranteed to work upon future upgrades of the Masking product because the API specification may change between versions and requests that conform to the old API specification may not work on the new API specification.

11.1.2.3.1 DefaultApiVersion

If the version is omitted from the base path of the request's URL, but wishes to be treated using a specific masking API version that is not the latest version, set the `DefaultApiVersion` application setting. If the `DefaultApiVersion` is not set and the version is omitted from the URL, the latest version of the API on that engine will be used.



The `DefaultApiVersion` application setting will not be applied to any requests made from within the masking engine. This means that the UI, api-client, and phone home will always use the latest API version supported on the engine.

11.1.3 API response escaping

In Masking API responses, a backslash character (\) is escaped with an additional backslash character (\\). Special attention should be paid to this behavior in scenarios where an API response is passed to another system as an input, for example, an automation system.

In such cases, a response might need special handling to convert the double backslash sequence (\\) back to a single backslash (\).

For example, consider the `POST /ssh-key` API for creating/installing an SSH Key. The result when the `POST /ssh-key` API is called with a file name that contains \ , such as `\key.txt` , is shown below.

Response Body:

```
{
  "errorMessage": "SSH Key file name should not contain [\\, ;, %, ?, :]"
}
```

11.2 API workflows

This section covers the following topics:

- [API calls for masking administration \(see page 892\)](#)
- [API calls for managing algorithms \(see page 893\)](#)
- [API calls for managing extended connectors \(see page 936\)](#)
- [API calls for ASDD profile set import and export \(see page 941\)](#)
- [API calls for managing classifiers \(see page 945\)](#)
- [API calls for managing profile set usage \(see page 949\)](#)
- [API calls for searching and filtering \(see page 952\)](#)
- [API calls for managing masking job driver support tasks \(see page 958\)](#)
- [API calls for creating an inventory \(see page 963\)](#)
- [API calls for creating and running masking jobs \(see page 966\)](#)
- [API calls involving file upload and download \(see page 972\)](#)
- [API call for generating support bundle \(see page 974\)](#)

11.2.1 API calls for masking administration

The Delphix Continuous Compliance Engine supports the following two types of administrative APIs:

- Analytics APIs
 - These APIs are for including Masking performance information in the support bundle and do not need to be used unless that information is requested.
- Application Setting APIs

- Application Setting APIs allow an administrator to change the Delphix Continuous Compliance Engine settings. Presently there are five categories of settings: analytics settings, LDAP settings, general settings, mask settings and profile settings. Over time, more settings will be added to give users direct control over the product's various settings. Below are the details of currently supported settings.

11.2.2 API calls for managing algorithms

This section covers the following topics:

- [Configuring algorithms \(see page 893\)](#)
- [Managing algorithm usage \(see page 894\)](#)
- [Migrating algorithms \(see page 898\)](#)
- [Binary lookup \(see page 900\)](#)
- [Character mapping \(see page 902\)](#)
- [Data cleansing \(see page 904\)](#)
- [Date replacement \(see page 905\)](#)
- [Date shift \(see page 907\)](#)
- [Dependent date shift \(see page 908\)](#)
- [Email \(see page 910\)](#)
- [Free text redaction \(see page 912\)](#)
- [Full name \(see page 914\)](#)
- [Mapping \(see page 916\)](#)
- [Min Max \(see page 918\)](#)
- [Name \(see page 921\)](#)
- [Numeric expression \(see page 923\)](#)
- [Payment card \(see page 925\)](#)
- [Regex decompose \(see page 926\)](#)
- [Secure lookup \(see page 929\)](#)
- [Segment mapping \(see page 932\)](#)
- [Tokenization \(see page 934\)](#)

11.2.2.1 Configuring algorithms

This section provides information on configuring algorithms using the API.

11.2.2.1.1 Masking client algorithm model

- **algorithmName** (maxLength=201)

String Equivalent to the algorithm name saved by the user through the GUI. For out of the box algorithms, this will be a similar name as that in the GUI, but presented in a more user-friendly format.

- **algorithmType**

String The type of algorithm Enum values: - BINARY_LOOKUP - CLEANSING - COMPONENT - LOOKUP - MAPPING - MINMAX - REDACTION - SEGMENT - TOKENIZATION

- **createdBy** (optional; readOnly; maxLength=255)

String The name of the user that created the algorithm

- **description** (optional; maxLength=255)

String The description of the algorithm

- **frameworkId**

Integer The frameworkId, corresponding to the framework that extensible algorithm is built upon. This field is to be used only for the Extensible Algorithms.

- **algorithmExtension** (optional)

Object Contains algorithm instance specific configuration parameters. See specific framework for more details.

11.2.2.1.2 Algorithm extension for extensible algorithms

It uses the generic Object, defined in the base AlgorithmExtension. Depending on the Extensible Algorithm design it currently supports following implementations (or their mix):

- **fileReference(s)** (optional, name is defined by Extensible Algorithm creator)

single fileReference or array[FileReference] A JSON formatted file reference or list of file references. Each file reference may be one of the following four options: - UUID value returned from the endpoint for uploading file to the Masking Engine - NFS mounted file URL - HTTP URL to external web located file - HTTPS URL to external web located file

- **calledExtendedAlgorithm(s)** (optional, name is defined by Extensible Algorithm creator)

single algorithmName or array[AlgorithmName] A JSON formatted name or a list of extensible algorithms names

11.2.2.2 Managing algorithm usage

11.2.2.2.1 Overview

The Masking Engine provides API endpoints to view and modify the usage of algorithms globally. These operations span all usage of an algorithm, including: database column and file format assignments across all environments, usage in domains, and references from other algorithms.

11.2.2.2.2 Viewing algorithm usage

The following API endpoint retrieves all usage of the algorithm specified in the request path:

```
algorithm GET algorithm/{algorithmName}/usage
```

This endpoint supports the following option in the query parameters:

- **includeAssignmentDetail** (optional, default=false)

boolean Enabling this option causes the API response to include a list of human-readable assignment detail objects, one for each individual usage of the algorithm in inventory. This can result in a very large response if the algorithm is heavily used. Algorithm usage in file formats will be reported once for each application of the file format to a file in inventory.

- **excludeChainedAlgorithms** (optional, default=false)

boolean Enabling this option causes the API response to exclude a list of algorithm references. This parameter will skip the computation of chained algorithms which will result in increased API performance.

- **environmentFilter** (optional)

String Report only usage occurring within the specified environment(s). This query option may be included multiple times, in which case usage for all matching environments will be reported. When the algorithm is used in a file format, all usage of that file format is reported so long as it is referenced by any environment matching the filter; this may include environments other than those selected by the filter. Filtering by environment excludes all domain and algorithm reference usage, as such usage is global rather than part of any particular environment.

- **rulesetFilter** (optional)

String Report only usage occurring within the specified ruleset(s). This query option may be included multiple times, in which case usage for all matching rulesets will be reported. When the algorithm is used in a file format, all usage of that file format is reported so long as it is referenced by any ruleset matching the filter; this may include rulesets other than those selected by the filter. Filtering by ruleset excludes all domain and algorithm reference usage, as such usage is global rather than part of any particular ruleset.

11.2.2.2.3 Updating algorithm usage

The following API endpoint updates all usage of the algorithm specified in the request path to use the new algorithm name supplied as a query parameter:

```
algorithm PUT algorithm/{algorithmName}/usage
```

This endpoint supports the following option in the query:

- **replacementAlgorithmName** (required, no default)

String The name of the algorithm that should replace the existing algorithm in all usage across the entire Masking Engine.

- **ignoreIncompatibleTypes** (optional, default=false)

boolean Setting this option to true will allow some algorithm replacements that would normally fail due to incompatible types to succeed. This may result in job failures if type conversions don't exist to convert the underlying data type to the type expected by the new algorithm.

- **environmentFilter** (optional)

String This options functions just like the environmentFilter option for the GET operation described above. Only usage matching the filter is updated. The update will fail if any file format referencing the algorithm is used from an environment that does not match the filter. Domain and algorithm reference usage is never updated when an environment filter is applied.

- **rulesetFilter** (optional)

String This options functions just like the rulesetFilter option for the GET operation described above. Only usage matching the filter is updated. The update will fail if any file format referencing the algorithm is used from a ruleset that does not match the filter. Domain and algorithm reference usage is never updated when a ruleset filter is applied.

The response body from the PUT request details all usage that was updated by the operation.



Globally updating algorithm usage can produce many inventory changes across multiple environments, and is not easily reversible when the replacement algorithm is already in use on the engine.

Delphix recommends performing the following steps *before* any update to algorithm usage via this API endpoint:

1. Perform the GET usage operation (described above) for both the existing and replacement algorithms. Carefully review the results and save them for future reference.
2. Export the engine's global settings and all affected environments prior to changing algorithm usage.



Algorithm compatibility checking of usage changes may still allow some replacements that could result in job failures using the new algorithm. Careful consideration should be given to whether the new algorithm can handle the data types and inputs for all usage of the algorithm being replaced.

11.2.2.2.4 Examples

Getting usage for an algorithm with one column assignment:

REQUEST

```
curl -X GET --header 'Accept: application/json' --header 'Authorization:
d46db68d-59f1-41e0-a128-c01bc920da30'
'http://masking-engine.example.com/masking/api/v5.1.10/algorithms/alg_6EBH8EGK/usage?
includeAssignmentDetail=false'
```

RESPONSE

```
{
  "algorithmName": "alg_6EBH8EGK",
  "columnMetadataIds": [
    11
  ],
  "fileFieldMetadataIds": [],
  "mainframeDatasetFieldMetadataIds": [],
  "domainNames": [
    "domain_6GXXQP60"
  ],
  "algorithmReferences": []
}
```

The same request with additional detail requested:

REQUEST

```
curl -X GET --header 'Accept: application/json' --header 'Authorization:
d46db68d-59f1-41e0-a128-c01bc920da30'
'http://masking-engine.example.com/masking/api/v5.1.10/algorithms/alg_6EBH8EGK/usage?
includeAssignmentDetail=true'
```

RESPONSE

```
{
  "algorithmName": "alg_6EBH8EGK",
  "columnMetadataIds": [
    11
  ],
  "fileFieldMetadataIds": [],
  "mainframeDatasetFieldMetadataIds": [],
  "domainNames": [
    "domain_6GXXQP60"
  ],
  "algorithmReferences": [],
}
```

```

"assignmentDetails": [
  {
    "assignmentType": "DATABASE_COLUMN",
    "environmentName": "env_ZBQ0XK09",
    "databaseRulesetName": "rule_POQRBZ44",
    "databaseTableName": "profile",
    "databaseColumnName": "last_name"
  }
]
}

```

Updating all usage of algorithm named `alg_6EBH8EGK` to `alg_82U5GUZB`:

REQUEST

```

curl -X PUT --header 'Content-Type: application/json' --header 'Accept: application/
json'
--header 'Authorization: d46db68d-59f1-41e0-a128-c01bc920da30'
'http://masking-engine.example.com/masking/api/v5.1.10/algorithms/alg_6EBH8EGK/usage?
replacementAlgorithmName=alg_82U5GUZB&ignoreIncompatibleTypes=false'

```

RESPONSE

```

{
  "columnMetadataIds": [
    11
  ],
  "fileFieldMetadataIds": [],
  "mainframeDatasetFieldMetadataIds": [],
  "domainNames": [
    "domain_6GXXQP60"
  ],
  "algorithmReferences": [],
  "assignmentDetails": []
}

```

11.2.2.3 Migrating algorithms

11.2.2.3.1 Overview

As Delphix continues to make continuous improvement to the algorithms included with the Masking Engine, some algorithm frameworks will have multiple versions available simultaneously. New API paths have been added to allow migration of existing algorithm instances from old frameworks to new ones. The migration mechanism creates a new algorithm with the same configuration as the existing algorithm, allowing the behavior and performance of the migrated algorithm to be evaluated before adoption of the new algorithm for production use.

In this release, the following algorithm migrations are available:

- **FROM:** `algorithmType=MAPPING` **TO:** `algorithmType=COMPONENT, pluginName=dlpx-core, frameworkName=Mapping`

The [algorithm usage APIs](#) (see page 894) can be used to conveniently transition usage from the old to the new algorithm instance created by the migration mechanism.

11.2.2.3.2 Listing available migrations

The following API endpoint returns a list of result objects describing each possible migration. One object is returned for every algorithm on the engine that can be migrated:

```
algorithm GET algorithm/migration
```

Each object in the response contains the name of the algorithm that can be migrated, as well as the `frameworkId` of the framework that the migrated algorithm would use.

11.2.2.3.3 Migrating algorithms to new frameworks

The following API endpoint creates a new algorithm named **newAlgorithmName** (from the API query parameters), by migrating from the algorithm named in the query path:

```
algorithm POST /algorithms/{algorithmName}/migration
```

This endpoint requires the following option in the query:

- **newAlgorithmName** (required, no default)

String The name of the new algorithm to be created by the migration.

This response from the API is an `AsyncTask` object that can be used to check the status and result of the migration.



Migration of algorithms with a large amount of state (ex. a mapping algorithm with many mappings) can take several minutes or longer to complete. The engine's `info.log` will contain log messages indicating that the migration operation is making progress. Mapping algorithm migration is estimated to take approximately 3 minutes per 1,000,000 mapping values associated with the source algorithm.

11.2.2.3.4 Examples

Listing available migrations:

REQUEST

```
curl -X GET --header 'Accept: application/json' --header 'Authorization:
3d2d6f53-4b1a-42b5-b4c0-33ec3d66082f'
'http://masking-engine.example.com/masking/api/v5.1.10/algorithms/migration'
```

RESPONSE

```
{
  "availableMigrations": [
    {
      "algorithmName": "alg_J24QXMN3",
      "frameworkId": 13
    }
  ]
}
```

Migrating a mapping algorithm from the legacy framework to the new framework:

REQUEST

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/
json'
--header 'Authorization: 3d2d6f53-4b1a-42b5-b4c0-33ec3d66082f'
'http://masking-engine.example.com/masking/api/v5.1.10/algorithms/alg_J24QXMN3/
migration?newAlgorithmName=new_J24QXMN3'
```

RESPONSE

```
{
  "asyncTaskId": 29,
  "operation": "ALGORITHM_MIGRATE",
  "reference": "alg_J24QXMN3",
  "status": "WAITING",
  "cancellable": false
}
```

11.2.2.4 Binary lookup

See [Binary Lookup](#) (see page 705) for more information about this algorithm framework.

11.2.2.4.1 Creating a binary lookup algorithm via API

1. Retrieve the **frameworkId** for the BinarySL Framework. This information can be retrieved using the following endpoint:

```
algorithm GET /algorithm/frameworks
```

The framework information should look similar to the following:

```
{
  "frameworkId": 9,
  "frameworkName": "Binary Lookup",
  "frameworkType": "BYTE_BUFFER",
  "plugin": {
    "pluginId": 7,
    "pluginName": "dlpx-core",
    "pluginAuthor": "Delphix Engineering",
    "pluginType": "EXTENDED_ALGORITHM"
  }
}
```

2. Create a Binary SL algorithm instance via the following endpoint:

```
algorithm POST /algorithms
```

Configure a new algorithm using the [JSON formatted input](#)³⁷⁵ similar to the following:

```
{
  "algorithmName": "newbinarysl",
  "algorithmType": "COMPONENT",
  "description": "Binary Secure Lookup Example",
  "frameworkId": 9,
  "pluginId": 7,
  "algorithmExtension": {
    "lookupFiles": [
      {
        "uri": "delphix-file://upload/
f_39bbf352139f4234873109ad4e6271e1/file1.png"
      },
      {
        "uri": "delphix-file://upload/
f_093b5c07f90e4b9dbddb0339b71703d3/file2.png"
      },
      {
        "uri": "delphix-file://upload/
f_8da2b97e201b4152b2befafc05612d8c/file3.png"
      }
    ]
  }
}
```

³⁷⁵ <https://delphixdocs.atlassian.net/continuous-compliance-10-0-0-0/docs/configuring-algorithms#masking-client-algorithm-model>

```
}

```

11.2.2.4.2 Binary SL algorithm extension

- **lookupFiles**(required, no default)

array[Object] A list of file reference UUID values returned from the endpoint for uploading files to the Masking Engine. There is a maximum limit of 50 files which can be uploaded into each instance of the algorithm

11.2.2.5 Character mapping

See [Character Mapping \(see page 707\)](#) for more information about this algorithm framework.

11.2.2.5.1 Creating a character mapping algorithm via API

1. Retrieve the **frameworkId** for the Character Mapping Framework. This information can be retrieved using the following endpoint:

```
algorithm GET /algorithm/frameworks

```

The framework information should look similar to the following:

```
{
  "frameworkId": 8,
  "frameworkName": "Chracter Mapping",
  "frameworkType" : "STRING",
  "plugin" :
  {
    "pluginId" : 7,
    "pluginName" : "dlpx-core"
  }
}
```

2. Create a Character Mapping algorithm instance via the following endpoint:

```
algorithm POST /algorithms

```

Configure a new algorithm using the [JSON formatted input \(see page 0\)](#) similar to the following:

```
{

```

```

"algorithmName": "Digits and A to F",
"algorithmType": "COMPONENT",
"frameworkId": 8,
"algorithmExtension": {
  "caseSensitive": true,
  "preserveRanges": null,
  "characterGroups": [
    "0123456789", "[a-fA-F]"
  ],
"minMaskedPositions": 1,
"preserveLeadingZeros": false
}

```

11.2.2.5.2 Character mapping algorithm extension

- **characterGroups**(required, no default)

Array of Strings A list of String values defining the characters to be masked. Each group must be either: - a Java regex style character group beginning with '[' - a String of the literal characters that comprise the group.

Duplication of characters within or among groups is not permitted.

- **caseSensitive**(default=true)

Boolean Whether the mapping should be case sensitive. When this is false, each group must be composed either: entirely of characters having no case; or of pairwise matching sets of LC and UC characters - example: [a-zA-Z], not [a-bC-D].

- **minMaskedPositions**(default=1, minimum=0)

Integer The minimum number of positions that must be replaced for masking to be considered successful. Non-conformant data handling is triggered whenever fewer positions are masked. Inputs containing only whitespace never trigger non-conformant data handling.

- **preserveRanges**(optional)

Array of PreserveRange objects A list of PreserveRange objects specifying regions of maskable characters to be preserved. Only maskable characters are considered when determining whether a position is preserved. Ranges are specified with position 0 representing the first maskable character.

- **preserveLeadingZeros**(default=false)

Boolean Whether to preserve leading '0' characters. This option may only be used when '0' is a masked character, and may not be used in conjunction with preserveRanges.

11.2.2.5.2.1 Character Mapping PreserveRange extension

- **start**(minimum=0, required, no default)

Integer The starting position of the preserve range, indexed starting with 0.

- **length**(minimum 1, required, no default)

Integer The length of the preserve range.

- **direction**(default="FORWARD")

String Defines the processing direction for this preserve range, with FORWARD starting at the beginning of input, and REVERSE starting at the end. Possible enum values: - FORWARD - process left to right - REVERSE - process right to left

11.2.2.6 Data cleansing

See [Data cleansing \(see page 714\)](#) for more information about this algorithm framework.

11.2.2.6.1 Creating a data cleansing algorithm via API

1. Retrieve the **frameworkId** for the Data Cleansing Framework. This can be done via the following endpoint:

```
algorithm GET /algorithm/frameworks
```

The framework information should look similar to the following:

```
{
  "frameworkId": 24,
  "frameworkName": "Data Cleansing",
  "frameworkType": "STRING",
  "plugin": {
    "pluginId": 7,
    "pluginName": "dlpx-core",
    "pluginAuthor": "Delphix Engineering",
    "pluginType": "EXTENDED_ALGORITHM"
  }
}
```

2. Upload a lookup file via the following endpoint:

```
fileUpload POST /file-uploads
```

Copy the **fileReferenceId** value returned in the Response Body.

3. Create a Data Cleansing algorithm via the following endpoint:

```
algorithm POST /algorithms
```

Using the [JSON formatted input](#) (see page 0), similar to the following example:

```
{
  "algorithmName": "demoDataCleansing",
  "algorithmType": "COMPONENT",
  "frameworkId": 24,
  "algorithmExtension": {
    "lookupFile": {
      "uri": "delphix-file://upload/f_52b19f8a9125435a83a1237fa53aeaf5/sample.txt"
    },
    "delimiter": "=",
    "caseSensitive": false,
    "trimWhitespace": true
  }
}
```

11.2.2.6.2 Data cleansing algorithm extension

- **lookupFile**(required)

String The fileReferenceId value returned from the fileUpload endpoint for uploading files to the Masking Engine. The file should contain a newline separated list of {value, replacement} pairs separated by the delimiter. No extraneous whitespace should be present.

- **delimiter**(required, minLength=1; maxLength=50; default="=")

String The delimiter string used to separate {value, replacement} pairs in the lookup file.

- **caseSensitive**(optional, default=true)

Boolean Whether the case of the input string must match the values in the lookup file.

- **trimWhitespace**(optional, default=true)

*Boolean Whether to trim leading and trailing whitespace from the input string. Note: This must be **true** to cleanse fixed-width files and fixed-length database data types such as CHAR and NCHAR.*

11.2.2.7 Date replacement

See [Date Replacement](#) (see page 717) for more information about this algorithm framework.

11.2.2.7.1 Creating a date replacement algorithm via API

1. Retrieve the **frameworkId** for the Date Replacement Framework. This information can be retrieved using the following endpoint:

```
algorithm GET /algorithm/frameworks
```

The framework information should look similar to the following:

```
{
  "frameworkId": 1,
  "frameworkName": "Date Replacement",
  "frameworkType" : "LOCAL_DATE_TIME",
  "plugin" :
  {
    "pluginId" : 7,
    "pluginName" : "dlpx-core"
  }
}
```

2. Create a Date Replacement algorithm via the following endpoint:

```
algorithm POST /algorithms
```

Configure a new algorithm using the [JSON formatted input \(see page 0\)](#) similar to the following:

```
{
  "algorithmName": "exampleDateReplacementAlgorithm",
  "algorithmType": "COMPONENT",
  "frameworkId" : 1,
  "algorithmExtension" :
  {
    "minDate": "2020-01-01 00:00:00",
    "maxDate": "2021-01-01 00:00:00",
    "unit": "DAYS"
  }
}
```

11.2.2.7.2 Date replacement algorithm extension

- **minDate**

String A date representing the minimum value that an input can be masked to. The range is inclusive of this value.

- **maxDate**

String A date representing the maximum value that an input can be masked to. The range is inclusive of this value.

- **unit**(default="SECONDS")

String A unit of time that determines what the output is truncated to. For example, when the unit is set to days, the years, months, and days may change, but the hours, minutes, and seconds will always be 00:00:00. Unit options supported by this framework: days, hours, minutes, and seconds.

11.2.2.8 Date shift

See [Date Shift \(see page 908\)](#) for more information about this algorithm framework.

11.2.2.8.1 Creating a date shift algorithm via API

1. Retrieve the **frameworkId** for the Date Shift Framework. This information can be retrieved using the following endpoint:

```
algorithm GET /algorithm/frameworks
```

The framework information should look similar to the following:

```
{
  "frameworkId": 5,
  "frameworkName": "Date Shift",
  "frameworkType": "LOCAL_DATE_TIME",
  "plugin": {
    "pluginId": 7,
    "pluginName": "dlpx-core"
  }
}
```

2. Create a Date Shift algorithm via the following endpoint:

```
algorithm POST /algorithms
```

Configure a new algorithm using the [JSON formatted input \(see page 893\)](#) similar to the following:

```
{
  "algorithmName": "exampleDateShiftAlgorithm",
  "algorithmType": "COMPONENT",
  "frameworkId": 5,
  "algorithmExtension": {
    "minRange": -3,
    "maxRange": 3,
    "unit": "MINUTES",
    "roll": "false"
  }
}
```

```
}
}
```

11.2.2.8.2 Date shift algorithm extension

- **minRange**

Integer A number representing the minimum range value from the input that the input can mask to. The range is inclusive of this value and must be an integer value. Negative values represent units of time in the past and positive values represent units of time in the future. Zero may be included in the range or as one of the range values, but the input will not mask to the same value.

- **maxRange**

Integer A number representing the maximum range value from the input that the input can mask to. The range is inclusive of this value and must be an integer value. Negative values represent units of time in the past and positive values represent units of time in the future. Zero may be included in the range or as one of the range values, but the input will not mask to the same value.

- **unit(default="DAYS")**

String A unit of time that determines what the range is expressed in. Only one unit of time can be specified for each algorithm created. Masked values will be returned with the same granularity as the specified unit. For example a range of 1-2 days will not return the same masked values as a range of 24-48 hours as a range of 1-2 days will return a value with the hours, minutes, and seconds intact but a range of 24-48 hours may return a value with a change in hours anywhere from 24 hours to 48 hours. Unit options supported by this framework: years, months, days, hours, minutes, and seconds.

- **roll(default="false")**

String A boolean that represents whether or not the specified time unit should roll which means that units of time larger and smaller than the specified unit will remain the same. When set to false, there is no guarantee that larger units of time remain the same. When set to true, all larger units of time will retain their same values and the specified unit may wrap around to the beginning. For example, a date at the end of March may wrap around to the beginning of March while keeping all larger units of time and smaller units of time intact. Unit options supported by this framework: months, days, hours, minutes, and seconds.

11.2.2.9 Dependent date shift

See [Dependent Date Shift \(see page 722\)](#) for more information about this algorithm framework.

11.2.2.9.1 Creating a Dependent Date Shift algorithm via API

1. Retrieve the **frameworkId** for the Dependent Date Shift Framework. This information can be retrieved using the following endpoint:

```
algorithm GET /algorithm/frameworks
```

The framework information should look similar to the following:

```
{
  "frameworkId": 3,
  "frameworkName": "Dependent Date Shift",
  "frameworkType": "GENERIC_DATA_ROW",
  "plugin": {
    "pluginId": 7,
    "pluginName": "dlpx-core"
  }
}
```

2. Create a Dependent Date Shift algorithm via the following endpoint:

```
algorithm POST /algorithms
```

Configure a new algorithm using the [JSON formatted input \(see page 0\)](#) similar to the following:

```
{
  "algorithmName": "DependentDateShiftTest",
  "algorithmType": "COMPONENT",
  "createdBy": "admin",
  "description": "Test of the DependentDateShiftAlgo",
  "frameworkId": 3,
  "pluginId": 7,
  "fields": [
    {
      "fieldId": 1,
      "name": "date1",
      "type": "LOCAL_DATE_TIME"
    },
    {
      "fieldId": 2,
      "name": "date2",
      "type": "LOCAL_DATE_TIME"
    }
  ],
  "algorithmExtension": {
    "roll": false,
    "unit": "DAYS",
    "maxRange": 5,
    "minRange": 3,
    "intervalRange": 2
  }
}
```

```
}

```

11.2.2.9.2 Dependent Date Shift Algorithm extension

- **minRange**

Integer This number represents the smallest number of time units that will be added to date1 when masking. The range is inclusive of this value. Negative values represent units of time in the past and positive values represent units of time in the future. If date1 is not provided, this is applied to date2.

- **maxRange**

Integer This number represents the largest number of time units that will be added to date1 when masking. The range is inclusive of this value. Negative values represent units of time in the past and positive values represent units of time in the future. If date1 is not provided, this is applied to date2.

- **unit**(default="DAYS")

String A unit of time that the range is expressed in. This unit is also used to determine the interval between date1 and date2. Supported units include years, months, days, hours, minutes, and seconds.

- **roll**(default="false")

String A boolean that represents whether or not the specified time unit should roll which means that units of time larger and smaller than the specified unit will remain the same. When set to false, there is no guarantee that larger units of time remain the same. Option only supported for months, days, hours, minutes, and seconds. This applies when masking date1. If date1 is not provided, this is applied to date2

- **intervalRange**

Integer A number representing the +/- range value to shift the interval inclusive of the range value. A value of 0 will not change the interval between dates. This number may not be less than 0. If the specified unit difference between date1 and date2 is within the bound of the intervalRange, only values will be provided such that the sign of the difference is preserved. For example, if the day difference between date1 and date2 is 2 and the specified intervalRange is 3, only values greater than -2 will be used (i.e.: -1 to 3). Otherwise, the full range of values will be used (i.e.: -3 to 3).

11.2.2.10 Email

See [Email](#) (see page 726) for more information about this algorithm framework.

11.2.2.10.1 Creating an email algorithm via API

1. Retrieve the **frameworkId** for the Email Framework. This information can be retrieved using the following endpoint:

```
algorithm GET /algorithm/frameworks
```

The framework information should look similar to the following:

```
{
  "frameworkId": 3,
  "frameworkName": "Email",
  "frameworkType": "STRING",
  "plugin": {
    "pluginId": 7,
    "pluginName": "dlpx-core",
    "pluginAuthor": "Delphix Engineering",
    "pluginType": "EXTENDED_ALGORITHM"
  }
}
```

- a. Lookup files should be provided via File Reference. Files can be uploaded via the following endpoint: fileUpload POST /file-uploads Alternatively, those files might also be provided via HTTP / HTTPS / NFS mount URLs.
2. Create an Email algorithm via the following endpoint:

```
algorithm POST /algorithms
```

Configure a new algorithm using the [JSON formatted input \(see page 0\)](#) similar to the following:

```
{
  "algorithmName": "ExampleEmailAlgorithm",
  "algorithmType": "COMPONENT",
  "frameworkId": 3,
  "algorithmExtension": {
    "nameAction": "LOOKUP",
    "domainAction": "REPLACEMENT",
    "nameAlgorithm": null,
    "nameLookupFile": {
      "uri": "delphix-file://upload/f_08bb469a2ddc407bb97a31e96ed0a76a/lookup.txt"
    },
    "domainAlgorithm": null,
    "domainReplacementString": "delphix.com"
  }
}
```

11.2.2.10.2 Email algorithm extension

- **nameAction**

NameAction The type of action to apply to the name portion of the email. Must be one of the following enum values: - *UNIQUE* - applies a SHA-256 hash of the entire input then Base32 encodes the hash value - *LOOKUP* - applies a secure lookup using the values provided in the lookup list - *APPLY_ALGORITHM* - the name portion is replaced by the output of another chained masking algorithm

The **UNIQUE** option may produce masked name portions with lengths up to 52 characters.

- **domainAction**

MaskAction The type of action to apply to the name portion of the email. Must be one of the following enum values: - *REPLACEMENT* - the domain portion is replaced by a fixed value - *APPLY_ALGORITHM* - the domain portion is replaced by the output of another chained masking algorithm

- **nameLookupFile**

FileReference A file reference to a UTF-8 encoded file containing newline separated replacement values for the name portion of the email.

- **nameAlgorithm**

AlgorithmInstanceReference A reference for the algorithm to use when "APPLY_ALGORITHM" is the *NameAction* type. The algorithm must have *maskingType* "STRING". The algorithm will never be passed a null or empty value to mask. See *AlgorithmInstanceReference Extension* below for more information.

- **domainAlgorithm**

AlgorithmInstanceReference A reference for the algorithm to use when "APPLY_ALGORITHM" is the *DomainAction* type. The algorithm must have *maskingType* "STRING". The algorithm will never be passed a null or empty value to mask. See *AlgorithmInstanceReference Extension* below for more information.

- **domainReplacementString**

String

The string to replace the domain portion when "REPLACEMENT" is the *DomainAction* type.

11.2.2.10.2.1 AlgorithmInstanceReference Extension

- **name**

String The algorithm instance name.

11.2.2.11 Free text redaction

See [Free Text Redaction](#) (see page 730) for more information about this algorithm framework.

11.2.2.11.1 Creating a free text redaction algorithm via API

1. Retrieve the **frameworkId** for the Free Text Redaction Algorithm Framework. This information can be retrieved using the following endpoint:

```
algorithm GET /algorithm/frameworks
```

The framework information should look similar to the following:

```
{
  "frameworkId": 8,
  "frameworkName": "Free Text Redaction",
  "frameworkType": "STRING",
  "plugin": {
    "pluginId": 7,
    "pluginName": "dlpx-core",
    "pluginAuthor": "Delphix Engineering",
    "pluginType": "EXTENDED_ALGORITHM"
  }
}
```

2. Upload the **Lookup File**(if any) for the Free Text Redaction Algorithm Framework. It can be done using the following endpoint:

```
fileUpload POST /file-uploads
```

The response with the reference UUID information should look similar to the following:

```
{
  "fileReferenceId": "delphix-file://upload/f_6426ea480db14c1ea9f83f7eb98f3c0e/lookupFile.txt"
}
```

3. Create a Free Text Redaction Algorithm instance via the following endpoint:

```
algorithm POST /algorithms
```

Configure a new algorithm using the [JSON formatted input](#) (see page 0) similar to the following:

```
{
  "algorithmName": "Free Text Redaction for masking addresses and zip codes",
  "algorithmType": "COMPONENT",
  "frameworkId": 8,
  "algorithmExtension": {
    "isDenyList": true,
    "lookupFile": {
      "uri": "delphix-file://upload/f_6426ea480db14c1ea9f83f7eb98f3c0e/lookupFile.txt"
    }
  }
}
```

```

    },
    "lookupFileRedactValue": "redact_value1",
    "regularExpressions": [
      {
        "patternString": "a|A"
      },
      {
        "patternString": "[0-9]{5}"
      }
    ],
    "regExRedactValue": "redact_value2"
  }
}

```

11.2.2.11.2 Free text redaction algorithm extension

- **isDenyList** (required)

Boolean Deny list redaction if true, allow list redaction if false.

- **lookupFile** (optional)

String The reference UUID value returned from the endpoint for uploading the lookup file to the Masking Engine.

- **lookupFileRedactValue** (optional)

String The value to use to redact items matching entries specified in the lookup file.

- **regExPatternList** (optional)

array[patternString]

- **patternString**(required)

String Java 8 style regular expression.

- **regExRedactValue** (optional)

String The value to use to redact items matching regular expression patterns.

11.2.2.12 Full name

See [Full Name](#) (see page 735) for more information about this algorithm framework.

11.2.2.12.1 Creating a full name algorithm via API

1. Find the FrameworkId for the Extensible SL Framework. That might be done via the following EndPoint:

```
algorithm GET /algorithm/frameworks
```

Plugin name is **dlpx-core**, the framework name is **Full Name**.

2. Involved algorithm references might be built using the name of the desired existing extensible String-type algorithm. For example: "firstNameAlgorithmRef" : { "name" : "dlpx-core:FirstName" }
3. Create an Extensible Name Algorithm via the following EndPoint:

```
algorithm POST /algorithms
```

Using the [JSON formatted input \(see page 0\)](#), similar to the following example:

```
{
  "algorithmName": "demo-FullName",
  "algorithmType": "COMPONENT",
  "description": "This is a new style FullName algorithm",
  "frameworkId" : 3,
  "algorithmExtension" :
  {
    "firstNameAlgorithmRef" : { "name" : "dlpx-core:FirstName" },
    "lastNameAlgorithmRef" : { "name" : "dlpx-core:LastName" },
    "maxLengthOfMaskedName" : 0,
    "ifSingleWordConsiderAsLastName" : true,
    "lastNameAtTheEnd" : true,
    "lastNameSeparators" : [ ",", " " ],
    "maxNumberFirstNames" : 2
  }
}
```

Fields description:

"algorithmName" - customer created algorithm name

"algorithmType" - should be "COMPONENT" for Extensible Algorithms

"description" - free text

"frameworkId" - the numeric value found in #1 above

"algorithmExtension" - the composite field, containing algorithm instance specific configuration parameters

11.2.2.12.2 Name algorithm extension

- **firstNameAlgorithmRef**(required)

AlgorithmReferenceId Must be an Algorithm Reference, pointing to an existing extensible algorithm of String type.

- **lastNameAlgorithmRef**(required)

AlgorithmReferenceId Must be an Algorithm Reference, pointing to an existing extensible algorithm of String type.

- **maxLengthOfMaskedName**(optional, default=0)

Integer Should be a non-negative number. The output (masked) value is forcibly trimmed to that length (by the number of characters).

- **ifSingleWordConsiderAsLastName**(optional)

Boolean If true consider single input word as a last name, otherwise as a first name. Default: true

- **lastNameAtTheEnd**(optional)

Boolean If true last name to be detected at the end of the input string, otherwise last name is at the beginning. Default: true

- **lastNameSeparators**(optional)

List [Char] List of the last name separators. Default: contains single value: comma ','

- **maxNumberFirstNames**(optional, default=2, minimum=1, maximum=4)

Integer Defines the max number of first and middle names to be masked. The rest would be ignored.

11.2.2.13 Mapping

See [Mapping](#) (see page 740) for more information about this algorithm framework.

11.2.2.13.1 Creating a mapping algorithm via API

1. Retrieve the **frameworkId** for the Mapping Framework. This information can be retrieved using the following endpoint:

```
algorithm GET /algorithm/frameworks
```

The framework information should look similar to the following:


```
{
  "frameworkId": 15,
  "frameworkName": "Mapping",
  "frameworkType": "STRING",
  "plugin": {
    "pluginId": 7,
    "pluginName": "dlpx-core"
  }
}
```

2. Create a Mapping algorithm instance via the following endpoint:

```
algorithm POST /algorithms
```

Configure a new algorithm using the [JSON formatted input](#) (see [page 0](#)) similar to the following:

```
{
  "algorithmName": "MyMappingAlgo",
  "algorithmType": "COMPONENT",
  "frameworkId": 15,
  "algorithmExtension": {
    "ignoreCharacters": [],
    "mappingSet": {
      "host": "mypostgreshost.mydomain.com",
      "port": 5432,
      "schema": "mySchema",
      "database": "myDb",
      "isRemote": true,
      "algorithmName": "mappingTestRemote",
      "propertiesRef": {
        "uri": "delphix-file://upload/f_6ce20b134d5c4891bf90ccf7bd22d9b1/
mapping.properties"
      }
    }
  }
}
```

 The above is an example of a remote mapping algorithm. See the extension options below for more information.

11.2.2.13.2 Mapping algorithm extension

- **ignoreCharacters** (optional; minimum=32; maximum=126)

array[Integer] The integer ASCII values of characters to ignore in the column data to map

- **mappingSet** (required)

mappingSet object An object that contains information about where the algorithm should find the mappings. See below for object property details.

11.2.2.13.3 MappingSet object

- **algorithmName** (required)

string The name of the algorithm this mappingSet corresponds to.

- **isRemote**

boolean Indicates if the mappings to be used for this algorithm are on the Masking Engine or if they are stored remotely. false if on the engine, true otherwise.

- **host**

string The host where the mapping database is running. Must be provided if isRemote is set to true.

- **port**

string The port to connect to the mapping database on the host. Must be provided if isRemote is set to true.

- **database**

string The name of the mapping database. Must be provided if isRemote is set to true.

- **schema**

string The schema where the mappings are. Must be provided if isRemote is set to true.

- **propertiesRef**

string The reference UUID value returned from the endpoint for uploading files to the Masking Engine. The file must be a properties file containing any further connection information for the database. Must be provided if isRemote is set to true.

11.2.2.14 Min Max

See [Min Max](#) (see page 754) for more information about this algorithm framework.

11.2.2.14.1 Creating a MinMax algorithm via API

1. Retrieve the **frame work Id** for the Minmax Date or Minmax Number Algorithm Framework. This information can be retrieved using the following endpoint:

```
algorithm GET /algorithm/frameworks
```

The framework information should look similar to the following:

```
{
  "frameworkId": 8,
  "frameworkName": "MinMax Date",
  "frameworkType": "LOCAL_DATE_TIME",
  "plugin": {
    "pluginId": 7,
    "pluginName": "dlpx-core",
    "pluginAuthor": "Delphix Engineering",
    "pluginType": "EXTENDED_ALGORITHM"
  }
}
```

Or:

```
{
  "frameworkId": 15,
  "frameworkName": "MinMax Number",
  "frameworkType": "BIG_DECIMAL",
  "plugin": {
    "pluginId": 7,
    "pluginName": "dlpx-core",
    "pluginAuthor": "Delphix Engineering",
    "pluginType": "EXTENDED_ALGORITHM"
  }
}
```

2. Create a Free Text Redaction Algorithm instance via the following endpoint:

```
algorithm POST /algorithms
```

Configure a new algorithm using the [JSON formatted input \(see page 0\)](#) similar to the following (for MinMax Date algorithm framework):

```
{
  "algorithmName": "MinMax Date algorithm for 2021 year",
  "algorithmType": "COMPONENT",
  "frameworkId": 8,
```

```

    "algorithmExtension": {
      "minDate": "2021-01-01 00:00:00",
      "maxDate": "2021-12-31 00:00:00",
      "nonConformingDataDefaultValue": "replacement_value1"
    }
  }

```

or the following (for MinMax Number algorithm framework):

```

{
  "algorithmName": "MinMax Number algorithm for normalizing age
range",
  "algorithmType": "COMPONENT",
  "frameworkId": 15,
  "algorithmExtension": {
    "minValue": 18,
    "maxValue": 65,
    "nonConformingDataDefaultValue": "replacement_value2"
  }
}

```

11.2.2.14.2 Minmax algorithm extension

- **minValue** (required)

Integer The minimum value for a Number range used in conjunction with maxValue. This field is used for "MinMax Number" framework only.

- **maxValue** (required)

Integer The maximum value for a Number range used in conjunction with and must be greater than minValue. This field is used for "MinMax Number" framework only.

- **minDate** (required)

date The minimum value for a Date range used in conjunction with maxDate. The Date must be specified in the following format: "yyyy-MM-dd HH:mm:ss". The Date will be interpreted as UTC. This field is used for "MinMax Date" framework only.

- **maxDate** (required)

date The maximum value for a Date range used in conjunction with and must be greater than minDate. The Date must be specified in the following format: "yyyy-MM-dd HH:mm:ss". The Date will be interpreted as UTC. This field is used for "MinMax Date" framework only.

- **nonConformingDataDefaultValue** (optional)

String The default replacement value for any value that is triggering non conforming data event handling. This field is only applicable when the underlying data to be masked is of type String and conversion to a Date or a Number is required.

11.2.2.15 Name

See [Name](#) (see page 773) for more information about this algorithm framework.

11.2.2.15.1 Creating a name algorithm via API

- Find the **frameworkId** for the Name Framework. This can be done via the following endpoint:

```
algorithm GET /algorithm/frameworks
```

The frameworkName is "**Name**" and the pluginName is "**dlpx-core**".

```
{
  "frameworkId": 10,
  "frameworkName": "Name",
  "frameworkType": "STRING",
  "description": "This Name algorithm masks input name (first or last)
using values from the supplied LookupFile",
  "plugin": {
    "pluginId": 7,
    "pluginName": "dlpx-core",
    "pluginAuthor": "Delphix Engineering",
    "pluginType": "EXTENDED_ALGORITHM"
  }
}
```

- Involved files (lookupFile, particlesToRemoveFile, and particlesToPreserveFile) should be provided via the File Reference. They can be uploaded via the following endpoint:

```
fileUpload POST /file-uploads
```

Alternatively, those files can also be provided via HTTP / HTTPS / NFS mount URLs.

- Create an Extensible Name Algorithm via the following endpoint:

```
algorithm POST /algorithms
```

Using the [JSON formatted input](#) (see page 0), similar to the following example:

```
{
  "algorithmName": "NameDemo",
  "algorithmType": "COMPONENT",
  "description": "This is a new style Name algorithm",
  "frameworkId": 10,
  "algorithmExtension": {
    "lookupFile": {
```

```

        "uri": "delphix-file://upload/f_85f082535d054ee8a11696a24ed86d65/
LN_LOOKUP_100K.txt"
    },
    "particlesToRemoveFile": {
        "uri": "delphix-file://upload/f_1cc829ceee324113ab16c4e750dfce12/
particlesToRemove.txt"
    },
    "particlesToPreserveFile": {
        "uri": "delphix-file://upload/f_1cc829ceee324113ab16c4e750dfce12/
particlesToPreserve.txt"
    },
    "inputCaseSensitive": false,
    "filterAccent": true,
    "maskedValueCase": "PRESERVE_LOOKUP_FILE",
    "maxLengthOfMaskedName": 0,
    "maxNumberNames": 2
    }
}

```

Fields description:

- **"algorithmName"**: User-defined algorithm name.
- **"algorithmType"**: Should be "COMPONENT" for Extensible Algorithms.
- **"description"**: User-defined, free text field.
- **"frameworkId"**: Numeric ID for the framework, provided in line #2 and #5 in the above excerpts.
- **"algorithmExtension"**: The composite field, containing algorithm instance specific configuration parameters.

11.2.2.15.2 Name algorithm extension

- **lookupFile** (required)

String Lookup file may be FileReferenceld in the one of the following four options: - UUID value returned from the endpoint for uploading file to the Masking Engine - NFS mounted file URL - HTTP URL to external web located file - HTTPS URL to external web located file

- **particlesToRemoveFile** (optional)

String File listing particles to remove may be FileReferenceld in the one of the following four options: - UUID value returned from the endpoint for uploading file to the Masking Engine - NFS mounted file URL - HTTP URL to external web located file - HTTPS URL to external web located file

- **particlesToPreserveFile** (optional)

String File listing particles to preserve may be FileReferenceld in the one of the following four options: - UUID value returned from the endpoint for uploading file to the Masking Engine - NFS mounted file URL - HTTP URL to external web located file - HTTPS URL to external web located file

- **inputCaseSensitive** (optional, default=false)

Boolean Setting "true" means input value case matter (i.e. "Peter" and "peter" might be masked to different values). Setting "false" (default) makes input value case insensitive ("Peter" and "peter" would be masked to the same value).

- **filterAccent** (optional, default=true)

Boolean

Setting "true" (default) means accented characters don't matter - similar input with and without accented characters are masked to the same values ("Adrián" and "Adrian" both mask to "John").

Setting "false" makes the input value accent sensitive ("Adrián" and "Adrian" would be masked to different values).

- **maskedValueCase** (optional, default="PRESERVE_INPUT")

String The output (masked) value case enforcing. Enum values: - PRESERVE_LOOKUP_FILE - use the unmodified replacement value.

- PRESERVE_INPUT - preserve case of input value. If mixed, use unmodified replacement value.

- ALL_LOWER - force the output to lowercase. - ALL_UPPER - force the output to uppercase.

- **maxLengthOfMaskedName** (optional, default=0)

Integer Should be a non-negative number. The output (masked) value is forcibly trimmed to that length (by the number of characters).

- **maxNumberNames** (optional, default=2, minimum=1, maximum=4)

Integer

Defines the max number of names to be masked and returned. The rest are removed.

11.2.2.16 Numeric expression

See [Numeric Expression](#) (see page 777) for more information about this algorithm framework.

11.2.2.16.1 Creating a numeric expression algorithm via API

1. Retrieve the **frameworkId** for the Numeric Expression Framework. This information can be retrieved using the following endpoint:

```
algorithm      GET /algorithm/frameworks
```

The framework information should look similar to the following:

```
{
  "frameworkId": 26,
  "frameworkName": "Numeric Expression",
  "frameworkType": "BIG_DECIMAL",
```

```

    "description": "Numeric Expression masks input by ... [truncated for
    brevity].",
    "plugin": {
      "pluginId": 7,
      "pluginName": "dlpx-core",
      "pluginAuthor": "Delphix Engineering",
      "pluginType": "EXTENDED_ALGORITHM"
    }
  }
}

```

2. Create a Numeric Expression algorithm instance via the following endpoint:

```
algorithm POST /algorithms
```

Configure a new algorithm using the [JSON formatted input](#) (see page 0) similar to the following:

```

{
  "algorithmName": "NumericExpressionTest",
  "algorithmType": "COMPONENT",
  "frameworkId": 26,
  "algorithmExtension": {
    "expression": "Math.floor(((input * randomPercentage) * 100.0) + 0.5) /
100.0",
    "inputType": "DOUBLE",
    "constants": [
      {
        "name": "randomPercentage",
        "value": "new java.util.Random(seed).doubles(0.1,
0.9).iterator().nextDouble()"
      }
    ],
    "nonConformingDataDefaultValue": "100.0"
  }
}

```

11.2.2.16.2 Numeric expression algorithm extension

- **expression**

*String One-line mathematical expression written in the Java programming language that references `input` (the current unmasked value), e.g. `input * 0.5` or `input + Math.random()`.*

- **inputType**

String ENUM(DOUBLE, LONG, BIG_DECIMAL) Data type that `input` conforms to within the expression.

`DOUBLE` (default) is double-precision floating point, `LONG` is long integer, and `BIG_DECIMAL` is `java.math.BigDecimal` object.

- **constants**(optional)

array[Constant] An array of `Constant` objects. Constants are variables that the expression can reference by name and whose values remain fixed for the life of a masking job. Constants can reference by name other constants defined before them.

- **nonConformingDataDefaultValue**(optional)

String Default masked value to be used if the unmasked input is not a numeric data type and can't automatically be converted to one.

11.2.2.16.2.1 Constant

- **name**

String Must be valid Java variable name. No two constants can have the same name, nor can "input" or "seed" be used as a constant name.

- **value**

String One-line Java expression that must return a value, which is not required to be numeric.

11.2.2.17 Payment card

See [Payment Card](#) (see page 786) for more information about this algorithm framework.

11.2.2.17.1 Creating a payment card algorithm via API

1. Retrieve the **frameworkId** for the Payment Framework. This information can be retrieved using the following endpoint:

```
algorithm GET /algorithm/frameworks
```

The framework information should look similar to the following:

```
{
  "frameworkId": 4,
  "frameworkName": "Payment Card",
  "frameworkType" : "STRING",
  "plugin" :
```

```
{
  "pluginId" : 7,
  "pluginName" : "dlpx-core"
}
```

2. Create a Payment Card algorithm via the following endpoint:

```
algorithm POST /algorithms
```

Configure a new algorithm using the [JSON formatted input \(see page 0\)](#) similar to the following:

```
{
  "algorithmName": "examplePaymentCardAlgorithm",
  "algorithmType": "COMPONENT",
  "frameworkId" : 4,
  "algorithmExtension" :
  {
    "minMaskedPositions" : 7,
    "preserve" : 4
  }
}
```

11.2.2.17.2 Payment card algorithm extension

- **minMaskedPositions**(default=1, minValue=0, maxValue=32)

Integer A value that represents the minimum number of positions that must be replaced for masking to be considered successful. A non-conformant data error is thrown when fewer positions are masked. The minimum value for this field is 0 and the default value is 1. The maximum value is 32.

- **preserve**(default=0, minValue=0, maxValue=32)

Integer A value that represents the number of maskable characters to preserve at the beginning of the input. Only maskable characters are considered when determining whether a position is preserved. The minimum value for this field is 0 and the default value is 0. The maximum value is 32.

11.2.2.18 Regex decompose

See [Regex Decompose \(see page 788\)](#) for more information about this algorithm framework.

11.2.2.18.1 Creating a regex decompose algorithm via API

1. Retrieve the **frameworkId** for the Regex Decompose Framework. This information can be retrieved using the following endpoint:

```
algorithm GET /algorithm/frameworks
```

The framework information should look similar to the following:

```
{
  "frameworkId": 5,
  "frameworkName": "Regex Decompose",
  "frameworkType": "STRING",
  "plugin": {
    "pluginId": 7,
    "pluginName": "dlpx-core",
    "pluginAuthor": "Delphix Engineering",
    "pluginType": "EXTENDED_ALGORITHM"
  }
}
```

2. Create a Regex Decompose algorithm via the following endpoint:

```
algorithm POST /algorithms
```

Configure a new algorithm using the [JSON formatted input \(see page 0\)](#) similar to the following:

```
{
  "algorithmName": "ExampleRegexDecomposeAlgorithm",
  "algorithmType": "COMPONENT",
  "frameworkId": 5,
  "algorithmExtension": {
    "trimInput": true,
    "requireMask": false,
    "maskPatterns": [
      {
        "regex": "([0-9+)-([a-z]+)",
        "actions": [
          {
            "type": "REDACT",
            "algorithm": null,
            "redactString": "asdf",
            "redactCharacter": null
          },
          {
            "type": "APPLY_ALGORITHM",
            "algorithm": { "name": "dlpx-core:CM Alpha-Numeric" }
          }
        ]
      }
    ]
  },
  "fallbackAction": null,
}
```

```

    "maxLength": 65536
  }
}

```

11.2.2.18.2 Regex decompose algorithm extension

- **maskPatterns**

Array of MaskPattern objects Defines the mask pattern(s) of the algorithm. See Regex Decompose MaskPattern Extension below for more information.

- **fallbackAction**

MaskAction The action that should be applied to the entire input if none of the defined regular expressions match. If no pattern matches and no fallbackAction is set, non-conformant data handling will be triggered. See Regex Decompose MaskAction Extension below for more information.

- **requireMask** (default="true")

String A boolean that represents whether the input must be masked. When this is true, patterns are matched until one changes the input. If no pattern can change the input and no fallbackAction is set, non-conformant data handling will be triggered for this value. If false, the first matching pattern will apply regardless of whether it changes the input. Any difference in value from the input is considered successful masking.

- **trimInput** (default="true")

String A boolean that represents whether to trim whitespace from the beginning and end of the input prior to processing. The same leading and trailing whitespace will be reintroduced into the masked value. This option is provided to simplify the regular expressions that can be used in maskPatterns, as they no longer must account for and preserve leading and trailing whitespace.

- **maxLength** (default=65536, minValue=1)

Integer A value that represents the maximum character length of input the algorithm will attempt to process. If the input length exceeds this value, non-conformant data handling will be triggered for this value.

11.2.2.18.2.1 Regex decompose maskpattern extension

- **regex**

String A Java 8 style regular expression used to match the masking input.

- **actions**

Array of MaskAction objects Defines the action(s) to be applied to the match or capturing group(s) when the regular expression matches. See Regex Decompose MaskAction Extension below for more information.

11.2.2.18.2.2 Regex decompose mask action extension

- **type**

String The type of action to the input that matches the regex. Must be one of the following enum values: - PRESERVE - the value or capturing group is not masked and remains unchanged - TRUNCATE - the value or capturing group is replaced with "" - REDACT - the value or capturing group is replaced by a value or repeated character - APPLY_ALGORITHM - the value or capturing group is replaced by the output of another chained masking algorithm

- **algorithm**

AlgorithmInstanceReference A reference for the algorithm to use when "APPLY_ALGORITHM" is the MaskAction type. The algorithm must have maskingType "STRING". The algorithm will never be passed a null or empty value to mask. The algorithm's **name** attribute should specify the algorithm's name exactly as it would be returned from a call to the API's `GET /algorithms` endpoint, e.g.

```
"algorithm": { "name": "dlpx-core:CM Alpha-Numeric" }
```

- **redactCharacter**

String The character to use to replace the input when "REDACT" is the MaskAction type. Each character in the portion of input matched is replaced with this character. Length of the matched input is preserved. Only one of redactCharacter or redactString may be specified for a given MaskAction.

- **redactString**

String The string to use to replace the input when "REDACT" is the MaskAction type. The entire matched portion is replaced with this string. Use of this option will cause the length of the value to change during masking unless the matched portion of input happens to have the same length of the redactString. Only one of redactCharacter or redactString may be specified for a given MaskAction.

11.2.2.19 Secure lookup

See [Secure Lookup \(see page 793\)](#) for more information about this algorithm framework.

11.2.2.19.1 Creating a secure lookup algorithm via API

1. Find the frameworkId for the Extensible SL Framework. This can be done via the following endpoint:

```
algorithm GET /algorithm/frameworks
```

Plugin name is **dlpx-core**, the framework name is **Secure Lookup**.

2. Upload Lookup File via the following endpoint:

```
fileUpload POST /file-uploads
```

Alternatively, the Lookup File might also be provided via HTTP / HTTPS / NFS mount URLs.

3. Create an Extensible SL Algorithm via the following endpoint:

```
algorithm POST /algorithms
```

Using the [JSON formatted input \(see page 0\)](#), similar to the following example:

```
{
  "algorithmName": "demoExtendedSL",
  "algorithmType": "COMPONENT",
  "frameworkId" : 1,
  "algorithmExtension" :
  {
    "lookupFile" : {
      "uri":"delphix-file://upload/f_7984ee9672b44e309f7ef5940f856e7c/
ColorsLF.txt"
    },
    "inputCaseSensitive" : true,
    "maskedValueCase" : "ALL_LOWER",
    "hashMethod" : "SHA256"
  }
}
```

Fields description:

- "algorithmName": customer created algorithm name
- "algorithmType": should be "COMPONENT" for Extensible Algorithms
- "description": free text
- "frameworkId": the numeric value found in #1 above
- "algorithmExtension": the composite field, containing algorithm instance specific configuration parameters

11.2.2.19.2 Exporting secure lookup values via API

Secure lookup values can now be exported from algorithms. These values can only be exported from algorithms of type **LOOKUP** or type **COMPONENT** where the framework name is **Secure Lookup**.

1. Find the algorithmCd of the algorithm instance to retrieve the values from. This may be done via the following endpoint:

```
algorithm GET /algorithms
```

2. Use the following endpoint to export the lookup values:


```
algorithm POST /algorithms/{algorithmName}/export-lookup-values
```

Info:

Lookup values cannot be exported from algorithms where the lookup values are provided via MOUNT or via HTTP/HTTPS.

3. A response similar to the following will be returned:

```
{
  "asyncTaskId": 55,
  "operation": "EXPORT_SL_VALUES",
  "reference": "EXPORT_SL_VALUES-c2VjdXJlbG9va3VwX2NNSGdZc2FQLnR4dA==",
  "status": "WAITING",
  "cancellable": false
}
```

4. Retrieve the "reference" from the response body in the previous step and use this value as the fileDownloadId for the following endpoint:

```
fileDownload GET /file-downloads/{fileDownloadId}
```

The response will contain the exported lookup values. Values will be returned in a plaintext file with newline-separated values.

11.2.2.19.3 Secure Lookup algorithm extension

- **lookupFile** (maxLength=255)

String Lookup file may be one of the following four options: - UUID value returned from the endpoint for uploading file to the Masking Engine - NFS mounted file URL - HTTP URL to external web located file - HTTPS URL to external web located file

- **inputCaseSensitive** (optional, default=false)

Boolean Setting "true" means input value case matter (i.e. "Peter" and "peter" might be masked to different values) Setting "false" (default) makes input value case-insensitive ("Peter" and "peter" would be masked to the same value)

- **maskedValueCase** (optional, default="PRESERVE_LOOKUP_FILE")

String The output (masked) value case enforcing. Enum values: - PRESERVE_LOOKUP_FILE - use the unmodified replacement value (default). - PRESERVE_INPUT - preserve case of input value. If mixed - use unmodified replacement value. - ALL_LOWER - force the output to lowercase. - ALL_UPPER - force the output to uppercase.

- **hashMethod** (optional, default="SHA256")

String The hash method used to select replacement values. Must be one of the following enum values: - SHA256 - the default hash method for extensible secure lookup - LEGACY - hash method used to mimic the legacy secure lookup behavior in the extensibility framework

11.2.2.20 Segment mapping

See [Segment Mapping](#) (see page 797) for more information about this algorithm framework.

11.2.2.20.1 Creating a segment mapping algorithm via API

1. Retrieve the **frameworkId** for the Segment Mapping Framework. This information can be retrieved using the following endpoint:

```
algorithm    GET /algorithm/frameworks
```

The framework information should look similar to the following:

```
{
  "frameworkId": 6,
  "frameworkName": "Segment Mapping",
  "frameworkType": "STRING",
  "description": "The Segment Mapping Algorithm will ... [truncated for
brevity].",
  "plugin": {
    "pluginId": 7,
    "pluginName": "dlpx-core",
    "pluginAuthor": "Delphix Engineering",
    "pluginType": "EXTENDED_ALGORITHM"
  }
}
```

2. Create a Segment Mapping algorithm instance via the following endpoint:

```
algorithm    POST /algorithms
```

Configure a new algorithm using the [JSON formatted input](#) (see page 0) similar to the following:

```
{
  "algorithmName": "SegmentMappingTest",
  "algorithmType": "COMPONENT",
  "frameworkId": 6,
  "algorithmExtension": {
    "segments": [
      {
```

```

        "length": 4,
        "segmentType": "MASK_NUMERIC",
        "inputValues": null,
        "maskValues": null
    },
    {
        "length": 1,
        "segmentType": "PRESERVE"
    },
    {
        "length": 2,
        "segmentType": "MASK_ALPHANUMERIC",
        "inputValues": "A,B,C,F-G",
        "maskValues": "Q-S,X,Y,Z"
    }
],
"ignoreCharacters": [],
"autoIgnoreCharacters": true,
"allowShortSegments": false,
"processPreserveBeforeIgnore": false
}

```

11.2.2.20.2 Segment mapping algorithm extension

- **segments** (required, minimum=1, maximum=10)

Array of Segment objects A list of Segment Mapping Segments defining the masking behavior in order. See [Segment Mapping Segment Extension](#) (see page 932) below for more information.

- **ignoreCharacters** (optional)

Array of Integers A list of integer ASCII values of characters to ignore. For example, `[44, 65]` would ignore commas and the letter 'A'. These are removed from input value before masking and restored to their original positions after masking.

- **autoIgnoreCharacters** (default=false)

Boolean Whether or not to ignore all non alpha-numeric characters. Use this as an alternative to specifying individual characters in `ignoreCharacters`.

- **allowShortSegments** (default=false)

Boolean Whether or not to allow masking of short MASK_NUMERIC segments. When set to false, a MASK_NUMERIC segment cannot be masked if it is shorter than the defined segment length. If a short MASK_NUMERIC segment is encountered, a NonConformantDataException will be triggered. When set to true, a short MASK_NUMERIC segment may be masked. **Note:** If set to true and a MASK_NUMERIC segment is defined, the algorithm is not reversible and cannot be used for tokenization/re-identification.

- **processPreserveBeforeIgnore** (default=false)

Boolean Whether or not to process *PRESERVE* segments before removing ignore characters. When set to false, ignore characters are removed from the input string first, and then all segments are processed in the order in which they are defined. When set to true, *PRESERVE* segment are processed first, before removing ignore characters, so the preserved segment positions are based on the original input string, which may include ignore characters. Afterwards, ignore characters are removed and the remaining string is masked according to the other segment definitions. **Setting this to true is not recommended, as it may cause some segments to be processed out of order.**

11.2.2.20.2.1 Segment mapping segment extension

- **length** (required, minimum=1, maximum=6)

Integer The length of the segment in characters.

- **segmentType** (required)

String The masking behavior for this segment. Enum values: - *MASK_ALPHANUMERIC* - mask letters to letters and digits to digits. Mappings are configured for each character position independently (e.g. AA -> GC, 'A' does not always mask to the same letter at each position) - *MASK_NUMERIC* - mask the entire segment as a single integer value to another integer value - *PRESERVE* - do not mask this segment - *CONSTANT* - mask any input value to a constant value

- **inputValues**

String Defines the input values to mask in this segment, provided as either individual values, ranges, or a combination thereof. This field is optional for *MASK_ALPHANUMERIC* and *MASK_NUMERIC*, and if left blank or omitted (null), the default value ranges are used. This field is not used for *PRESERVE* or *CONSTANT*. For a *MASK_ALPHANUMERIC* segment, the default value range is '0-9,A-Z'. You can specify something like 'A-F,P,R,1-5,7,9'. For a *MASK_NUMERIC* segment, the default value range is 0 to the max integer that can fit into the segment length (ex: 000-999 for a segment of length 3). You can specify integer values and ranges, like '10,30,50-875'. The masking will only look to mask these values and will preserve any other values.

- **maskValues**

String Defines the values to mask to for this segment. This is defined the same way as *inputValues* and has the same default value ranges. This field is optional for *MASK_ALPHANUMERIC* and *MASK_NUMERIC*, and if left blank or omitted (null), the default value ranges are used. This field is required for *CONSTANT* and not used for *PRESERVE*. **Note:** if the *inputValues* and *maskValues* are not the same, then the algorithm is not reversible and cannot be used for tokenization/re-identification.

11.2.2.21 Tokenization

See [Tokenization](#) (see page 806) for more information about this algorithm framework.

11.2.2.21.1 Creating a tokenization algorithm via API

1. Retrieve the **frameworkId** for the Tokenization Framework. This information can be retrieved using the following endpoint:

```
algorithm GET /algorithm/frameworks
```

The framework information should look similar to the following:

```
{
  "frameworkId": 13,
  "frameworkName": "Tokenization",
  "frameworkType": "STRING",
  "plugin": {
    "pluginId": 7,
    "pluginName": "dlpx-core",
    "pluginAuthor": "Delphix Engineering",
    "pluginType": "EXTENDED_ALGORITHM"
  }
}
```

2. Create a Tokenization algorithm instance via the following endpoint:

```
algorithm POST /algorithms
```

Configure a new algorithm using the [JSON formatted input \(see page 0\)](#) similar to the following:

```
{
  "algorithmName": "exampleTokenization",
  "algorithmType": "COMPONENT",
  "frameworkId": 13,
  "algorithmExtension": {
    "ivLength": 16,
    "fallback": "CHARACTER_MAPPING",
    "cmCharacterGroups": [
      "[A-Za-z0-9+/"
    ],
    "cmMinMaskedPositions": 1
  }
}
```

11.2.2.21.2 Tokenization algorithm extension

- **ivLength**(default=16, minimum=0, maximum=16)

Integer The length of the initialization vector (IV) used for AES in CBC-CTS mode. The default length is 16, which offers the most security. The tradeoff is that this increases the length of the masked result. Selecting a lower IV length decreases the length of the masked result. It is recommended that you only select an IV length of 0 if you require the masked value for each input to be consistent between jobs and for the same input to only mask to one output.

- **fallback**(required, no default)

String This specifies how to handle masking a value where the encrypted result does not fit in the column size. If an AES encrypted result is too long to fit into the field, there are two fallback options: - NONE - the job fails if the masked result is too long - CHARACTER_MAPPING - the Character Mapping algorithm is used to tokenize the value, which produces a result that is the same length as the input

11.2.2.21.2.1 Extension for Character Mapping fallback

- **cmCharacterGroups**(default=["[A-Za-z0-9+/]"])

Array of Strings A list of String values defining the characters to be masked. Each group must be either: - a Java regex style character group beginning with '[' - a String of the literal characters that comprise the group.

Duplication of characters within or among groups is not permitted.

- **cmMinMaskedPositions**(default=1, minimum=0)

Integer The minimum number of positions that must be replaced for masking to be considered successful. Non-conformant data handling is triggered whenever fewer positions are masked. Inputs containing only whitespace never trigger non-conformant data handling.

11.2.3 API calls for managing extended connectors

11.2.3.1 Introduction

This section details how to manage extended database connectors, including how to manage driver support tasks on a masking job.

1. [Installing a driver support plugin \(see page 936\)](#)
2. [Installing a JDBC driver \(see page 939\)](#)
3. [Creating an extended database connector \(see page 940\)](#)
4. [Managing masking job driver support tasks \(see page 958\)](#)



Installing a JDBC driver with a driver support is only possible via the web API.

11.2.3.2 Installing a driver support plugin

11.2.3.2.1 Install driver support jar on masking engine

1. Select `POST /file-uploads`

2. Click "Choose File" and select desired driver support jar

The response will look similar to the following with a return status of 200:


```
{
  "fileReferenceId": "delphix-file://upload/f_xxxx/sampleDriverSupport.jar"
}
```

11.2.3.2.2 Create driver support plugin

1. Select `POST /plugins`
2. **fileReferenceId**: delphix-file://upload/f_xxxx/sampleDriverSupport.jar
3. **pluginName**: whatever desired name
4. **pluginType**: DRIVER_SUPPORT

The response will look similar to the following with a return status of 200:

```
{
  "pluginId": 9,
  "pluginName": "Sample Plugin",
  "pluginAuthor": "Sample Plugin Author",
  "pluginType": "DRIVER_SUPPORT",
  "originalFileName": "driverSupport.jar",
  "originalFileChecksum":
"f8398c0768ecf7709c6992b3f048f9da8be640285b3ccc968973949ca3cceb02",
  "installDate": "2021-04-21T15:29:01.982+00:00",
  "installUser": 5,
  "builtIn": false,
  "pluginVersion": "1.5.0",
  "pluginObjects": [
    {
      "objectIdentifier": "1",
      "objectName": "Disable Constraints",
      "objectType": "DRIVER_SUPPORT_TASK"
    },
    {
      "objectIdentifier": "2",
      "objectName": "Disable Triggers",
      "objectType": "DRIVER_SUPPORT_TASK"
    },
    {
      "objectIdentifier": "3",
      "objectName": "Drop Indexes",
      "objectType": "DRIVER_SUPPORT_TASK"
    }
  ]
}
```

 The `objectIdentifier` field refers to the ID of the task. Specifying the ID of the tasks is required to [enable/disable tasks](#) (see page 958) on a masking job. `objectIdentifier` (task ID) has no bearing on the task execution order. The task order is determined by the order the tasks are added to `getTasks` in the [Driver Support Plugin implementation](#). (see page 1051)


11.2.3.2.3 Create JDBC driver that uses driver support plugin

1. Select `POST /jdbc-drivers` (or `PUT /jdbc-drivers/{jdbcDriverId}` to update existing JDBC driver)
2. Form the request body as follows:

```
{
  "driverName": "HANA driver",
  "driverClassName": "com.sap.db.jdbc.Driver",
  "fileReferenceId": "delphix-file://upload/f_xxxx/sampleJdbcDriver.zip",
  "driverSupportId": 9
}
```

The response will look similar to the following with a return status of 200:

```
{
  "jdbcDriverId": 8,
  "driverName": "HANA driver",
  "driverClassName": "com.sap.db.jdbc.Driver",
  "version": "2.4",
  "uploadedBy": "admin",
  "uploadDate": "2021-04-27T20:34:47.748+00:00",
  "checksum": "a5b7cf1323b71398e68fd583cd4f40ef8a5f4212ae94b63e95c904ed226d4c7b",
  "builtIn": false,
  "loggerInstalled": true,
  "driverSupportId": 9
}
```

 If the referenced driver support plugin is being used by existing masking jobs that have tasks enabled, extra validation is performed. In the case of updating a driver support plugin or updating a JDBC driver to use a different driver support, the driver support plugin must implement all enabled tasks on any existing masking job. If the other driver support does not implement all

enabled tasks, the update will fail. In the case of deleting a driver support plugin, the delete will fail if the driver support plugin is being used by any existing masking jobs that have tasks enabled.


11.2.3.3 Installing a JDBC driver

11.2.3.3.1 Install JDBC driver zip on masking engine

1. Select `POST /file-uploads`
2. Click "Choose File" and select desired JDBC driver zip

The response will look similar to the following with a return status of 200:

```
{
  "fileReferenceId": "delphix-file://upload/f_xxxx/sampleJdbcDriver.zip"
}
```

 Note that you can also install a JDBC driver [via the UI \(see page 0\)](#).

11.2.3.3.2 Create JDBC driver without driver support

1. Select `POST /jdbc-drivers`
2. Format the request body as follows:

```
{
  "driverName": "HANA driver",
  "driverClassName": "com.sap.db.jdbc.Driver",
  "fileReferenceId": "delphix-file://upload/f_xxxx/sampleJdbcDriver.zip",
}
```

The response will look similar to the following with a return status of 200:

```
{
  "jdbcDriverId": 8,
  "driverName": "HANA driver",
  "driverClassName": "com.sap.db.jdbc.Driver",
  "version": "2.4",
}
```


```

    "uploadedBy": "admin",
    "uploadDate": "2021-04-27T20:34:47.748+00:00",
    "checksum": "a5b7cf1323b71398e68fd583cd4f40ef8a5f4212ae94b63e95c904ed226d4c7b",
    "builtIn": false,
    "loggerInstalled": true,
  }

```


11.2.3.3.3 Create JDBC driver with driver support

To create a JDBC driver with driver support, follow the same process, but add `driverSupportId` to the request body. This is used to specify the ID of the driver support plugin to associate with the JDBC driver.

 If the JDBC driver's referenced driver support plugin tasks are enabled on any existing masking job, validation on update is done in order to prevent changing the driver support plugin to another one unless it implements all enabled tasks. If the other driver support does not implement all enabled tasks, the update will fail.

11.2.3.4 Creating an extended database connector

11.2.3.4.1 Creating an extended database connector

 This assumes an application and environment already exists, to which you can add this extended connector.

1. Select `POST /database-connectors`
2. Format response body as follows:

```

{
  "connectorName": "hana db",
  "databaseType": "EXTENDED",
  "environmentId": 1,
  "jdbc": "JDBC_SERVER_URL",
  "username": "USERNAME",
  "password": "PASSWORD",
  "kerberosAuth": false,
  "jdbcDriverId": 7,
  "enableLogger": false
}

```

The response will look similar to the following with a return status of 200:

```
{
  "databaseConnectorId": 1,
  "connectorName": "hana db",
  "databaseType": "EXTENDED",
  "environmentId": 1,
  "jdbc": "JDBC_SERVER_URL",
  "username": "USERNAME",
  "kerberosAuth": false,
  "jdbcDriverId": 7,
  "enableLogger": false
}
```

11.2.3.5 Managing masking job driver support tasks

For information on managing masking driver support tasks, see [API Calls for Managing Masking Job Driver Support Tasks](#). (see page 958)

11.2.4 API calls for ASDD profile set import and export

A profile set defines the set of classifiers or expressions that will be used to identify sensitive information in the rule set when a profiling job is run. Refer to [Discovering your sensitive data](#) (see page 526) for an overview of profile sets and related concepts. More information about ASDD profile set import and export can be found in the [ASDD profile set import and export article](#) (see page 646). More information about configuring profile sets can be found in the [Configuring Profile Sets article](#) (see page 603).

In order to perform ASDD profile set import and export using the API client, you should have some familiarity with REST APIs and JSON data encoding. Access the API Client, as described in the [Masking API Client](#) (see page 878) section, and authenticate by pressing the **Authorize** button at the upper right part of the screen.

The ASDD profile set import and export endpoints utilize the file upload and download endpoints, respectively.

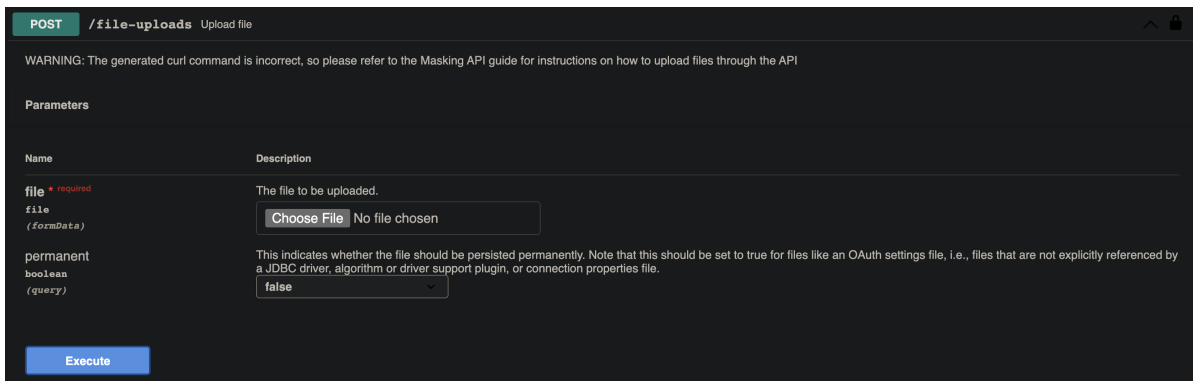


The generated curl command for file upload and download visible in the API client is incorrect. If using curl over the API client is desired, see [API calls involving file upload and download](#) (see page 972) for more information on how to format file upload and download curl requests correctly.

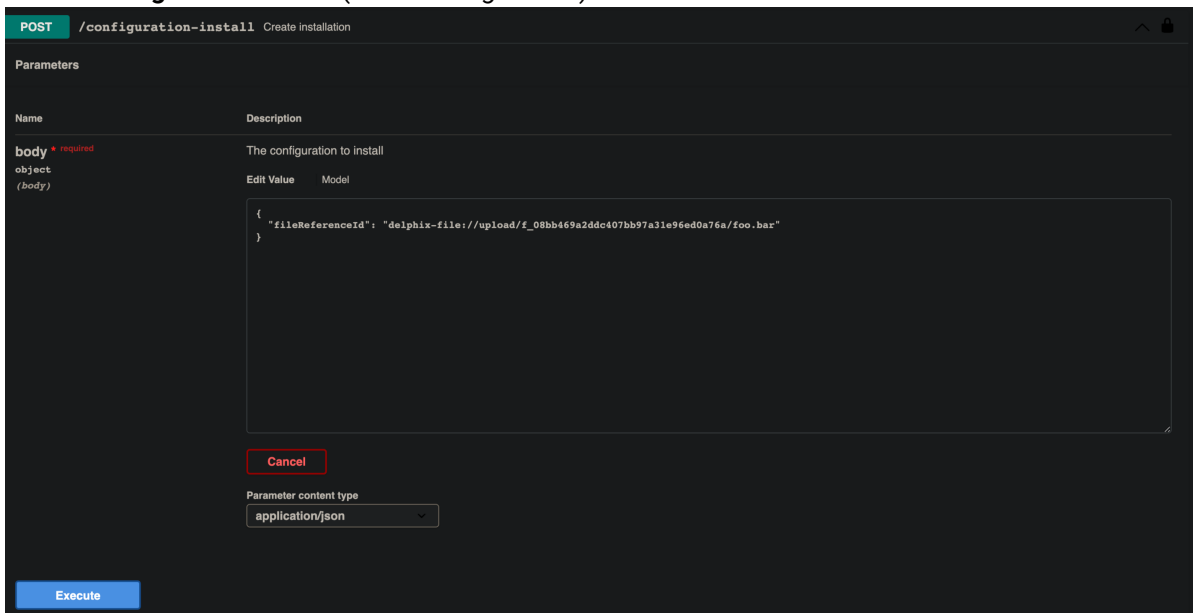
11.2.4.1 ASDD profile set import

Import requires two API calls:

1. **POST /file-upload** (under *fileUpload*)



2. POST /configuration-install (under configuration)



These API path's purposes are as expected based on the path and operation; the purpose of **POST /file-upload** is to upload the profile set configuration zip file to the engine file system, and the purpose of **POST /configuration-install** is to unpack the zip file and import the profile set configuration onto the engine.

11.2.4.1.1 Example

In this example, a new profile set is configured that has a new domain and a few classifiers (one PATH classifier, one LIST and one TYPE). The content of the Standard.json file is as follows:

```
{
  "profileSetName": "Custom ASDD Profile Set",
  "domains": [
    {
      "domain": "CUSTOM_PII",
      "maskingAlgorithmName": "dlpx-core:CM Alpha-Numeric"
    }
  ],
}
```

```

"classifiers": [
  {
    "domain": "CUSTOM_PII",
    "name": "Capital - Path",
    "type": "PATH",
    "properties": {
      "paths": [
        {
          "allowPartialMatch": true,
          "caseSensitive": false,
          "fieldValue": "(?i)capital",
          "matchStrength": 0.67,
          "matchType": "REGEX",
          "parentValue": ""
        },
        {
          "allowPartialMatch": true,
          "caseSensitive": false,
          "fieldValue": "(?i)capital[ _-]??(city)?",
          "matchStrength": 0.5,
          "matchType": "REGEX",
          "parentValue": ""
        }
      ],
      "rejectStrength": 0.0
    }
  },
  {
    "domain": "CUSTOM_PII",
    "name": "Capital - Type",
    "type": "TYPE",
    "properties": {
      "allowedTypes": [
        {
          "minimumLength": 10,
          "typeName": "String"
        }
      ],
      "matchAutoIncrementingColumn": false
    }
  },
  {
    "domain": "CUSTOM_PII",
    "name": "Capitals - List",
    "type": "LIST",
    "properties": {
      "rejectStrength": 0.5,
      "valueLists": [
        {
          "file": "file://de_capitals.txt",
          "matchStrength": 1.0
        }
      ]
    }
  }
]

```

```

    }
  }
]
}

```

The above profile set configuration file (Standard.json) and the file used by the LIST classifier (de_capitals.txt) are contained in a zip file that is uploaded to the engine filesystem via **POST /file-upload** (with the default permanent query parameter value of *false*). The API response is:

```

{
  "fileReferenceId": "delphix-file://upload/f_8f90c13fb1b9434abdbe871da894a2a8/custom_profile_set_1.zip",
  "filename": "custom_profile_set_1.zip",
  "fileSize": 3724,
  "persistenceType": "OBJECT/TEMPORARY"
}

```

The fileReferenceId value returned by **POST /file-upload** is used for the **POST /configuration-install** request to initiate profile set import:

```

{
  "fileReferenceId": "delphix-file://upload/f_8f90c13fb1b9434abdbe871da894a2a8/custom_profile_set_1.zip"
}

```

The API response is:

```

{
  "fileReferenceId": "delphix-file://upload/f_8f90c13fb1b9434abdbe871da894a2a8/custom_profile_set_1.zip",
  "requiresRestart": false
}

```

11.2.4.2 ASDD profile set export

Export requires two API calls:

1. **POST /profile-sets/{profileSetId}/export**
2. **GET /file-downloads/{fileDownloadId}**

This API path's purpose is as expected based on the path and operation; the purpose of **POST /profile-sets/{profileSetId}/export** is to export the given profile set's configuration in a zip file in the same format expected for import. This API call is asynchronous, so it spawns an async task with a file reference that you can then download via **GET /file-downloads/{fileDownloadId}** when the task is complete. The API response for the custom profile set imported above (**POST /profile-sets/5/export**) is:

```

{

```

```


"asyncTaskId": 1,
"operation": "EXPORT_PROFILE_SET",
"reference": "EXPORT_PROFILE_SET-cHJvZm1sZV9zZXRFNS56aXA=",
"status": "WAITING",
"cancellable": false
}

```

GET /file-downloads/EXPORT_PROFILE_SET-cHJvZm1sZV9zZXRFNS56aXA= then provides a link to download the file in the response body.

11.2.5 API calls for managing classifiers

Classifier instances define logic used by the ASDD profiler to identify sensitive information. Refer to [Discovering your sensitive data \(see page 526\)](#) for an overview of classifiers and related concepts. Each classifier instance is based on a **classifier framework** that implements the recognition logic. An overview of the available frameworks is available under the [classifier concept section \(see page 527\)](#). More information about managing classifiers can be found in the [Managing classifiers article \(see page 616\)](#).

 Creating or modifying profile sets can also be done via the API and requires only two API calls. See [ASDD profile set import and export \(see page 646\)](#) for usage instructions.

In order to manage classifiers using the API client, you should have some familiarity with REST APIs and JSON data encoding. Access the API Client, as described in the [Masking API client \(see page 878\)](#) section, and authenticate by pressing the **Authorize** button at the upper right part of the screen. Locate the API paths related to classifiers:

classifier		^
GET	/classifiers Get all classifiers	∨ 🔒
POST	/classifiers Create classifier	∨ 🔒
GET	/classifiers/{classifierId} Get classifier by ID	∨ 🔒
PUT	/classifiers/{classifierId} Update classifier by ID	∨ 🔒
DELETE	/classifiers/{classifierId} Delete classifier by ID	∨ 🔒
POST	/classifiers/{classifierId}/copy Copy classifier by ID	∨ 🔒
POST	/classifiers/{classifierId}/export-files Export files used for specified classifier	∨ 🔒
GET	/classifiers/frameworks Get all classifier frameworks	∨ 🔒
GET	/classifiers/frameworks/{frameworkId} Get classifier framework by frameworkId	∨ 🔒
POST	/classifiers/search Search classifiers	∨ 🔒

Each of these API path's purposes are as expected based on the operation; GET to view the configuration of existing classifiers, PUT to modify the configuration of an existing classifier, POST to create a new classifier, and DELETE to delete a classifier.

11.2.5.1 Retrieving classifier framework configurations

The **classifiers/frameworks** paths allow retrieval of information about the available classifier frameworks. In particular, making a request to these endpoints with `include_schema=true` will return open API style

descriptions of the schema for the classifier frameworks. It is also necessary to use this API to map classifier framework type names, such as 'PATH' or 'REGEX', to the numeric frameworkid when creating classifier instances.

When creating a new classifier, it can be helpful to first perform a GET operation that retrieves the configuration of an existing classifier instance, using the intended framework as a starting point.

11.2.5.2 Example: Creating a new PATH classifier

In this example, a database contains some columns named **snack_pref1**, **snack_pref2**, etc. The columns contain sensitive user data that should be masked, thus, a good regex for recognizing these columns would be **snack_pref[0-9]+**. A **SNACK_PREF** domain has been created with an appropriate algorithm for this type of data. To match the column name to profile, the type of classifier needed is **PATH**.

First, perform a GET operation on the **classifiers/frameworks** path. In the output below, the respective **frameworkid** of the **PATH** classifier is **3**.

```
{
  "_pageInfo": {
    "numberOnPage": 4,
    "total": 4
  },
  "responseList": [
    {
      "frameworkId": 1,
      "frameworkName": "REGEX",
      "description": "The regex framework can be used to specify one or more regular
expressions to match the data in a field."
    },
    {
      "frameworkId": 2,
      "frameworkName": "LIST",
      "description": "The list framework can be used to specify one or more value
lists to match the data in a field."
    },
    {
      "frameworkId": 3,
      "frameworkName": "PATH",
      "description": "The path framework can be used to specify exact values or
regular expressions to match the name of a field."
    },
    {
      "frameworkId": 4,
      "frameworkName": "TYPE",
      "description": "The type framework can be used to specify valid types and type
lengths for fields to rule out invalid data types and lengths during classification."
    }
  ]
}
```

After determining that **classifier 1** has **frameworkId=3**, perform the GET operation on **classifiers/1**:


```
{
  "classifierId": 1,
  "classifierName": "Account Number - Path",
  "frameworkId": 3,
  "domainName": "ACCOUNT_NO",
  "createdBy": "System",
  "builtIn": true,
  "classifierConfiguration": {
    "paths": [
      {
        "matchType": "REGEX",
        "fieldValue": "(?i)(?>(account|acct|acct)_?-? ?(number|num|nbr|no|user))$",
        "parentValue": "",
        "caseSensitive": false,
        "matchStrength": 0.67,
        "allowPartialMatch": true
      }
    ],
    "rejectStrength": 0
  }
}
```

To exemplify, this configuration is edited, replacing several configuration values to create a new classifier:

```
{
  "classifierName": "Snack Preference - Path",
  "frameworkId": 3,
  "domainName": "SNACK_PREF",
  "classifierConfiguration": {
    "paths": [
      {
        "matchType": "REGEX",
        "fieldValue": "snack_pref[0-9]+",
        "parentValue": "",
        "caseSensitive": false,
        "matchStrength": 0.67,
        "allowPartialMatch": false
      }
    ],
    "rejectStrength": 0
  }
}
```

This body can then be used with a POST operation to the **classifiers** path to create the new classifier. The API response will include the newly assigned **classifierId**.

 LIST type classifiers

Note that LIST type classifiers require one or more input files to define the value lists for recognition. These files must first be uploaded by doing a POST to the [fileUpload](#) (see page 972) API endpoint³⁷⁶. The resulting **fileReferenceId** values may then be used for fields of type FILE in the classifier configuration when creating the classifier.

11.2.5.3 Downloading files associated with classifiers

Using the **classifiers/{classifierId}/export-files** endpoint, files associated with classifiers can be downloaded by specifying the **classifierId**. This endpoint will return an async task similar to the following:

```
{
  "asyncTaskId": 2,
  "operation": "EXPORT_CLASSIFIER_FILES",
  "reference": "EXPORT_CLASSIFIER_FILES-Y2xhc3NpZml1c180NV92YWx1ZV9saXN0cy56aXA=",
  "status": "RUNNING",
  "startTime": "2023-03-21T18:45:59.390+00:00",
  "cancellable": false
}
```

You can check the status of this task by using the **asyncTask/{asyncTaskId}** endpoint.

```
{
  "asyncTaskId": 2,
  "operation": "EXPORT_CLASSIFIER_FILES",
  "reference": "EXPORT_CLASSIFIER_FILES-Y2xhc3NpZml1c180NV92YWx1ZV9saXN0cy56aXA=",
  "status": "SUCCEEDED",
  "startTime": "2023-03-21T18:45:59.390+00:00",
  "endTime": "2023-03-21T18:45:59.520+00:00",
  "cancellable": false
}
```

Once this task has succeeded, the **fileDownload/{fileDownloadId}** endpoint can be used to download the files where **fileDownloadId** is the **reference** provided in the **classifiers/{classifierId}/export-files** API response. The files are returned in a .zip that contains all files associated with the specified classifier.

11.2.5.4 Searching and Filtering Classifiers

The **classifiers/search** endpoint allows for searching and filtering of classifiers. More information on syntax can be found at [API calls for searching and filtering](#) (see page 952).

³⁷⁶ <https://delphixdocs.atlassian.net/continuous-compliance-12-0-0-0/docs/api-calls-involving-file-upload-and-download>

11.2.6 API calls for managing profile set usage

11.2.6.1 Overview

The Masking Engine provides an API endpoint to view the usage of profile sets globally.

11.2.6.2 Viewing profile set usage

The following API endpoint retrieves all usages of profile sets:

```
profileSet GET profile-sets/usage
```

This endpoint supports the following options in the query parameters:

- **includeAssignmentDetail** (required, default=false)

boolean Enabling this option causes the API response to include a list of human-readable assignment detail objects, one for each individual usage of the profile set on the engine. This can result in a very large response if there are many profile sets and/or there is heavy usage of the profile sets.

- **profileSetType** (optional)

String Report only usage of profile sets with type ASDD or LEGACY. ASDD profile sets are comprised of classifiers which are used by the new ASDD profiler. LEGACY profile sets are comprised of profile expressions and profile type expressions used by the old profile.

- **connectorType** (optional)

String Report only usage of profile sets used by jobs of the specific connector types: DATABASE, FILE, or MAINFRAME. Note that Mainframe DB2 is categorized under the DATABASE option.

- **environmentFilter** (optional)

String Report only usage occurring within the specified environment(s). This query option may be included multiple times, in which case usage for all matching environments will be reported.

- **profileSetFilter** (optional)

String Report only usage occurring of the specified profile set(s). This query option may be included multiple times, in which case usage for all matching profile sets will be reported.

11.2.6.3 Examples

Getting usage for all profile sets:

REQUEST

```
curl -X 'GET' \
  'http://masking-engine.example.com/masking/api/v5.1.26/profile-sets/usage?
includeAssignmentDetail=false' \
  -H 'accept: application/json' \
  -H 'Authorization: aa20e962-0bcf-4e65-bb44-8028dd3544ce'
```

RESPONSE

```
{
  "responseList": [
    {
      "profileSetName": "ASDD Standard",
      "profileSetId": 4,
      "jobIds": [
        1
      ]
    },
    {
      "profileSetName": "Financial - Legacy",
      "profileSetId": 1,
      "jobIds": []
    },
    {
      "profileSetName": "HIPAA - Legacy",
      "profileSetId": 2,
      "jobIds": []
    },
    {
      "profileSetName": "Standard",
      "profileSetId": 3,
      "jobIds": [
        2
      ]
    }
  ]
}
```

The same request with additional details:

REQUEST

```
curl -X 'GET' \
  'http://masking-engine.example.com/masking/api/v5.1.26/profile-sets/usage?
includeAssignmentDetail=true' \
  -H 'accept: application/json' \
  -H 'Authorization: aa20e962-0bcf-4e65-bb44-8028dd3544ce'
```

RESPONSE

```

{
  "responseList": [
    {
      "profileSetName": "ASDD Standard",
      "profileSetId": 4,
      "jobIds": [
        1
      ],
      "assignmentDetails": [
        {
          "assignmentType": "DATABASE",
          "asddProfileSet": true,
          "environmentName": "TestEnvironment",
          "environmentId": 1,
          "jobName": "TestDatabaseJob",
          "jobId": 1,
          "rulesetName": "DatabaseRuleset",
          "rulesetId": 5
        }
      ]
    },
    {
      "profileSetName": "Financial - Legacy",
      "profileSetId": 1,
      "jobIds": [],
      "assignmentDetails": []
    },
    {
      "profileSetName": "HIPAA - Legacy",
      "profileSetId": 2,
      "jobIds": [],
      "assignmentDetails": []
    },
    {
      "profileSetName": "Standard",
      "profileSetId": 3,
      "jobIds": [
        2
      ],
      "assignmentDetails": [
        {
          "assignmentType": "FILE",
          "asddProfileSet": false,
          "environmentName": "TestEnvironment",
          "environmentId": 1,
          "jobName": "TestFileJob",
          "jobId": 2,
          "rulesetName": "FileRuleset",
          "rulesetId": 2
        }
      ]
    }
  ]
}

```

```
]
}
```

11.2.7 API calls for searching and filtering

API endpoints are available for certain objects that allow searching and filtering. The body of these endpoints takes a single key-value pair {"**filter_expression**": "<expression>"} that should conform to the filter-query language specified below.

The following objects support searching and filtering:

- [classifiers](#) (see page 616)
- [profileExpressions](#) (see page 630)
- [profileTypeExpressions](#) (see page 630)
- executions

11.2.7.1 Comparison operators

Operator	Description	Example
CONTAINS	Substring or membership testing for string and list attributes respectively.	field3 CONTAINS 'foobar', field4 CONTAINS TRUE
IN	Tests if field is a member of a list literal. List can contain a maximum of 100 values.	field2 IN ['Goku', 'Vegeta']
GE	Tests if a field is greater than or equal to a literal value	field1 GE 1.2e-2
GT	Tests if a field is greater than a literal value	field1 GT 1.2e-2
LE	Tests if a field is less than or equal to a literal value	field1 LE 9000
LT	Tests if a field is less than a literal value	field1 LT 9.02
NE	Tests if a field is not equal to a literal value	field1 NE 42
EQ	Tests if a field is equal to a literal value	field1 EQ 42

11.2.7.2 Search operator

The SEARCH operator filters for items that have any filterable attribute that contains the input string as a substring. The comparison is done case-insensitively. This is not restricted to attributes with string values. Specifically `SEARCH '12` would match an item with an attribute with an integer value of `123`.

11.2.7.3 Logical operators

Logical operators are ordered by precedence.

Operator	Description	Example
NOT	Logical NOT (right associative)	NOT filed1 LE 9000
AND	Logical AND (left associative)	field1 GT 9000 AND field2 EQ 'Goku'
OR	Logical OR (left associative)	field1 GT 9000 OR field2 EQ 'Goku'

11.2.7.4 Grouping

Parentheses '()' can be used to override operator precedence.

For example:

- NOT (field1 LT 1234 AND field2 CONTAINS 'foo')

11.2.7.5 Literal Values

Literal	Description	Example
Nil	Represents the absence of a value	nil, Nil, nll, NIL
Boolean	true/false boolean	true, false, True, False, TRUE, FALSE
Number	Signed integer and floating point numbers. Also supports scientific notation.	0, 1, -1, 1.2, 0.35, 1.2e-2, -1.2e+2
String	Single or double quoted	"foo", "bar", "foo bar", 'foo', 'bar', 'foo bar'

Literal	Description	Example
Datetime	Formatted according to RFC3339 ³⁷⁷	2018-04-27T18:39:26.397237+00:00
List	Comma-separated literals wrapped in square brackets	[0], [0, 1], ['foo', "bar"]

11.2.7.6 Limitations

- Not all fields of the objects are filterable or searchable. The allowed fields for the specific object are listed in the body description of that object API endpoint.
- For a filterable field of the form *field1*[*field2*], the query within **filter_expression** should start with the "contains" operator.
 - For example, for a filterable-field of "domain[domain_name]" the query body should be **{ "filter_expression": "domain contains {domain_name EQ 'Account_NO'}" }**
- A maximum of 8 unique identifiers may be used inside a filter expression.

11.2.7.7 Filtering usage examples

Below is a sample 'fruit_inventory' table containing information that will be filtered using the above syntax.

id	name	color	size	quantity	in_season
1	apple	red	medium	4	false
2	watermelon	red	large	1	true
3	strawberry	red	small	10	true
4	orange	orange	medium	7	false
5	kiwi	green	small	3	false
6	raspberry	red	small	20	false
7	lemon	yellow	medium	2	true

³⁷⁷ <https://www.ietf.org/rfc/rfc3339.txt>

id	name	color	size	quantity	in_season
8	lime	green	small	8	false
9	pineapple	yellow	large	3	true
10	blueberry	blue	small	132	true

11.2.7.7.1 Example 1:

This example uses the CONTAINS operator to search for the substring "berry" in the name field.

```
{"filter_expression": "name CONTAINS 'berry'"}
```

This query returns the following objects:

id	name	color	size	quantity	in_season
3	strawberry	red	small	10	true
6	raspberry	red	small	20	false
10	blueberry	blue	small	132	true

11.2.7.7.2 Example 2:

This example uses the GT operator to filter for quantities greater than 5 and the EQ operator to filter for size "small".

```
{"filter_expression": "quantity GT 5 AND size EQ 'small'"}
```

This query returns the following objects:

id	name	color	size	quantity	in_season
3	strawberry	red	small	10	true
6	raspberry	red	small	20	false

id	name	color	size	quantity	in_season
8	lime	green	small	8	false
10	blueberry	blue	small	132	true

11.2.7.7.3 Example 3:

This example uses the NOT operator to filter for objects where the color is not in list ['red', 'orange', 'green'].

```
{"filter_expression": "NOT color IN ['red','orange','green']"}
```

This query returns the following objects:

id	name	color	size	quantity	in_season
7	lemon	yellow	medium	2	true
9	pineapple	yellow	large	3	true
10	blueberry	blue	small	132	true

11.2.7.7.4 Example 4:

This example uses the EQ operator to filter for objects where the boolean value of in_season is equal to true.

```
{"filter_expression": "in_season EQ true"}
```

This query returns the following objects:

id	name	color	size	quantity	in_season
2	watermelon	red	large	1	true
3	strawberry	red	small	10	true
7	lemon	yellow	medium	2	true

id	name	color	size	quantity	in_season
9	pineapple	yellow	large	3	true
10	blueberry	blue	small	132	true

11.2.7.7.5 Example 5:

This example uses a combination of many operators with grouping to filter for objects that are green, small, and have a quantity greater than or equal to 8, or objects that are medium, not in season, and are in list ['apple', 'lemon'].

```

{"filter_expression": "(color EQ 'green' AND size EQ 'small' AND quantity GE 8) OR (size EQ 'medium' AND in_season EQ false AND name IN ['apple', 'lemon'])"}
    
```

This query returns the following objects:

id	name	color	size	quantity	in_season
1	apple	red	medium	4	false
8	lime	green	small	8	false

11.2.7.7.6 Example 6:

This example demonstrates the filter query on an 'order' table (described below), which is joined to the 'fruit_inventory' table (above) by the fruit-'name' column. Use the 'contains' query only when the allowed filterable_attribute in the API mentions

explicitly like 'order[order_id]' instead of 'order_id'.

order_id	name	order_quantity
1	apple	2
2	strawberry	5
3	lime	7

```
{"filter_expression": "order contains {name EQ 'lime'}"}
```

This query returns the following objects:

order_id	name	order_quantity
3	lime	7

11.2.8 API calls for managing masking job driver support tasks

Enabling driver support tasks is possible for built-in Oracle and MSSQL connectors as well as extended connectors that [have a JDBC driver that uses a driver support plugin \(see page 936\)](#) at the following endpoints:

- Masking jobs - POST /masking-jobs and PUT /masking-jobs/{maskingJobId}
- Reidentification jobs - POST /reidentification-jobs and PUT /reidentification-jobs/{reidentificationJobId}
- Tokenization jobs - POST /tokenization-jobs and PUT /tokenization-jobs/{tokenizationJobId}

Disabling driver support tasks is possible for built-in Oracle and MSSQL connectors as well as extended connectors that [have a JDBC driver that uses a driver support plugin \(see page 936\)](#) at the following endpoints:

- PUT /masking-jobs/{maskingJobId}
- PUT /reidentification-jobs/{reidentificationJobId}
- PUT /tokenization-jobs/{tokenizationJobId}

i The order of the tasks returned in `enabledTasks` in the Job APIs' responses is not indicative of the task execution order. The task order is determined by the order the tasks are added to `getTasks` in the [Driver support plugin implementation. \(see page 1051\)](#)

The following instructions to enable driver support tasks on an Oracle masking job can be used to enable driver support tasks for applicable reidentification and tokenization jobs as well.

11.2.8.1 View the tasks implemented by driver support plugin

1. Select GET /plugin (or GET /plugin/{pluginId} if the plugin ID of the driver support is known).
2. Change `pluginType` query parameter to `DRIVER_SUPPORT` (default is `EXTENDED_ALGORITHM`)

3. The response should include the full list of driver support plugins on the masking engine. If the engine only has the builtin Oracle driver support plugin installed, the response will look as follows:

```
{
  "_pageInfo": {
    "numberOnPage": 1,
    "total": 1
  },
  "responseList": [
    {
      "pluginId": 8,
      "pluginName": "dlpx-oracle-driver-support",
      "pluginAuthor": "Delphix Engineering",
      "pluginType": "DRIVER_SUPPORT",
      "originalFileName": "delphix-oracle-driver-support-plugin-1.0.0.jar",
      "originalFileChecksum":
"17b06f2fd888888e26a634d501b4ac9be5a91a7f50000a995934145c7afe7e12",
      "installDate": "2021-10-24T18:08:50.868+00:00",
      "builtIn": true,
      "pluginVersion": "1.0.0",
      "description": "This plugin provides built-in driver support
functionality for the Oracle JDBC driver that ships with the Delphix Masking
Engine.",
      "pluginObjects": [
        {
          "objectIdentifier": "1",
          "objectName": "Disable Constraints",
          "objectType": "DRIVER_SUPPORT_TASK"
        },
        {
          "objectIdentifier": "2",
          "objectName": "Drop Indexes",
          "objectType": "DRIVER_SUPPORT_TASK"
        },
        {
          "objectIdentifier": "3",
          "objectName": "Disable Triggers",
          "objectType": "DRIVER_SUPPORT_TASK"
        }
      ]
    }
  ]
}
```

11.2.8.2 Create masking Job that enables tasks

i This assumes a ruleset using the desired connector already exists. The following example demonstrates the creation of an in-place masking job on a built-in Oracle connector. This also assumes you know the ID of the task that you want to enable and have execute as part of a given masking job. To enable tasks to execute as part of a masking job on an *extended* connector, you need to ensure the ruleset points to an extended connector that is using a JDBC driver with a driver support and include the property `enabledTasks` in your request.

1. Select `POST /masking-jobs` to create a masking job using the ruleset you created earlier that targets the desired connector.
2. Format the request body as follows to enable Disable Constraints, Drop Indexes and Disable Triggers per the `objectIdentifier` values returned from the GET Plugin API endpoint:

```
{
  "jobName": "Oracle IP job",
  "rulesetId": 1,
  "jobDescription": "Job description",
  "enabledTasks": [
    {
      "taskId": 1
    },
    {
      "taskId": 2
    },
    {
      "taskId": 3
    }
  ]
}
```

The response will look similar to the following with a return status of 200:

```
{
  "maskingJobId": 1,
  "jobName": "Oracle IP job",
  "rulesetId": 1,
  "rulesetType": "table",
  "createdBy": "admin",
  "createdTime": "2021-04-27T21:29:46.043+00:00",
  "feedbackSize": 50000,
  "jobDescription": "Job description",
  "maxMemory": 1024,
```

```

"minMemory": 1024,
"multiTenant": false,
"numInputStreams": 1,
"onTheFlyMasking": false,
"databaseMaskingOptions": {
  "batchUpdate": true,
  "commitSize": 10000,
  "disableConstraints": false,
  "dropIndexes": false,
  "disableTriggers": false,
  "numOutputThreadsPerStream": 1,
  "truncateTables": false
},
"failImmediately": false,
"enabledTasks": [
  {
    "taskId": 1
  },
  {
    "taskId": 2
  },
  {
    "taskId": 3
  }
],
"streamRowLimit": 20000
}

```

11.2.8.3 Disable tasks

To disable the Disable Triggers task on an Oracle masking job, the request body to `PUT /masking-jobs/1` should exclude the `taskId` of the task to disable. Using the above request body as an example, Disable Triggers has a task ID of 3 so the request body to `PUT /masking-job/1` should exclude the object in `enabledTasks` with `"taskId": 3`. The request body should thus be:

```
{
  "jobName": "Oracle IP job",
  "rulesetId": 1,
  "jobDescription": "Job description",
  "onTheFlyMasking": false,
  "enabledTasks": [
    {
      "taskId": 1
    },
    {
      "taskId": 2
    }
  ]
}
```

The Oracle masking job will now only have Disable Constraints and Drop Indexes enabled (in this example, their respective task IDs are 1 and 2). The response will look similar to the following with a return status of 200:

```
{
  "maskingJobId": 1,
  "jobName": "Oracle IP job",
  "rulesetId": 1,
  "rulesetType": "table",
  "createdBy": "admin",
  "createdTime": "2021-04-27T21:29:46.043+00:00",
  "feedbackSize": 50000,
  "jobDescription": "Job description",
  "maxMemory": 1024,
  "minMemory": 1024,
  "multiTenant": false,
  "numInputStreams": 1,
  "onTheFlyMasking": false,
  "databaseMaskingOptions": {
    "batchUpdate": true,
    "commitSize": 10000,
    "disableConstraints": false,
    "dropIndexes": false,
    "disableTriggers": false,
    "numOutputThreadsPerStream": 1,
    "truncateTables": false
  },
  "failImmediately": false,
  "enabledTasks": [
    {
      "taskId": 1
    },
    {
      "taskId": 2
    }
  ],
}
```



```
"streamRowLimit": 20000
}
```

11.2.9 API calls for creating an inventory

Below are examples of requests you might enter and responses you might receive from the Masking API client. For commands specific to your masking engine, work with your interactive client at `http://<myMaskingEngine>/masking/api-client/`



HTTPS (SSL/TLS) is recommended, but for explanatory purposes these examples use insecure HTTP



In all code examples, replace `<>` with the hostname or IP address of your virtual machine.

11.2.9.1 Fetch table names from database connector

Object references you will need:

- The ID of the database connector to fetch tables for



This database connector ID (1, in this example) is included in the PATH for this operation, NOT the payload.

11.2.9.1.1 REQUEST

```
curl -X GET --header 'Accept: application/json' --header 'Authorization:
7c856e3d-5b20-4261-b5fe-cc2ffcee5ae0'
'http://<myMaskingEngine>/masking/api/database-connectors/1/fetch'
```

11.2.9.1.2 RESPONSE

```
[ "ALL_COLUMNS", "DBVERIFICATION_TABLE" ]
```

11.2.9.1.3 More info

```
http://<myMaskingEngine>/masking/api-client/#!/databaseConnector/fetchTableMetadata
```

11.2.9.1.4 Example

See how to use this in the context of a script [here \(see page 987\)](#).

11.2.9.2 Create table metadata

Object references you will need:

- The name of the table to create the metadata for
- The ruleset ID

11.2.9.2.1 REQUEST

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' --header 'Authorization: 7c856e3d-5b20-4261-b5fe-cc2ffcee5ae0' -d '{ "tableName": "ALL_COLUMNS", "rulesetId": 2 }' 'http://<myMaskingEngine>/masking/api/table-metadata'
```

11.2.9.2.2 RESPONSE

```
{ "tableMetadataId": 2, "tableName": "ALL_COLUMNS", "rulesetId": 2 }
```

11.2.9.2.3 More info

```
http://<myMaskingEngine>/masking/api-client/#!/tableMetadata/createTableMetadata
```

11.2.9.2.4 Example

See how to use this in the context of a script [here \(see page 987\)](#).

11.2.9.3 Get All column metadata belonging to table metadata

Object references you will need:

- The table metadata ID to get the columns for

i This table metadata ID (2, in this example) is included in the QUERY STRING for this operation, NOT the payload.

11.2.9.3.1 REQUEST

```
curl -X GET --header 'Accept: application/json' --header 'Authorization:
7c856e3d-5b20-4261-b5fe-cc2ffcee5ae0'
'http://<myMaskingEngine>/masking/api/column-metadata?table_metadata_id=2'
```

11.2.9.3.2 RESPONSE

```
[ { "columnMetadataId": 12, "columnName": "schoolnme",
"tableMetadataId": 2, "columnLength": 50, "isMasked": false,
"isPrimaryKey": false, "isIndex": false, "isForeignKey": false }, ... ]
```

Note that the above response has been truncated due to its length for the purposes of this documentation.

11.2.9.3.3 More info

<http://<myMaskingEngine>/masking/api-client/#!/columnMetadata/getAllColumnMetadata>

11.2.9.3.4 Example

See how to use this in the context of a script [here \(see page 987\)](#).

11.2.9.4 Update column metadata with algorithm assignment

Object references you will need:

- Column metadata ID for the column you wish to update

i Tip
This column metadata ID (20, in this example) is included in the PATH for this operation, NOT the payload.

- Since the names can vary in the API and UI, you should use the names obtained through the API (these may not align with the UI).
- Algorithm name
- Domain name

11.2.9.4.1 REQUEST

```
curl -X PUT --header 'Content-Type: application/json' --header 'Accept:
application/json' --header 'Authorization:
7c856e3d-5b20-4261-b5fe-cc2ffcee5ae0' -d '{ "algorithmName":
"AddrLine2Lookup", "domainName": "ADDRESS_LINE2", "isProfilerWritable": false }'
'http://<myMaskingEngine>/masking/api/column-metadata/20'
```

11.2.9.4.2 RESPONSE

```
{ "columnMetadataId": 20, "columnName": "l2_address",
"tableMetadataId": 2, "algorithmName": "AddrLine2Lookup", "domainName":
"ADDRESS_LINE2", "columnLength": 512, "isMasked": true, "isProfilerWritable": false,
"isPrimaryKey":
false, "isIndex": false, "isForeignKey": false, "domainAssignedBy": "admin_user"
}
```

11.2.9.4.3 More info

<http://<myMaskingEngine>/masking/api-client/#!/columnMetadata/updateColumnMetadata>

11.2.9.4.4 Example

See how to use this in the context of a script [here \(see page 987\)](#).

11.2.10 API calls for creating and running masking jobs

Below are examples of requests you might enter and responses you might receive from the Masking API client. For commands specific to your masking engine, work with your interactive client at <http://<myMaskingEngine>/masking/api-client/>



In all code examples, replace **<myMaskingEngine>** with the hostname or IP address of your virtual machine.



HTTPS (SSL/TLS) is recommended, but for explanatory purposes these examples use insecure HTTP.

11.2.10.1 Creating a masking job

Object references you will need:

- The ID of the ruleset for which you wish to create the masking job

REQUEST

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' --header 'Authorization: e23bad24-8760-4091-a131-34f235d9b2d6' -d '{ "jobName": "some_masking_job", "rulesetId": 7, "jobDescription": "This example illustrates a MaskingJob with just a handful of the possible fields set. It is meant to exemplify a simple JSON body that can be passed to the endpoint to create a MaskingJob.", "feedbackSize": 100000, "onTheFlyMasking": false }'
'http://<myMaskingEngine>/masking/api/masking-jobs'
```

11.2.10.1.1 RESPONSE

```
{ "jobId": 1, "jobName": "some_masking_job", "rulesetId": 7, "createdBy": "Axistech", "createdTime": "2017-07-04T00:31:00.952+0000", "environmentId": 2, "feedbackSize": 100000, "jobDescription": "This example illustrates a MaskingJob with just a handful of the possible fields set. It is meant to exemplify a simple JSON body that can be passed to the endpoint to create a MaskingJob.", "maxMemory": 1024, "minMemory": 1024, "multiTenant": false, "numInputStreams": 1, "onTheFlyMasking": false }
```



The response includes the ID of the newly created job ("jobId").

11.2.10.1.2 More info

<http://<myMaskingEngine>/masking/api-client/#!/maskingJob/createMaskingJob>

11.2.10.2 Running a masking job

Create a new execution of a masking job.

Object references you will need:

- The ID of the job you want to run

11.2.10.2.1 REQUEST

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' --header 'Authorization: e23bad24-8760-4091-a131-34f235d9b2d6' -d '{"jobId": 1}' 'http://<myMaskingEngine>/masking/api/executions'
```

11.2.10.2.2 RESPONSE

```
{ "executionId": 1, "jobId": 1, "status": "RUNNING" }
```

11.2.10.2.3 More info

<http://<myMaskingEngine>/masking/api-client/#!/execution/createExecution>

11.2.10.3 Checking the status of a masking job

Object references you will need:

- The ID of the execution you want to check (IN THE PATH)



This execution id (1, in this example) is included in the PATH for this operation, NOT the payload.

11.2.10.3.1 REQUEST

```
curl -X GET --header 'Accept: application/json' --header 'Authorization: 8935f7f7-6de6-40ba-80d8-d8956b71248b'
```

```
'http://<myMaskingEngine>/masking/api/executions/1'
```

11.2.10.3.2 RESPONSE

```
{
  "executionId": 1,
  "jobId": 1,
  "status": "SUCCEEDED",
  "rowsMasked": 1000,
  "rowsTotal": 1000,
  "startTime": "2019-02-14T21:51:13.253+0000",
  "endTime": "2019-02-14T21:51:54.956+0000"
}
```

11.2.10.3.3 More info

<http://<myMaskingEngine>/masking/api-client/#!/execution/getExecutionById>

11.2.10.4 Retrieving execution events related to a masking job

Object references you will need:

- The ID of the execution you want to check (as a URL parameter).




This execution id (1, in this example) is specified as a URL parameter for this operation.

The execution-events endpoint returns execution events for a specified job execution. These execution events report failures or warnings associated with the masking job execution. NOT specifying the execution in the URL parameter will retrieve all execution events for all masking jobs.

11.2.10.4.1 REQUEST

```
curl -X GET --header 'Accept: application/json' --header 'Authorization:
8935f7f7-6de6-40ba-80d8-d8956b71248b'
'http://<myMaskingEngine>/masking/api/execution-events?execution_id=1&page_number=1'
```

11.2.10.4.2 RESPONSE

 `eventType` can be `JOB_ABORTED`, `UNMASKED_DATA`, `MASKING_FALLBACK` or `FILE_PATTERN_NO_MATCH`, the last of which is new as of 21.0.0.0. New execution event types are subject to be added in any Delphix Engine release. New execution event types will be specified in the Release Notes of the release in which it was introduced.

If custom scripts are being used to process execution events by type, they must be adjusted accordingly to account for new execution event types.

```
{
  "_pageInfo": {
    "numberOnPage": 1,
    "total": 1
  },
  "responseList": [
    {
      "executionEventId": 1,
      "executionId": 1,
      "eventType": "UNMASKED_DATA",
      "severity": "WARNING",
      "cause": "PATTERN_MATCH_FAILURE",
      "count": 1000,
      "timeStamp": "2019-02-14T21:51:51.790+0000",
      "executionComponentId": 1,
      "maskedObjectName": "RCHARS64_T1_0",
      "algorithmName": "DateShiftVariable"
    }
  ]
}
```

11.2.10.4.3 More info

<http://<myMaskingEngine>/masking/api-client/#!/execution-events/getAllExecutionEvents>

11.2.10.5 Retrieving non-conformant data samples associated with an execution Event

Object references you will need:

- The ID(s) of the execution event(s) you want to check (as one or more URL parameters).

- This execution event id (1, in this example) is specified as a URL parameter for this operation.

The non-conformant-data-sample endpoint returns non-conformant data samples for a specified job execution event. These non-conformant data samples will report the data patterns that caused the non-conformant data execution event to help identify why data is not getting masked. NOT specifying an execution event in the URL parameter will retrieve all non-conformant data samples events for all masking jobs.

11.2.10.5.1 REQUEST

```
curl -X GET --header 'Accept: application/json' --header 'Authorization:
8935f7f7-6de6-40ba-80d8-d8956b71248b'
'http://<myMaskingEngine>/masking/api/non-conformant-data-sample?
execution_event_id=1&page_number=1'
```

11.2.10.5.2 RESPONSE

```
{
  "_pageInfo": {
    "numberOnPage": 7,
    "total": 7
  },
  "responseList": [
    {
      "dataSampleId": 1,
      "executionEventId": 1,
      "dataSample": "LLLLL",
      "count": 200
    },
    {
      "dataSampleId": 2,
      "executionEventId": 1,
      "dataSample": "LLLLLL",
      "count": 400
    },
    {
      "dataSampleId": 3,
      "executionEventId": 1,
      "dataSample": "LLLL",
      "count": 80
    },
    {
      "dataSampleId": 4,
```

```

    "executionEventId": 1,
    "dataSample": "LLLLLLLL",
    "count": 100
  },
  {
    "dataSampleId": 5,
    "executionEventId": 1,
    "dataSample": "LLLLLLLLLLLLL",
    "count": 50
  },
  {
    "dataSampleId": 6,
    "executionEventId": 1,
    "dataSample": "LLLLLLLLLLLL",
    "count": 10
  },
  {
    "dataSampleId": 7,
    "executionEventId": 1,
    "dataSample": "LLLLLLLLL",
    "count": 40
  }
]
}

```

11.2.10.5.3 More info

<http://<myMaskingEngine>/masking/api-client/#!/non-conformant-data-sample/getAllNon-conformantDataSamples>

11.2.11 API calls involving file upload and download

11.2.11.1 File download

API calls involving file download through API client are noteworthy because if the request fails, the API client will continue to show the "loading" icon indefinitely.

To avoid this, make all file download calls through CURL instead. An example of a file download call using CURL is below.

```

curl -X GET --header 'Accept: application/octet-stream' --header
'Authorization: ec443730-124e-4958-a872-324a975bb500'
-o "/home/user/downloads"
'http://<myMaskingEngine>/masking/api/file-downloads/EXPORT-
ZXhwb3J0X2RvY3VtZW50X2dGZU9JMVYxLmpzb24%3D'

```

The `-o` flag from above specifies the location to save the file to.

11.2.11.2 File upload

API calls involving file upload are noteworthy because the generated curl from the Masking API client will be **missing the parameter referencing the file**; as such, those commands from the Masking API client **will not work**.

Instead, below are examples of working requests and responses for API calls involving file upload.

For commands specific to your masking engine, work with your interactive client at `http://<myMaskingEngine>/masking/api-client/`



HTTPS (SSL/TLS) is recommended, but for explanatory purposes these examples use insecure HTTP.



In all code examples, replace `<myMaskingEngine>` with the hostname or IP address of your virtual machine.

11.2.11.3 Creating a file format

11.2.11.3.1 REQUEST

```
curl -X POST --header 'Content-Type: multipart/form-data' --header
'Accept: application/json' --header 'Authorization:
d1313dd8-2ed9-4699-8e88-2b6a089ae2a6' -F
fileFormat=@/path/to/file_format/delimited_format.txt -F
fileFormatType=DELIMITED
'http://<myMaskingEngine>/masking/api/file-formats'
```

11.2.11.3.2 RESPONSE

```
{ "fileFormatId": 123, "fileFormatName": "delimited_format.txt",
  "fileFormatType": "DELIMITED"
}
```

11.2.11.3.3 More info

<http://<myMaskingEngine>/masking/api-client/#!/fileFormat/createFileFormat>

11.2.11.4 Creating an SSH Key

11.2.11.4.1 REQUEST

```
curl -X POST --header 'Content-Type: multipart/form-data' --header
'Accept: application/json' --header 'Authorization:
d1313dd8-2ed9-4699-8e88-2b6a089ae2a6' -F
sshKey=@/path/to/ssh_key/this_file_name_is_your_ssh_key_name.txt
'http://<myMaskingEngine>/masking/api/ssh-keys'
```

11.2.11.4.2 RESPONSE

```
{ "sshKeyName": "this_file_name_is_your_ssh_key_name.txt"
}
```

11.2.11.4.3 More info

<http://<myMaskingEngine>/masking/api-client/#!/sshKey/createSshKey>

11.2.12 API call for generating support bundle

Below are examples of requests you might enter and responses you might receive from the Masking API client. For commands specific to your masking engine, work with your interactive client at <http://<myMaskingEngine>/masking/api-client/>



In all code examples, replace **<myMaskingEngine>** with the hostname or IP address of your virtual machine.



HTTPS (SSL/TLS) is recommended, but for explanatory purposes these examples use insecure HTTP.

11.2.12.1 Generating a support bundle


No arguments are required for launching a support bundle generation task

11.2.12.1.1 REQUEST

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' --header 'Authorization: 5f745517-d4ce-45de-afb3-6be06205188f' 'http://<myMaskingEngine>/masking/api/v5.2.0/support-bundle'
```

11.2.12.1.2 RESPONSE

```
{
  "asyncTaskId": 5,
  "operation": "SUPPORT_BUNDLE_GENERATE",
  "reference": "",
  "status": "RUNNING",
  "startTime": "2022-06-02T16:33:42.792+00:00",
  "cancellable": true
}
```

-  The response includes the ID of the launched asynchronous task (“asyncTaskId”) which runs the scripts collecting the Support Bundle information and packs it into tar.gz file.

11.2.12.2 Reading the async task result


Object references you will need:

- The ID of the asyncTask retrieved from the response of the `supportBundle` endpoint (in the response example above equals 5)

Retrieve the result of the async task by running `asyncTask GET /async-tasks/{asyncTaskId}` endpoint.

11.2.12.2.1 REQUEST

```
curl -X GET --header 'Accept: application/json' --header
'Authorization: 5f745517-d4ce-45de-afb3-6be06205188f'
'http://<myMaskingEngine>/masking/api/v5.2.0/async-tasks/6'
```

-  Gathering the support bundle information might be a relatively long task, running minutes (depending on the amount of the accumulated information on the Masking Engine). While it's not finished the response will have "status": "RUNNING" and "reference":"" (i.e. empty).

After the task is finished the response will look like:


```
{
  "asyncTaskId": 6,
  "operation": "SUPPORT_BUNDLE_GENERATE",
  "reference": "SUPPORT_BUNDLE-d\lpx-support-564db0c0-162b-c22f-f2ed-
b17ff6b933b0-20220602-16-48-03.tar.gz",
  "status": "SUCCEEDED",
  "startTime": "2022-06-02T16:48:02.969+00:00",
  "endTime": "2022-06-02T16:50:25.960+00:00",
  "cancellable": true
}
```

11.2.12.3 Retrieving the generated support bundle file

The Support Bundle file may be downloaded using the `fileDownload GET /file-downloads/{fileDownloadId}` API endpoint.

Object references you will need:

- The reference provided in the succeeded async task response, to be used as `fileDownloadId` input argument.

-  Getting the Support Bundle file to the browser memory might take few minutes (depending on the generated Support Bundle size).

- The `Response Content Type` field should be set to `application/octet-stream` value. If it's left on the default `application/json` than the downloaded file wouldn't be recognized as a valid tar file.

11.2.12.3.1 REQUEST

```
curl -X GET --header 'Accept: application/octet-stream'
--header 'Authorization: 5f745517-d4ce-45de-afb3-6be06205188f'
'http://<myMaskingEngine>/masking/api/v5.2.0/file-downloads/SUPPORT_BUNDLE-dlpx-
support-564db0c0-162b-c22f-f2ed-b17ff6b933b0-20220602-16-48-03.tar.gz'
```

11.2.12.3.2 RESPONSE

The `Response Body` is represented as a clickable download URL, for example: `Download SUPPORT_BUNDLE-dlpx-support-564db0c0-162b-c22f-f2ed-b17ff6b933b0-20220602-16-48-03.tar.gz`

Here you have 2 options:

- Click on that link, and the Support Bundle tar file would be downloaded to your default download directory.
- Use the above curl command to download the support bundle file. To keep the same file name you need to add `-O` (capital letter O) argument to this curl command, for example:

```
$ curl -X GET --header 'Accept: application/octet-stream' --header 'Authorization:
5f745517-d4ce-45de-afb3-6be06205188f' 'http://<myMaskingEngine>/masking/api/v5.2.0/
file-downloads/SUPPORT_BUNDLE-dlpx-support-564db0c0-162b-c22f-f2ed-
b17ff6b933b0-20220602-16-48-03.tar.gz' -O
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
 100 17.5M    0 17.5M    0     0    735k      0  --:--:--  0:00:24  --:--:--  782k
```

OR

```
$ curl -X GET --header 'Accept: application/octet-stream' --header 'Authorization:
5f745517-d4ce-45de-afb3-6be06205188f' 'http://<myMaskingEngine>/masking/api/v5.2.0/
file-downloads/SUPPORT_BUNDLE-dlpx-support-564db0c0-162b-c22f-f2ed-
b17ff6b933b0-20220602-16-48-03.tar.gz' --output support_bundle.tar.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
```

100 17.5M 0 17.5M 0 0 660k 0 ---:---:-- 0:00:27 ---:---:-- 426k



If you choose not to use the curl command but clicking to the download URL - the downloaded file has autogenerated suffix added to the name of the file, for example:

`application_octet-stream_SUPPORT_BUNDLE-dlpx-support-564db0c0-162b-c22f-f2ed-b17ff6b933b0-20220602-16-48-03.tar.gz_blob_http___<>` You might use that file as is, or rename it to the desired name. The recommendation is to leave the .tar.gz extension.

11.2.12.3.3 More info

- Only one support bundle generation task can be running at a time.
- Support Bundle generation is cancellable (via asyncTask `PUT /async-tasks/{asyncTaskId}/cancel` endpoint).

11.3 API examples

This section covers the following topics:

- [loginCredentials](#) (see page 979)
- [helpers](#) (see page 979)
- [apiHostInfo](#) (see page 982)
- [Configure enclosure escape character](#) (see page 983)
- [createApplication](#) (see page 986)
- [createEnvironment](#) (see page 986)
- [createInventory](#) (see page 987)
- [create DatabaseConnector](#) (see page 988)
- [create DatabaseRuleset](#) (see page 989)
- [getBillingUsage](#) (see page 989)
- [getAuditLogs](#) (see page 990)
- [getSyncableObjects](#) (see page 991)
- [getSyncableObjectsExport](#) (see page 991)
- [profileTypeExpressions](#) (see page 993)
- [runMaskingJob](#) (see page 994)
- [getDatabaseUsage](#) (see page 995)

11.3.1 loginCredentials

```
#!/bin/bash

#
# This file contains all the login information for the masking engine.
#

# Login credentials for the Masking Engine.
USERNAME="myUsername"
PASSWORD="myPassword"

# Login into a masking engine
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/
json' -d '\{ \ "username": "myUsername", \ "password": "myPassword" \}' 'http://
<myMaskingEngine>/masking/api/login'
```

11.3.2 helpers

```
#!/bin/bash

#
# This file contains helpers for the various Masking API cookbook scripts.
# This script uses jq to process JSON. More information can be found here - https://stedolan.github.io/jq/.
#

# Login and set the correct $AUTH_HEADER.
login() {
    echo "* logging in..."
    LOGIN_RESPONSE=$(curl -s $SSL_CERT -X POST -H 'Content-Type: application/json' -H
'Accept: application/json' --data @- $MASKING_ENGINE/login <<EOF
{
    "username": "$USERNAME",
    "password": "$PASSWORD"
}
EOF
) || die "Login failed with exit code $?"
    check_error "$LOGIN_RESPONSE"
    TOKEN=$(echo $LOGIN_RESPONSE | jq -r '.Authorization')
    AUTH_HEADER="Authorization: $TOKEN"
}

# Get all applications and select the first one. Place the applicationName in
$APPLICATION_ID.
get_application_id() {
```

```

    echo "* getting all applications and selecting first one"
    APPLICATIONS_RESPONSE=$(curl -s $SSL_CERT -X GET -H ""$AUTH_HEADER"" -H
'Content-Type: application/json' $MASKING_ENGINE/applications)
    check_error "$APPLICATIONS_RESPONSE"
    NUM_APPLICATIONS=$(echo $APPLICATIONS_RESPONSE | jq -r '._pageInfo.total')
    check_empty $NUM_APPLICATIONS "found no applications to use"
    APPLICATION_ID=$(echo $APPLICATIONS_RESPONSE | jq -r
'.responseList[0].applicationName')
    echo "using application '$APPLICATION_ID'"
}
# Get all environments and select the first one. Place the environmentId in
$ENVIRONMENT_ID.
get_environment_id() {
    echo "* getting all environments and selecting first one"
    ENVIRONMENTS_RESPONSE=$(curl -s $SSL_CERT -X GET -H ""$AUTH_HEADER"" -H
'Content-Type: application/json' $MASKING_ENGINE/environments)
    check_error "$ENVIRONMENTS_RESPONSE"
    NUM_ENVIRONMENTS=$(echo $ENVIRONMENTS_RESPONSE | jq -r '._pageInfo.total')
    check_empty $NUM_ENVIRONMENTS "found no environments to use"
    ENVIRONMENT_ID=$(echo $ENVIRONMENTS_RESPONSE | jq -r
'.responseList[0].environmentId')
    echo "using environment '$ENVIRONMENT_ID'"
}
# Get all database connectors and select the first one. Place the databaseConnectorId
in $CONNECTOR_ID.
get_connector_id() {
    echo "* getting all database connectors and selecting first one"
    CONNECTORS_RESPONSE=$(curl -s $SSL_CERT -X GET -H ""$AUTH_HEADER"" -H 'Content-
Type: application/json' $MASKING_ENGINE/database-connectors)
    check_error "$CONNECTORS_RESPONSE"
    NUM_CONNECTORS=$(echo $CONNECTORS_RESPONSE | jq -r '._pageInfo.total')
    check_empty $NUM_CONNECTORS "found no db connectors to use"
    CONNECTOR_ID=$(echo $CONNECTORS_RESPONSE | jq -r
'.responseList[0].databaseConnectorId')
    echo "using database connector '$CONNECTOR_ID'"
}
# Get all database rulesets and select the first one. Place the databaseRulesetId in
$RULESET_ID.
get_ruleset_id() {
    echo "* getting all database rulesets and selecting first one"
    RULESETS_RESPONSE=$(curl -s $SSL_CERT -X GET -H ""$AUTH_HEADER"" -H 'Content-
Type: application/json' $MASKING_ENGINE/database-rulesets)
    check_error "$RULESETS_RESPONSE"
    NUM_RULESETS=$(echo $RULESETS_RESPONSE | jq -r '._pageInfo.total')
    check_empty $NUM_RULESETS "found no db rulesets to use"
    RULESET_ID=$(echo $RULESETS_RESPONSE | jq -r '.responseList[0].databaseRulesetId')
    echo "using database ruleset '$RULESET_ID'"
}
# Get all database tables for a database connector specified by $CONNECTOR_ID. Select
the first one and place in $TABLE_NAME.
get_table() {

```

```

    echo "* getting all tables for connector '$CONNECTOR_ID' and selecting first one"
    TABLES_RESPONSE=$(curl -s $SSL_CERT -X GET -H ""$AUTH_HEADER"" -H 'Content-
Type: application/json' $MASKING_ENGINE/database-connectors/$CONNECTOR_ID/fetch)
    check_error "$TABLES_RESPONSE"
    NUM_TABLES=$(echo $TABLES_RESPONSE | jq -r '. | length')
    check_empty $NUM_TABLES "found no tables to use"
    TABLE_NAME=$(echo $TABLES_RESPONSE | jq -r '.[0]')
    echo "using table '$TABLE_NAME'"
}

# Get all column metadata for table metadata specified by $TABLE_METADATA_ID. Select
the first one and place in $COLUMN_METADATA_ID.
get_column_metadata_id() {
    echo "* getting all column metadata belonging to table metadata
'$TABLE_METADATA_ID' and selecting the first one"
    COLUMNS_RESPONSE=$(curl -s $SSL_CERT -X GET -H ""$AUTH_HEADER"" -H 'Content-
Type: application/json' $MASKING_ENGINE/column-metadata?)
    table_metadata_id=$TABLE_METADATA_ID
    check_error "$COLUMNS_RESPONSE"
    NUM_COLUMNS=$(echo $COLUMNS_RESPONSE | jq -r '. | length')
    check_empty $NUM_COLUMNS "found no columns to use"
    COLUMN_METADATA=$(echo $COLUMNS_RESPONSE | jq -r '.responseList[0]')
    COLUMN_METADATA_ID=$(echo $COLUMN_METADATA | jq -r '.columnMetadataId')
    echo "using column '$COLUMN_METADATA_ID'"
}

# Get all masking jobs and select the first one. Place the jobId in $MASKING_JOB_ID.
get_masking_job_id() {
    echo "* getting all masking jobs and selecting first one"
    MASKINGJOB_RESPONSE=$(curl -s $SSL_CERT -X GET -H ""$AUTH_HEADER"" -H 'Content-
Type: application/json' $MASKING_ENGINE/masking-jobs)
    check_error "$MASKINGJOB_RESPONSE"
    NUM_MASKINGJOB=$(echo $MASKINGJOB_RESPONSE | jq -r '._pageInfo.total')
    check_empty $NUM_MASKINGJOB "found no masking jobs to use"
    MASKING_JOB_ID=$(echo $MASKINGJOB_RESPONSE | jq -r
'.responseList[0].maskingJobId')
    echo "using masking job '$MASKINGJOB_ID'"
}

# run_masking_job and save execution id in $MASKING_EXECUTION_ID.
run_masking_job() {
    echo "* running masking job '$MASKING_JOB_ID'..."
    MASKINGJOB_RESPONSE1=$(curl $SSL_CERT -X POST -H ""$AUTH_HEADER"" -H 'Content-
Type: application/json' -H 'Accept: application/json' --data @- $MASKING_ENGINE/
executions <<EOF
    {
        "jobId": "$MASKING_JOB_ID"
    }
EOF
)
    echo "Response for Masking job is: '$MASKINGJOB_RESPONSE1'"
    MASKING_EXECUTION_ID=$(echo $MASKINGJOB_RESPONSE1 | jq -r '.executionId')

```

```

}

# get_execution_status in $MASKING_EXECUTION_STATUS.
get_execution_status() {
    echo "* Getting execution details.....for execution id = $1"
    MASKINGJOB_RESPONSE=$(curl -s $SSL_CERT -X GET -H '"$AUTH_HEADER"' -H 'Content-
Type: application/json' $MASKING_ENGINE/executions/$1)
    check_error "$MASKINGJOB_RESPONSE"
    MASKING_EXECUTION_STATUS=$(echo $MASKINGJOB_RESPONSE | jq -r '.status')
    echo "Execution status for id= $1 is '$MASKING_EXECUTION_STATUS'"
}

# Check if $1 is equal to 0. If so print out message specified in $2 and exit.
check_empty() {
    if [ $1 -eq 0 ]; then
        echo $2
        exit 1
    fi
}

# Check if $1 is an object and if it has an 'errorMessage' specified. If so, print
the object and exit.
check_error() {
    # jq returns a literal null so we have to check against that...
    if [ "$(echo "$1" | jq -r 'if type=="object" then .errorMessage else "null"
end')" != 'null' ]; then
        echo $1
        exit 1
    fi
}

# Print the message and exit the program.
die() {
    echo
    "*****"
    echo "$(basename $0) ERROR: $*" >&2
    echo
    "*****"
    exit 1
}

```

11.3.3 apiHostInfo

```

#!/bin/bash

#
# This file contains all the host information for the masking engine. Additionally,
# this file allows configuration of SSL if desired.
#

```

```

# update host name
HOST="myMaskingEngine.com"
API_PATH="masking/api"

# To connect via SSL, set $SSL to "on" and update the port if necessary (default 8443)
.
# Additionally, you must update the path to the ssl certificate.
SSL="off"
SSL_PORT="8443"
# update cert name
SSL_CERT_PATH="self-signed.cer"

if [ "$SSL" = "on" ]
then
    MASKING_ENGINE="https://$HOST:$SSL_PORT/$API_PATH"
    SSL_CERT="--cacert $SSL_CERT_PATH"
else
    MASKING_ENGINE="http://$HOST/$API_PATH"
    SSL_CERT=""
fi

```

11.3.4 Configure enclosure escape character

```

#!/bin/bash

#
# This script uses the Masking Engine APIs to configure the enclosure escape
# character feature.
# The script uses the /login API to obtain an authentication token and then uses the
# PUT /file-metadata API.
#
# To use this script, you must set DOUBLE_ENCLOSURE,
# CUSTOM_ENCLOSURE_ESCAPE_CHARACTER, ESCAPE_ENCLOSURE_ESCAPE_CHARACTER and RULESET_ID
# accordingly

source ./apiHostInfo.bash
eval $(cat ./loginCredentials.bash)
source ./helpers.bash

helpFunction() {
    echo ""
    echo "Usage: $0 -h HELP -d DOUBLE_ENCLOSURE -c CUSTOM_ENCLOSURE_ESCAPE_CHARACTER
-e ESCAPE_ENCLOSURE_ESCAPE_CHARACTER -r RULESET_ID"
    echo -e "\t-h Show the usage of the script"
    echo -e "\t-d Set the value for DOUBLE_ENCLOSURE"
    echo -e "\t-c Set the value for CUSTOM_ENCLOSURE_ESCAPE_CHARACTER"
}

```

```

    echo -e "\t-e Set the value for ESCAPE_ENCLOSURE_ESCAPE_CHARACTER"
    echo -e "\t-r Set the value for RULESET_ID"
    echo -e "\n\tAdditional Note:"
    echo -e "\t1: The default value for parameter D=true, no need to set the value if
you want to set enclosure escape character same as enclosure character."
    echo -e "\t2: If parameter D=true then custom enclosure escape character value
will be ignored."
    echo -e "\t3: The default value for parameter E=false, change accordingly as per
the requirement."
    echo -e "\t4: If parameter R is blank, it means changes will be applicable for
all rulesets. Pass the R={RULESET_ID} if you want to update the settings only for the
given ruleset. Example R=1"
    exit 1 # Exit script after printing help
}

# Set DOUBLE_ENCLOSURE=true if you want to set enclosure escape character same as
enclosure character,
# and if DOUBLE_ENCLOSURE=true then CUSTOM_ENCLOSURE_ESCAPE_CHARACTER value will be
ignored.
DOUBLE_ENCLOSURE=true
# Replace * with your custom escape character if you want to set custom enclosure
escape character
# and also DOUBLE_ENCLOSURE=false need to set
CUSTOM_ENCLOSURE_ESCAPE_CHARACTER="*"
# Modify ESCAPE_ENCLOSURE_ESCAPE_CHARACTER value accordingly.
ESCAPE_ENCLOSURE_ESCAPE_CHARACTER=false
# Comment this RULESET_ID if you want to update for all delimited file ruleset for
which enclosure is defined.
#RULESET_ID=1

while getopts "h:d:c:e:r:" opt; do
    case "$opt" in
        h) helpFunction exit ;;
        d) DOUBLE_ENCLOSURE="$OPTARG" ;;
        c) CUSTOM_ENCLOSURE_ESCAPE_CHARACTER="$OPTARG" ;;
        e) ESCAPE_ENCLOSURE_ESCAPE_CHARACTER="$OPTARG" ;;
        r) RULESET_ID="$OPTARG" ;;
        ?) helpFunction ;; # Print helpFunction in case parameter is non-existent
    esac
done

# Print helpFunction in case parameters are empty
if [ -z "$DOUBLE_ENCLOSURE" ] || [ -z "$CUSTOM_ENCLOSURE_ESCAPE_CHARACTER" ] || [ -z
"$ESCAPE_ENCLOSURE_ESCAPE_CHARACTER" ]; then
    echo "Some or all of the parameters are empty"
    helpFunction
fi

login

echo "Calling GET /file-metadata API"

```

```

if [[ -z "$RULESET_ID" ]] || [ "$RULESET_ID" = "null" ] || [ "$RULESET_ID" = "" ];
then
    echo "Configuring the enclosure escape character feature for all File Ruleset."
    FILE_METADATA_RESPONSE=$(curl $SSL_CERT -X GET -H ""$AUTH_HEADER"" -H 'Accept:
application/json' ""$MASKING_ENGINE/file-metadata'')
else
    echo "Configuring the enclosure escape character feature for File
Ruleset(RULESET_ID=$RULESET_ID)"
    FILE_METADATA_RESPONSE=$(curl $SSL_CERT -X GET -H ""$AUTH_HEADER"" -H 'Accept:
application/json' ""$MASKING_ENGINE/file-metadata?ruleset_id=$RULESET_ID'')
fi

i=0
while true; do
    ENCLOSURE=$(jq '.responseList['$i'] .enclosure' <<<"$FILE_METADATA_RESPONSE")

    if [ "$DOUBLE_ENCLOSURE" = true ]; then
        CUSTOM_ENCLOSURE_ESCAPE_CHARACTER=$ENCLOSURE
    fi

    UPDATED_FILE_METADATA_RESPONSE=$(jq '.responseList['$i'] .enclosureEscapeCharacte
r='$CUSTOM_ENCLOSURE_ESCAPE_CHARACTER' <<<"$FILE_METADATA_RESPONSE")
    UPDATED_FILE_METADATA_RESPONSE=$(jq '.responseList['$i'] .escapeEnclosureEscapeCh
aracter='$ESCAPE_ENCLOSURE_ESCAPE_CHARACTER' <<<"$UPDATED_FILE_METADATA_RESPONSE")
    FILE_METADATA_RESPONSE=$UPDATED_FILE_METADATA_RESPONSE
    FILE_METADATA_OBJECT=$(jq '.responseList['$i']' <<<"$FILE_METADATA_RESPONSE")
    FILE_METADATA_ID=$(jq '.responseList['$i'] .fileMetadataId' <<<"$FILE_METADATA_RE
SPONSE")

    if [[ -z "$FILE_METADATA_ID" ]] || [ "$FILE_METADATA_ID" = "null" ]; then
        break
    else
        if [[ ! -z "$ENCLOSURE" ]] && [ ! "$ENCLOSURE" = "null" ] && [ ! "$ENCLOSURE"
= "" ]; then
            echo "Calling $MASKING_ENGINE/file-metadata/$FILE_METADATA_ID API to
update enclosureEscapeCharacter=$CUSTOM_ENCLOSURE_ESCAPE_CHARACTER and
escapeEnclosureEscapeCharacter=$ESCAPE_ENCLOSURE_ESCAPE_CHARACTER"
            UPDATE_RESPONSE=$(curl $SSL_CERT -X PUT -H ""$AUTH_HEADER"" -H
'Content-Type: application/json' -H 'Accept: application/json' -d ""$FILE_METADATA_O
BJECT"" ""$MASKING_ENGINE/file-metadata/$FILE_METADATA_ID'')
            check_error "$UPDATE_RESPONSE"
        fi
    fi
    ((i++))
done

echo

```

11.3.5 createApplication

```
#!/bin/bash

#
# This script will login and create an application. It depends on helpers in the
# helpers script as well as host and login
# information found in apiHostInfo and loginCredentials, respectively.
#

source apiHostInfo
eval $(cat loginCredentials)
source helpers

login

echo "* creating application 'App123'..."
curl $SSL_CERT -X POST -H ""$AUTH_HEADER"" -H 'Content-Type: application/json' -H
'Accept: application/json' --data @- $MASKING_ENGINE/applications <<EOF
{
  "applicationName": "App123"
}
EOF

echo
```

11.3.6 createEnvironment

```
#!/bin/bash

#
# This script will login and create an environment with an application. It depends on
# helpers in the helpers
# script as well as host and login information found in apiHostInfo and
# loginCredentials, respectively.
#

source apiHostInfo
eval $(cat loginCredentials)
source helpers

login

#
# When deciding which application to place the environment in we simply choose the
# first application found. You are
```



```

# encouraged to modify this to suit your needs. Please see get_application_id in
helpers for more information.
#
get_application_id

echo "* creating environment 'newEnv' in application '$APPLICATION_ID'..."
curl $SSL_CERT -X POST -H '$AUTH_HEADER' -H 'Content-Type: application/json' -H
'Accept: application/json' --data @- $MASKING_ENGINE/environments <<EOF
{
  "environmentName": "newEnv",
  "application": "$APPLICATION_ID",
  "purpose": "MASK"
}
EOF

echo

```

11.3.7 createInventory

```

#!/bin/bash

#
# This script will login and create an environment with an application. It depends on
helpers in the helpers
# script as well as host and login information found in apiHostInfo and
loginCredentials, respectively.
#

source apiHostInfo
eval $(cat loginCredentials)
source helpers

login

#
# When deciding which application to place the environment in we simply choose the
first application found. You are
# encouraged to modify this to suit your needs. Please see get_application_id in
helpers for more information.
#
get_application_id

echo "* creating environment 'newEnv' in application '$APPLICATION_ID'..."
curl $SSL_CERT -X POST -H '$AUTH_HEADER' -H 'Content-Type: application/json' -H
'Accept: application/json' --data @- $MASKING_ENGINE/environments <<EOF
{
  "environmentName": "newEnv",
  "application": "$APPLICATION_ID",
  "purpose": "MASK"
}

```

```

}
EOF

echo

```

11.3.8 create DatabaseConnector

```

#!/bin/bash

#
# This script will login and create a database connector in an environment. It
# depends on helpers in the helpers
# script as well as host and login information found in apiHostInfo and
# loginCredentials, respectively.
#

source apiHostInfo
eval $(cat loginCredentials)
source helpers

login

#
# When deciding which environment to place the connector in we simply choose the
# first environment found. You are
# encouraged to modify this to suit your needs. Please see get_environment_id in
# helpers for more information.
#
get_environment_id

echo "* creating database connector 'connector' in environment '$ENVIRONMENT_ID'..."
curl $SSL_CERT -X POST -H '$AUTH_HEADER' -H 'Content-Type: application/json' -H
'Accept: application/json' --data @- $MASKING_ENGINE/database-connectors <<EOF
{
  "connectorName": "connector",
  "databaseType": "ORACLE",
  "environmentId": $ENVIRONMENT_ID,
  "host": "myHost",
  "password": "myPassword",
  "port": 1234,
  "schemaName": "MYSCHEMA",
  "sid": "mySID",
  "username": "MYUSERNAME"
}
EOF

echo

```

11.3.9 create DatabaseRuleset

```
#!/bin/bash

#
# This script will login and create a database ruleset for a database connector. It
# depends on helpers in the helpers
# script as well as host and login information found in apiHostInfo and
# loginCredentials, respectively.
#

source apiHostInfo
eval $(cat loginCredentials)
source helpers

login

#
# When deciding which database connector we will use, we simply choose the first
# database connector found. You are
# encouraged to modify this to suit your needs. Please see get_connector_id in
# helpers for more information.
#
get_connector_id

echo "* creating database ruleset 'myRuleset' in db connector '$CONNECTOR_ID'..."
curl $SSL_CERT -X POST -H '$AUTH_HEADER' -H 'Content-Type: application/json' -H
'Accept: application/json' --data @- $MASKING_ENGINE/database-rulesets <<EOF
{
  "rulesetName": "myRuleset",
  "databaseConnectorId": $CONNECTOR_ID
}
EOF

echo
```

11.3.10 getBillingUsage

```
#!/bin/bash

#
# This script is an "out of the box" script that goes through
# Login and GET /billing-usage with the authentication
# token from Login
#
```

```

source apiHostInfo
eval $(cat loginCredentials)
source helpers

login

START_DATE=yyyy-mm-dd
END_DATE=yyyy-mm-dd

echo "* GET /billing-usage for date range $START_DATE - $END_DATE from
$EXPORT_ENGINE"
EXPORT_RESPONSE=$(curl $SSL_CERT -X GET -H ""$AUTH_HEADER"" -H 'Accept:
application/json' $MASKING_ENGINE/billing-usage?
start_date=$START_DATE&end_date=$END_DATE) || die "curl failed with exit code $?"
echo $EXPORT_RESPONSE

```

11.3.11 getAuditLogs

```

#!/bin/bash

#
# This script is an "out of the box" script that goes through
# Login and GET /audit-logs with the authentication
# token from Login
#
source apiHostInfo
eval $(cat loginCredentials)
source helpers

login

echo "* GET /audit-logs from $EXPORT_ENGINE"
EXPORT_RESPONSE=$(curl $SSL_CERT -X GET -H ""$AUTH_HEADER"" -H 'Accept:
application/json' $MASKING_ENGINE/audit-logs)

# Calculate the number of audit log entries and the proximity to the entry limit.
AUDIT_ENTRY_COUNT=$(jq '._pageInfo.total' <<<"$EXPORT_RESPONSE")
MAX_ENTRIES=1000000
DIFFERENCE=$((MAX_ENTRIES-AUDIT_ENTRY_COUNT))

# Retrieve the date of the oldest audit entry retained.
OLDEST_DATE=$(jq '.responseList[1].activityTime' <<<"$EXPORT_RESPONSE")

echo "There are $AUDIT_ENTRY_COUNT entries in the audit log. After $DIFFERENCE more
audits you will hit the $MAX_ENTRIES limit and will begin to overwrite entries
starting from the oldest, which was created on: $OLDEST_DATE"

```

11.3.12 getSyncableObjects

```
#!/bin/bash

#
# This script is an "out of the box" script that goes through
# Login and GET /syncable-objects with the authentication
# token from Login
#

source apiHostInfo
eval $(cat loginCredentials)
source helpers

login

echo "* GET /syncable-objects from $EXPORT_ENGINE"
EXPORT_RESPONSE=$(curl $SSL_CERT -X GET -H "$AUTH_HEADER" -H 'Accept:
application/json' $MASKING_ENGINE/syncable-objects)
echo $EXPORT_RESPONSE
```

11.3.13 getSyncableObjectsExport

```
#!/bin/bash

#
# This script will log in and get all syncable objects on
# the Masking Engine and then, given a grouping command, save the
# exported document in a file and export all syncable objects
# in the indicated group
#
# Grouping command:
# algoType: -t <LOOKUP | BINARYLOOKUP | SEGMENT | TOKENIZATION | KEY>
# algoCd: -n <RegexForAlgoName>
#
# Currently the response from GET /syncable-objects is saved
# to getobj_response.json, and the grouped input for /export
# in grouped_export_list.json, and the final export response
# into export_response.json. But of course, this can script
# can be modified to save to other specified places.
#

source apiHostInfo
eval $(cat loginCredentials)
source helpers
```

```

login

echo "* GET /syncable-objects"
GETOBJ_RESPONSE=$(curl $SSL_CERT -X GET -H ""$AUTH_HEADER"" -H 'Content-Type:
application/json' $MASKING_ENGINE/syncable-objects)
echo $GETOBJ_RESPONSE > "./getobj_response.json"

# Create a temporary export list file
GROUPED_EXPORT_LIST="./grouped_export_list.json"
echo "[]" > $GROUPED_EXPORT_LIST

if [[ $1 == "-t" ]]; then
    ALGO_TYPE=$2
    echo "* Filter for all syncable objects of algorithm type $ALGO_TYPE"

    jq -c '.responseList[]' getobj_response.json | while read i; do
        if [[ $(echo $i | jq '.objectType') == \"$ALGO_TYPE\" ]]; then
            # The key to getting the correct json format here was to use
            # the --argjson instead of --arg. --arg will stringify everything
            # and escape all special characters like {, ", etc.
            echo $(cat $GROUPED_EXPORT_LIST | jq --argjson obj "$i" '. |= . + [$obj]') >
$GROUPED_EXPORT_LIST
        fi
    done
elif [[ $1 == "-n" ]]; then
    ALGO_NAME_REGEX=$2
    echo "* Filter for all syncable objects where algorithmCd matches the regex
$ALGO_NAME_REGEX"

    jq -c '.responseList[]' getobj_response.json | while read i; do
        if [[ "$(echo $i | jq '.objectIdentifier.algorithmName')" =~
\"$ALGO_NAME_REGEX\" ]]; then
            echo $(cat $GROUPED_EXPORT_LIST | jq --argjson obj "$i" '. |= . + [$obj]')
> $GROUPED_EXPORT_LIST
        fi
    done
fi

echo "* Export syncable objects from $GROUPED_EXPORT_LIST"
EXPORT_RESPONSE=$(curl $SSL_CERT -X POST -H ""$AUTH_HEADER"" -H 'Content-Type:
application/json' -H 'Accept: application/json' -d "$(<$GROUPED_EXPORT_LIST)"
$MASKING_ENGINE/export)

# Save the grouped export response into a file
echo $EXPORT_RESPONSE > export_response.json
echo '* Completed exporting. Check "export_response.json" for the export document.
This export document json object will be what you literally put in as the input for
import'
```

11.3.14 profileTypeExpressions

11.3.14.1 Add a new type expression

```
#!/bin/bash

#
# This script will login and create a profile type expression. It depends on helpers
# in the helpers script as well as host and login
# information found in apiHostInfo and loginCredentials, respectively.
#

source apiHostInfo
eval $(cat loginCredentials)
source helpers

login

curl $SSL_CERT -X POST -H ""$AUTH_HEADER"" -H 'Content-Type: application/json' -H
'Accept: application/json' --data @- $MASKING_ENGINE/profile-type-expressions <<EOF
{
  "domainName": "FIRST_NAME",
  "expressionName": "FirstNameType",
  "dataType": "String",
  "minDataLength": 5
}
EOF

echo
```

To be effective, a Profile Type Expression has to be part of a profile set. A type expression can be added to a profile set with the profile-sets endpoint. For example, if some Profile Type Expressions were created and have ids 57 and 48, we can use the PUT method on the profile-set endpoint to update an existing profile set so that it includes the new profile type expression. This is shown below, where the profile set has id 42.

```
#!/bin/bash

source apiHostInfo
eval $(cat loginCredentials)
source helpers

login

curl $SSL_CERT -X PUT -H ""$AUTH_HEADER"" -H 'Content-Type: application/json' -H
'Accept: application/json' --data @- $MASKING_ENGINE/profile-sets/42 <<EOF
{
```

```

"profileSetName": "FINDS_ALL_SENSITIVE_DATA",
"profileExpressionIds": [
  4,
  8,
  12,
  13,
  27
],
"profileTypeExpressionIds": [
  57,
  58
]
}
EOF

```

11.3.14.2 Delete a type expression

Deleting a type expression is done using the DELETE method on the profile-type-expression endpoint. The expression must be removed from any profile sets it's a part of before it can be deleted.

```

#!/bin/bash

#
# This script will login and delete a profile type expression. It depends on helpers
# in the helpers script as well as host and login
# information found in apiHostInfo and loginCredentials, respectively.
#

source apiHostInfo
eval $(cat loginCredentials)
source helpers

login

echo "* creating application 'App123'..."
curl $SSL_CERT -X DELETE -H ""$AUTH_HEADER"" -H 'Content-Type: application/json' -H
'Accept: application/json' --data @- $MASKING_ENGINE/profile-type-expressions/57

echo

```

11.3.15 runMaskingJob

This script will login and run a masking job. It depends on helpers in the helpers script as well as host and login information found in apiHostInfo and loginCredentials, respectively.

```

#!/bin/bash
source apiHostInfo
eval $(cat loginCredentials)

```



```
source helpers
```

```
login
```

When deciding which masking job to run, we simply choose the first masking job found. You are encouraged to modify this to suit your needs. Please see `get_masking_job_id` in `helpers` for more information.

```
get_masking_job_id
```

```
echo "* running masking job '$MASKING_JOB_ID'..."
curl $SSL_CERT -X POST -H ""$AUTH_HEADER"" -H 'Content-Type: application/json' -H
'Accept: application/json' --data @- $MASKING_ENGINE/executions <<EOF
{
    "jobId": "$MASKING_JOB_ID"
}
EOF
echo
```

If a masking job is called by a PowerShell hook script, the following command **MUST** be added to the script using the **PowerShell -File** prefix, **file path**;, and the **exit \$LASTEXITCODE** suffix.

```
PowerShell -File C:\Users\HomeFolder\AddUser.ps1; exit $LASTEXITCODE
```

If this is not added then Delphix will not know if the script ran or completed. For more information, please visit this [SQL Server PowerShell Script Error Handling](#)³⁷⁸ documentation.

11.3.16 getDatabaseUsage

```
#!/bin/bash

#
# This script is an "out of the box" script that goes through
# Login and GET /database-usage with the authentication
# token from Login
#

source apiHostInfo
eval $(cat loginCredentials)
source helpers

login

echo "* GET /database-usage from $EXPORT_ENGINE"
EXPORT_RESPONSE=$(curl $SSL_CERT -X GET -H ""$AUTH_HEADER"" -H 'Accept:
application/json' $MASKING_ENGINE/database-usage) || die "curl failed with exit code
$?"
echo $EXPORT_RESPONSE
```

³⁷⁸ <https://delphixdocs.atlassian.net/continuous-data/docs/using-pre-and-post-scripts-with-sql-server-dsources>

12 Authoring extensible plugins

This section covers the following articles:

- [Introduction \(Authoring extensible plugins\)](#) (see page 996)
- [General plugin structure](#) (see page 999)
- [Setting up your development environment](#) (see page 1005)
- [Algorithms \(Authoring extensible plugins\)](#) (see page 1007)
- [Driver supports](#) (see page 1050)
- [Managing plugins using the API client](#) (see page 1065)
- [Installing a plugin onto the Delphix masking engine](#) (see page 1065)
- [Secure plugin deployment](#) (see page 1066)
- [Terminology](#) (see page 1068)

12.1 Introduction (Authoring extensible plugins)

The SDK was formerly referred to as the *Masking Algorithm SDK*, but it is now referred to as the *Masking Extensible SDK*, as of SDK version 1.5.0, as it now allows for the development of different types of extensible plugins. As of Delphix release 6.0.3.0, the Delphix Masking Engine supports the installation of plugins, written in Java, that provide new masking algorithms; and as of 6.0.9.0, driver support plugins. The former feature is referred to as Extensible Algorithms and the latter is referred to as Extensible Driver Supports. This section of the documentation details all aspects of masking algorithm and driver support plugin usage and development. The *Guided Tour* portion of the workflows section for [Extensible Algorithms](#) (see page 1013) and [Extensible Driver Supports](#) (see page 1013) walk the user through the basic process of building a simple plugin and installing it onto the Delphix Masking Engine. Other sections explore in-depth topics such as making algorithms configurable, consuming input files, etc.

This documentation assumes the reader has some familiarity with Java development as well as operation of the Delphix Masking Engine via both the UI and Web API Client. The reader should also understand the security requirements associated with any new algorithms being developed.

12.1.1 Before getting started

This documentation assumes you have a functional Java 8 development environment. Instructions for setting up a basic development environment are [here](#) (see page 1005).

You should also download the [Extensible SDK binary package](#)³⁷⁹ from the Delphix [download site](#)³⁸⁰ and unpack it into a new directory on your development system. This directory - the root of the unpacked archive - will be referred to as ***sdk_root***.

It's helpful to add the binaries directory to your PATH. On a UNIX like system, this command will add the SDK utilities to PATH:

³⁷⁹ <https://download.delphix.com/folder/574/Delphix%20Product%20Releases/Masking%20SDK>

³⁸⁰ <http://download.delphix.com>

```
$ PATH=$PATH:$(pwd)/sdkTools/bin
```

It is presumed that the SDK bin directory is in the user's PATH throughout this documentation.

12.1.2 SDK features

The Extensible SDK provides a number of useful functions that aid development of new algorithms and driver supports for the Delphix Masking Engine. It is available on the Delphix software [download site](#)³⁸¹.

- Creation of empty "skeleton" projects, with build files - the maskScript *init* sub-command
- Creation of empty class files for algorithms and driver supports - the maskScript *generate* sub-command
- Testing of masking algorithms and driver supports without a masking engine
 - The maskApp CLI (*only algorithms*)
 - The maskScript *mask* sub-command (*both algorithms and driver supports*)
- Uploading of plugins to the masking engine - the maskScript *install* sub-command
- Sample algorithms and driver supports that illustrate the usage of key features of the Masking Plugin API

12.1.3 Versions Compatibility

The SDK shares some key elements with the Masking Engine, so in order for the SDK to provide behaviors as close as possible to the Masking Engine, use the SDK version which corresponds to the Masking Engine where you are planning to use the created algorithm(s). The SDK and the Masking Engine use a common Masking API which provides the mechanisms to run the extensible algorithms. Masking algorithms built on the SDK using the latest Masking API will not necessarily run on an older Masking Engine version.

Delphix Release	Masking API*	Extensible SDK*
6.0.3	1.0.0	-
6.0.4	1.1.0	1.0.0
6.0.5	1.1.0	1.1.0
6.0.6	1.2.0	1.2.0
6.0.7	1.3.0	1.3.0
6.0.8	1.4.0	1.4.0

³⁸¹ <http://download.delphix.com>

Delphix Release	Masking API*	Extensible SDK*
6.0.9	1.5.0	1.5.0
6.0.10	1.6.0	1.6.0
6.0.11	1.6.0	1.6.0
6.0.12	1.7.0	1.7.0
6.0.13	1.8.0	1.8.0
6.0.14	1.9.0	1.9.0
6.0.15	1.10.0	1.10.0
6.0.16	1.11.0	1.11.0
6.0.17	1.12.0	1.12.0
7.0	1.13.0	1.13.0
8.0	1.14.0	1.14.0
9.0	1.15.0	1.15.0
10.0	1.16.0	1.16.0
11.0	1.17.0	1.17.0
12.0	1.18.0	1.18.0
13.0	1.19.0	1.19.0
14.0, 15.0	1.20.0	1.20.0
16.0	1.21.0	1.21.0

Delphix Release	Masking API*	Extensible SDK*
17.0+	(same as Masking Engine)	(same as Masking Engine)



Prior to Delphix Release 6.0.9 and SDK release 1.5.0, Masking API was referred to as Algorithm API and Extensible SDK as Algorithm SDK.

Several other sources of information are available to aid in plugin development:

- The README.md file under docs in the Extensible SDK download archive
- The [Masking Plugin API Javadoc](#)³⁸²
- Invoke **maskScript** (located under *sdkTools/bin* in the SDK download) with the -h option for usage help
- Type help at the **maskApp** (also under *sdkTools/bin* in the SDK download) command prompt

12.2 General plugin structure

This section covers the following topics:

- [Introduction \(General plugin structure\)](#) (see page 999)
- [Dependency management](#) (see page 1000)
- [Plugin Metadata](#) (see page 1002)
- [Versioning](#) (see page 1003)

12.2.1 Introduction (General plugin structure)

This section describes the structure of the plugin Java archive (JAR) files used to extend the Continuous Compliance Engine with additional algorithms. This includes the **MaskingAlgorithm** interface that classes providing new algorithm code must implement, and various other metadata present in the plugin JAR required for the plugin to be usable. It also discusses some aspects of build dependencies and common pitfalls involved when adding new 3rd party dependencies.

Plugins for the Masking Engine should be self-contained. This means they should included all Java classes necessary to run, with a few critical exclusions. The Java classes that comprise the Masking Plugin API itself are the exception; these must be excluded from the plugin JAR to ensure that the plugin properly uses the API classes present on the Masking Engine (or SDK during the test process). This is described in more detail in the [dependency management section](#). (see page 1000)

³⁸² <https://maskingdocs.delphix.com/maskingPluginAPIJavadoc/>

12.2.2 Dependency management

A vast assortment of third-party Java libraries are available, expanding the set of ready-to-use functionality well beyond what is already a rich standard library. Plugins for the Delphix Masking engine are able to make use of external libraries, but a number of guidelines should be followed to ensure proper function and compatibility. Note that the plugin classloader uses a plugin-first loading strategy for dependencies.

12.2.2.1 How to properly use and embed external libraries

When using an external library in a plugin for the Delphix Masking engine, consider these guidelines:

- The plugin JAR should contain all external libraries. This is commonly referred to as a "fat JAR". This prevents the plugin code from inadvertently linking with copies of the same library that might happen to be part of the Masking Engine's codebase, leading to potential version conflicts and unpredictable behavior across upgrades.
- **However**, a small set of packages defining the interface between plugins and the Delphix Masking Engine **must not** be embedded in the JAR. It is critical that, for these packages, the plugin code link against the same classes already loaded by the engine. These packages are:
 - com.delphix.masking:masking-algorithm-api
 - com.fasterxml.jackson.core:jackson-annotations
 - com.google.code.findbugs:jsr305
 - junit:junit

If the externally created plugin uses any of the mentioned libraries, the exact same libraries versions should be used by the plugin author, as the ones used by the SDK. The way to find those versions is:

```
- in the installed SDK find the following gradle file under `samples` directory:
  * gradle.properties
```

It contains the versions of the SDK provided external libraries, for example:

```
googleGuavaVer=28.0-jre
maskingAlgoVer=1.3.0
jacksonVer=2.9.5
junitVer=4.12
```

Looking to those versions author should decide what version of corresponding library to use (if it is required by their design).

- Plugins consuming third-party libraries should be thoroughly tested, as it is not uncommon that library code will attempt to use permissions not granted by the plugin sandbox. If this is the case, there is currently no way to modify the constraints under which the plugin code is executed.
- The plugin author, not *Delphix*, is responsible for ensuring that any license files or other forms of attribution required by any embedded software are handled properly.

- The entity deploying the plugin, not *Delphix*, is responsible for ensuring the organization operating the Masking Engine has obtained the necessary licenses or rights to use any embedded software.

12.2.2.2 Example build file

The following fragments, derived from the sample algorithm *build.gradle* file, illustrate how to correctly build a plugin using the gradle build system:

```

jar {
    from {
        configurations.runtimeClasspath.collect { it.isDirectory() ? it :
zipTree(it) }
    }
    includeEmptyDirs = false

    manifest {
        attributes(
            (PluginMetadata.PLUGIN_NAME_KEY)          : "SampleAlgorithms",
            (PluginMetadata.AUTHOR_NAME_KEY)         : "Sample Author",
            (PluginMetadata.PLUGIN_VERSION_KEY)      : "1.0.0 ${getGitHash}",
            (PluginMetadata.ALGORITHM_API_VERSION_KEY): maskingAlgoVer,
            'Build-Timestamp': new java.text.SimpleDateFormat("yyyy-MM-
dd'T'HH:mm:ss.SSSZ").format(new Date()),
            'Created-By'      : "Gradle ${gradle.gradleVersion}",
            'Build-Jdk'       : "${System.properties['java.version']} ($
{System.properties['java.vendor']} ${System.properties['java.vm.version']})",
            'Build-OS'        : "${System.properties['os.name']} $
{System.properties['os.arch']} ${System.properties['os.version']}",
        )
    }
}

dependencies {
    compileOnly ('com.google.code.findbugs:jsr305:3.0.2')
    compileOnly ('com.delphix.masking:masking-algorithm-api:' + maskingAlgoVer)
    compileOnly ('com.fasterxml.jackson.core:jackson-annotations:' + jacksonVer)

    compile 'com.google.guava:guava:' + googleGuavaVer

    testImplementation 'com.google.code.findbugs:jsr305:3.0.2'
    testImplementation 'com.delphix.masking:masking-algorithm-api:' + maskingAlgoVer
    testImplementation 'com.fasterxml.jackson.core:jackson-annotations:' + jacksonVer
    testImplementation 'junit:junit:' + junitVer
    testImplementation "com.google.truth:truth:" + googleTruthVer
}

```

How this works:

- The "from { ... }" property of the **jar** section instructs gradle to included all classed need at runtime in the plugin JAR file.

- In the **dependencies** section, packages comprising the interface between the plugin and Masking Engine are listed as *compileOnly*. This excludes them from the runtime environment and causes them to be omitted from the plugin JAR file.
- The third-party code dependency on the popular Google Guava library is listed as *compile*, causing it to be included in the plugin JAR file.



Variables defining the package dependency versions are typically read from the *gradle.properties* file.

12.2.2.3 Using a shadow JAR to resolve dependency issues

This configuration adds the shadow JAR, which helps to relocate dependencies, avoiding potential conflicts. Below is an example of how to configure it in your build.gradle file. In this example, the `org.dom4j` package is relocated to `shadow.org.dom4j`.

```
buildscript {
    repositories {
        gradlePluginPortal()
    }

    dependencies {
        classpath ("com.github.jengelman.gradle.plugins:shadow:5.2.0")
    }
}

plugins {
    id 'com.github.johnrengelman.shadow' version '5.2.0'
    id 'java'
}

shadowJar {
    relocate 'org.dom4j', 'shadow.org.dom4j'
}
```

12.2.3 Plugin Metadata

When a plugin is built for use with the Masking Engine, it is critical that certain metadata be included in the plugin JAR file. This includes certain attributes in the JAR's manifest, as well as the service discovery file used to determine which classes in the JAR are directly usable by the Extensibility Framework.

12.2.3.1 Manifest Attributes

Java archives carry metadata attributes in the manifest file, located at *META-INF/MANIFEST.MF* in the archive file. Some of these attributes are required or at minimum quite useful in a plugin's manifest.

The following attributes carry special meaning to the extensibility framework when present in a plugin's manifest. Care must be taken to ensure they are set to valid and meaningful values for any plugins intended for production use. Additional attributes may be supported in the future. Any future attributes introduced for anything beyond a purely informational purpose will be of the format "Delphix-*" to avoid conflict with any preexisting usage.

12.2.3.1.1 Recognized Manifest Attributes

Attribute	Meaning	Example Value
Delphix-Plugin-Name	The default name of the plugin. This name will be used on the Masking Engine unless overridden at plugin install time. All plugin names beginning with the string "dlpx" are reserved for future use by modules delivered with the Delphix Masking Engine product.	SamplePlugin
Implementation-Vendor	The individual or organization that authored the plugin module.	Sample Inc.
Implementation-Version	The version of the plugin. This is an entirely free-form string, limited to 255 characters.	1.0.0-SNAPSHOT
Delphix-Algorithm-API-Version	The version of the Delphix Masking Plugin API used by the plugin. This value must be present and represent a valid API version.	1.0.0



The maskScript *init* sub-command adds logic to the gradle build files to ensure that the project build inserts the correct attribute values into the plugin manifest.

12.2.4 Versioning

In order to ensure compatibility between each software component in the extensible algorithms system, careful use of versioning must be enforced. The goals are twofold; first, to ensure that the version of each software module is visible, and second, to ensure that incompatible plugins are detected and prevented from

running. The software modules particular to extensible algorithms - the Masking Plugin API and Masking Algorithm SDK, use a version number in the following format: Major.Minor.Micro, with an option *-TEXT* notation.

Version information for the plugin, including the plugin version and the version of the Masking Plugin API it was built against, are embedded in the plugin JAR file as [metadata](#) (see [page 1002](#)).

12.2.4.1 Table of Versioned Objects

The following table explains how each software module is versioned, and what enforcement takes place:

Software Module	Example Values	Details
Delphix Masking Engine	6.0.3.0	None, however the Masking Algorithm API version is fixed for each particular Delphix Masking Engine release.
Masking Plugin API	1.0.0	The version of the Masking Plugin API used to build a plugin must be embedded in each Plugin. This is done automatically when plugins are built using the Masking Algorithm SDK. Currently, as only one version of the Masking Plugin API exists, the only enforcement in place ensures that a valid version has been embedded in the plugin metadata. In the future, should it be necessary to make backward incompatible changes to the Masking Plugin API, a support matrix will be established and enforced.
Masking Algorithm SDK	1.0.0	Each version of the Masking Algorithm SDK will have a maximum version of the Masking Plugin API it can handle, and will refuse to work with future versions. This is not currently enforced.
Plugins	Author defined	No enforcement. The plugin version will be visible using the Masking API GET operation on the <i>Plugin</i> endpoint.

12.2.4.2 Ensuring Plugin Compatibility

As the implementation of a plugin evolves, it's natural for the software to change. Changes like bug fixes, performance improvements, etc. can typically be made transparently, without endangering backward compatibility. Challenges can arise when it is necessary to change the configuration schema for a framework within a plugin or remove certain algorithm frameworks or instances from a plugin. This section will detail some best practices for maximizing backward compatibility in each case.

In this case, backward compatibility means that it is possible to upgrade the plugin to a new version on the masking engine without removing the previous version of the plugin. While it is always possible to replace a plugin by removing the old plugin and installing a new one, this requires manual reconstruction of any inventory that references the algorithms based on the old plugin, which can be very labor-intensive.

12.2.4.2.1 Schema Changes

A schema change means altering the set of configuration parameters exposed by an algorithm framework. For example, this might be done to add a case-insensitive flag to an algorithm that processes Strings. In order to make this kind of change while preserving backward compatibility, the following rules must be followed:

- Existing configurable variables cannot be removed or modified. These are the class fields with to which the `@JsonProperty` annotation has been applied.
- New configurable variables may be added, but they must have a default value, so that applying a JSON document lacking a value for the new field results in a valid instance.

If changes must be made that do not meet these requirements, it may be preferable to expose the new or modified functionality as a new algorithm framework, rather than changing the existing one.

12.2.4.2.2 Component Removal

Component removal means removing an algorithm framework or a built-in instance provided by an existing framework. When this happens, updating to the new version of the plugin will be blocked if any of the removed objects are in use by the Delphix Masking Engine. This includes references in Inventory, Domains, and any File Formats. In addition, the presence of any user-created algorithms based on a framework which is removed in the new plugin version will block updating.

12.2.4.3 Plugin Naming

One last compatibility concern is the potential for a plugin name to clash with the name of plugins delivered by the Delphix Masking Engine product. Such a clash would make it impossible to upgrade the engine to a new version without first removing the conflicting user installed plugin. To avoid this concern, avoid embedding any default plugin name beginning with the string "dlpx".

12.3 Setting up your development environment

This section describes the step-by-step process for setting up the development environment that was used to develop and test many of the procedures in this Guided Tour. A rich set of tools exist to support Java development, so this is by no mean the only development environment possible.



As of version 6.0.3.0, the Delphix Masking Engine's JVM version is 8, so it's important to build a plugin compatible with Java 8.

12.3.1 Downloading and installing tools

- Download and install the Oracle [Java JDK](#)³⁸³ or [OpenJDK](#)³⁸⁴ from the AdoptOpenJDK Project. Make sure you install **Java 8** version.
- Download and install [IntelliJ IDEA Community Edition](#)³⁸⁵ for your OS. These instructions are known to work for version 2019.3.

12.3.2 Creating a new project

Identify the root directory of your project code. For example, if you used the instructions to create a new [algorithm](#) (see page 1014) or [driver support](#) (see page 1053) project, this directory is referred to as **proj_dir**.

Start IntelliJ IDEA. A pop-up should appear.

1. Select **Project Settings > Project > Project SDK > New > JDK**
2. Provide the path to the JDK you installed earlier (e.g. /Library/Java/JavaVirtualMachines/jdk1.8.0_60.jdk/Contents/Home). Select **OK**.
3. For **Project language level**, set to "8 - Lambda, type annotations etc."
4. In the IntelliJ IDE, select **Import Project** from the pop-up.
5. Provide the path to the root of your project, for example, **proj_dir**.
6. Select **Import project from external model** and **Gradle**, and then select **Next**.
7. After the project is imported, a directory tree is shown on the left panel.

At this point, the development environment should be ready to use.

12.3.3 Enabling remote debugging

It is often useful to enable remote debugging, which allows the IDEs debugger to attach to a running **maskApp** or **maskScript** process. To enable remote debugging, certain environment variables must be set. In both cases, the value `5005` can be replaced with the value of any open TCP port.

For **maskApp**

```
MASK_APP_OPTS='-Xdebug -Xnoagent -Djava.compiler=NONE
-Xrunjdpw:transport=dt_socket,server=y,suspend=n,address=5005'
```

For **maskScript**

383 <https://www.oracle.com/technetwork/java/javase/downloads/index.html>

384 <https://adoptopenjdk.net/?variant=openjdk8&jvmVariant=hotspot>

385 <https://www.jetbrains.com/idea/download/>

```
MASK_SCRIPT_OPTS='-Xdebug -Xnoagent -Djava.compiler=NONE
-Xrunjdwp:transport=dt_socket,server=y,suspend=y,address=5005'
```



These settings will cause the maskScript to suspend at startup to allow time to attach the debugger.

12.4 Algorithms (Authoring extensible plugins)

As of release 6.0.3.0, the Continuous Compliance Engine supports the installation of plugins, written in Java, that provide new masking algorithms. This feature is referred to as Extensible Algorithms. This section of the documentation details all aspects of masking algorithm plugin usage and development. The *Guided Tour* portion of the [workflows section](#) (see page 1013) walks the user through the basic process of building a simple plugin and installing it onto the Continuous Compliance Engine. Other sections explore in-depth topics such as making algorithms configurable, consuming input files, etc.

This documentation assumes the reader has some familiarity with Java development as well as operation of the Delphix Masking Engine via both the UI and Web API Client. The reader should also understand the security requirements associated with any new algorithms being developed.

The Extensible Algorithms framework is designed to replace the custom algorithm (aka. mapplets) feature by providing richer functionality, greatly simplifying algorithm development, and ensuring long-term maintainability of plugins. The end-of-support of custom algorithms will occur in release 6.0.15.0 of the Continuous Compliance Engine.

12.4.1 SDK Features

The Masking Algorithm SDK provides a number of useful functions that aid development of new algorithms for the Continuous Compliance Engine. It is available on the Delphix software [download site](#)³⁸⁶.

- Creation of empty "skeleton" projects, with build files - the maskScript *init* sub-command
- Creation of empty class files for algorithms - the maskScript *generate* sub-command
- Testing of masking algorithms without a masking engine
 - The maskApp CLI
 - The maskScript *mask* sub-command
- Uploading of plugins to the masking engine - the maskScript *install* sub-command
- Sample algorithms that illustrate the usage of key features of the Masking Plugin API

12.4.2 Getting more information

Several other sources of information are available to aid in plugin development:

³⁸⁶ <http://download.delphix.com/>

- The README.md file under docs in the Algorithm SDK download archive
- The [Masking Plugin API Javadoc](#)³⁸⁷
- Invoke **maskScript** (located under *sdkTools/bin* in the SDK download) with the -h option for usage help
- Type help at the **maskApp** (also under *sdkTools/bin* in the SDK download) command prompt

12.4.3 The MaskingAlgorithm Java interface

Any Java class that should be recognized as a masking algorithm (whether standalone or configurable) must implement the **MaskingAlgorithm** interface. This interface is parameterized with the data type that the algorithm masks, which defines the input and output data type of the **mask** method. The full details of this interface are described in the [masking plugin API Java](#)³⁸⁸ document.

12.4.3.1 Core data types

The Delphix Continuous Compliance Engine is designed to support a wide and extensible set of data sources, which naturally encode data in a variety of different formats. In order to simplify algorithm development while maintaining the ability to mask data from many sources, a core set of data formats have been identified to likely require different masking treatment, and the Extensible Algorithm framework ensures that all data is converted to and from these types (as needed). These types define the allowed parameterization of the **MaskingAlgorithm** Java interface.

Each masking algorithm class is defined to mask exactly one of the following data types:

- Binary data - **java.nio.ByteBuffer**
- String data - **java.lang.String**
- Numeric data - **java.math.BigDecimal**
- Date time data - **java.time.LocalDateTime**
- Multi-column data - **com.delphix.masking.api.plugin.utils.GenericDataRow** (See Multi-Column Masking section)

Each algorithm is expected to input, process, and emit objects of one of the above Java types, but is free to use any intermediate types (as needed) to access library methods. Since that is frequently the case, that data of one type is stored in databases or documents in a type other than its most natural native type (ex. dates stored in VARCHAR fields, or numbers stored as text in a CSV file), the masking framework that executes these algorithms is capable of performing a number of automatic type conversions, detailed in the next section. This allows algorithms written to process one data type to handle data of other types, with no additional work required of the algorithm author.

Supported automatic type conversions

Algorithm native type	Supported type	Notes
ByteBuffer	String	Algorithm receives the UTF-8 encoded value of the String and is expected to return a valid UTF-8 ByteBuffer.

³⁸⁷ <https://maskingdocs.delphix.com/maskingPluginAPIJavadoc/>

³⁸⁸ <https://maskingdocs.delphix.com/maskingPluginAPIJavadoc/>

Algorithm native type	Supported type	Notes
LocalDateTime	String	The correct date format must be assigned to the field or column in the masking inventory.
LocalDateTime	Compatible numeric types	A compatible date format, such as <i>yyyyMMdd</i> , must be assigned to the column in inventory.
BigDecimal	All numeric types	Unconverted to BigDecimal. Values out of range (after masking) are truncated to fit the range of the underlying type.
BigDecimal	String	String value is converted to a number.

12.4.3.2 Special case values

In order to allow algorithms to implement special handling for null, empty, and special case values, these values are presented to the masking algorithm unmodified. Algorithms should be prepared to process the full range of input values possible for the input type. In practice, this means that most **mask** method implementations will begin with a null check on the *input* value prior to attempting to use the input – for example, calling `input.length()` or similar. It is perfectly acceptable and common to return null in the case where the mask input is null.

12.4.3.3 Method overview

This section provides a high-level overview of the methods in the **MaskingAlgorithm** interface. For complete details, consult the [masking plugin API Java](https://maskingdocs.delphix.com/maskingPluginAPIJavadoc/)³⁸⁹ document included in the Algorithm SDK archive.

- *getName* and *getDescription* - These methods are used to determine the name and description of frameworks and algorithm instances included in the plugin. For user-created instances, these methods are never called.
- *getDefaultInstances* and *getAllowFurtherInstances* - These methods control the set of instances of the algorithm framework that are defined by the plugin, and whether the user should be allowed to create additional instances.
- *validate* - This method is called after configuration is applied to allow the algorithm class to check whether the injected configuration is valid.
- *setup* and *tearDown* - These methods are called before the algorithm object is used for masking, and after, respectively. Typically, any resources, such as input files, are acquired during *setup* and released during *tearDown*.

³⁸⁹ <https://maskingdocs.delphix.com/maskingPluginAPIJavadoc/>

- *mask* - This is the method that does the actual data masking in the algorithm class. The input and output values are parameterized for type safety as described above
- *maskBatch* - This method is called to perform masking in situations when it is possible for the caller to build a collection of input values to mask in a single method call. A default implementation is provided that simply calls the *mask* method on each value in the batch.
- *listMultiColumnFields* - This method needs to be implemented only for Multi-Column Algorithms. It returns a list of **AlgorithmLogicalField** objects that define the set of fields that the multi-column algorithm masks.

The following methods are available but deprecated:

- *listMaskedFields* - This method needs to be implemented for Multi-Column Algorithms. It returns a map of field names (`String`) to the Core Data Type. This method does not need to be implemented if not implementing a Multi-Column Algorithm. Implement *listMultiColumnFields* instead.
- *listReadOnlyFields* - Similar to `listMaskedFields` but optional for Multi-Column Algorithms. Fields returned by this method are read-only and cannot be changed. Implement *listMultiColumnFields* instead.

12.4.3.4 The life cycles of algorithm objects

The Extensibility framework uses objects classes implementing **MaskingAlgorithm** interface for several distinct purposes. These object life cycles are as follows:

12.4.3.4.1 Plugin discovery

This occurs when the extensibility framework evaluates the capabilities present in a **MaskingAlgorithm** class.

1. Java object creation - an object of the algorithm class is created
2. *getName* - determines framework name
3. *getDescription* - determines framework description
4. *getDefaultInstances*- determines all plugin-provided algorithm instances. For each instance:
 - a. *getName* - determines instance name
 - b. *getDescription* - determines instance description
 - c. *validate* - ensure object passes validation
 - d. Serialize configurable fields - these are saved as a JSON document defining the instance's configuration
 - e. Disposal - the Java object is discarded
5. *getAllowFurtherInstances* - determines whether the framework is visible in the *algorithm/framework* API endpoint
6. Disposal - the Java object is discarded

12.4.3.4.2 User algorithm creation

This life cycle occurs whenever a user attempts to create a new instance of a plugin algorithm framework. The algorithm definition is saved only if each step succeeds.

1. Java object creation - an object of the algorithm class is created
2. Configuration injection - the values in the user-provided JSON document are injected into the object
3. *validate* - the object's *validate* method is called
4. Disposal - the Java object is discarded



The *setup* method is not executed when a user-defined instance is created.

12.4.3.4.3 Algorithm use

This is the life cycle of an algorithm object when used to mask data.

1. Java object creation - an object of the algorithm class is created
2. Configuration injection - the saved JSON document defining this instance is injected in the object
3. *setup* - the *setup* method is called once
4. *mask* - the *mask* method is called on each value to be masked
5. *tearDown* - the *tearDown* method is called once
6. Disposal - the Java object is discarded



It should be noted that a distinct Java object is created for each application of a masking algorithm during Job execution. For algorithms that create or load a large amount of state, this can result in significant memory usage storing redundant data for each instance. This can be avoided using a class level static cache to store data; the instance name, which can be retrieved during *setup* from the **ComponentService** interface object, can be used as an access key for data cached in this way.

12.4.3.5 Multi-column masking

It is possible to write an algorithm that masks data that depends on other column(s) values. In order to account for the different possible data types, we use an object called a `GenericDataRow`.

12.4.3.5.1 Generic data

A `GenericDataRow` is a map of field names (`String`) to `GenericData` objects. Each `GenericData` object contains the value, along with methods to return the respective typed object. When accessing the value from a `GenericDataObject` it will be necessary to read it into a Core Data Type. To do so, use one of the following methods:

- `getStringValue()`
- `getBigDecimalValue()`
- `getLocalDateTimeValue()`
- `getByteBufferValue()`

Once the value has been masked it should be re-set by calling `setValue` and passing as an argument the value as a Core Data Type.

12.4.3.6 Batch masking

Batch masking is a feature that can improve algorithm performance significantly when high latency operations are employed as part of the masking process. Accessing an external resource like a database or API introduces significant execution latency compared executing Java code; batching incurs only a single round-trip latency while masking many values. Batching also allows the interchange of values between data rows during masking.

12.4.3.6.1 Batch masking support in jobs

Batching is currently supported for these job types:

- All Database masking jobs
- Delimited File jobs
- Fixed-Width File jobs

Batch size is equal to the job's `Row Limit` divided by 5, or equal to 2000 when the `Row Limit` is disabled; this is the guaranteed lower bound for batch size, assuming at least that number of inputs are available and no conditional record types are present. The final batch when processing a table or file may be up to twice the normal batch size.



For file jobs, the presence of conditional record types will cause batch sizes to be unpredictable, as the availability of records for batch execution will naturally vary based on how many records actually match the criteria for each record type. Algorithms that require a minimum batch size, such as Secure Shuffle, may fail in this case.

12.4.3.6.2 Using Batch Masking in an algorithm implementation

An algorithm implementation can customize how batches of values are masked by overriding the *maskBatch* method in the `MaskingAlgorithm` interface. There is no reason to implement this method unless there is a benefit to processing multiple values in a single operation. A common example of this is when the algorithm is accessing an external API to perform masking; in this case, masking multiple inputs per method call allows the access latency of the API to be incurred only once for the entire batch of inputs.

The *maskBatch* method is called with a `MaskingBatch` object parameterized by the same Java type used in the `MaskingAlgorithm` interface definition. The `MaskingBatch` object provides the following methods to facilitate masking:

- *size* - returns the size of the batch of values
- *getValue* - returns the value to be masked at a particular index in the batch
- *setValue* - sets the mask result at a particular index in the batch
- *setError* - indicates that an error occurred when masking the input value at a particular index in the batch

The default implementation of *maskBatch* in the `MaskingAlgorithm` interface provides a simple example of how to use these methods.



The masking engine will not utilize the *maskBatch* method or create a batch with size greater than 1 in all cases. Batch masking is only supported for some job configurations, so it is critical that the *mask* method also be implemented for all algorithms. It is strongly recommended that the *mask* and *maskBatch* method be implemented to produce the same mask results given the same inputs.

12.4.4 SDK Workflows (Algorithms)

This section is intended to walk a developer through several workflows using the Delphix Algorithm SDK, such as creating a new algorithm plugin and installing it on a Continuous Compliance Engine. Once an algorithm plugin has been installed, the included algorithms function as expected; they may be assigned to domains and inventory in the normal fashion.

In order to develop and deploy algorithm plugins, you will interact primarily with two tools - the Masking API client, and the Masking Algorithm SDK. The Masking API client is a long-standing feature that allows interactive execution of API operations on the Continuous Compliance Engine, while the Masking Algorithm SDK is a new software package created specifically to aid in algorithm development.

12.4.4.1 Outline for a guided tour

By following the steps in the outline below, you can tour the basic functionality provided by the Extensible Algorithm feature and Algorithm SDK.

1. Create an algorithm plugin by choosing one of two options:
 - a. [Building the sample algorithm project](#) (see page 1014)
 - b. [Creating and building your own algorithm project](#) (see page 1014)
2. [Run the algorithm plugin using maskApp](#) (see page 1017)
3. [Install the newly created plugin on the Continuous Compliance Engine](#) (see page 1065)
4. [View and manage the plugins on a Continuous Compliance Engine using the API Client](#) (see page 1065)
5. [Upload multiple plugin in SDK](#) (see page 1020)

12.4.4.2 Building the sample plugin (SDK workflows/Algorithms)

The Algorithm SDK contains a buildable Sample Algorithm Plugin with a number of functional algorithms illustrating the features of the Extensibility Framework. These simple commands build the plugin containing the sample algorithms.

Starting from **sdk_root**:

```
$ cd samples
$ ./gradlew :algorithm:jar
```

This creates the Sample Algorithm plugin JAR file **sdk_root/samples/build/libs/algorithm.jar**.

The Sample Algorithm project provides a convenient way to see a working example plugin.

i While it is possible to modify these algorithms by changing the Java source and rebuilding the plugin, when starting a new project to develop one or more standalone algorithms, it is highly recommended that you [create your own project](#) (see page 1014) rather than modifying files in the Sample Algorithm project subtree. This will prevent the loss of customizations to the project build files should you chose to install a new version of Masking Algorithm SDK over your existing SDK directory.

12.4.4.3 Creating a New Project (SDK workflows/Algorithms)

This section describes how to create a brand new Java project for a new masking algorithm plugin. We will use the maskScript utility to create a skeleton project and an empty algorithm class in that project.

12.4.4.3.1 Creating the Project

Before you begin, you'll want to pick a name for your project, and an **empty** directory (outside of the Masking SDK source tree) where your project will be created. Once you've done this, run this **maskScript** command:

```
$ maskScript init -t algorithm -d <project path> -n <project name> -a <author name>
-v <version>
```

For example, this command will create a project named *demoProject* in the *demo-proj* subdirectory of your home directory.

```
$ maskScript init -d $HOME/demo-proj -n demoProject -a "Demo Author" -v 1.0.0
```

For the rest of this section, we'll assume a new project has been created under **proj_dir**. Change your working directory to **proj_dir**. You'll notice that the project is created with a sample algorithm file **proj_dir/src/main/java/com/sample/SampleAlgorithm.java**. It's possible to build this into a usable plugin by running:

```
$ cd <proj_dir>
$ ./gradlew jar
```

But let's create our own, brand new algorithm.

12.4.4.3.2 Creating an Algorithm Class

For this part of the tour, we're going to create a new algorithm named Clobber. First, we'll run the maskScript utility to create a skeleton class file:

```
$ cd <proj_dir>
$ maskScript generate algorithm -p com.delphix.demo -c Clobber -v String -s .
```

By convention, the class file Clobber.java will be created under a sub-directory path based on the package name, so it might be helpful to use the find command to locate it:

```
$ find . -name Clobber.java
./src/main/java/com/delphix/demo/Clobber.java
```

The initial content of this file is:

```
$ cat ./src/main/java/com/delphix/demo/Clobber.java

package com.delphix.demo;

import com.delphix.masking.api.plugin.MaskingAlgorithm;
import java.lang.String;
import javax.annotation.Nullable;

public class Clobber implements MaskingAlgorithm<String> {

    /**
     * Masks String object
```

```

    * @param input The String object to be masked. This method should handle null
    inputs.
    * @return Returns the masked value.
    */
    @Override
    public String mask(@Nullable String input) {
        // TODO: change the default implementation.
        return input;
    }

    /**
     * Get the recommended name of this Algorithm.
     * @return The name of this algorithm
     */
    @Override
    public String getName() {
        // TODO: Change this if you'd like to name your algorithm differently from the
        Java class.
        return "Clobber";
    }
}

```

12.4.4.3.3 Customizing the Algorithm Class

The first thing to notice about the skeleton algorithm is that the mask method just returns the input. This means no masking will be done, so this will certainly need to change. We're going to create an algorithm that overwrites the entire input String with the first letter of that String. This replaces the skeleton *mask* method with:

```

@Override
    public String mask(@Nullable String input) {
        // Always be ready to handle null or empty input
        if (input == null || input.length() < 2) {
            return input;
        }

        char firstChar = input.charAt(0);
        StringBuilder result = new StringBuilder();

        for (int i = 0; i < input.length(); i++) {
            result.append(firstChar);
        }

        return result.toString();
    }
}

```

The algorithm name "Clobber" is fine, so we can just delete the TODO comment in the getName() method. Now, we'll rebuild the project to include this new algorithm in the plugin JAR:

```
$ ./gradlew jar
```

This creates or updates the plugin JAR file *proj_dir*/build/libs/demoProject.jar

12.4.4.4 Service discovery (SDK workflows/Algorithms)

Java service discovery is used to determine which classes in the plugin JAR present relevant functionality to the Delphix Masking Engine. When a plugin is loaded, the file *com.delphix.masking.api.plugin.MaskingComponent* under *META-INF/services* in the JAR is consulted for a list of classes that implement the **MaskingComponent** interface. As **MaskingAlgorithm** includes this interface, each algorithm in the plugin will be discovered this way. In the future, this mechanism may be expanded to support additional types of components beyond algorithms.



When the `maskScript generate` sub-command is used to create a new algorithm class, the service discovery metadata file is automatically updated.

If an algorithm class is missing from the services file, it will not be usable when the plugin is loaded. It is essentially invisible to the extensibility framework. If a class is mentioned in this file but not present in the JAR, the plugin will fail to load. There is a fallback during plugin loading that will scan the entire JAR for algorithms if the services file is not present. This fallback may be removed in the future and should not be relied on.

12.4.4.5 Running an Algorithm using the SDK tools (SDK workflows/Algorithms)

It will often be more convenient to use the SDK utilities to test an algorithm since this avoids the need to install or update your plugin, create masking inventory, and execute jobs on the Continuous Compliance Engine. This can be done interactively using `maskApp`, or entirely from the command line using `maskScript`.

12.4.4.5.1 Using `maskApp` to Test an Algorithm

The `maskApp` application is interactive and may be launched with no parameters from the shell:

```
$ maskApp
```

After a moment, the application will print a banner and prompt for input. Currently, the only sub-command supported is `mask`. In order to use this command, you'll need to know the location of the plugin JAR file on the filesystem.

```
MASKING-APP:> mask -j <plugin_jar_location>
```

You will be prompted to select an algorithm. This example runs `maskApp` in *proj_dir* and uses the JAR file created with the [Create a new project](#) (see page 1014) workflow:

```

MASKING-APP:> mask -j build/libs/demoProject.jar
/Users/*****/demo-proj/build/libs/demoProject.jar
Loaded plugin demoProject version 1.0.0 (API version: 1.0.0) from [/Users/*****/
demo-proj/build/libs/demoProject.jar]
13:17:00.707 [main] INFO  global - Loaded plugin demoProject: Plugin {'embeddedName':
'demoProject', 'version': '1.0.0', 'author': 'Delphix Dev', 'apiVersion': '1.0.0'}
Framework:
* [0] Clobber
  [1] SampleAlgorithm
Select an algorithm framework: 0
Instance:
* [0] demoProject:Clobber
Select an instance of algorithm framework: Clobber: 0
Selected algorithm: com.delphix.demo.Clobber(Clobber) instance: demoProject:Clobber,
data type: STRING
Input value to be masked('null' for null, 'doneMasking' to finish): Test1
Masked value: TTTT
Input value to be masked('null' for null, 'doneMasking' to finish): test2
Masked value: tttt
Input value to be masked('null' for null, 'doneMasking' to finish): 1 more test
Masked value: 1111111111
Input value to be masked('null' for null, 'doneMasking' to finish):

```

When you invoke the *mask* sub-command, you will first be presented with a list of possible frameworks (aka. Algorithm Components) to choose from. These correspond with the Java classes that implement the **MaskingAlgorithm** interface in the plugin file. Once you have selected a framework, you will be presented with a list of each pre-defined instance to choose from. If the algorithm supports creating new instances, that option will be present as well. Once an instance is selected, you're ready to enter test values and see the masked result.



In order to be usable, each class that implements **MaskingComponent** must also be listed in the appropriate service description file. Refer to [this section \(see page 1017\)](#) for details.

The selection marked with a star is the default selection; you may always press return at the prompt to make the default selection.



When a Multi-Column algorithm is selected, the prompt will contain the order to provide the input values in. They should be placed on a single row, separated by a comma.

```
Enter CSV-formatted values for the following columns in the following
order: date1(LOCAL_DATE_TIME), date2(LOCAL_DATE_TIME)
```


Missing Algorithms

If an algorithm seems to be missing from the list, or an algorithm's behavior does not seem to match the latest version of the code, it may be necessary to rebuild the plugin JAR:

```
$ cd ; ./gradlew="";>
```

12.4.4.5.2 Running an Algorithm using maskScript

Algorithms may be also be tested using the SDK **maskScript**. The **maskScript** utility is non-interactive, which lets you conveniently process input files using a masking algorithm. Each input line is considered a separate value to be masked. The algorithm framework and instance are selected using command-line options. This example uses the "Redaction X" algorithm instance from the Sample Algorithm plugin. This plugin can be built using the process described [here \(see page 1014\)](#).

Create a small sample input file:

```
$ echo "Adam
Amy
Brandon" > test_input.txt
```

Mask each line of the file to standard output:

```
$ cat test_input.txt | maskScript mask -j algorithm/build/libs/algorithm.jar -n
"Sample Plugin:Redaction Z"
Jun 19, 2020 1:51:54 PM com.delphix.masking.api.provider.LogService info
INFO: Loaded plugin Sample Plugin: Plugin {'embeddedName': 'Sample Plugin', 'version':
'1.0.0', 'author': 'Delphix', 'apiVersion': '1.0.0'}
ZZZZ
ZZZ
ZZZZZZZ
Jun 19, 2020 1:51:54 PM com.delphix.masking.api.provider.LogService info
INFO: StringRedaction: Masked a total of 3 values
```

- When masking using a Multi-Column algorithm, the inputs in the file must be provided in the same format they would be provided when using the `mask` subcommand of the `maskApp` (i.e.: comma-separated list of expected values). If unsure of the order, use the `mask` subcommand in the `maskApp` to see what the expected order is.

redirecting Information Messages

To remove all informational message from the output, redirect standard error to /dev/null or an alternate location:

```
$ cat test_input.txt | maskScript mask -j algorithm/build/libs/algorithm.jar -n "Sample Plugin:Redaction Z" 2>/dev/null
```

12.4.4.6 Installing multiple plugins onto the Delphix Masking engine (SDK workflows/Algorithms)

Starting SDK version 1.4.0 (corresponding to the Masking Engine version 6.0.8.0) Delphix has implemented multiple plugins upload within SDK, where the Delphix provided dlpX-core plugin is uploaded by default. That gives an option to chain multiple extensible algorithms, even if those are based different plugins.

12.4.4.6.1 Load multiple algorithm plugins

1. Using maskApp to Test an Algorithm The **maskApp** application is interactive and may be launched with no parameters from the shell: `$ maskApp` After a moment, the application will print a banner and prompt for input. Currently, the only sub-command supported is `mask`. In order to use this command, you'll need to know the location of the plugin JAR file on the filesystem.
2. Upload a desired algorithm plugin `bash MASKING-APP:> mask -j <>` This example runs *maskApp* in **proj_dir** and uses the JAR file created with the [Create a new project \(see page 1014\)](#) workflow:

```
MASKING-APP:> mask -j ./algorithm/build/libs/plugin1.jar
/Users/testuser/delphix/masking-algorithm-sdk/./algorithm/build/libs/plugin1.jar
Loading security manager
Loaded plugin dlpX-core version 1.4.0 (API version: 1.4.0) from [/Users/testuser/delphix/masking-algorithm-sdk/./sdkTools/build/install/maskApp/lib/delphix-algorithm-plugin-1.4.0.jar, /Users/testuser/delphix/masking-algorithm-sdk/./algorithm/build/libs/plugin1.jar]
Loaded plugin plugin1 version 1.4.0 (API version: 1.4.0) from [/Users/testuser/delphix/masking-algorithm-sdk/./sdkTools/build/install/maskApp/lib/delphix-algorithm-plugin-1.4.0.jar, /Users/testuser/delphix/masking-algorithm-sdk/./algorithm/build/libs/plugin1.jar]
Framework:
* [0] dlpX-core:CM Numeric
  [1] dlpX-core:Character Mapping
  [2] dlpX-core:Date Replacement
  [3] dlpX-core:Date Shift
  [4] dlpX-core:Date Shift Discrete
  [5] dlpX-core:Date Shift Variable
  [6] dlpX-core:Dependent Date Shift
  [7] dlpX-core:FullName
```

```

[8] dlpX-core:Name
[9] dlpX-core:Payment Card
[10] dlpX-core:Regex Decompose
[11] dlpX-core:Secure Lookup
[12] plugin1:Byte Array Redaction
[13] plugin1:Date Redaction
[14] plugin1:Number Redaction
[15] plugin1:Randomized Masking
[16] plugin1:StringRedaction
Select an algorithm framework:

```

[-] Algorithm framework name is now prefixed with the plugin name (that framework is originated from). There are always `dlpx-core` plugin frameworks present, since those are provided by default by the Masking Engine. Previously one could only chain the algorithm with the other algorithm provided by the same plugin. Now it is possible to chain plugin's algorithm with the algorithm instance(s), based on the default `dlpx-core` plugin.

It is also possible to upload another plugin along with the already loaded ones: instead of selecting an algorithm framework from the above menu - step back by pressing *Ctrl-C*. That will bring you back to the `MASKING-APP:>` menu (with the `UserInterruptedException` notification). That error message will be taken care of in a future releases, providing better way for this step back.

```

org.jline.reader.UserInterruptedException
Details of the error have been omitted. You can use the stacktrace command to print
the full stacktrace.
MASKING-APP:>

```

Here it is possible to use similar `mask -j <>` command to upload another plugin:

```

MASKING-APP:> mask -j ./algorithm/build/libs/plugin2.jar
/Users/testuser/delphix/masking-algorithm-sdk/./algorithm/build/libs/plugin2.jar
Loading security manager
Loaded plugin dlpX-core version 1.4.0 (API version: 1.4.0) from [/Users/testuser/
delphix/masking-algorithm-sdk/./sdkTools/build/install/maskApp/lib/delphix-algorithm-
plugin-1.4.0.jar, /Users/testuser/delphix/masking-algorithm-sdk/./algorithm/build/
libs/plugin2.jar]
Loaded plugin plugin2 version 1.4.0 (API version: 1.4.0) from [/Users/testuser/
delphix/masking-algorithm-sdk/./sdkTools/build/install/maskApp/lib/delphix-algorithm-
plugin-1.4.0.jar, /Users/testuser/delphix/masking-algorithm-sdk/./algorithm/build/
libs/plugin2.jar]
21:17:37.426 [main] INFO global - Loaded plugin plugin2: Plugin {'embeddedName':
'plugin2', 'version': '1.4.0', 'author': 'Sample Plugin Author', 'apiVersion':
'1.4.0'}
Framework:
* [0] dlpX-core:CM Numeric

```

```

[1] dlpx-core:Character Mapping
[2] dlpx-core:Date Replacement
[3] dlpx-core:Date Shift
[4] dlpx-core:Date Shift Discrete
[5] dlpx-core:Date Shift Variable
[6] dlpx-core:Dependent Date Shift
[7] dlpx-core:FullName
[8] dlpx-core:Name
[9] dlpx-core:Payment Card
[10] dlpx-core:Regex Decompose
[11] dlpx-core:Secure Lookup
[12] plugin1:Byte Array Redaction
[13] plugin1:Date Redaction
[14] plugin1:Number Redaction
[15] plugin1:Randomized Masking
[16] plugin1:StringRedaction
[17] plugin2:MultiColumnDateAlgorithm
[18] plugin2:Numeric Mapping
[19] plugin2:RedactionDB
[20] plugin2:RedactionFile
[21] plugin2:StringHashedLookup
Select an algorithm framework:

```

Example above shows 3 plugins uploaded:

- dlpx-core (default)
- plugin1 (uploaded by customer)
- plugin2 (uploaded by customer)

Now it is possible choosing any of those plugins frameworks and their related algorithm instances, as well chaining of those algorithms instances to any configurable extensible algorithm (within the loaded plugins). That behavior simulates Masking Engine where multiple plugins are uploaded.

The technique of algorithms chaining is out of scope of the current description. It's the same as chaining the algorithms belonging to the same plugin. Please refer to the [Algorithm chaining page](#) (see page 1041) for chaining details and examples.

12.4.4.7 Retrieving information about installed plugins (SDK workflows/Algorithms)

The GET endpoints are useful for getting information about plugins. After following the steps in [this section](#) (see page 1065) to install the Sample Algorithm plugin, the GET operation will return (elided for brevity):

```

{
  "pluginId": 7,
  "pluginName": "Delphix Sample",
  "originalFileName": "algorithm.jar",
  "originalFileChecksum":
"74df61f436aceb80107c22964c027d32a565d0100de36c7fa42f528327cf2e2a",
  "installDate": "2020-06-19T20:15:58.239+0000",
  "installUser": 5,
  "builtIn": false,
  "pluginVersion": "1.0.0",

```

```

"pluginObjects": [
  {
    "objectIdentifier": "2",
    "objectName": "StringRedaction",
    "objectType": "ALGORITHM_FRAMEWORK"
  },
  {
    "objectIdentifier": "3",
    "objectName": "RedactionFile",
    "objectType": "ALGORITHM_FRAMEWORK"
  },
  {
    "objectIdentifier": "5",
    "objectName": "StringHashedLookup",
    "objectType": "ALGORITHM_FRAMEWORK"
  },
  {
    "objectIdentifier": "7",
    "objectName": "Randomized Masking",
    "objectType": "ALGORITHM_FRAMEWORK"
  },
  {
    "objectIdentifier": "Delphix Sample:Byte Array Redaction",
    "objectName": "Delphix Sample:Byte Array Redaction",
    "objectType": "ALGORITHM"
  },
  {
    "objectIdentifier": "Delphix Sample:Date Redaction",
    "objectName": "Delphix Sample:Date Redaction",
    "objectType": "ALGORITHM"
  },
  {
    "objectIdentifier": "Delphix Sample:Number Redaction",
    "objectName": "Delphix Sample:Number Redaction",
    "objectType": "ALGORITHM"
  },
  {
    "objectIdentifier": "Delphix Sample:Numeric Mapping",
    "objectName": "Delphix Sample:Numeric Mapping",
    "objectType": "ALGORITHM"
  },
  ...
]
}

```

For each plugin, the plugin metadata, including `pluginId`, `pluginName` and `originalFileChecksum` are displayed first. This is followed by a list of algorithm frameworks included in the plugin, then a list of algorithm instances included in the plugin. The list of frameworks will contain only those frameworks that support the creation of additional algorithm instances as described in [this section](#). (see [page 1024](#))

12.4.5 Configurability

12.4.5.1 Introduction

The Extensible Algorithms feature supports the creation of algorithm frameworks. When an algorithm class is constructed to be a framework, the Continuous Compliance Engine operator may create additional instances of the algorithm - with different configurations - after the plugin has been installed. New instances are created by supplying a JSON document describing a new instance using the POST method of the Algorithm endpoint in the Masking Web API. This may be done using the Masking API client. The JSON schema for configuration is determined by which data members in the framework class are marked as configurable, and may vary from framework to framework.

New algorithms created using the SDK skeleton generator are not, by default, configurable using this mechanism. It is necessary to modify the default implementation of the `allowFurtherInstances` method and mark one or more public data members as configurable. This is described in detail in [the next section](#). (see [page 1024](#))

This part of the documentation illustrates what options are available when creating an algorithm to define whether and what kind of configuration is required, and what, if any, default instances should be created. It will also describe how to create instances of a plugin provided algorithm framework using the Masking API Client.

12.4.5.2 Making an algorithm configurable

To integrate an algorithm as a configurable framework within the Continuous Compliance Engine, specific **requirements** must be met:

1. **Method requirement:** The algorithm class should have the `getAllowFurtherInstances` method implemented to return `true`.
2. **Data member annotation:** The algorithm class must include one or more public data members. These members are crucial for configuration and must be annotated with `@JsonProperty` from the `com.fasterxml.jackson.annotation.JsonProperty` package. This annotation enables the data members to be recognized and processed as configurable parameters.

Additional discretionary notes

- **JSON handling and schema consistency:** Most JSON processing tasks are handled by the Masking Plugin API instead of the plugins directly. This ensures uniformity in JSON document and schema interpretation. For each algorithm marked as configurable, the SDK or Continuous Compliance Engine inspects the class annotations to identify which parameters can be configured.
- **Instance creation and JSON application:** When creating a new instance of the algorithm, the system tries to apply the user-provided JSON configuration to the algorithm class object. This process includes a level of validation, ensuring that the provided JSON aligns with the expected schema as inferred from the annotated fields. Note that there are certain limitations in this validation, as detailed in the corresponding section below.

12.4.5.2.1 Example configurable algorithm explained

The concept of configurability can be illustrated using one of the sample algorithms from the SDK as an example, `StringRedaction.java` in this case:

```
package sample.masking.algorithm.redaction;
...

public class StringRedaction implements MaskingAlgorithm<String> {
    private String name = "StringRedaction";

    @JsonProperty(value = "redactionCharacter", required = true)
    public String redactionCharacter = "specified";

    @Override
    public String getName() {
        return name;
    }

    @Override
    public Collection<MaskingComponent> getDefaultInstances() {
        StringRedaction instanceX = new StringRedaction();
        instanceX.name = "Redaction X";
        instanceX.redactionCharacter = "X";
        return Arrays.asList(instanceX);
    }

    @Override
    public boolean getAllowFurtherInstances() {
        return true;
    }

    @Override
    public String getDescription() {
        return String.format(
            "Redact String by overwriting with '%s' character",
            redactionCharacter);
    }

    @Override
    public String mask(@Nullable String input) throws MaskingException {
        if (input == null) {
            return null;
        }
        StringBuilder returnVal = new StringBuilder();

        for (int i = 0; i < input.length(); i++) {
            returnVal.append(redactionCharacter);
        }
        return returnVal.toString();
    }
}
```

```

    }

    @Override
    public void validate() throws ComponentConfigurationException {
        if (redactionCharacter == null || redactionCharacter.length() != 1) {
            throw new ComponentConfigurationException(
                "redactionCharacter must be a single character");
        }
    }
}

```

This algorithm is designed for straightforward redaction of input strings, with a unique feature allowing the redaction character to be customized. Here's how it functions:

- **Configurable parameters:** The class includes a public field named `redactionCharacter`, marked with the `@JsonProperty` annotation. This field can be configured with different values, and a default value is set to ensure that the `getDescription` method provides an appropriate description in both the framework and individual instance contexts.
- **Default instance specification:** The `getDefaultInstances` method in the class defines a default instance of this algorithm. This is achieved by returning a list of objects configured with 'X' as their redaction character. The API framework then converts these object configurations into JSON, which it stores for future use when an instance of the "Redaction X" algorithm is requested.
- **Enabling additional instances:** Setting the `getAllowFurtherInstances` method to return true allows the creation of additional instances of this algorithm. This capability is particularly useful once the plugin is integrated into the Continuous Compliance Engine using the Masking API through an API client.
- **Configuration validation:** The class implements a `validate` method to ensure that the configuration values provided are appropriate. Specifically, for the `redactionCharacter` field, the validation restricts the string length to a single character, ensuring the redaction functionality works as intended.

12.4.5.2.2 Frameworks, instances, and configuration injection

When used as a framework, the algorithm class is instantiated and used without any configuration injection. In the example above, that means that the `getDescription` method will return, "redact String by overwriting with 'specified' character" when the algorithm framework is evaluated. Similarly, `getName` will return "StringRedaction", the name of the framework.

When a runnable algorithm instance is needed, the algorithm class is instantiated, and all saved configuration is injected before any methods are called. This configuration is gathered in one of two ways:

- For statically provided instances embedded in the plugin, the configurable fields of each object returned by the `getDefaultInstances` method are serialized to JSON and saved. Again, only the values of public fields marked with the `@JsonProperty` annotation are extracted this way.

- When the user creates a new algorithm instance using the Masking Web API, the contents of the `algorithmExtension` field of the POST or PUT request is validated and saved for future injection whenever that particular algorithm instance is needed in the future.

In the above example, when algorithm instance "Redaction X" is created, the saved values will be injected, so `redactionCharacter` will have the value 'X'.

12.4.5.2.3 Validation of configuration values

The major JSON handling libraries, for performance reasons, conduct minimal validation during object deserialization. This means that many `@JsonProperty` annotation aspects are not strictly enforced. For instance, even if a property is flagged as `required`, an object can still be deserialized successfully without that property in the input JSON.

While there are libraries to enhance JSON validation within the framework, implementing them complicates the distinction between validations handled by the API framework and those required in the component's `validate` method. Consequently, the framework currently performs only essential input validation. It falls upon plugin authors to thoroughly validate all aspects of an object's configuration in their `validate` method implementation, particularly on the presence of required fields (i.e. non-empty and non-null values).

Despite this enforcement lapse of `@JsonProperty` annotation properties, it is important not to disregard them. These properties are included in the auto-generated schema for each framework, accessible through the SDK's `maskApp` and the algorithm/framework endpoint in the masking API client. Their visibility in these schemas might be helpful for future UI development.

12.4.5.2.4 Default interface implementations

The masking API defines default implementations of `getDefaultInstances` and `getAllowFurtherInstances` as follows:

```

default Collection<ComponentInstanceDescription> getDefaultInstances() {
    return Collections.singletonList(this);
}

default boolean getAllowFurtherInstances() {
    return getDefaultInstances() == null || getDefaultInstances().isEmpty();
}

```

This means that if neither of these methods is overridden by the masking algorithm class, a single instance capturing whatever default values exist for configurable fields is created by default.

Only algorithm classes that define `getAllowFurtherInstances` to return `true` appear as Algorithm Frameworks on the Continuous Compliance Engine.

12.4.5.2.5 Build dependencies for configurable algorithms

When the `maskScript init` sub-command is used to create a new project, the initial build files may not include the dependencies required for the Jackson `@JsonProperty` annotation. This can be corrected by adding this line to `proj_root/gradle.properties`:

```
jacksonVer=2.15.2
```

The Jackson version is 2.15.2 at the time of authoring, though, be sure to use the latest stable version to remain up-to-date with security fixes in the library.

And this line to the `dependencies*` section at the end of `proj_root***/build.gradle`:

```
compileOnly ('com.fasterxml.jackson.core:jackson-annotations:' + jacksonVer)
```

The set of Jackson annotations tested and supported for use in algorithm plugin classes are:

- `@JsonProperty`
- `@JsonPropertyDescription`
- `@JsonFormat` (Useful in specifying formats for Date fields)

12.4.5.3 Using an Algorithm Framework

When a plugin algorithm supports configuration, it is possible to create new instances of the algorithm on the Continuous Compliance Engine by specifying the desired configuration. This is done using the engine's [Masking Web API](#) (see page 878). Configurable algorithms may also be tested using the **maskApp** and **maskScript** utilities, by providing the desired configuration in an input file or at the command line.

12.4.5.3.1 Creating New Algorithm Instances Using the maskApp SDK Utility

When the `maskApp` utility's `mask` command is invoked and a configurable algorithm is selected, the option will be presented to create a new algorithm instance. This is done by choosing `::Create New Instance::`. The algorithm's configuration schema is displayed, and then a valid JSON input must be provided to create the new instances. Rather than entering the literal JSON, the `@` symbol may be used to load the JSON from a file (**@file-path**).

What follows is an example of loading the Sample Algorithm plugin, creating a new instance of the `StringRedaction` framework and masking test values with the new algorithm instance.

```
$ maskApp
... Startup Messages ...
MASKING-APP:> mask -j algorithm/build/libs/algorithm.jar
/Users/jleser/ws/algorithm-sdk/algorithm/build/libs/algorithm.jar
Loaded plugin Delphix Sample version 1.0.0 dea904c (API version: 1.0.0) from [/Users/
jleser/ws/algorithm-sdk/algorithm/build/libs/algorithm.jar]
```

```

16:44:47.743 [main] INFO  global - Loaded plugin Delphix Sample: Plugin {'embeddedName': 'Delphix Sample', 'version': '1.0.0 dea904c', 'author': 'Delphix', 'apiVersion': '1.0.0'}
Framework:
* [0] Byte Array Redaction
  [1] Date Redaction
  [2] Number Redaction
  [3] Numeric Mapping
  [4] Randomized Masking
  [5] RedactionFile
  [6] StringHashedLookup
  [7] StringRedaction
Select an algorithm framework: 7
Instance:
* [0] Delphix Sample:Redaction X
  [1] Delphix Sample:Redaction Y
  [2] Delphix Sample:Redaction Z
  [3] ::Create New Instance::
Select an instance of algorithm framework: StringRedaction: 3
The JSON schema of the selected framework is:
{
  "type" : "object",
  "id" : "urn:jsonschema:sample:masking:algorithm:redaction:StringRedaction",
  "properties" : {
    "redactionCharacter" : {
      "type" : "string",
      "required" : true
    }
  }
}
Enter config(Prefix with '@' for file location)(Blank for no config): {
"redactionCharacter" : "+" }
Enter instance name: RedactPlus
Algorithm Configuration: {"redactionCharacter":"+"}
Selected algorithm:
sample.masking.algorithm.redaction.StringRedaction(StringRedaction) instance:
RedactPlus, data type: STRING
Input value to be masked('null' for null, 'doneMasking' to finish): Test
Masked value: ++++
Input value to be masked('null' for null, 'doneMasking' to finish): One
Masked value: +++
Input value to be masked('null' for null, 'doneMasking' to finish): TwoThree
Masked value: ++++++++

```

12.4.5.3.2 Creating New Algorithm Instances on the Continuous Compliance Engine

New instances of plugin frameworks may be created using the Continuous Compliance Engine's Web API's *algorithm* endpoint. This is similar to creating any other algorithm using the *algorithm* API endpoint and may be performed using the API client. Unlike when an algorithm is created using older, built-in frameworks like Secure Lookup:

- The value for *algorithmType* in the JSON request is always "COMPONENT". This is now the default value, so this field may be omitted.
- A value for the field *frameworkId* must be included - this is the integer ID of the framework as provided in the plugin description retrievable using the GET operation on the plugin endpoint, or GET on the *algorithm/frameworks* endpoint.
- The *algorithmExtension* field's contents are used directly as the JSON configuration for the algorithm instance. Unlike other algorithm types, this field does not have a fixed schema for COMPONENT type algorithms. The required schema may be retrieved using the [procedure described below](#). (see page 0)

This example API request, POSTed to the algorithm endpoint, creates a new instance of the StringRedaction algorithm (described above), named "RedactStar" using '*' as the redaction character. In this case, the sample algorithm plugin JAR has already been uploaded, and the StringRedaction framework has id 19:

```
{
  "algorithmName": "RedactStar",
  "algorithmType": "COMPONENT",
  "description": "Redact with the star character",
  "frameworkId" : 19,
  "algorithmExtension" : {
    "redactionCharacter": "*"
  }
}
```

12.4.5.3.3 Discovering the algorithm extension API Field Schema

The Masking Web API *algorithm/framework* endpoint has the ability to show the JSON Schema for each algorithm framework implemented using the extensibility mechanism. By default, the schema is not included, but by setting *include_schema* true, the schema may be retrieved. Here is the GET API result, including schema, for the StringRedaction framework used above:

```
{
  "frameworkId": 19,
  "frameworkName": "StringRedaction",
  "frameworkType": "STRING",
  "plugin": {
    "pluginId": 47,
    "pluginName": "algorithm"
  },
  "extensionSchema": {
    "id": "urn:jjsonschema:sample:masking:algorithm:redaction:StringRedaction",
    "properties": {
      "redactionCharacter": {
        "type": "string",
        "required": true
      }
    }
  }
}
```

This schema is generated automatically using the annotated public fields in the framework class.

12.4.5.4 Using Multi-Column Algorithms

To be able to configure and use the Multi-Column (MC) Algorithms one should be familiar with the following themes:

- Extensible Algorithms in general
- Their creation using the [Masking SDK](#) (see page 1008)
- Extensible Algorithms [Plugin installation](#) (see page 1065)
- [Masking API Client](#) (see page 878) (optional)

12.4.5.4.1 Logical Fields

A sample instance (serving as an example) of the MC algorithms is in the Masking SDK distribution, named "MultiColumnDateAlgorithm". That framework (the instance is based on) defines two fields:

```
@Override
public List<AlgorithmLogicalField> listMultiColumnFields() {
    /*
     * Here we define the column names to be used in the algorithm. These names
     * are only used to reference the
     * columns within the algorithm and do not need to correspond to the names
     * of the columns on the data source.
     * For example, our data source may call these 2 fields "dateOfBirth" and
     * "dateOfDeath", however within the
     * algorithm implementation they will be referenced as "startDate" and
     * "endDate" (see mask method to see how
     * this is used).
     */
    return ImmutableList.of(
        new AlgorithmLogicalField("startDate", MaskingType.LOCAL_DATE_TIME),
        new AlgorithmLogicalField("endDate", MaskingType.LOCAL_DATE_TIME));
}
```

In that example, the fields "startDate" and "endDate" are logical fields, defined by the framework. If one doesn't have access to the source code of the framework, it's possible to find the logical field names (and their types) using the *Masking API: GET /algorithms/{algorithmName}* endpoint.

The API provides a five-argument constructor for **AlgorithmLogicalField** that allows for fields to be marked as: *read-only* and/or *optional*, as well as to provide a short documentation string for the field's usage. The Extensibility SDK provides an example algorithm that demonstrates this called *MultiColumnRedaction.java*.


Let's suppose you already have an instance of multi-column Algorithm installed. That might happen in any of the following two cases:

- The Plugin you've installed contains a default instance for MC algorithms.
- The Plugin you've installed contains only a framework for configurable MC algorithms. In that case, you've configured an instance of the algorithm.

Let's take as an example "MultiColumnDateAlgorithm" algorithm mentioned above (plugin is named "sample" in that example). Retrieving its info using the `GET /algorithms/{algorithmName}` endpoint returns:

```
{
  "algorithmName": "Sample Plugin:MultiColumnDateAlgorithm",
  "algorithmType": "COMPONENT",
  "isTokenizationSupported": false,
  "pluginId": 11,
  "fields": [
    {
      "fieldId": 5,
      "name": "startDate",
      "type": "LOCAL_DATE_TIME",
      "isReadOnly": false,
      "isOptional": false
    },
    {
      "fieldId": 6,
      "name": "endDate",
      "type": "LOCAL_DATE_TIME",
      "isReadOnly": false,
      "isOptional": false
    }
  ],
  "algorithmExtension": {}
}
```

Here we can see the information structure for the logical fields, defined by the current framework. We will use that data when configuring the Inventory fields.

 Previous versions of the Extensibility API required two methods - `listMaskedFields` and `listReadOnlyFields` - to be implemented when creating a multi-column algorithm. These methods are now deprecated, and `listMultiColumnFields` is preferred way for multi-column algorithms to define their fields. However, existing algorithms that use the old methods should continue to function normally.

12.4.5.4.2 Configuring column metadata for MC algorithm

To configure the involved column (i.e. masked and read-only columns) - we should update the column's metadata with the following information:

```
"algorithmFieldId"
"algorithmGroupNo"
"algorithmName"
"domainName"
```

The last two fields are the regular configuring fields for masked columns. Let's look closer to the newly introduced fields for MC:

- `algorithmFieldId` is a `fieldId` for the corresponding logical field. For example for "startDate" from the example above its value is 5.
- `algorithmGroupNo` is a group number (integer) for the columns treated by the same algorithm instance. It is introduced for cases where we might have multiple columns of a similar type, which are masked by the different Masking Jobs using the same algorithm. In such a case that's important to unite the columns per algorithm run, by assigning the same group number.

There are two supported methods to configure the `columnMetadata` for the masked table inventory:

- Via API
- Via UI

12.4.5.4.2.1 Configuring `columnMetadata` for MC algorithms via API

Below is the example of the column metadata before it's configured for MC algorithm:

```
Response Body
{
  "columnMetadataId": 63,
  "columnName": "DATA00",
  "tableMetadataId": 19,
  "dataType": "VARCHAR2",
  "columnLength": 100,
  "isMasked": false,
  "isProfilerWritable": true,
  "isPrimaryKey": false,
  "isIndex": false,
  "isForeignKey": false
},
```

Let's associate that field with the logical field `startDate` (`fieldId=5`) from the snapshot above, by adding the mentioned fields:

```
Response Body

{
  "columnMetadataId": 63,
  "columnName": "DATA00",
  "tableMetadataId": 19,
  "algorithmName": "sample:MultiColumnDateAlgorithm",
  "algorithmFieldId": 5,
  "algorithmGroupNo": 1,
  "domainName": "DOB",
  "dataType": "VARCHAR2",
  "dateFormat": "yyyy-MM-dd",
  "columnLength": 100,
  "isMasked": true,
  "isProfilerWritable": true,
  "isPrimaryKey": false,
  "isIndex": false,
  "isForeignKey": false,
  "domainAssignedBy": "admin_user"
},
```

i For the masked column, the *isMasked* field should be manually changed to *true*, while for read-only field it stays *false*.

If at this point an inventory for the masked table is checked in the UI - the configured (via API) inventory will be displayed there:

Overview Connector Rule Set **Inventory**

Home > Environments > test1 > Inventory > test

test Import Export

Filter By: All Fields Masked Fields Auto User

Column	Data Type	Algorithm	Edit
DATA00	VARCHAR2 (100)	sample:MultiCo...nDateAlgorit...	
DATA01	VARCHAR2 (100)		
ID	NUMBER (38)		

Select Rule Set: test

Filter Contents: Search By Name, Search Alphabetically

12.4.5.4.2.2 Configuring columnMetadata for MC algorithms via UI

The same columnMetadata configuration can also be made via the UI. As with other algorithms one has to choose the Domain and Algorithm values, applied to the current column. If a Multi-Column algorithm has been chosen, the following additional two fields will need to be filled out:

- **Select Logical Field** dropbox, where the corresponding logical field to be selected.
- **Algorithm Group** window, where algorithmGroupNo value to be entered.

The screenshot shows a dialog box titled "Edit Properties" with the following fields:

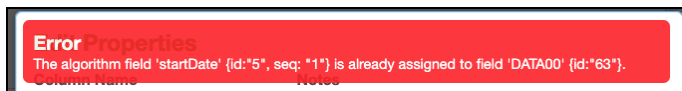
- Column Name:** DATA00
- Notes:** (Empty text area)
- ID Method:** Auto
- Domain:** DOB
- Algorithm:** sample:MultiColumnDateAl...
- Select Logical Field:** startDate
- Algorithm Group:** 1
- Type:** LOCAL_DATE_TIME
- Date Format:** yyyy-MM-dd

Buttons for "Cancel" and "Save" are located at the bottom right.

i In the UI configuration for columnMetadata, the customer shouldn't mark the *isMasked* field (as via the API in the example above). It's taken care automatically since ME knows the associated logical field is being masked or used as a read-only.

12.4.5.4.3 Error Management

There are different configuration errors possible while setting the MC algorithms. The configuration process prevents as many misconfigurations as possible, but some configuration errors can only be detected when a job is executed. For example, if trying to associate a second column to the same (already busy) logical field will result in a configuration error similar to:



In case there is a missed association with the required logical field - that type of error isn't recognized during the configuration, but only during the job execution (which will fail due to that misconfiguration).

Please find below an example of the monitor job error report:

▲ Error Report ✕

Execution Events Learn More		
Event	Cause	Description
JOB_ABORTED	UNHANDLED_EXCEPTION	Exception:Algorithm 'sample:MultiColumnDateAlgorithm' requires 2 fields [endDate, startDate] but found 1 fields [startDate]. Missing fields: [endDate]
75_53.log		
<pre> 2020-12-14 17:24:16,233 [thread] INFO com.dmsuite.dmsApplicator.masking.XMLGenerator executeMarshalling - Generate request xml started successfully. 2020-12-14 17:24:16,782 [thread] INFO com.dmsuite.dmsApplicator.masking.XMLGenerator executeMarshalling - Generate Request xml done successfully. 2020-12-14 17:24:16,810 [thread] INFO com.dmsuite.dmsApplicator.masking.transformation.MaskingMarshalling createKettleXML - Generate Transformation XML started successfully </pre>		

12.4.5.4.4 Limitations for the MC Algorithms

1. Currently, it's possible to run the MC Algorithms only on a single table. Masking multiple tables columns by MC Algorithms is not supported.
2. XML File masking does not support MC algorithms.
3. VSAM File masking does not support MC algorithms.
4. Some types of misconfiguration errors (as described above) are only detected during job execution.

12.4.6 Service interfaces (Algorithms)

12.4.6.1 Introduction

The Extensible Algorithms framework makes a number of services available to the algorithm implementation. This prevents the algorithm from having to re-implement code to perform certain routine tasks and facilitates seamless integration with the Masking Engine. This functionality is exposed to the algorithm class via the **ComponentService** interface.

Whenever a new Masking Algorithm instance is required for masking, the extensibility framework first injects any saved configuration, then invokes the objects *setup* method. This method is passed a reference to an object that implements **ComponentService**. The algorithm's *setup* method can then use this object to access a number of provider methods:

- *getInstanceName* - Get the name of this instance. Because the instance name it is not typically a configurable field in the algorithm, the *getName* method will not correctly return the name of an algorithm instance, even after JSON configuration injection. This method will always return the correct instance name as known to the Masking Engine.
- *openInputFile* - Access the contents of a file, as described in [this section](#) (see page 1037)
- *getAlgorithmByName* - Get a usable instance of another algorithm, as described in [this section](#) (see page 1041)
- *getCryptoService* - Access cryptographic methods based on the algorithm's key, as described in [this section](#) (see page 1044)
- *getLogService* - Get a logger object, as described in [this section](#) (see page 1046)



Getting More Information

Refer to the *com.delphix.masking.api.provider* package in the [Javadoc](#)³⁹⁰ for detailed information.

12.4.6.2 Accessing Files

It is often the case that a masking algorithm will require a large library of input values - for example, a set of replacement names or account numbers. In other cases, it may be desirable to store the configuration of a particularly complex algorithm in a format other than JSON, perhaps in order to leverage pre-existing code. To support these use cases, the extensible algorithm framework allows algorithms to access input files from a variety of sources.

12.4.6.2.1 Opening Input Files

Algorithms may access files in several locations, both on the engine and over the network. In all cases, access is achieved by invoking the *openInputFile* method of the **ComponentService** object passed to the algorithm's *setup* method to acquire an **InputStream**. The file's location must be specified by an **FileReference** object visible in the object's public fields. This object is passed to the *openInputFile* method. The value of the **FileReference** may be made configurable using the `@JsonProperty` annotation. The *openInputFile* method accepts a URI style syntax that combines standard URL notation for web resources with custom URI types for engine and JAR located files. The following formats are supported for **FileReference** values:

URI Format	Description
<code>http[s]:// <host>[:<port>] /<path></code>	To open files located on a remote web server
<code>jar://file/ <filepath></code>	To open a file located in the algorithm plugin JAR
<code>delphix-file:// upload/<file reference details></code>	To open a file uploaded using Delphix Masking Engine's <i>fileUpload</i> endpoint. The result of POST ing to the <i>fileUpload</i> API endpoint is a URI in this format that should be used exactly as-is for the <i>uri</i> value of the FileReference .

³⁹⁰ <https://maskingdocs.delphix.com/maskingPluginAPIJavadoc/>

URI Format	Description
<pre>delphix-file:// mount/ <mountType>/ <mountId>/<file path></pre>	To open a file located on a NFS/CIFS mount server that has been mounted inside the Delphix Masking Engine using mountFilesystem endpoint

12.4.6.2.2 Example Algorithm

```
public class RedactionFile implements MaskingAlgorithm<String> {
    private String redactionCharacter = null;

    @JsonProperty(value = "file", required = true)
    public FileReference file;
    @Override
    public String getName() {
        return "RedactionFile";
    }

    @Override
    public String mask(@Nullable String input) throws MaskingException {
        if (input == null) {
            return null;
        }
        StringBuilder returnVal = new StringBuilder();
        for (int i = 0; i < input.length(); i++) {
            returnVal.append(redactionCharacter);
        }
        return returnVal.toString();
    }

    @Override
    public void setup(@NonNull ComponentService serviceProvider) {
        InputStream inputStream = serviceProvider.openInputFile(file);
        try (Scanner scanner = new Scanner(inputStream,
Charset.defaultCharset().name())) {
            redactionCharacter = scanner.nextLine().trim();
        } catch (Exception e) {
            e.printStackTrace();
            throw new RuntimeException("Unable to parse input file", e);
        }
    }

    @Override
```

```

public void validate() throws ComponentConfigurationException {
    GenericReference.checkRequiredReference(file, "file");
}
}

```

Some methods have been omitted for brevity.

This example algorithm is very similar to the `StringRedaction` class discussed earlier, in that it redacts strings by replacing with a same-length string of the redaction character. This variant reads the character to use for redaction from an input file, the location of which is specified in the algorithm's configuration. This is all done in the `initialize` method:

- The value of the variable `file` is public and marked configurable with `@JsonProperty`.
- The file reference is passed to `serviceProvider.openInputFile` during setup - this allows the algorithm to ingest input files in any supported location.
- The redaction character is read from the input stream and stored in instance variable `redactionCharacter` for use in the `mask` method.
- This class's `validate` method uses the static method `GenericReference.checkRequiredReference` provided by the Masking Plugin API to check the file reference for validity.

12.4.6.3 Accessing Database Servers (JDBC)

It is often the case that a masking algorithm will require access to a large amount of data such as lookup values for masking input data or storing states of the algorithm. To support these use cases, the extensible algorithm framework allows algorithms to access database servers using JDBC connections.

12.4.6.3.1 Opening Database Connection

Algorithms access the database by using [extensible drivers](#) (see page 0). Access is achieved by invoking the `openJdbcConnection` method of the `ComponentService` object passed to the algorithm's `setup` method to acquire a `Connection` (`java.sql.Connection`) object. The file's location must be specified by an `JdbcReference` object visible in the object's public fields. This object is passed to the `openJdbcConnection` method. The value of the `JdbcReference` may be made configurable using the `@JsonProperty` annotation. The `JdbcReference` object requires following fields

1. `jdbcDriverId`: The driver Id of the JDBC Driver uploaded using the JDBC Driver API (`/jdbc-drivers`).
2. `url`: The JDBC URL that will be used to connect to the server. Please don't use this to pass the credentials.
3. `credFileReference`: A `FileReference` object that contains the location of the JSON file that stores the credentials. The schema of the file is:

```

{
  "username": "USERNAME",
  "password": "PASSWORD"
}

```

The file must be a *mount* type **FileReference** object. To see how to create a mount type **FileReference** object, refer to [Accessing Files](#) (see page 1037). The **Connection** object is kept open throughout the execution of the algorithm unless it is closed by the algorithm itself.

12.4.6.3.2 Example Algorithm

```
public class RedactionDB implements MaskingAlgorithm<String> {
    private String redactionCharacter = null;

    @JsonProperty(value = "jdbc", required = true)
    @JsonPropertyDescription("A reference to a database containing a table
redaction_character")
    public JdbcReference jdbc;

    private static final String GET_REDACTION_CHARACTER = "SELECT redact FROM
redaction_character LIMIT 1";

    @Override
    public String getName() {
        return "RedactionDB";
    }

    @Override
    public String mask(@Nullable String input) throws MaskingException {
        if (input == null) {
            return null;
        }
        StringBuilder returnVal = new StringBuilder();
        for (int i = 0; i < input.length(); i++) {
            returnVal.append(redactionCharacter);
        }
        return returnVal.toString();
    }

    @Override
    public void setup(@NonNull ComponentService serviceProvider) {
        try (Connection conn = serviceProvider.openJdbcConnection(jdbc);
            PreparedStatement stmt = conn.prepareStatement(GET_REDACTION_CHARACTER)) {
            ResultSet resultSet = stmt.executeQuery();
            List<String> redactionChars = new ArrayList<>();
            if (resultSet.next()) {
                redactionCharacter = resultSet.getString("redact");
            } else {
                throw new RuntimeException("Couldn't find redaction character");
            }
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
    }
}
```

```

    }

    @Override
    public void validate() throws ComponentConfigurationException {
        GenericReference.checkRequiredReference(jdbc, "jdbc");
    }
}

```

Some methods have been omitted for brevity.

This example algorithm is very similar to the `StringRedaction` class discussed earlier, in that it redacts strings by replacing with a same-length string of the redaction character. This variant reads the character to use for redaction from a table in a database, the connection information for which is specified in the algorithm's configuration. This is all done in the `initialize` method:

- The value of the variable "jdbc" is public and marked configurable with `@JsonProperty`.
- The jdbc reference is passed to `serviceProvider.openJdbcConnection` during setup.
- The redaction character is read from the table `redaction_character` and stored in the instance variable `redactionCharacter` for use in the `mask` method.
- This class's `validate` method uses the static method `GenericReference.checkRequiredReference` provided by the Masking Plugin API to check the jdbc reference for validity.

12.4.6.4 Algorithm chaining

The extensible algorithm framework allows algorithms to instantiate and call other algorithms. This is useful to allow for the composition and reuse of algorithm behaviors. This feature is referred to as algorithm chaining.

12.4.6.4.1 Calling other algorithms

In order to make use of this feature, the caller algorithm must acquire an object of the algorithm class it wishes to call by requesting it by instance name using the `getAlgorithmByName` method of the **ComponentService** object. This is done during the execution of the algorithm's `setup` method.

This method requires that the caller specify two values:


1. A reference to the algorithm instance. This must be stored in an **AlgorithmInstanceReference** object whose value is the name of the algorithm instance. This is `algorithmName` in the Masking API, occasionally referred to as "algorithmCd" or "algorithm code". The **AlgorithmInstanceReference** object must be referenced in a `public` field in the algorithm object.
2. The type of data the returned algorithm object should mask, selected from the core types supported by the extensible algorithm framework. Type adaptation is not currently supported in this context, so the algorithm's native type must be the type requested using `getAlgorithmByName`.

Once an algorithm object has been obtained using `getAlgorithmByName`, a reference to the algorithm object may be kept and that algorithm's `mask` method called as needed.

Examples:

- You are creating algorithm instance A via the Masking API Client Algorithm endpoint, and algorithm A uses *getAlgorithmByName* to find algorithm B during *setup*. For the creation of algorithm A to succeed, algorithm B **must** already exist on the Delphix Masking Engine.
- You are installing a plugin that would create the same algorithm A as a static instance. This will fail if algorithm instance B is not also provided by an algorithm class in the same plugin.

Because it is difficult to predict what algorithm names exist on a Delphix Masking Engine, it is advised that the names of any algorithms used for chaining be supplied in the algorithm's JSON configuration. Hard-coding names of algorithms passed to *getAlgorithmByName* directly in the Java source creates dependencies that are not visible except in the error message that results when the caller algorithm fails to initialize, as described in the second example scenario above. Hard-coded references to other algorithms provided by the same plugin should have the value `":algorithmName"`. The ":" character tells the API to fill in this plugin's name when searching for the instance.

 Algorithm instances provided by plugins (via the *getDefaultInstances* method) are prohibited from having dependencies on algorithm instances provided by other plugins. A way to safely implemented this kind of dependency may be added in the future.

12.4.6.4.2 Example algorithm

```
public class RandomizedStringMasking implements MaskingAlgorithm<String> {
    private List<MaskingAlgorithm<String>> algorithmList = new ArrayList<>();
    private Iterator<Integer> randomStream;

    @JsonProperty(value = "algorithmNames", required = true)
    public List<AlgorithmInstanceReference> algorithms;

    @Override
    public String getName() {
        return "Randomized Masking";
    }

    @Override
    public Collection<MaskingComponent> getDefaultInstances() {
        RandomizedStringMasking myInstance =
            new RandomizedStringMasking() {
                @Override
                public String getName() {
                    return "Randomized Redaction";
                }
                @Override
                public String getDescription() {
                    return "Apply a random redaction algorithm from { X, Y, Z }";
                }
            };
        myInstance.algorithms =
```



```

        Arrays.asList(
            new AlgorithmInstanceReference(":Redaction X"),
            new AlgorithmInstanceReference(":Redaction Y"),
            new AlgorithmInstanceReference(":Redaction Z"));

    return Collections.singletonList(myInstance);
}

@Override
public void validate() throws ComponentConfigurationException {
    if (algorithms == null || algorithms.isEmpty()) {
        throw new ComponentConfigurationException(
            "Value for field algorithmNames is missing or empty");
    }
    for (AlgorithmInstanceReference ref : algorithms) {
        GenericReference.checkRequiredReference(ref, "algorithms");
    }
}

@Override
public void setup(@NonNull ComponentService serviceProvider) {
    for (AlgorithmInstanceReference algorithm : algorithms) {
        algorithmList.add(serviceProvider.getAlgorithmByName(algorithm,
MaskingType.STRING));
    }
    randomStream = new Random().ints(0, algorithmList.size()).iterator();
}

@Override
public String mask(@Nullable String s) throws MaskingException {
    return algorithmList.get(randomStream.next()).mask(s);
}

```

Some methods have been omitted for brevity.

This algorithm is configured with a list of other String masking algorithms and masks by calling another algorithm from that list at random. This randomization is not based on the algorithm key, so results will not be consistent across masking runs. In addition, this framework defines a default instance that chooses randomly between algorithms "Redaction X", "Redaction Y" or "Redaction Z" included in the same plugin.

The algorithm's public fields include a list of **AlgorithmInstanceReference** objects, made configurable by the `JsonProperty` annotation.

This algorithm's `setup` method does the following:

- For each algorithm name, it calls `getAlgorithmByName` to instantiate a usable algorithm object, saving them in `algorithmList`.
- It initializes a random number generator to produce integers corresponding to each index in `algorithmList`.

This algorithm's `mask` method selects an algorithm at random from `algorithmList` and calls its `mask` method on the input value, returning the result.

This algorithm's `getDefaultInstances` method creates a single instance that chooses between three algorithms. Each algorithm reference begins with ':', indicating that these algorithms should be found in the

same plugin as this algorithm. The `getName` and `getDescription` methods of the returned object are overridden to provide values different from those of the framework itself.

12.4.6.5 Using cryptographic keys

eCryptography is useful in algorithm development for a range of purposes, from straightforward encryption of value to shuffling collections and permuting data in a manner that is consistent across masking jobs. The extensible algorithm framework automatically provides each algorithm with a cryptographic key. This key is wrapped by a service provider object that implements the **CryptoService** interface, providing a number of useful operations based on the algorithm's key. It is also possible to retrieve the raw key assigned to the algorithm as an array of bytes.

Similar to working with files, there is a **KeyReference** type that represents a reference to the key. This is present to support access to keys stored in alternative locations (ex. a key vault) in the future. Currently, the only supported value for these references is "", which indicates that the per-algorithm key stored on the Delphix Masking Engine should be used.



When working with the Masking SDK `maskApp` and `maskScript` utilities, each algorithm's key is a stable hash of its algorithm name, but maybe temporarily set to a random value using the `-K` flag.

12.4.6.5.1 Using the CryptoService provider

The first step any algorithm that wishes to use its algorithm key must take is to retrieve a handle to a cryptographic service provider during initialization. This is done by calling the **ComponentService** object's `getCryptoService` method. The returned provider wraps the key. The operations supported by the **CryptoService** interface are as follows:

- `getRawKey` - retrieve the raw key associated with this provider as a byte array.
- `wrap` - wraps an array of bytes to create a **CryptoService** object. This is useful for accessing **CryptoService** methods when the algorithm's key is stored in an alternative location or hard-coded in the algorithm source.
- `deriveNewKey` - derive a new key by permuting this provider's key using SHA-256. A new **CryptoService** object is returned wrapping the new key. The zero-argument version of this method returns the same key each time it is called on the same provider - in order to create multiple, different keys, a different salt must be provided to each method call. It is advisable that whenever an algorithm wishes to use cryptography for multiple purposes, new and distinct keys be derived for each purpose.
- `computeHashedLookupIndex` - compute an integer value from 0 to (modulus - 1) by hashing the input value + key. This method is designed to allow randomized, but consistent, lookups into a replacement table based on the input value.
- `shuffleList` and `shuffleListNoCollisions` - these methods shuffle their argument **List** in-place using the key to seed the randomization. The "noCollisions" variant ensures that no object in the list remains in its original position.

12.4.6.5.2 Example algorithm

```

public class StringHashedLookup implements MaskingAlgorithm<String> {
    private List<String> replacements;
    private CryptoService crypto;

    public KeyReference key = new KeyReference();

    @JsonProperty("replacementFile")
    public FileReference replacementFile;

    @Override
    public void validate() throws ComponentConfigurationException {
        GenericReference.checkRequiredReference(replacementFile, "replacementFile");
    }

    @Override
    public void setup(@NonNull ComponentService serviceProvider) {
        replacements = new ArrayList<>();

        String line;
        try (InputStream is = serviceProvider.openInputFile(replacementFile);
            BufferedReader reader =
                new BufferedReader(new InputStreamReader(is, "UTF_8"))) {
            while ((line = reader.readLine()) != null) {
                replacements.add(line);
            }
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
        crypto = serviceProvider.getCryptoService(key);
    }

    @Override
    public String mask(@Nullable String input) {
        if (input == null || input.length() == 0) {
            return input;
        }
        return replacements.get(((int) crypto.computeHashedLookupIndex(input,
            replacements.size())));
    }
}

```

Some methods have been omitted for brevity.

This example algorithm functions very similarly to the existing Secure Lookup algorithm, except it employs a different hash method from the new **CryptoService** provider.

- The algorithm is configured with an input file by supplying a public, annotated **FileReference** field *replacementFile*.

- In the *setup* method, the replacement file is ingested and saved as a list of values.
- Additionally in *setup*, the cryptographic service provider is initialized using the default key reference, accessing the algorithm's key.
- The mask method uses the *computeHashedLookupIndex* method to compute the index of the replacement to use from the *replacements* list.

12.4.6.6 Logging

It is possible for a plugin algorithm to write information into the job logs, and consequently, Continuous Compliance Engine logs. This is accomplished by using calling the *getLogService* method of the **ServiceProvider** interface provided at the algorithm setup. The resulting **LogService** object may be used to make logging entries at various levels of severity. The available log levels are ERROR, WARNING, INFO, and DEBUG.

The log interface is provided to allow for debugging output during algorithm development, and for reporting of statistical or similar values detailing the overall operation of the algorithm, typically in the *tearDown* method.



Logging Security Warning

An algorithm **must** never log unmasked values (the *input* argument to the *mask* method) to the log files. The job and Masking Engine log files may be retrieved by engine users and are included support bundles.



Logging Verbosity

Algorithms also should not log progress messages or other verbose details, especially from the *mask* method, as this will fill the log files with messages and may impact job performance. There is a rate-limiting mechanism that limits the volume of messages each algorithm can write over time, but any amount of routine logging is likely to diminish the overall usefulness of the logs by obscuring more important messages.

12.4.6.6.1 Example code

This example is take from the StringRedaction sample algorithm provided with the SDK:

```
public class StringRedaction implements MaskingAlgorithm<String> {
    ...
    private LogService logger;
    ....

    @Override
    public String mask(@Nullable String input) throws MaskingException {
        if (input == null) {
```

```

        return null;
    }

    if (random.nextDouble() < 0.1) {
        logger.info("{0}: Masked {1} values", getName(), count);
    }

    StringBuilder returnVal = new StringBuilder();

    for (int i = 0; i < input.length(); i++) {
        returnVal.append(redactionCharacter);
    }
    count++;
    return returnVal.toString();
}

@Override
public void validate() throws ComponentConfigurationException {
    if (redactionCharacter == null || redactionCharacter.length() != 1) {
        throw new ComponentConfigurationException(
            "redactionCharacter must be a single character");
    }
}

@Override
public void setup(@NonNull ComponentService serviceProvider) {
    logger = serviceProvider.getLogService();
}

@Override
public void tearDown() {
    logger.info("{0}: Masked a total of {1} values", getName(), count);
    count = 0;
}

```

Some methods and fields elided for the sake of brevity

The relevant details here:

- The *setup* method uses the provided **ComponentService** object to get a **LogService** instance, saving it as *logger*.
- The *mask* method calls the logger's *info* method to write informational messages at random during execution. This kind of "progress" logging may be useful during development but should be removed for algorithms before production deployment.
- The *tearDown* method calls the *info* method of the logger again to record the total number of values masked.

12.4.7 Security considerations

It is important that only well-crafted and trustworthy plugin modules are installed on the Continuous Compliance Engine; otherwise, the security of the appliance and masked data may be compromised. This section contains information for developers on how to ensure that their algorithm plugins function securely, as well as for engine administrators to ensure that only trusted plugins are installed and executed on the engine.

12.4.7.1 Algorithm implementation

This section details a number of security considerations developers should be aware of when creating plugin algorithms for the Continuous Compliance Engine.

12.4.7.1.1 The security sandbox

During execution, all plugin code is sandboxed using the Java Security Manager. Plugins are granted all permissions except for the following non-FilePermission:

Class	Target	Action
java.net.SocketPermission	localhost:-	accept, connect, listen, resolve
java.lang.RuntimePermission	exitVM	
java.lang.RuntimePermission	createClassLoader	
java.lang.RuntimePermission	accessClassInPackage.sun	
java.lang.RuntimePermission	setSecurityManager	
java.security.SecurityPermission	setPolicy	
java.security.SecurityPermission	setProperty.package.access	

With regards to FilePermissions, `read` access is granted to all, though `write` is only allowed for the following directories:

- the masking user's home directory (`System.getProperty("user.home")`)
- the JVM's default temp directory (`System.getProperty("java.io.tmpdir")`)

Please note that both of these locations are shared, so care will need to be taken to avoid collisions.

The set of permissions granted to plugins is static and cannot be modified. To facilitate testing, the same security restrictions are applied when plugins are run using the *maskApp* or *maskScript* utilities in the Masking SDK (with the exception of the `SocketPermission` and all instances of `writeFilePermission`).

12.4.7.1.2 Handling errors

One important aspect of ensuring that an algorithm securely masks sensitive data is proper handling any errors that might occur during algorithm execution.

One particular category of error that might occur is when the input value does not match the format expected by the algorithm. Perhaps an account number masking algorithm is applied to a column containing free-text comments, or an image blurring algorithm is applied to non-image binary data. This is referred to as Non-conformant data. The Algorithm Extension Plugin API defines how an algorithm may trigger the Non-conformant data handling mechanisms built into the Masking Engine.

12.4.7.1.2.1 Reporting non-conformant data

Whenever a Non-conformant input value is encountered, and the algorithm cannot mask it, the algorithm *mask* method should throw an exception of class **NonConformantDataException** supplied by the Masking Plugin API. This triggers the Non-conformant data reporting mechanism of the masking engine. The **String** value used to construct this exception **must not** include the unmasked input value, as this would result in the sensitive value being saved in the Masking Engine logs and made visible in the engine UI. A redacted sample of the Non-conformant data will be saved automatically by the reporting mechanism.

12.4.7.1.2.2 Handling other errors

In general, other code errors should be handled as responsibly as possible by the algorithm implementations, following these guidelines:

- Under no circumstances should the unmasked input values (the *input* argument to the *mask* method) be included in any **Exception** thrown. Exception details are recorded in the engine logs, making them visible to the engine operator and subject to potential disclosure in support bundles. Similarly, exceptions should not simply be re-thrown as **NonConformantDataException** as the original exception's message may contain the sensitive value.
- Whenever possible, configuration problems should be reported in the *validate* or *setup* method, rather than the *mask* method. Waiting until the *mask* method has run to report an error allows the masking job to run, potentially leaving the database table or file partially masked.
- An algorithm should **never** fail in such a way that sensitive values pass through without being masked. In such cases, non-conformant can be reported as described above.

12.4.7.1.3 Logging

The extensibility framework provides the capability for an algorithm to create a [logger](#) (see page 1046) in order to write diagnostic messages to the Continuous Compliance Engine logs. **Under no circumstances should unmasked data (any input argument values to the mask method) be logged.** Logged messages are visible to users via the UI and web API, and may be disclosed in support bundles. It is recommended that production algorithms never log in the *mask* method, for both performance and security reasons.

Additionally, plugin code should *never* read or write any of the *System* input or output streams. Specifically, these are *System.in*, *System.out*, and *System.err*. All logging should be done using the provided logging interfaces.

12.4.7.1.4 Handling secret credentials and keys

The JSON document describing the configuration of each algorithm is stored unencrypted on the Continuous Compliance Engine and made visible to users with access privileges through the UI and web API. For these reasons, secret values of any kind should **never** be part of an algorithm's configuration, regardless of whether the algorithm is user-created or built into a plugin. This includes secret keys, as well as access credentials or API keys that might be used to access remote systems. The only mechanism available as of release 6.0.3.0 that would allow an algorithm to load a sensitive value without the risk of compromise is reading the value from a file stored on an NFS or CIFS [mounted filesystem](#). (see [page 1037](#))



A feature to allow plugins to securely access managed credentials will be added in a future Delphix release.

Secret values (keys) or seeds that drive the output "randomization" an algorithm should not be embedded in the algorithm code. Instead, the algorithm's assigned key should be accessed via the [CryptoService interface](#) (see [page 1044](#)). Static secrets of this kind of risk disclosure should the plugin JAR file be disclosed. There are also risks associated with the algorithm producing the same masking results in all cases, especially if the plugin is to be used for masking by multiple organizations.

12.5 Driver supports

12.5.1 Introduction

As of release 6.0.9.0, the Continuous Compliance Engine supports the installation of driver support plugins, written in Java, that provide tasks to execute before/after masking jobs on extended database connectors. Note that this feature requires creating/updating an uploaded JDBC driver to reference the driver support plugin, which is only possible via the web API. Thus creating an extended database connector using that JDBC driver and a corresponding masking job will allow you to enable whatever available tasks that are implemented by the driver support on the job, which you can do via the web API and UI. This process is detailed further [here](#) (see [page 936](#)). This feature is referred to as Extensible Driver Supports. This section of the documentation details all aspects of masking driver support plugin usage and development. The *Guided Tour* portion of the [workflows section](#) (see [page 1052](#)) walks the user through the basic process of building a simple plugin and installing it onto the Continuous Compliance Engine. Other sections explore topics such as the [DriverSupport interface](#) (see [page 1051](#)) and [service interface](#). (see [page 1036](#))

This documentation assumes the reader has some familiarity with Java development as well as operation of the Delphix Masking Engine via both the UI and Web API Client. The reader should also understand the security requirements associated with any new driver supports being developed.

12.5.1.1 SDK features

The Extensible SDK provides a number of useful functions that aid development of new driver supports for the Continuous Compliance Engine. It is available on the Delphix software [download site](#).³⁹¹

- Creation of empty "skeleton" projects, with build files - the maskScript *init* sub-command
- Testing of the execution of driver support tasks on a database without a masking engine
 - The maskScript *taskExecute* sub-command (**NOTE:** If you want to verify that the **preJobExecute** part of the task was successfully executed, you will want to comment out the reversal of the task in **postJobExecute**, or vice versa. Otherwise, set up your [development environment](#) (see page 1005), add a breakpoint and use the debugger to pause after **preJobExecute** execution.)
- Uploading of plugins to the masking engine - the maskScript *install* sub-command
- Sample driver support for MSSQL extended database connector

12.5.1.2 Getting more information

Several other sources of information are available to aid in plugin development:

- The <http://README.md> file under docs in the Extensible SDK download archive
- The [Masking Plugin API Javadoc](#)³⁹²
- Invoke **maskScript** (located under *sdkTools/bin* in the SDK download) with the -h option for usage help

12.5.2 The DriverSupport Java interface

Any Java class that should be recognized as a driver support plugin must implement the **DriverSupport** interface. The full details of this interface are described in the [Masking Plugin API Javadoc](#)³⁹³.

12.5.2.1 Method overview

This section provides a high-level overview of the methods in the **DriverSupport** interface. For complete details, consult the Masking Plugin API Javadoc included in the Algorithm SDK archive.

- *getTasks* - This method is used to determine the list of available tasks to execute on a corresponding data source. The order in which the tasks are added to the list of tasks indicates the order in which the tasks will be executed on the target data source.

12.5.2.2 The life cycles of driver support objects

The Extensibility framework uses objects classes implementing **DriverSupport** interface for several distinct purposes. These object life cycles are as follows:

³⁹¹ <https://download.delphix.com/folder/574/Delphix%20Product%20Releases/Masking%20SDK>

³⁹² <https://maskingdocs.delphix.com/maskingPluginAPIJavadoc/>

³⁹³ <https://maskingdocs.delphix.com/maskingPluginAPIJavadoc/>

12.5.2.2.1 Plugin discovery

This occurs when the extensibility framework evaluates the capabilities present in a **DriverSupport** class.

1. Java object creation - an object of the driver support class is created
2. *getTasks*- determines all available tasks
 - *getTaskName* - get the name of each task
3. Disposal - the Java object is discarded

12.5.2.2.2 Driver support use

This is the life cycle of a driver object when executing a masking job.

1. Java object creation - an object of the driver support class is created
2. Configuration injection - the masking inventory is used to instantiate a JobInfo object and the database connection is used to instantiate the Connection object (the target SQL connection)
3. *setup* - the *setup* method is called once
4. *preJobExecute* - the *preJobExecute* method is called once before executing the transformation
5. *postJobExecute* - the *postJobExecute* method is called once after executing the transformation
6. Disposal - the Java object is discarded

12.5.3 SDK workflows (Driver supports)

12.5.3.1 Introduction

This section is intended to walk a developer through several workflows using the Delphix Extensible SDK, such as creating a new algorithm or driver support plugin and installing it on a Continuous Compliance Engine.

In order to develop and deploy driver support plugins, you will interact primarily with two tools - the Masking API client, and the Masking Extensible SDK. The Masking API client is a long-standing feature that allows interactive execution of API operations on the Continuous Compliance Engine, while the Masking Extensible SDK is a software package created specifically to aid in driver support development.

12.5.3.2 Outline for a guided tour

By following the steps in the outline below, you can tour the basic functionality provided by the Extensible Driver Support feature and Extensible SDK.

1. Create a driver support plugin by choosing one of two options:
 - a. [Building the sample driver support project](#) (see page 1053)
 - b. [Creating and building your own driver support project](#) (see page 1053)

2. [Test the driver support plugin using maskScript](#) (see page 1057)
3. [Install the newly created plugin on the Continuous Compliance Engine](#) (see page 1065)
4. [View and manage the plugins on a Continuous Compliance Engine using the API Client](#) (see page 1065)

12.5.3.3 Building the sample plugin (SDK workflows/Driver supports)

The Extensible SDK contains a buildable Sample Driver Support Plugin with a functional driver support illustrating the features of the Extensibility Framework. These simple commands build the plugin containing the sample driver support.

Starting from *sdk_root*:

```
$ cd samples
$ ./gradlew :driverSupport:jar
```

This creates the Sample Driver Support plugin JAR file *sdk_root*/samples/build/libs/driverSupport.jar.

The Sample Driver Support project provides a convenient way to see a working example plugin.

i While it is possible to modify these driver supports by changing the Java source and rebuilding the plugin, when starting a new project to develop one, it is highly recommended that you [create your own project](#) (see page 1053) rather than modifying files in the Sample Driver Support project subtree. This will prevent the loss of customizations to the project build files should you chose to install a new version of Masking Extensible SDK over your existing SDK directory.

12.5.3.4 Creating a new project (SDK workflows/Driver supports)

This section describes how to create a brand new Java project for a new masking driver support plugin. We will use the maskScript utility to create a skeleton project and an empty driver support class in that project.

12.5.3.4.1 Creating the project

Before you begin, you'll want to pick a name for your project, and an **empty** directory (outside of the Masking SDK source tree) where your project will be created. Once you've done this, run this **maskScript** command:

```
$ maskScript init -t driverSupport -d <project path> -n <project name> -a <author name> -v <version>
```

For example, this command will create a project named *demoProject* in the *demo-proj* subdirectory of your home directory.

```
$ maskScript init -t driverSupport -d $HOME/demo-proj -n demoProject -a <plugin
author's name> -v <version>
```

For the rest of this section, we'll assume a new project has been created under **proj_dir**. Change your working directory to **proj_dir**. You'll notice that the project is created with a sample driver support file **proj_dir/src/main/java/com/sample/masking/driverSupport/MSSQLDriverSupport.java**. It's possible to build this into a usable plugin by running:

```
$ cd <proj_dir>
$ ./gradlew jar
```



This sample driver support project is not intended to be used in a production environment and is only meant to serve as an example

12.5.3.4.2 Creating a driver support class

Run the maskScript utility to create a skeleton class file:

```
$ cd <proj_dir>
$ maskScript generate driverSupport -p com.delphix.demo -c <class_name> -s .
```

By convention, the class file .java will be created under a sub-directory path based on the package name, so it might be helpful to use the find command to locate it:

```
$ find . -name <class_name>.java
./src/main/java/com/delphix/demo/<class_name>.java
```

The initial content of this file is:

```
package com.delphix.demo;

import com.delphix.masking.api.driverSupport.DriverSupport;
import com.delphix.masking.api.driverSupport.Task;
import com.delphix.masking.api.driverSupport.jobInfo.JobInfo;
import com.delphix.masking.api.provider.ComponentService;
import com.delphix.masking.api.provider.LogService;
import java.sql.Connection;
import java.util.ArrayList;
import java.util.List;

public class <class_name> implements DriverSupport {
```

```

/**
 * This method serves as a directory of Task objects provided by this plugin.
 *
 * @return an ordered list of tasks. The order that tasks are added to the
returning list is the
 *     order that they will be executed in.
 */
@Override
public List<Task> getTasks() {
    // TODO: return list of implemented task objects
    List tasks = new ArrayList<>();
    tasks.add(new ExampleTask());

    return tasks;
}

public class ExampleTask implements Task {
    private JobInfo jobInfo;
    private LogService logService;
    private Connection targetConnection;

    @Override
    public String getTaskName() {
        return "Example Task";
    }

    @Override
    public void setup(ComponentService serviceProvider) {
        this.jobInfo = serviceProvider.getJobInfo();
        this.targetConnection =
serviceProvider.getTargetConnection();
        this.logService = serviceProvider.getLogService();
    }

    @Override
    public void preJobExecute() {
        // TODO: implement code to execute BEFORE masking job runs.
    }

    @Override
    public void postJobExecute() {
        // TODO: implement code to execute AFTER masking job runs.
    }
}
}

```

12.5.3.4.3 Implementing the driver support class

The first thing to notice about the skeleton driver support class is that the `getTasks` method just returns an array of tasks with a single no-op task called `ExampleTask`. This means no actual additional

transaction will be performed on the target data as part of a masking job, so this will certainly need to change.

It is recommended that you change the task class to a name that more accurately reflects what the task does as well as the string returned from the method `getTaskName`. Delete the `TODO` comments in

In order to rebuild the project to generate the driver support plugin JAR, you'll need to first update `settings.gradle` to include the project directory:

```

/*
 * Copyright (c) 2019, 2021 by Delphix. All rights reserved.
 */

pluginManagement {
    resolutionStrategy {
        eachPlugin {
            if ( requested.id.id == 'com.diffplug.gradle.spotless' ) {
                useModule( "com.diffplug.spotless:spotless-plugin-
gradle:$spotlessVer" )
            }
        }
    }
}

rootProject.name = '<proj_dir>'
include 'sdkTools'
include 'algorithm'
include 'assemble'
include 'driverSupport'

```

Then to generate the driver support plugin JAR:

```
$ ./gradlew jar
```

This creates or updates the plugin JAR file `proj_dir/build/libs/.jar`

12.5.3.5 Service discovery (SDK workflows/Driver supports)

Java service discovery is used to determine which classes in the plugin JAR present relevant functionality to the Delphix Masking Engine. When a plugin is loaded, the file `com.delphix.masking.api.plugin.DriverSupport` under `META-INF/services` in the JAR is consulted for a list of classes that implement the **DriverSupport** interface.



When the `maskScript generate` sub-command is used to create a new driver support class, the service discovery metadata file is automatically updated.

If a driver support class is missing from the services file, it will not be usable when the plugin is loaded. It is essentially invisible to the extensibility framework. If a class is mentioned in this file but not present in the JAR, the plugin will fail to load.

12.5.3.6 Executing a driver support task using the SDK (SDK workflows/Driver supports)

It will often be more convenient to use the SDK utilities to test a driver support since this avoids the need to install or update your plugin, create or update a jdbc driver to reference the driver support plugin, and execute jobs on the Delphix Masking Engine. This can be done from the command line using `maskScript`.

12.5.3.6.1 Using `maskScript` to test a driver support task

The `maskScript` utility is non-interactive, which lets you execute a task on a given data source. The jdbc driver, driver support and task are selected using command-line options. This example uses the Sample Driver Support plugin. This plugin can be built using the process described [here](#)³⁹⁴.

Create a task set up json file that corresponds to the specific table and desired database with the contents:

```
{
  "tableMetadata": [
    {
      "name": "Person",
      "schema": "dbo",
      "columns": [
        {
          "name": "column_pk"
        },
        {
          "name": "column_name_1"
        },
        {
          "name": "column_name_2"
        },
        {
          "name": "column_name_3"
        }
      ]
    }
  ],
  "jdbcConnection": {
    "username": "USERNAME",
    "password": "PASSWORD",
    "host": "jdbc:sqlserver://HOST:1433;databaseName=DB_NAME",
    "propertyFilePath": ""
  }
}
```

³⁹⁴ <https://delphixdocs.atlassian.net/continuous-compliance-10-0-0-0/docs/sdk-workflows-building-the-sample-plugin>

Execute the task by indicating the name of the desired task, driver support filepath, task set up json, and jdbc driver:

```
$ maskScript taskExecute -n "Task Name" -j /path/to/driverSupport.jar -c /path/to/task-setup.json -l /path/to/jdbcDriver.jar
```



In order to be usable, the class that implements **DriverSupport** must also be listed in the appropriate service description file. Refer to [this section](#) (see page 1056) for details.

Use any available database management tool like [DbVisualizer](#)³⁹⁵ to connect to the database and verify that the task was successfully executed.

12.5.3.7 Retrieving information about installed plugins (SDK workflows/Driver supports)

The GET endpoints are useful for getting information about plugins. After following the steps in [this section](#) (see page 1065) to install the Sample Driver Support plugin, the GET operation will return (elided for brevity):

```
{
  "pluginId": 9,
  "pluginName": "Sample Plugin",
  "pluginAuthor": "Sample Plugin Author",
  "pluginType": "DRIVER_SUPPORT",
  "originalFileName": "driverSupport.jar",
  "originalFileChecksum":
"f8398c0768ecf7709c6992b3f048f9da8be640285b3ccc968973949ca3cceb02",
  "installDate": "2021-04-21T15:29:01.982+00:00",
  "installUser": 5,
  "builtIn": false,
  "pluginVersion": "1.5.0",
  "pluginObjects": [
    {
      "objectIdentifier": "1",
      "objectName": "Disable Constraints",
      "objectType": "DRIVER_SUPPORT_TASK"
    },
    {
      "objectIdentifier": "2",
      "objectName": "Disable Triggers",
      "objectType": "DRIVER_SUPPORT_TASK"
    }
  ]
}
```

³⁹⁵ <https://www.dbvis.com/>


```

        "objectIdentifier": "3",
        "objectName": "Drop Indexes",
        "objectType": "DRIVER_SUPPORT_TASK"
    }
]
},
...

```

i The `objectIdentifier` field refers to the ID of the task. The order in which the tasks are returned from the API is the order in which the tasks will be executed; the `objectIdentifier` (task ID) has no bearing on the task execution order.

For each plugin, the plugin metadata, including `pluginId`, `pluginName` and `originalFileChecksum` are displayed first. This is followed by a list of tasks included in the plugin.

12.5.4 Service Interface (Driver supports)

12.5.4.1 Introduction

The Extensible Driver Supports framework makes certain services available to the driver support implementation. This prevents the driver support from having to re-implement code to perform certain routine tasks and facilitates seamless integration with the Masking Engine. This functionality is exposed to the driver support class via the **ComponentService** interface.

Whenever a new Masking driver support instance is required for masking, the extensibility framework first injects any saved configuration, then invokes the object's `setup` method. This method is passed a reference to an object that implements **ComponentService**. The driver support's `setup` method can then use this object to access a number of provider methods:

- `getInstanceName` - Get the name of this instance. Because the instance name it is not typically a configurable field in the driver support, the `getName` method will not correctly return the name of an driver support instance, even after JSON configuration injection. This method will always return the correct instance name as known to the Masking Engine.
- `getTargetConnection` - Gets a `java.sql.Connection` that is made using the target database connector.
- `getJobInfo` - Gets a `jobInfo` object, which maps the names of tables, schemas, and columns that are in the masking ruleset.
- `getLogService` - Get a logger object, as described in [this section \(see page 1046\)](#)

i Getting more information

Refer to the *com.delphix.masking.api.provider* package in the [Javadoc](#)³⁹⁶ for detailed information.

12.5.4.2 Accessing masking engine rulesets

The `JobInfo` object represents the database connector's inventory on the masking engine. It contains all of the columns that are going to be masked along with the table and/or schema that they belong to.

12.5.4.2.1 Example driver support task

```
public class DropIndexes implements Task {
    private JobInfo jobInfo;
    private LogService logService;
    private Connection targetConnection;

    @Override
    public String getTaskName() {
        return "Drop Indexes";
    }

    @Override
    public void setup(ComponentService serviceProvider) {
        this.jobInfo = serviceProvider.getJobInfo();
        this.targetConnection = serviceProvider.getTargetConnection();
        this.logService = serviceProvider.getLogService();
    }

    /**
     * This method is to structure all of the columns belonging to the jobInfo.
     *
     * @return A String of comma separated column names.
     */
    private String getCommaSeparatedColumnNames() {
        StringBuilder resultStringBuilder = new StringBuilder();
        for (TableInfo table : jobInfo.getTables()) {
            resultStringBuilder.append(
                table.getColumns().stream()
                    .map(ColumnInfo::getName)
                    .map(this::singleQuoted)
            );
        }
        String commaSeparatedResult = resultStringBuilder.toString();
    }
}
```

³⁹⁶ <https://maskingdocs.delphix.com/maskingPluginAPIJavadoc/>

```

        return commaSeparatedResult.substring(0,
        commaSeparatedResult.length() - 1);
    }
}

```

Some methods have been omitted for brevity.



See the [Javadocs](#)³⁹⁷ for further information on the JobInfo, SchemaInfo, TableInfo and ColumnInfo interfaces.

12.5.4.3 Accessing database server (JDBC)

Driver support plugins will require access to the target database table on which its selected tasks will be run as part of a masking job. The extensible driver support framework allows driver supports to access database servers using JDBC connections, utilizing the existing masking web API. The same connection that is built during the test connection endpoint (`POST /database-connectors/{connector_id}/test`) on the masking engine is the same connection that will be returned by the service provider's **getTargetConnection** method.

12.5.4.3.1 Example driver support task

```

public class DisableTriggers implements Task {
    ...
    private Connection targetConnection;
    ...

    @Override
    public String getTaskName() {
        return "Disable Triggers";
    }

    @Override
    public void setup(ComponentService serviceProvider) {
        this.jobInfo = serviceProvider.getJobInfo();
        this.targetConnection = serviceProvider.getTargetConnection();
        this.logService = serviceProvider.getLogService();
    }

    ...

    @Override
    public void preJobExecute() throws MaskingException {

```

³⁹⁷ <https://maskingdocs.delphix.com/maskingPluginAPIJavadoc/>

```

    long start = System.currentTimeMillis();
    this.triggersOnMaskedTables = findEnabledTriggersOnMaskedTables();
    try (Statement statement = targetConnection.createStatement()) {
        for (Map.Entry<String, String> entry : triggersOnMaskedTables.entrySet())
        {
            String triggerName = entry.getKey();
            String tableName = entry.getValue();
            String disableTriggersStatement =
                String.format(MODIFY_TRIGGERS_SQL, "DISABLE", triggerName,
                    tableName);
            try {
                statement.execute(disableTriggersStatement);
            } catch (SQLException e) {
                String errorMessage = "...";
                logService.error(errorMessage + e);
                throw new MaskingException(errorMessage, e);
            }
        }
    } catch (SQLException e) {
        String errorMessage = "Error creating a statement on target connection.";
        logService.error(errorMessage + e);
        throw new MaskingException(errorMessage, e);
    }
}

```

Some methods have been omitted for brevity.

12.5.4.4 Logging (Service interfaces)

It is possible for a driver support plugin to write information into the app logs, and consequently, Continuous Compliance Engine logs. This is accomplished by using calling the *getLogService* method of the **ServiceProvider** interface provided at the driver support setup. The resulting **LogService** object may be used to make logging entries at various levels of severity. The available log levels are ERROR, WARNING, INFO, and DEBUG.

The log interface is provided to allow for debugging output during driver support development, and for reporting of statistical or similar values detailing the overall operation of the driver support, typically in the *tearDown* method.



Logging Verbosity

Driver support tasks also should not log progress messages or other verbose details, especially from the **preJobExecute** or **postJobExecute** methods, as this will fill the log files with messages and may impact job performance. There is a rate-limiting mechanism that limits the volume of messages each driver support can write over time, but any amount of routine logging is likely to diminish the overall usefulness of the logs by obscuring more important messages.

12.5.4.4.1 Example code

This example is taken from the MSSQL sample Disable Constraints driver support task provided with the SDK:

```

public class DisableConstraints implements Task {
    ...
    private LogService logService;
    ...

    @Override
    public String getTaskName() {
        return "Disable Constraints";
    }

    @Override
    public void setup(ComponentService serviceProvider) {
        this.jobInfo = serviceProvider.getJobInfo();
        this.targetConnection = serviceProvider.getTargetConnection();
        this.logService = serviceProvider.getLogService();
    }

    ...

    @Override
    public void preJobExecute() throws MaskingException {
        long start = System.currentTimeMillis();
        disableConstraints();
        logService.info(
            String.format(
                "Total execution to disable all constraints on masked tables
took %s ms.",
                String.valueOf(System.currentTimeMillis() - start)));
    }

    /** This function enables all constraints on the target database table. */
    private void disableConstraints() throws MaskingException {
        this.enabledConstraints = findEnabledConstraints();
        try (Statement statement = targetConnection.createStatement()) {
            for (ConstraintMetadata constraint : enabledConstraints.values()) {
                logService.info(
                    String.format(
                        "Starting to disable constraint: \"%s\" on table
\"%s\"",
                        constraint.getName(),
                        constraint.getQualifiedTableName()));
                try {
                    String builtSqlStatement =
                        String.format(
                            ALTER_CONSTRAINT_STATEMENT,
                            constraint.getQualifiedTableName(),
                            constraint.getDisableAction(),

```

```

        constraint.getName(),
        ");");
        logService.info(builtSqlStatement);
        statement.execute(builtSqlStatement);
    } catch (SQLException e) {
        String errorMessage =
            String.format(
                "Error disabling constraint: \"%s\" on table
\"%s\".",
                constraint.getName(),
constraint.getQualifiedTableName());
        logService.error(errorMessage + e);
        throw new MaskingException(errorMessage, e);
    }
    logService.info(
        String.format(
            "Finished disabling constraint: \"%s\" on table
\"%s\".",
            constraint.getName(),
constraint.getQualifiedTableName()));
    }
    } catch (SQLException e) {
        String errorMessage =
            String.format(
                "Error creating statement on target connection %s: ",
                targetConnection.getClass());
        logService.error(errorMessage + e);
        throw new MaskingException(errorMessage, e);
    }
    }
    ...

@Override
public void postJobExecute() throws MaskingException {
    long start = System.currentTimeMillis();
    enableConstraints(); // comment this out if testing of the task execution via
the SDK is desired
    logService.info(
        String.format(
            "Total execution to enable all constraints on masked tables
took %s ms.",
            System.currentTimeMillis() - start));
}

```

Many methods and fields elided for the sake of brevity

The relevant details here:

- The *setup* method uses the provided **ComponentService** object to get a **LogService** instance, saving it as *logService*.
- The *disableConstraints* method calls the logger's *info* method to write informational messages at random during execution. This kind of "progress" logging may be useful during development but should be removed for driver supports before production deployment. It also calls the logger's *error*

method in the event of a failure to connect to the data source or otherwise execute the task on the given data source.

12.6 Managing plugins using the API client

The Continuous Compliance Engine's web API includes a *plugin* endpoint for managing plugins:

plugin		Show/Hide	List Operations	Expand Operations
GET	/plugin			Get all plugins
POST	/plugin			Install plugin
DELETE	/plugin/{pluginId}			Delete plugin
GET	/plugin/{pluginId}			Get plugin detail by pluginId
PUT	/plugin/{pluginId}			Update plugin

12.6.1 Displaying information about installed plugins

The GET endpoints are useful for getting information about plugins. After following the steps in [this section](#) (see page 1065) to install the plugin, the GET operation will allow you to retrieve information about the installed plugins. To know what response and information to expect, please see the respective documentation for [driver supports](#) (see page 1058) and [algorithms](#) (see page 1058).

12.6.2 Other plugin endpoint operations

In addition, to GET, the *plugin* endpoint supports the other CRUD operations:

- POST - install a new plugin
- PUT - update an existing plugin
- DELETE - remove a plugin from the system

The POST and PUT operations both require a *fileReference* value representing the plugin file to be installed or updated. These values are the result of using the *fileUpload* endpoint to upload the plugin JAR file to the Masking Engine.

In order to install a new version of this plugin, one could use the PUT operation, or, assuming the algorithm or driver support plugin are not in use, simply DELETE the plugin and POST a new version (or install using the SDK maskScript). Both PUT and DELETE operations require the pluginId value listed for each plugin using the GET operation. Refer to [this section](#) (see page 1003) for details to help the plugin author ensure that new versions of a plugin can successfully install over an existing version using the PUT operation.

12.7 Installing a plugin onto the Delphix masking engine

Once you've successfully built a plugin, it's possible to upload it using the *fileUpload* endpoint in the Masking Engine's API Client, then install the plugin using the *plugin* endpoint. The SDK's **maskScript** includes a sub-

command to automate this process. Replace "admin" with your username if you prefer to install the plugin as another user.

```
$ maskScript install -j <path to plugin JAR> -H <engine hostname> -u admin
```

For example, if you've chosen to build the included Sample Algorithm Plugin in its standard location, and the IP address of your Delphix Masking Engine is 10.0.0.1, this command would install the Sample Algorithm Plugin onto your engine:

```
$ maskScript install -j algorithm/build/libs/algorithm.jar -H 10.0.0.1 -u admin
```

You will be prompted for the Delphix Masking Engine user's password.

Upon success, this command will display the JSON response from the API request, including details about the installed plugin as well as a list of the frameworks and algorithms that were installed.

When installing a plugin using the **maskScript**, the **-n** option may be used to override the plugin name on the Masking Engine. This may be used to install two plugins with the same built-in name on the engine at once (for example, two different versions of the same plugin), but should usually be avoided due to the potential confusion that can result from installing the same plugin on multiple engines with different names.



The Web API Client may also be used to manage the plugins installed on the Delphix Masking Engine, as described in this [section](#) (see page 1065). Also, algorithms support installing [multiple plugins](#) (see page 1020) on a masking engine.

12.8 Secure plugin deployment

It is absolutely vital that only known plugin modules from trusted vendors be installed on the Delphix Masking Engine. A bad plugin may include algorithms that malfunction, possibly by failing to mask data or entering a loop consuming CPU or memory resource. This can lead to job failure, the engine UI becoming unresponsive, or failure to properly mask sensitive data in the case of [algorithms](#) (see page 1048). Plugin execution is sandboxed using the Java Security Manager to guard against malfunctioning code. However, JVM security has historically proven susceptible to allowing untrusted modules to run with the danger of malicious code gaining enhanced or full access to the system running the JVM.

With these considerations in mind, this section describes steps the Delphix Masking Engine administrator can take to ensure that only trusted plugins are executed.

12.8.1 Using roles to restrict plugin installation

This [section](#) (see page 248) describes how to define roles and assign roles to Delphix Masking Engine users. The new profile privilege **Plugins** controls which users are able to install new plugins on to the engine. It is advised that only users that **need** the ability to install plugin modules onto the engine be granted roles that include this privilege.

12.8.2 Verifying the SHA256 hash of installed plugins

When the Masking Web API Client *plugin* endpoint is used to GET the details of a plugin, the field *originalFileChecksum* contains the SHA256 hash of the plugin file installed. This may be compared to a vendor-supplied list of known plugin hashes to verify that a plugin installed on the Delphix Masking Engine has not been tampered with.

For example:

```
{
  "pluginId": 9,
  "pluginName": "demoPlugin",
  "originalFileName": "demoProject.jar",
  "originalFileChecksum":
  "65053d20874ec7929d219b24bdf98ac5b6f7b06ac6bab59712cf78971be135c9",
  "installDate": "2020-06-24T18:19:42.534+0000",
  "installUser": 5,
  "builtIn": false,
  "pluginVersion": "1.0.0",
  "pluginObjects": [
    {
      "objectIdentifier": "demoPlugin:Clobber",
      "objectName": "demoPlugin:Clobber",
      "objectType": "ALGORITHM"
    },
    {
      "objectIdentifier": "demoPlugin:SampleAlgorithm",
      "objectName": "demoPlugin:SampleAlgorithm",
      "objectType": "ALGORITHM"
    }
  ]
}
```

Most UNIX like operating systems provide a way to compute the same hash of a file on the command line.

Apple OSX Example:

```
$ shasum -a 256 demoProject.jar
65053d20874ec7929d219b24bdf98ac5b6f7b06ac6bab59712cf78971be135c9  demoProject.jar
```

Ubuntu Linux Example:

```
$ sha256sum demoProject.jar
65053d20874ec7929d219b24bdf98ac5b6f7b06ac6bab59712cf78971be135c9  demoProject.jar
```

At the time this document was written, there are no known means that would allow an attacker to produce a plugin module with different content, but the same SHA256 hash value of a particular file.

12.9 Terminology

12.9.1 Terminology

Algorithm instance - An algorithm instance is a fully-formed algorithm, which may be assigned to mask data in your masking Inventory. Algorithm instances are uniquely identified by their algorithmName in the Masking API, which is sometimes referred to as "algorithm code" or algorithmCd.

Algorithm component - This term refers to a Java class within an algorithm plugin that implements the MaskingAlgorithm Java interface.

Algorithm framework - This term refers to a family of algorithms on the Delphix Masking Engine. It is necessary to create an instance of an algorithm framework in order to use it - for example, FirstNameLookup is an instance of the Secure Lookup (aka. SL) algorithm framework.

Delphix algorithm SDK - A toolkit authored by Delphix to support the development of algorithm plugins. This includes a CLI for testing algorithms, a skeleton generator for creating empty plugin projects and algorithm classes, and sample algorithms illustrating various use cases.

Delphix masking API - This refers to the set of web APIs offered by the Delphix Masking Engine over HTTP/HTTPS. This API is sometimes referred to as the V5 APIs (referencing their current major version number) or Masking Web API.

Delphix masking plugin API - A package containing the set of Java interfaces that may be implemented in and consumed by a plugin for the Delphix Masking Engine. In order for a plugin to supply algorithms, one or more classes in the plugin must implement the MaskingAlgorithm interfaces provided by this API. This component also includes some common utilities used to load and run plugins on the engine and in the Masking SDK. The JAR containing the appropriate version of the Delphix Masking Plugin API classes has been embedded in the Algorithm SDK zip file.

Plugin - A JAR file containing classes that implement interfaces usable to extend the Delphix Masking Engine. Currently, only masking algorithms may be included in plugins. Plugins also contain self-descriptive metadata to facilitate their use on the engine.

Multi-Column (MC) algorithm - An algorithm that can take as input more than one field and mask one or all the inputted fields, computing the masked value using any of the fields provided. An MC Algorithm can also take in read-only fields that it does not modify but uses to compute a masked value for another field. The type of the input specified for an MC Algorithm is GENERIC_DATA_ROW, though all the fields must specify one of the "standard" masking types (STRING, BIG DECIMAL, etc).