Structured Data Manager

Software Version 23.2.0

API Reference Guide

opentext™

Document Release Date: June 2023
Software Release Date: June 2023

Legal notices

Documentation updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for updated documentation, visit https://www.microfocus.com/support-and-services/documentation/.

Support

Visit the MySupport portal to access contact information and details about the products, services, and support that OpenText offers.

This portal also provides customer self-solve capabilities. It gives you a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the MySupport portal to:

- View information about all services that Support offers
- Submit and track service requests
- Contact customer support
- Search for knowledge documents of interest
- View software vulnerability alerts
- Enter into discussions with other software customers
- Download software patches
- Manage software licenses, downloads, and support contracts

Many areas of the portal require you to sign in. If you need an account, you can create one when prompted to sign in.

Contents

About this document

The OpenText Structured Data Manager API enables you to modify certain behaviors of the software. This guide provides information about:

- the Groovy script API files supplied with Structured Data Manager
- how to run the Groovy script API files

Intended audience

This guide is intended for users running the Groovy script API files.

Prerequisites

Prerequisites for using this product include:

- Knowledge of operating systems
- Database knowledge
- Application knowledge

Related documentation

Document Name	Description
Structured Data Manager Certification Matrix	Provides information about supported Operating Systems, databases, browsers, software integrations and other technology stacks.
Structured Data Manager Concepts Guide	Explains the major concepts of database archiving in general and Structured Data Manager in particular.
Structured Data Manager Developer's Guide	Explains how to use the Designer component to design, build, test, and deploy your archiving projects.
Structured Data Manager Installation Guide	Explains how to install the product.
Structured Data Manager Release Notes	Lists any items of importance that were not captured in the regular documentation.
Structured Data Manager Runtime Guide	Explains how to use the Web Console component to run, monitor, and administer business flows that move data to and from the database.
Structured Data	Explains how to diagnose and resolve errors, and provides a list of

Document Name	Description
Manager Troubleshooting Guide	common errors and solutions.
Structured Data Manager Tutorial	Provides step-by-step instructions to build a sample archiving module, deploy, run, and troubleshoot errors in it.
Structured Data Manager Upgrade Guide	Explains how to upgrade the product and archive schema generated by the earlier versions of the product.
Structured Data Manager Discovery Guide	Explains the purpose, how to install and use Discovery.

Chapter 1: Use the Groovy script API

OpenText provides pre-packaged Groovy script API calls you can use to customize your Structured Data Manager.

In this chapter:

About Groovy scripts in Structured Data Manager

You can run the Groovy script APIs from the command line by entering all the necessary parameters, or you can edit the scripts to create reusable customizations.

The following are instructions for:

- Retrieve valid values
- Modify the createArchiveAccess job
- Modify running jobs
- Register pre-created owner mappings
- Modify cartridges and components

> **TIP:** OpenText strongly recommends implementing API calls using the Groovy script APIs. If you are upgrading from a previous version of the software, and are using Javascript APIs, see the *Structured Data Manager 6.1 API Reference Guide*.

Retrieve valid values

Use the following to query for valid values for your customizations:

Groovy script file name	For a list of required parameters and description of returned values, see
getActions.groovy	getActions
getActionParams.groovy	getActionParams
getActionParamTypes.groovy	getActionParamTypes
getActionTypes.groovy	getActionTypes
getAllJobs.groovy	getAllJobs
getJobParams.groovy	getJobParams

Modify the createArchiveAccess job

Use the following to modify the createArchiveAccess job:

Groovy script file name	For a list of required parameters, see
addAddedDependency.groovy	addAddedDependency
addObjectExclusion.groovy	addObjectExclusion
addObjectOwnerPair.groovy	addObjectOwnerPair
addPrimaryObject.groovy	addPrimaryObject
addTextReplacer.groovy	addTextReplacer
cloneDatabaseLinks.groovy	cloneDatabaseLinks
removeAddedDependency.groovy	removeAddedDependency
removeObjectExclusion.groovy	removeObjectExclusion
removePrimaryObject.groovy	removePrimaryObject
removeTextReplacer.groovy	removeTextReplacer

Modify running jobs

If you need to cancel a job or mark a certain action to be skipped so you can complete or restart a job, then use one of the following Groovy scripts.

> **NOTE:** OpenText recommends contacting OpenText Support before using these scripts.

Groovy script file name	For a list of required parameters, see
forceSkipAction.groovy	forceSkipAction
cancelJob.groovy	cancelJob

Register pre-created owner mappings

If you want to pre-create a history schema or archive database, or set an archive access owner name, you must register the owner mapping prior to installing a cartridge.

Use the following scripts to manipulate the owner mappings:

Groovy script file name	For a list of required parameters, see
getOwnerMappings.groovy	getOwnerMappings
addHistoryOwnerMapping.groovy	addHistoryOwnerMapping
removeHistoryOwnerMapping.groovy	removeHistoryOwnerMapping

NOTE: For information on how to pre-create the History schema, see Custom archive schema or database in the *Structured Data Manager Installation Guide*.

Modify cartridges and components

Use the following to manipulate configuration settings.

Groovy script file name	For a list of required parameters, see
createProductPopertyType.groovy	createProductPropertyType
getProductConfigs.groovy	getProductConfigs
getProductConfigValue.groovy	getProductConfigValue
setProductConfigValue.groovy	setProductConfigValue
getCartridgeConfigs.groovy	getCartridgeConfigs
getCartridgeConfigvalue.groovy	getCartridgeConfigValue
listBusinessFlowCartridges	listBusinessFlowCartridges
setCartridgeConfigValue.groovy	setCartridgeConfigValue

Run Structured Data Manager Groovy scripts

Use the following syntax to run the Groovy scripts. Each Groovy script and its parameters are described in the following chapters.

NOTE: All parameters are case-sensitive.

1. Open a command line window.
2. Run the script using the following job launch syntax:

Operating System	Command syntax
UNIX	`launch_groovyscript.sh -e <environment_name> -f <path> <filename>.groovy [<parameters>]`
Windows	`launch_groovyscript.bat -e <environment_name> -f <path> <filename>.groovy [<parameters>]`

Parameter	Description
environment_	The name of the environment as defined in the Web Console.

Parameter	Description
name	
path	The full path to directory containing the Groovy script. • *<install_directory>*/obt/scripts/usecases/ • *<install_directory>*/obt/scripts where *<install_directory>* is where you installed the software.
filename	The name of the Groovy script file you want to run. For example, forceSkipAction. For more information, see About Groovy scripts in Structured Data Manager.
parameters	Any parameters required for the running of the Groovy script. Parameters are case-sensitive. If the parameter value includes spaces, then put the value in double quotes. If no spaces are in the parameter value, then no quotes are required. To determine the parameters needed, look up the specific API script, or run the script with no parameters. A usage message indicating the required parameters will appear.

3. Type the encryption key when prompted. The encryption key is case sensitive.

Chapter 2: JobConfiguration

Use the following JobConfiguration Groovy scripts to modify database to database archiving jobs.

- getActions, below
- getActionParams, below
- getActionParamTypes, on the next page
- getActionTypes, on the next page
- getAllJobs, on the next page
- getJobParams, on page 13
- getMappedTable, on page 13

The scripts are located in the following directory:

```
<install_directory>/obt/scripts
```

where <install_directory> is where you installed the software.

> **NOTE:** All parameters are case-sensitive.

getActions

This script retrieves the actions for a given job.

Syntax

```
getActions.groovy <jobName>
```

Parameters

Parameter	Type	Description
jobName	String	Job name of the format: batch@job. Value exists and can be determined by using getAllJobs, on the next page.

getActionParams

This script returns the following for every parameter in the list:

```
(name=<parameter_name>, type=<parameter_type>, val=<parameter_value>)
```

Parameter	Description
parameter_name	The name of the parameter.
parameter_type	CONSTANT

Parameter	Description
	PASS_THROUGH GROUP_RUN_ID
parameter_value	The value defined for the parameter.

Syntax

```
getActionParams.groovy <jobName> <actionName>
```

Parameters

Parameter	Type	Description
jobName	String	Job name of the format: batch@job. Value exists and can be determined by using getAllJobs, below.
actionName	String	Name of the action. Value exists and can be determined by using getActions, on the previous page.

getActionParamTypes

This script retrieves all action parameter types defined in the repository.

Syntax

```
getActionParamTypes.groovy
```

getActionTypes

This script retrieves all action types defined in the repository.

Syntax

```
getActionTypes.groovy
```

getAllJobs

This script retrieves all jobs defined in the database. If the partialJobName is null, then it returns all job names defined in the database. If the partialJobName is not null, then it returns a set of job names matching the query criteria.

Syntax

```
getAllJobs.groovy <partialJobName>
```

Parameters

Parameter	Type	Description
partialJobName	String	The query criteria to return the job names selected. If you want all job names, then leave the parameter empty.

getJobParams

This script retrieves all job parameters for the given job name.

Syntax

```
getJobParams.groovy <jobName>
```

Parameters

Parameter	Type	Description
jobName	String	Job name of the format: batch@job. Value exists and can be determined by using getAllJobs, on the previous page.

getMappedTable

This script returns a fully-qualified selection table name for the given table.

Syntax

```
getMappedTable.groovy <BF_Name> <AppsPack_Name> <Catalog_Name> <Schema_Name>
<Table_Name> <Table_Type> [Table_Identifier]
```

Parameters

Parameter	Type	Description
Table_Name	string	The name of the OLTP table.
Table_Type	string	Optional. The type of table. Acceptable values are: • oltp—Contains original table instances in the cartridge. • selectionViewOLTP—Name of the view used during reading of rows from the OLTP table and selectionOLTP table. • selectionOLTP—Contains information about rows that are selected for archiving. • selectionHistory—Contains information on rows from the target database that will be selected by the undo job.

Parameter	Type	Description
		• selectionViewHistory—Name of the view based on the selectionHistory table and history table which will be used during reading of history data on undo/redo. • eligibilityOLTP—Contains information on eligible rules. • row_counts—Contains the count of rows that are archived. • custselOLTP—Contains custom selection information. • exclusionOLTP—Contains information about the rows that are excluded from archive selection.
Table_Identifier	string	The table identifier used in Designer if more than one instance of the same table is used in a model. If this parameter is null, then the first instance of an OLTP table in a cartridge is used in matching the mapped table. **NOTE:** For table types eligibilityOLTP, row_counts, exclusionOLTP, and custselOLTP, all instances of an OLTP table are mapped to the same table. Thus for these table types, this parameter has no effect and could be null.
BF_Name	string	The name of the business flow.
AppsPack_Name	string	The name of the cartridge within the business flow.
Catalog_Name	string	The name of the SQL Server catalog.
Schema_Name	string	The name of the schema for the table.

Chapter 3: ArchiveAccessConfiguration

Use the following Groovy scripts to customize the createArchiveAccess job.

The scripts are located in the following directory:

```
<install_directory>/obt/scripts
```

where <install_directory> is where you installed the software.

> **NOTE:** All parameters are case-sensitive.

Constants

The following constants are used to define, develop, or customize a job.

- TABLE
- VIEW
- PROCEDURE (Oracle only)
- STORED_PROCEDURE (SQL Server only)
- PROXY_TABLE
- SEQUENCE

- PACKAGE
- PACKAGE_BODY
- FUNCTION
- SYNONYM

addAddedDependency

This script adds an additional database object dependency to those returned from the database metadata. On some databases, all dependencies might not be available in the metadata.

For example, on SQL Server, dependencies across catalogs are not available. Also, if database objects are dropped and recreated, the dependencies may be lost.

This script allows you to add the missing metadata.

Syntax

```
addAddedDependency.groovy <dependentCatalog> <dependentSchema> <dependentName>
<dependentType> <referencedCatalog> <referencedSchema> <referencedName>
<referencedType>
```

Parameters

Parameter	Type	Description
dependentCatalog	String	For SQL Server, this is the database name. For Oracle, this is always " " (empty string).
dependentSchema	String	Owner of the dependent object.
dependentName	String	Name of the dependent object.
dependentType	String	Object type. Valid values are defined in Constants, on the previous page.
referencedCatalog	String	For SQL Server, this is the database name. For Oracle, this is always " " (empty string).
referencedSchema	String	Owner of the referenced object.
referencedName	String	Name of the referenced object.
referencedType	String	Object type. Valid values are defined in Constants, on the previous page.

To remove, see removeAddedDependency, on page 24.

addExcludedIndex

This script disables the creation of an index that exists on an OLTP managed table, on the history table.

For example, in a distributed archive, you should not use more than 20 indexes on the history table, as there is an Oracle Optimizer bug that causes performance issues in archive access, if the history table has over 20 indexes.

Indexes that are used for purge performance need not be created on the history tables. You can exclude any additional indexes using this script.

Syntax

```
addExcludedIndex.groovy <catalog> <tableSchema> <tableName> <IndexCatalog>
<IndexSchema> <IndexName>
```

Parameters

Parameter	Type	Description
catalog	String	For SQL Server, this is the database name. For Oracle, this is always " " (empty string).
tableSchema	String	Owner of the dependent object.
tableName	String	Name of the dependent object.
IndexCatalog	String	For SQL Server, this is the database name. For Oracle, this is always " " (empty string).
IndexSchema	String	Owner of the referenced object.
IndexName	String	Name of the referenced object.

addObjectExclusion

This script excludes an object from the schema cloning. Excluded objects are ignored and not created, dropped, or aliased.

If the exclusion is cascaded, then all objects directly or indirectly dependent on the object are excluded.

Syntax

```
   addObjectExclusion.groovy <catalog> <schema> <name> <objectType>
<cascadeExclusion> [<transparencyLayer>] [<skipOwnerCheck>]
```

Parameters

Parameter	Type	Description
catalog	String	For SQL Server, this is the database name. For Oracle, this is always " " (empty string).
schema	String	Owner of the object.
name	String	Name of the object to exclude.
objectType	String	Object type. Valid values are defined in Constants, on page 15.
cascadeExclusion	Boolean	True—to cascade the exclusion for all objects directly or indirectly dependent on the object. False—to exclude only the named object.
transparencyLayer	String	(Optional when skipOwnerCheck parameter is not specified) Name of the transparency layer. Enter " " to leave blank.
skipOwnerCheck	Boolean	(Optional parameter) True—doesn't verify if object owner (schema) is present in the config file. (**<OBT_HOME>/config/exclusionObjectOwners.properties**) False—verifies if the object owner is present in the config file. The default value is False.

To remove, see removeObjectExclusion, on page 24.

addObjectOwnerPair

This script maps the source schema to the archive access schema. By default, the name of the archive access schema is the name of the OTLP schema appended with "_AA".

OLTP schema	Archive Access schema
MyOLTP	MyOLTP_AA

If you want to designate a different name for the Archive Access schema, run this script before installing a cartridge.

The script applies the mapping for the new schema name, and the Deployment Assistant uses the schema name you choose.

Syntax

```
addObjectOwnerPair.groovy <oltpCatalog> <oltpSchema> <aaCatalog> <aaSchema>
[<transparencyLayer>]
```

Parameters

Parameter	Type	Description
oltpCatalog	String	OLTP catalog for SQL Server. For Oracle, this is always " " (empty string).
oltpSchema	String	OLTP schema for Oracle. OLTP user for SQL Server.
aaCatalog	String	Archive access catalog for SQL Server. For Oracle, this is always " " (empty string).
aaSchema	String	Archive access schema for Oracle. Archive access user for SQL Server.
transparencyLayer	String	Name of the transparency layer (optional).

To remove, see removeObjectOwnerPair, on page 25.

addPrimaryObject

This script adds an additional object to the list of primary objects. This script is normally used when you want the object cloned, rather than having a synonym or proxy created for it.

Syntax

```
addPrimaryObject.groovy <catalog> <schema> <name> <objectType>
```

Parameters

Parameter	Type	Description
catalog	String	For SQL Server, this is the database name. For Oracle, this is always " " (empty string).
schema	String	Owner of the object.
name	String	Name of the object.
objectType	String	Object type. Valid values are defined in Constants, on page 15.

To remove, see removePrimaryObject, on page 26.

addTextReplacer

This script adds a text replacement action on the object creation string. The search pattern is a Java regular expression matching a portion of the object creation string. That portion of the string is replaced with the replacement text.

Syntax

```
addTextReplacer.groovy <catalog> <schema> <name> <objectType> <sequence>
<searchPattern> <replacementText> <required> <globalReplace>
```

Parameters

Parameter	Type	Description
catalog	String	For SQL Server, this is the database name. For Oracle, this is always " " (empty string).
schema	String	Owner of the object.
name	String	Name of the object to which you are applying the text modifier.
objectType	String	Object type. Valid values are defined in Constants, on page 15.
sequence	int	Numerical value used to indicate when to run the text replacer. There might be multiple text replacers on an object. This indicates the order they should be applied.
searchPattern	String	A Java regular expression that matches a portion of the object creation string.
replacementText	String	Exact text to replace the text matched.
required	boolean	Values are: • True to throw an exception if the text is not found. • False to ignore any situation where text is not found. OpenText recommends setting this to True.
globalReplace	boolean	Values are: • True to replace all instances of the search string. • False to replace the first instance of the search string.

Example

You can use a text replacer with the USE_ROWIDTOCHAR configuration parameter. There might be some views or stored procedures referencing the ROWID pseudo column on an Oracle table. When

you replace the underlying table with a union view, you are no longer able to access this.

The USE_ROWIDTOCHAR configuration parameter causes createArchiveAccess to add a new column named ROW_ID. ROW_ID selects the ROWID values from each table of the union view. If existing views and packages are referencing ROWID, then they are not referencing ROW_ID.

In this case, it is necessary to add a text replacer to replace ROWID with ROW_ID.

Example

```
<install_directory>/obt/bin/launch_groovyscript.sh -e <env_ID> -f <install_
directory>/obt/scripts/addTextReplacer.groovy
"" SCOTT ORD VIEW 0 "ROWID" "ROW_ID" false false
```

where <install_directory> is where you installed the software.

After successful completion of this script, check the updated OBTSC_TEXT_REPLACER table in the OBT-REP schema, and then run the createArchiveAccess job.

To remove, see removeTextReplacer, on page 26.

cloneDatabaseLinks

The cloneDatabaseLinks script allows you to copy database links from your Oracle OLTP database when you create database transparency. Copying the database links ensures that applications, stored procedures, and functions that use those links continue to function correctly.

The cloneDatabaseLinks script:

- Clones all private database links from all of the source schemas used by a deployed business flow to the corresponding archive access schema. Passwords for all private links passwords are prompted for when the links are created.
- Clones all public database links when the archive access schema is not located on the source schema. If the archive access schema is located on the same server, the public database links already exist and do not need to be cloned.
- Creates only one database link with the same name. Any duplicate links are skipped and a warning message is displayed.
- Clones only links used by a source schema, if a source schema is provided.

Syntax

```
cloneDatabaseLinks.groovy [schema]
```

Parameters

Parameter	Type	Description
schema	String	Optional schema name. If the schema name is included, then only links used by that particular schema are cloned.

generateLaCleanupStmts

The generateLaCleanupStmts script generates clean-up SQL statements. Run this statement on History to clean up incomplete transactions. That is, transactions that were copied to the History table but not deleted from the OLTP database.

Use this script only if the archive job needs to be cancelled after data was copied to the History table through a TABLE_PARALLEL option. The statements can be either TRUNCATE or DELETE statements where archived data in the History table is retained through a temporary table. The generated file name starts with obtpa_cln_hist_run.

Syntax

```
generateLaCleanupStmts.groovy <groupRunningJobId> <directoryName> <cleanupMethod>
<batchSize>
```

Parameters

Parameter	Type	Description
groupRunningJobID	long	Child ID - Archive job ID as defined on the Console Job Summary page.
directoryName	String	Directory name, including the directory path that will contain the generated statements.
cleanupMethod	String	Clean-up method to use. Specify either TRUNCATE or DELETE.
batchSize	int	The number of row sequences processed by the statement. The number of rows processed will be less than or equal to the batchSize. The default is 0. This parameter is only applicable when the clean-up method is DELETE.

Example

```
<install_directory>/obt/bin/launch_groovyscript.sh -e DefEnv -f
../scripts/generateLaCleanupStmts.groovy 5 C:\MYDirectory\cleanup TRUNCATE
```

where <install_directory> is where you installed the software.

The script generates cleanup statements in a file that starts with the name obtpa_cln_hist_ run<Group_Run_Id>_trunc<>_<tableOwner>_<tableName>.sql. The file will be in the directory C:\MYDirectory\cleanup.

A file will be generated for each table that needs to be cleaned. These SQL files should be run on the relocation schema for History.

Example

```
<install_directory>/obt/bin/launch_groovyscript.sh -e DefEnv -f
../scripts/generateLaCleanupStmts.groovy 5 C:\MYDirectory\cleanup DELETE 10000
```

where <install_directory> is where you installed the software.

The script generates cleanup statements in a file that starts with the name obtpa_cln_hist_ run<Group_Run)Id>_del< >_<tableOwner>_<tableName>.sql in the directory C:\MYDirectory\cleanup.

When the script is done, there is a file for each table that needs to be cleaned. These SQL files should be run on the relocation schema for History. Each delete statement will delete 10,000 or less rows per commit.

generateLaPurgeSnapshotStmts

The generateLaPurgeSnapshotStmts script generates purge statements that should be run on the History table to clean up history transactions on the History snapshot table. That is, transactions that where copied to the History snapshot tables when SNAPSHOT_TYPE was set to WITH_HISTORY.

You should run this script if the snapshot type has switched from WITH_HISTORY to WITHOUT_ HISTORY.

Syntax

```
generateLaPurgeSnapshotStmts.groovy <cartridgeName> <fileName>
```

Parameters

Parameter	Type	Description
cartridgeName	String	The name of the cartridge that has its snapshot type set.
fileName	String	The file name and directory path of the file that will contain the generated purge statements.

Example

```
<install_directory>/obt/bin/launch_groovyscript.sh -e DefEnv -f
../scripts/generateLaPurgeSnapshotStmts.groovy ORDER_PURGE $I_TOP/log/x.sql
C:\MYDirectory\purgeSnapShot.sql
```

where <install_directory> is where you installed the software.

The script generates deletes statements in the file purgeSnapShot.sql to purge history rows from the snapshot table that belongs to the cartridge ORDER_PURGE. The purgeSnapShot.sql must be run on the relocation schema of History.

genericSqlConnection

The genericSqlConnection script gets SQL connections to the repository.

Syntax

```
genericSqlConnections.groovy
```

removeAddedDependency

This script removes a dependency added with addAddedDependency.

Syntax

```
removeAddedDependency.groovy <dependentCatalog> <dependentSchema> <dependentName>
<dependentType><referencedCatalog> <referencedSchema> <referencedName>
<referencedType>
```

Parameters

Parameter	Type	Description
dependentCatalog	String	For SQL Server, this is the database name. For Oracle, this is always " " (empty string).
dependentSchema	String	Owner of the dependent object.
dependentName	String	Name of the dependent object.
dependentType	String	Object type. Valid values are defined in Constants, on page 15.
referencedCatalog	String	For SQL Server, this is the database name. For Oracle, this is always " " (empty string).
referencedSchema	String	Owner of the referenced object.
referencedName	String	Name of the referenced object.
referencedType	String	Object type. Valid values are defined in Constants, on page 15.

NOTE: All parameters are case-sensitive.

removeObjectExclusion

This script removes exclusions made with addObjectExclusion.

Syntax

```
removeObjectExclusion.groovy <catalog> <schema> <name> <objectType>
```

Parameters

Parameter	Type	Description
catalog	String	For SQL Server, this is the database name. For Oracle, this is always " " (empty string).
schema	String	Owner of the object.
name	String	Name of the object to exclude.
objectType	String	Object type. Valid values are defined in Constants, on page 15.

NOTE: All parameters are case-sensitive.

removeObjectOwnerPair

This script removes owner pair objects added with addObjectOwnerPair.

Syntax

```
removeObjectOwnerPair.groovy <oltpCatalog> <oltpSchema> <aaCatalog> <aaSchema>
[<transparencyLayer>]
```

Parameters

Parameter	Type	Description
oltpCatalog	String	OLTP catalog for SQL Server. For Oracle, this is always " " (empty string).
oltpSchema	String	OLTP schema for Oracle. OLTP user for SQL Server.
aaCatalog	String	Archive access catalog for SQL Server. For Oracle, this is always " " (empty string).
aaSchema	String	Archive access schema for Oracle. Archive access user for SQL Server.
transparencyLayer	String	Name of the transparency layer (optional).

NOTE: All parameters are case-sensitive.

removePrimaryObject

This script removes a primary object added with addPrimaryObject.

Syntax

```
removePrimaryObject.groovy <catalog> <schema> <name> <objectType>
```

Parameters

Parameter	Type	Description
catalog	String	For SQL Server, this is the database name. For Oracle, this is always " " (empty string).
schema	String	Owner of the object.
name	String	Name of the object.
objectType	String	Object type. Valid values are defined in Constants, on page 15.

NOTE: All parameters are case-sensitive.

removeTextReplacer

This script removes a text replacement action created with addTextReplacer.

Syntax

```
removeTextReplacer.groovy <catalog> <schema> <name> <objectType> <sequence>
```

Parameters

Parameter	Type	Description
catalog	String	For SQL Server, this is the database name. For Oracle, this is always " " (empty string).
schema	String	Owner of the object.
name	String	Name of the object to which you are applying the text modifier.
objectType	String	Object type. Valid values are defined in Constants, on page 15.
sequence	int	Numerical value used to indicate when to execute the text replacer. There might be multiple text replacers on an object.

> **NOTE:** All parameters are case-sensitive.

Chapter 4: RuntimeJobConfiguration

The RuntimeJobConfiguration Groovy script API contains runtime functions.

> **CAUTION:** Use these scripts only on the advice of OpenText Customer Support. These scripts do not clean up the job, can put the system in an unknown state, and cause failures in later job runs.

These functions are used to work around problems in an already installed environment.

- forceSkipAction, below
- cancelJob, on the next page
- interruptJob, on the next page

The scripts are located in the following directory:

```
<install_directory>/obt/scripts
```

where <install_directory> is where you installed the software.

> **NOTE:** All parameters are case-sensitive.

forceSkipAction

This script skips a failed action in a failed job.

> **NOTE:** OpenText recommends using this script only after several attempts have been made at normal error fixing and rerunning of the failed job, and with the help of OpenText Customer Support. This script can put the system in an unknown state and cause it to behave unpredictably.

Before using this script, make sure the job has no running operating system process or database session.

If a job failed due to a hard failure, such as a system or database crash, the Job Monitor might report the job as Running. You must verify the job is no longer running using other methods.

When the job is recovered, the failed action is skipped, and the job continues on to the following action. After this method has been used, the Job Monitor shows the status of the task as Skipped.

Syntax

```
forceSkipAction.groovy <jobName> <actionName> <groupRunningJobId>
```

Parameters

Parameter	Type	Description
jobName	String	Job name of the format: batch@job. Value exists and can be determined by using getAllJobs, on page 12.

Parameter	Type	Description
actionName	String	Name of the action. Value exists and can be determined by using getActions, on page 11.
groupRunningJobID	long	GroupID as defined on the Console Job Summary page.

cancelJob

This script allows you to update the status of a job in the Job Monitor to read “Cancelled.” OpenText recommends running this job only with permission from OpenText Customer Support.

Syntax

```
cancelJob.groovy <groupRunningJobID>
```

Parameters

Parameter	Type	Description
groupRunningJobID	long	GroupID as defined on the Console Job Summary page.

interruptJob

This script allows you to interrupt or pause a currently running job. Use this script to stop a business flow when the Kill Job button does not display in the Web Console. Any children under the GroupID specified by the `groupRunningJobID` parameter are also stopped. The Business Flow is put into the suspended state, allowing you to restart it later.

Syntax

```
interruptJob.groovy <groupRunningJobID>
```

Parameters

Parameter	Type	Description
groupRunningJobID	long	GroupID of the business flow that displays when the user clicks on the running business flow in the Monitor tab of Web Console.

Chapter 5: OwnerMapping

The OwnerMapping Groovy scripts allow you to change the name for the following:

- Oracle history schema (also called the target schema)
- Oracle archive access schema
- SQL Server history database (also called the target database)
- SQL Server archive access database

You should change the names when:

- Your environment requires that a particular name be used.
- You want to pre-create the name.

The OwnerMapping Groovy scripts allow you to register owner mappings before you deploy a business flow. The following scripts allow you to manipulate owner mappings:

- getOwnerMappings, below
- addHistoryOwnerMapping, below
- removeHistoryOwnerMapping , on the next page

The scripts are located in the following directory:

```
<install_directory>/obt/scripts
```

where <install_directory> is where you installed the software.

For information on how to pre-create a History schema or database, see the *Structured Data Manager Installation Guide*.

> **NOTE:** All parameters are case-sensitive.

getOwnerMappings

This call retrieves configured owner mappings from the Repository.

Syntax

```
getOwnerMappings.groovy
```

addHistoryOwnerMapping

This call maps the OLTP owner to the History owner, if you want to pre-create the following:

- History schema for Oracle
- History database for SQL Server

You must add owner mapping for the OLTP database to the pre-created History before cartridge installation. You must ensure that the owner mapping is not already defined for the OLTP database, and that the pre-created History owner name is not in use.

> **NOTE:** The `addHistoryOwnerMapping` and `removeHistoryOwnerMapping` help you to define (or remove) required mapping between the OLTP owner and History owner. While doing so, ensure the following points:
>
> - Don't create mapping directly on the history schema as an owner.
> - Once the mapping is created between OLTP owner and the History owner, it cannot be created again.
> - Once the cartridge or business flow is deployed, the mapping cannot be modified or removed.
> - To modify the mapping, remove existing mapping (using `removeHistoryOwnerMapping`) and use `addHistoryOwnerMapping` as desired.
> - Uninstalling a cartridge does not remove existing owner mappings.

Syntax

```
addHistoryOwnerMapping.groovy <oltpSchema> <historySchema> [<oltpCatalog>]
[<historyCatalog>]
```

Parameters

Parameter	Type	Description
oltpSchema	String	OLTP schema for Oracle. OLTP user for SQL Server.
historySchema	String	History schema for Oracle. History user for SQL Server.
oltpCatalog	String	OLTP catalog for SQL Server only.
historyCatalog	String	History catalog for SQL Server only.

To remove, see removeHistoryOwnerMapping , below

removeHistoryOwnerMapping

This call removes a History owner mapping.

An existing owner mapping can be removed only before a cartridge is deployed. After a cartridge is deployed, owner mappings cannot be modified.

Uninstalling a cartridge does not remove existing owner mappings.

Syntax

```
removeHistoryOwnerMapping.groovy <oltpSchema> [<oltpCatalog>]
```

Parameters

Parameter	Type	Description
oltpSchema	String	OLTP schema for Oracle. OLTP owner for SQL Server.
oltpCatalog	String	OLTP catalog name for SQL Server only.

NOTE: All parameters are case-sensitive.

Chapter 6: Configuration

The following Groovy scripts allow you to manipulate configuration settings for all database to file and database to database cartridges, as well as for individual cartridges.

The scripts are located in the following directory:

```
<install_directory>/obt/scripts
```

where <install_directory> is where you installed the software.

> **NOTE:** All parameters are case-sensitive.

createProductPropertyType

This call creates a new property type for database to database cartridges or database to file cartridges.

Syntax

```
createProductPropertyType.groovy <propTypeName> <description> <scalarType>
<displayType> <mandatory> <isSensitive> <updateable> <copyable> <sequence>
<displayName> <supportsAppspackOverride> <installedProductId> [defaultValue]
[lowValue] [highValue][propertyGroupId] [propertyLovId] [jobParameterXml]
```

> **NOTE:** All parameters must be listed in order. To skip an optional parameter, use double quotes in place of the parameter value. For example, "".

Parameters

Parameter	Type	Description
propTypeName	String	The name of the product property type. The propTypeName and the displayName can be different.

Parameter	Type	Description
description	String	The description of the property type.
scalarType	String	Determines the scalar type for the property. Acceptable values are BOOLEAN, INTEGER, DATE, NUMBER, or STRING.
displayType	String	Determines the display type. Acceptable values are SLIDER or FIELD.
mandatory	String	Y—Makes the property mandatory. N—Makes the property optional.
isSensitive	String	Y—Makes the property case-sensitive. N—Makes the property non-case-sensitive.
updateable	String	Y—Allows the property to be updated. N—Prohibits the property from being updated.
copyable	String	Y—Allows the property to be copied. N—Prohibits the property from being copied.
sequence	String	Sequence number used to order properties of the same group.
displayName	String	The name that will be displayed in the Web Console.
supportsAppspackOverride	String	Y—Allows the property to be overridden for each individual cartridge. N—Prohibits the property from being overridden.
installedProductId	String	The ID of the installed database to database or database to file product. The ID is found by querying the obtrep_installed_products table.
defaultValue	String	Optional. The default value for the created property.
lowValue	String	Optional. The low value for the created property. Use to define the range of an INTEGER scalar type.
highValue	String	Optional. The high value for the created property. Use to define the range of an INTEGER scalar type.
propertyGroupId	String	Optional. The group ID of the property. The ID is found by querying the obtcfg_property_groups table.
propertyLovId	String	Optional. The list of value ID of the property. The ID is found by querying the obtcfg_property_lovs table.
jobParameterXml	String	For internal use only.

Windows example

```
launch_groovyscript.bat -e MyEnv -f
"C:\SDM\SDM760\obt\scripts\createProductPropertyType.groovy" "NewFormQueryOnly"
"For new QUERY_ONLY attribute." "STRING" "FIELD" "Y" "N" "Y" "N" "1" "Set forms
to Query Only." "N" "2" "Y" "" "" "1" "" ""
```

getProductConfigs

This call retrieves the names of database to file or database to database configuration properties.

Syntax

```
getProductConfigs.groovy <product ID>
```

Parameters

Parameter	Type	Description
product ID	String	LA—Database to database archiving. EA—Database to file archiving.

getProductConfigValue

This call retrieves a database to file or database to database configuration property value parameter.

Syntax

```
getProductConfigs.groovy <product ID> <property name>
```

Parameters

Parameter	Type	Description
product ID	String	LA—Database to database archiving. EA—Database to file archiving.
property name	String	The short name of the configuration property. Prints the current product configuration property value. Refer to Property names, on page 38.

setProductConfigValue

This call sets a database to file or database to database configuration property value.

Syntax

```
setProductConfigValue.groovy <product ID> <property name> <value>
```

Parameters

Parameter	Type	Description
product ID	String	LA—Database to database archiving. EA—Database to file archiving.
property name	String	The short name of the configuration property. Prints the current product configuration property value. See Property names, on page 38.
value	String	The value to set.

getCartridgeConfigs

This call lists the names of cartridge configuration properties.

Syntax

```
getCartridgeConfigs.groovy <product ID> <cartridge name>
```

Parameters

Parameter	Type	Description
product ID	String	LA—Database to database archiving. EA—Database to file archiving.
cartridge name	String	The short name of the cartridge.

getCartridgeConfigValue

This call returns the current cartridge configuration property value.

If the value does not exist for the individual cartridge, but a value for the same property name does exist for the default database to file or database to database value, then the corresponding database to file or database to database configuration property value is printed instead.

Syntax

```
getCartridgeConfigValue.groovy <product ID> <cartridge name> <property name>
```

Parameters

Parameter	Type	Description
product ID	String	LA—database to database archiving.

Parameter	Type	Description
		EA—database to file archiving.
cartridge name	String	The short name of the cartridge.
property name	String	The short name of the configuration property. See Property names, on the next page.

listBusinessFlowCartridges

This call lists the cartridges in a business flow and can be paired with setCartridgeConfigValue to update properties for all of the cartridges in a business flow programmatically.

Syntax

```
listBusinessFlowCartidges.groovy BUSINESS_FLOW_NAME
```

Example

```
//Get all cartridges within the current business flow
def bflist = getBusinessFlowService().listBusinessFlowCartridges (BUSINESS_FLOW_
NAME);
```

setCartridgeConfigValue

This call sets a cartridge configuration property value.

Syntax

```
setCartridgeConfigValue.groovy <product ID> <cartridge name> <property name>
<value>
```

Example

```
//Get all cartridges within the current business flow
def bflist = getBusinessFlowService().listBusinessFlowCartridges (BUSINESS_FLOW_
NAME);
//For each cartridge, set the 'Schema Mapping File for Upload' property
bflist.each() { map -> getConfigurationService().setCartridgeConfigValue
(map.cartridgeName,'SchemaMappingsFileNameForUpload', 'd:/qfiniti/Ref.txt')
```

Parameters

Parameter	Type	Description
product ID	String	LA—database to database archiving. EA—database to file archiving.

Parameter	Type	Description
cartridge name	String	The short name of the cartridge.
property name	String	Name of a cartridge parameter, for example, `SchemaMappingsFileNameForUpload`. Refer to Property names, below.
value	String	The value you want to set for the parameter, for example, `d:/qfiniti/Ref.txt)`.

Property names

The following table lists the available cartridge properties:

Property	Type	Description
VERIFY_ROW_COUNTS	Boolean	Indicates whether to perform verification of row counts between the current job and its corresponding selection job. Set to true or false. True is the default value.
ALLOW_MASKED_ DATA_ON_UNDO_ RELOAD	Boolean	Indicates whether to allow copying of masked data into the source database during undo and reload jobs. Set to true or false. False is the default value. If set to true, column data that is masked by a non-reversible masking function will be copied into the source database, possibly resulting in a data corruption for masked columns.
SEL_BATCH_SIZE	Number	Controls the number of driving table rows per transaction. This is used for selection operations that select related parent and child row ids from the source database into selection tables, which are read by data movement operation to move the source rows to the destination. A value of zero causes all rows to be inserted in the same transaction.
BATCH_SIZE	Number	Controls the number of driving table rows per transaction. This is used for data movement operations that operate on related parent and child rows in the same transaction. Used by partitioned and database to file movements only. The total number of rows operated on can be much larger than the value entered, depending on the characteristics of the data. A value of zero disables intermediate commits.
ELIGIBILITY_	Boolean	Globally enables or disables the recording of the

Property	Type	Description
ANALYTICS		cause for excluding records from the archive. Disabling the analytics improves performance. Enabling it allows querying of the analytics tables for information on record eligibility.
NUM_WORKERS	Number	Defines the default maximum number of job workers for tasks that can take advantage of parallelism.
COMPRESSION_ALG	String	Specifies the compression algorithm to apply to the created files. Valid values are NONE and GZIP.
SOURCE_LOCATION	String	Specifies the source database name.
EXTRACT_FORMAT	String	Specifies the extract file format.
PRESERVE_TEMP_ FILES	Boolean	Specifies whether temporary files should be preserved.
UNMASK_ON_UPLOAD	Boolean	Unmask data on upload if mask is reversible.
WRITE_XSD_ SUMMARY	Boolean	Indicates whether to write XSD and Summary files even when there are no data files.
STORAGE_RETENTION	Number	Normally this is the number of days the storage system is to retain the extracted data. Leave empty to specify the storage system's default retention behavior.
NUM_WORK_UNIT	Number	Defines the number of units amongst which the total amount of work will be divided. Each worker picks up a whole unit at a time to ensure clear progress indication and manage the total work in units for the job engine.
MAX_STMT_COUNT	Number	Defines the maximum number of database statements which can be combined in a single query.
MAX_DELETE_COUNT	Number	Defines the maximum number of rows to be deleted in a single request to the database.
POPULATE_INDEX_ TABLES	Number	Specifies whether Index Tables should be populated during Copy from DB to BE. If false, then Delete will not work.
POPULATE_USER_ INDEX_TABLES	Number	Specifies whether User-Index Tables should be populated during Copy from DB to BE. Note: If false, then User-Defined Queries will not work.

Property	Type	Description
UNIFY_MTU_ SELECTIONS	Boolean	Unify selections in multiple table uses (MTU) into one selection table, and remove duplicate rows.
CARDINALITY_VALID	Boolean	Indicates whether to validate that the extracted data does not violate cardinality constraints in the Model instance definition.
CHECKSUM_ALG	Boolean	Indicates whether to run the checksum algorithm on created files. The supported algorithms are MD5 and SHA-256.
CHECKSUM_VALID	Boolean	Indicates whether to validate that the XML file checksums have not changed.
ARCHIVE_ CONSISTENCY_VALID	Boolean	Indicates whether to validate that the archive as a whole is consistent.
XML_SCHEMA_VALID	Boolean	Indicates whether to validate that the XML files do not violate their XML schema.
ROWCOUNT_VALID	Boolean	Indicates whether to verify that rowcounts in the XML files match those in the database.
APPSPACK_DELETE_ VALID	Boolean	Indicates whether to verify that the Cartridge version used during database deletion is the same as the one used during database extraction.
VALIDATE_DATA_ UNCHANGED	Boolean	Indicates whether to validate that data selected for deletion has not changed on OLTP Schema.
VALIDATE_DELETE_ COUNT	Boolean	Indicates whether to verify the number of rows that will be deleted against the expected number.
RUN_OUTPUT_OPTION	String	This parameter controls how much PDM produces diagnostic output in the "PDM server side log."
START_TABLE_ALIAS	String	This parameter specifies to which table the "Start Partition List" belongs. If the table appears multiple times in the model, you must specifically use the alias the designer assigns to the table.
START_PARTITION_ LIST	String	This parameter is for a comma separated list of partitions, all belonging to the same table as specified by "Start Table Alias". This list of partitions will be moved when the cartridge runs.
UploadAddColumn Enabled	Boolean	Enables upload job to add a column to target table if it exists in source but not in target.
UploadModifyColumn Enabled	Boolean	Enables upload to modify a column if the source column differs from the existing target column.

Property	Type	Description
UploadDropColumn Enabled	Boolean	Enables upload to drop a column if it exists in the target table, but not in the source. Enabling this feature in a production environment is not recommended.
NUMBER_OF_ ATTEMPTS	Number	Number of attempts to read a file from S3 if "no such bucket" or "no such key" exception occurs while reading.
GroupFilePrefix	String	Is the prefix for the group data files.
GroupFileSuffix	String	Is the suffix for the group data files.
GroupXSDFilePrefix	String	Is the prefix for the group XML Schema file.
GroupXSDFileSuffix	String	Is the suffix for the group XML Schema file.
SummaryFilePrefix	String	Is the prefix for the summary files.
SummaryFileSuffix	String	Is the suffix for the summary files.
SummaryXSDFilePrefix	String	Is the prefix for the summary XML Schema file.
SummaryXSDFileSuffix	String	Is the suffix for the summary XML Schema file.
SchemaMappingsFile NameForUpload	String	Schema mappings defined in this file will be used to upload to destination database.

Glossary

active database
The database from which you plan to move or copy data. Typically, this database is your online transaction processing (OLTP) or production database. In a two- or three-tiered configuration, the active database resides on tier one and is the source for data movement operations.

active environment
The Web Console views and acts upon only one environment at a time, the active environment. To switch the active environment, you use the Change Active option in the Web Console.

activity
In Designer, a component of a business flow, which is added by using the toolbar. For example, you can add archive and reload activities to your business flow. Note, activities in a business flow are different from what you see at runtime and therefore do not necessarily map directly to what you see in the Web Console.

advanced selection
A method of data selection that discovers all of the interrelated rows from multiple tables and conceptually places them in the same application partition for archiving.

annotation
In Designer, a comment associated with the project, or one of its objects or components. These comments are collected and published in a PDF file when you right click a project or business flow and select Generate Documentation.

application partitioning
The concept of partitioning related rows together during data selection, regardless of whether they are in one or more tables. Application partitioning is unique to Structured Data Manager and contrasts with the more common table partitioning offered by the database management software, which only groups related rows from one table.

archive data store
The location where the data is to be archived. Can be a separate database, separate space on the same database, or an XML file. In a two-tiered configuration, the archive data store resides on tier two and can be a database or XML. In a three-tiered configuration, the archive data store is a database on tier two and XML on tier three, and is the target for data movement operations.

archive query server
The component that provides SQL access to XML database archives.

Consolidated Archive
A managed, scalable repository that consolidates electronic communications, attachments, and files, and provides complete control over corporate information assets, facilitating compliance with internal corporate governance policies and procedures as well as externally mandated laws and regulations.

business flow
A series of activities, such as archive operations and scripts, that run in sequence. You build business flows in Designer.

business flow status
The Web Console shows the last run of each business flow. The states are Complete/Error/Running.

cartridge
An instance of model- or schema-based eligibility criteria used to move or copy data from one location to another. Cartridges capture the application and business rules to ensure referential integrity of the data. For any one model in your project, you may have many cartridges that use it.

chaining table
The lower level table in a many-to-one or a many-to-many relationship between higher level and lower level tables in the model hierarchy.

classification
The Content Manager (formerly Records Manager) classification to be applied to the data moved by Structured Data Manager. This classification specifies where to place the data when it is ingested by Content Manager. For more information, see the Content Manager documentation.

collection
The configuration of a directory location and file pattern to match a set of archived XML files, thus allowing SQL access to the archived data.

comma separated values (CSV)
A database-to-file output format that stores the data as values separated by commas and a metadata file. Each line in the CSV file corresponds to a row in a table. Within a line, fields are separated by commas, each field belonging to one table column. CSV files provide a simple format that many applications can import.

command
Command files or JavaScript files launched by the Web Console on your behalf with status displays.

condition
In Designer, the way you branch your business flow to run or skip an activity based on some criteria.

configuration parameter
A type of parameter that has its values set by an administrator (someone who has repository privileges from the Web Console) through the administrator interface. Typically, this type of parameter represents values that should be changed very infrequently, perhaps only at deployment time.

console user
The Web Console identifies individual users, who are distinct from database users. The properties for a user are User Name, Full Name, Password, Enabled, Description, Email, Phone, and Privileges.

console user name
The login name associated with a Web Console user.

constraint
A column or a list of columns that enables you to identify rows in the database and relate them to one another.

Content Manager
Enterprise document and records management software designed to simplify the capture, management, security, and access to information. Content Manager enables organizations to more easily comply with regulations and corporate policies, and it helps secure information from inappropriate access and misuse.

custom properties
User-created name/value pairs in cartridges and business flows. These values are exposed at runtime as parameters.

customization
A change that an administrator or DBA makes to a project provided by a third party, typically for a packaged application like Oracle PeopleSoft or Oracle E-Business Suite. As long as the customization is allowable by the project, the user can merge the customization into newer revisions of the third party project.

customization mode
A Designer mode that provides visual cues to indicate customizations in the model. In a project with locked files, customization mode is on by default, but you can toggle it on and off from the toolbar in the model editor.

data access cartridge
A cartridge that provides lightweight query access to retired or archived data. Data access cartridges are designed by the archive developer but can be run by business users with no technical expertise.

data masking
The process of replacing private or confidential data during movement with a specified mask. You can choose from pre-defined masks that are part of OpenText or create your own mask. A mask may or may not

be reversible upon reload from the archive data store.

data movement
The method used by Structured Data Manager to actually move data (transactional, bulk or partitioned for database to database, and copy or archive for database to file).

data transparency
The ability to access archived data through your standard application interfaces for data access. Data transparency enables users to access archived data as though it were still in the active database.

database constraint
A constraint that exists in the database and can be discovered and referenced from Designer.

database to database
A movement in which data goes from an active database to an archive database, or separate tablespaces inside the active database. Typically, the archive database is located on cheaper storage devices.

database to file
A movement in which data goes from an active database to a file (XML, JSON or CSV format), which is offline but still accessible through SQL using the archive query server and a client tool of your choice.

deployment assistant
The user interface component within Designer used to deploy or generate business flows.

description
A technical description created by the developer for her own reference. These descriptions do not appear in the generated PDF file for the cartridge or business flow.

Designer
The user interface component used to develop, test, and deploy your archiving solution. Designer is a powerful graphical development environment for archive solutions.

distributed instance
A configuration option for database-to-database archiving where the data you archive is stored on a separate database from the source or active database.

DRE
See Dynamic Reasoning Engine (DRE).

driving table
A driving object is a root of a model hierarchy. Its relationship to the child tables drives the selection of transactions.

dynamic list of values
A list of values for a parameter that obtains its members from a SELECT statement that returns identifiers and labels.

dynamic parameter
A type of parameter that has its value set by a Groovy script that runs at deployment time to obtain a value. For example, this type of parameter can supply the type or version of a database or application, which can be obtained programmatically at deployment time.

Dynamic Reasoning Engine (DRE)
A platform technology that uses high performance pattern-matching algorithms to search for content stored in OpenText repositories. Performs core information operations for contextual analysis and concept extraction, enabling solutions for the categorization, summarization, personalization, hyperlinking, and retrieval of all forms of information.

environment
The source and (optional) target credentials against which you plan to run commands. You can define multiple environments within your installation to identify various source and target databases.

error
One of the ways in which you can interrupt a business flow. Error indicates that the business flow failed for some reason.

exclusive rules
One of the ways in which Structured Data Manager determines whether to include or exclude rows from the archive operation. Exclusive rules require all rows in the constraint table to match for inclusion. Exclusive rules exclude the instance if the condition on any child is false, like STATUS='CLOSED'.

exit
One of the ways in which you can interrupt a business flow. You can exit successfully or with a warning.

export
The way that you save an Structured Data Manager project to an exchange format (.hdp) from the File menu. See also *import*.

export data
The way that a user can send data to CSV format from Preview using the toolbar item.

generate documentation
The process of collecting and grouping all annotations into a PDF file that also describes the business flow or cartridge structure.

history schema
For database-to-database archiving, the schema in the target database where the archived data is stored.

IDOL
See Intelligent Data Operating Layer (IDOL).

import
The way that you transfer projects from exchange format (.hdp) into the Project Navigator. You can also use import to migrate cartridges created in 5.1 to 6.x. See also *export*.

inclusive rules
One of the ways in which Structured Data Manager determines whether to include or exclude rows from the archive operation. Inclusive rules require only one row in the constraint table to match the rule and be included. Inclusive rules include the instance if the condition on any child is true, like PRODUCT_RECALLED='Y'.

indexing cartridge
A cartridge that indexes your data for better searching. For example, you might associate an indexing cartridge with a database-to-file archiving cartridge to improve performance when querying the archive data files.

Intelligent Data Operating Layer (IDOL)
An information processing layer that collects indexed data from connectors and stores it in a structure optimized for fast processing and retrieval, integrating unstructured, semi-structured, and structured information from multiple repositories.

interrupt
The way to stop or pause a business flow (pause, error, exit with warning, exit successfully).

local deployment
The generation and deployment of your cartridge or business flow to an environment on your local, Designer client. Deployment files are generated locally and then deployed to the designated, local environment.

lookup table
A table that contains helpful non-transactional information. For example, non-transactional information could be status definitions, or the name of the sales representative.

managed table
A table in the model that is copied and then purged from the active database by a cartridge. Transactional, chaining, and driving tables in a model are all typically managed tables.

model
A model identifies the tables and table relationships representing a business entity or related business entities. A project can have multiple models. Each model contains a driving table and all of its child and descendent tables.

model compatibility
Each model in your project can have one or more dynamic parameters associated with it to verify the compatibility with the target environment. If the compatibility parameter

returns false, then the cartridge referencing the model will not deploy or run and throw an error. For example, the script could return false for Oracle 10.2 and true for Oracle 11.1 to indicate that a cartridge referencing the model can only deploy and run against Oracle 11.1.

model-based cartridge
A cartridge that moves data based upon a defined data model with relationships. This type of cartridge is typically used for ongoing archive operations.

non-intrusive environment
In a non-intrusive environment, data is archived without an interface schema and a generic JDBC driver is used. A non-intrusive environment enables you to copy or archive data from read-only sources, which is especially helpful in cases where the data is associated with older technologies that might not support basic SQL statements or when the database administrator or company policy prohibits write access to the production environment.

OLTP database
The online transaction processing database that typically is your active or source database.

pause
One of the ways in which you can interrupt a business flow. Pausing suspends the business flow while awaiting operator intervention.

reload
The act of taking data from an archive data store and loading it into the active database.

remote deployment
The generation and deployment of your cartridge or business flow to an environment on a system that is remote from your Designer client. Deployment files are generated locally and then deployed to the designated, remote environment.

repository
The location that holds business flow metadata, product configuration data, and data collected during runtime. The repository can be located on your active database or another logical database.

rule
Qualifications added to the model in order to include or exclude data based on certain criteria. For example, you might add a rule to exclude from archiving any orders that are not yet closed.

runtime parameter
A type of parameter that has its values set by the operator executing the job in Console or on the command line. Typically, this type of parameter represents operational values that tend to change frequently and therefore need to be set each time the job is run.

schema-based cartridge
A cartridge that moves data based upon the database schema rather than a defined data model with relationships. This type of cartridge is typically used for database retirement or the cleanup of orphan tables.

selection
The form of data selection to use (standard or advanced) for choosing data. When deploying a cartridge or adding it to a business flow, you must specify the selection method.

single instance
A configuration option for database-to-database archiving where the data you archive is stored on the same database (Oracle) or the same server (SQL Server) as the source or active database.

source
The location (database) from which you are copying or moving data.

SQL access server
See *archive query server*.

standard selection
A method of data selection that restricts itself to the rows identified by the model. Unlike advanced selection, it does not attempt to traverse related rows across multiple tables.

structured records management
A type of solution that extracts structured data from a source application and moves it into XML format. The XML is then ingested into the corporate records management system for long term management and eventual disposal according to corporate policy.

table use
A database table, view, or synonym that is referenced in Designer, for example, in the model. The same table can be used multiple times in a model. For example, a table could be appear as a transactional table and a lookup table in the same model.

target
The location (database or XML) to which you are copying or moving data.

tier
A level in your database archiving configuration. You can have two- or three-tiered configurations. In a two-tiered configuration, tier one contains your active database and tier two your archive data store, which can be a database or XML. In a three-tiered configuration, tier one contains your active database, tier two an archive database, and tier three XML.

transactional data movement
Transactional movement uses set-based data movement and is the default method of movement.

transactional table
A table that contains information about the business transaction. For example, a transactional table might contain detailed tax or payment information related to each business transaction.

unique identifiers (UIDs)
A 16 hexadecimal identifier calculated based on the content of a Designer file. This value is used to determine if the user has customized key pieces of a project.

unmanaged table
A table in a model that is copied but not purged from the active database by a cartridge. Lookup tables in a model are typically unmanaged tables.

Vertica
Column-oriented SQL database management software for storing and analyzing structured data. Used to manage large, fast-growing volumes of data and provide fast query performance for data warehouses and other query-intensive applications.

virtual constraint
A constraint that you define in Designer that only exists within Structured Data Manager.

Web Console
A browser-based interface where you can create and manage your deployment environments, and deploy, run, administer, and monitor your business flows.

Index

Send documentation feedback

If you have comments about this document, you can contact the documentation team by email. If an email client is configured on this system, click the link above and an email window opens with the following information in the subject line:

Feedback on OpenText Structured Data Manager 23.2.0 API Reference Guide

Add your feedback to the email and click **Send**.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to swpdl.sdm.docfeedback@microfocus.com.

We appreciate your feedback!