AWS Schema Conversion Tool Command Line Interface (CLI) Guide

August 1, 2018

Table of Contents

AWS Schema Conversion Tool Console

AWS Schema Conversion Tool has a console mode that allows user to run everything that he can from UI using a command line interface. This mode allows to reduce memory because of excluded UI side.

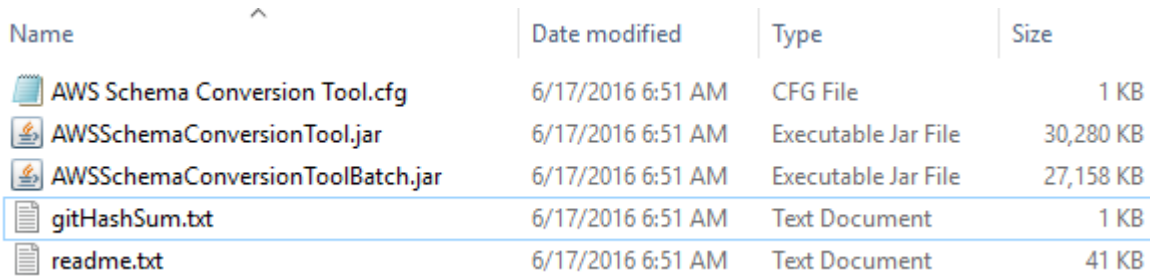
Run console from Windows

Prerequisites:

- Make sure that you have Java JRE installed your PC. The update of JRE is 8u72 and later.
- Have the batch script ready.

The batch script is a list that AWS Schema Conversion Tool uses consequently. It should be defined in xml.

1. Open the terminal, and go to the folder where AWS Schema Conversion Tool is installed.
2. Move to the App folder that is level below AWS Schema Conversion Tool.



Name	Date modified	Type	Size
AWS Schema Conversion Tool.cfg	6/17/2016 6:51 AM	CFG File	1 KB
AWSSchemaConversionTool.jar	6/17/2016 6:51 AM	Executable Jar File	30,280 KB
AWSSchemaConversionToolBatch.jar	6/17/2016 6:51 AM	Executable Jar File	27,158 KB
gitHashSum.txt	6/17/2016 6:51 AM	Text Document	1 KB
readme.txt	6/17/2016 6:51 AM	Text Document	41 KB

3. Run from CMD: `java -Xmx1G -cp AWSSchemaConversionToolBatch.jar com.amazon.sct.batch.BatchExecutor execute "C:\backup\SCTTest\Batch.xml"`

where batch.xml is batch script that was defined for running the batch.

NOTE: When you are defining BatchScript, it is not necessary to use all commands from SampleBatchScript.xml. You can use only what you need. For example, Create Project, Connect to Source, Connect to Target, Transform, and Save. All these commands will be executed by the application one after another in the order defined in batch.xml

NOTE: You can change memory for JM using param -Xmx[JVM] in the bat file.

You also will be able to see process how amazonmigrationstudiobatch processed all commands in the terminal/cmd output:

```
GENERAL MANDATORY  Windows 8.1 6.3 x86 575
GENERAL     INFO  Executing NewProjectCommand ...
GENERAL     INFO    Project name: Batch_Project
GENERAL     INFO    Project directory: c:\Batch\Batch_Project
GENERAL     INFO    Source: ORACLE
GENERAL     INFO    Target: MYSQL
Deleting Regular File: Batch_Project.gb
Deleting Directory: Batch_Project
GENERAL     INFO  NewProjectCommand complete
GENERAL     INFO
GENERAL     INFO  Executing OpenProjectCommand ...
GENERAL     INFO    Project file: c:\Batch\Batch_Project\Batch_Project.gb
GENERAL     INFO  projectModel.getVersion()=1.0
GENERAL     INFO  OpenProjectCommand complete
GENERAL     INFO
GENERAL     INFO  Executing ConnectCommand ...
GENERAL     INFO    Vendor: ORACLE
```

Run console from Linux

Prerequisites:

- Make sure that you have Java JRE installed your PC. The update of JRE is 8u72 and later.
- Have the batch script ready.

NOTE: UI Components needed. If UI Components not installed PDF and CSV report generation will not work correctly

The batch script is a list that AWS Schema Conversion Tool uses consequently. It should be defined in xml.

1. Open the terminal, and go to the folder where AWS Schema Conversion Tool is installed.
2. Move to the App folder that is level below AWS Schema Conversion Tool.

3. Run from terminal: `java -Xmx50G -cp app/AWSSchemaConversionToolBatch.jar com.sct.aws.batch.BatchExecutor execute "batch.xml"`

   where batch.xml is BatchScript that was defined for running the batch.

NOTE: When you are defining BatchScript, it is not necessary to use all commands from SampleBatchScript.xml. You can use only what you need. For example, Create Project, Connect to Source, Connect to Target, Transform, and Save. All these commands will be executed by the application one after another in the order defined in batch.xml

NOTE: You can change memory for JM using param -Xmx[JVM] in the bat file.

You also will be able to see process how amazonmigrationstudiobatch processed all commands in the terminal/cmd output:

```
2017-07-21 17:41:59.614 [   1]     GENERAL MANDATORY Linux 3.10.0-123.el7.x86_64 amd64 Java version: 1.8.0_121
2017-07-21 17:41:59.616 [   1]     GENERAL MANDATORY Used memory: 3.79 MB, Free memory: 25.21 MB, Total memory: 2
2017-07-21 17:41:59.616 [   1]     GENERAL MANDATORY Library list:
    AWSSchemaConversionToolBatch.jar

2017-07-21 17:41:59.632 [   1]     GENERAL INFO    Attached GC listener to Copy
2017-07-21 17:41:59.632 [   1]     GENERAL INFO    Attached GC listener to MarkSweepCompact
2017-07-21 17:42:02.736 [   1]     GENERAL INFO    NewProjectCommand started...
2017-07-21 17:42:02.736 [   1]     GENERAL INFO      Project name: AWS_batch_project
2017-07-21 17:42:02.736 [   1]     GENERAL INFO      Project directory: /home/user/project/AWS_batch_project
2017-07-21 17:42:02.736 [   1]     GENERAL INFO      Source: Oracle
2017-07-21 17:42:02.736 [   1]     GENERAL INFO      Target: PostgreSQL
2017-07-21 17:42:03.245 [   1]     GENERAL INFO    project_settings.project_folder changed
    from: null
      to: /home/user/project/AWS_batch_project
2017-07-21 17:42:03.336 [   1]     GENERAL INFO    project_settings saved.
2017-07-21 17:42:03.337 [   1]     GENERAL INFO    NewProjectCommand finished in 0:00:00.602 sec with memory cons
2017-07-21 17:42:03.338 [   1]     GENERAL INFO    NewProjectCommand statistics:
    GENERAL: 0:00:00.601 sec
```

Batch command file template

To work with console interface need to put all commands inside XML file that should be formatted in specific format:

```
<?xml version="1.0" encoding="UTF-8"?>
<tree>
  <instances>
    <BatchJob>
        PUT COMMANDS HERE
    </BatchJob>
  </instances>
</tree>
```

List of commands in console mode

NOTE: AWS SCT for Console used commands from XML file, and run it step-by-step the same as UI works, so you can't connect to source/target before you create or open a project.

All names of possible values should be the same as you can see in UI.

Possible Vendors list:

```
ORACLE
AURORA_POSTRGRESQL
DB2LUW
AURORA_MYSQL
MSSQL
POSTGRESQL
MYSQL
ORACLEDWH
REDSHIFT
TERADATA
GREENPLUM
NETEZZA
```

Full name to object should be setted using Categories, so for table it could be:
Schema.SOME_SCHEMA.Tables.SOME_TABLE

Create project command

Describes new project creation. Should have all needed properties to create new project.

```
 <NewProjectCommand targetVendor="$targetVendor" projectName="$projectName"
directory="$projectFolder" sourceVendor="$sourceVendor"></NewProjectCommand>
```

Allowed values:

sourceVendor: ORACLE, MSSQL, POSTGRESQL, MYSQL, DB2LUW, ORACLEDWH, TERADATA, NETEZZA, GREENPLUM, REDSHIFT

targetVendor: ORACLE, MSSQL, POSTGRESQL, MYSQL, AURORA_MYSQL, AURORA_POSTGRESQL, REDSHIFT

directory – path on file system. Depends on OS version
projectName – name of project

Open project command

Describes project opening. Should have properties for open project.

```
<OpenProjectCommand projectFilePath="$projectFileFullPath"></OpenProjectCommand>
```

Allowed values:

projectFilePath – path on file system. Depends on OS version

Save project command

Command that save project to predefined location. Project location will be taked from New Project Command or Open Project Command

```
<SaveProjectCommand></SaveProjectCommand>
```

Connect to Source command

Describes source database connection.

```
<ConnectCommand password="$passwordSourse" connectionType="$sourceConnectionType"
vendor="$sourceVendor" origin="$typeSource" serverName="$connectionSourse"
serverPort="$portSourse" userName="$userNameSourse" sid="$vendorSourse"
processMode="$treeNodeLoadMode" recursionDepth="$treeNodeLoadDepth"></ConnectCommand>
```

Allowed values:

origin: SOURCE, TARGET
For Oracle:
connectionType: BASIC_SERVICE_NAME, BASIC_SID, TNS_ALIAS, TNS_CONNECT_IDENTIFIER
NOTE: For TNS_ALIAS need to specify tnsAlias="" and tnsFilePath="". For TNS_CONNECT_IDENTIFIER need to specify tnsConnectIdentifier=""

For Other vendors:

connectionType: BASIC_SID

processMode – value that defines how detailed should be loading done. Possible values: MAIN, EXTENDED

recursionDepth – value that defines how deep need to load tree. Only Integer values allowed. For Example if selected to load all schemas with recurstion Depth 1, it will load only Schema names + 1 level down to table names.

Connect command with SSL mode

Sample of connect command usage with SSL.

```
<ConnectCommand _I_D="2dc585c9-e723-4015-9a6d-de12cd29c1c4" origin="SOURCE"
serverName="52.36.56.194" password="min_privs" serverPort="1522" userName="min_privs"
vendor="ORACLE" useSsl="true" encryptData="false" sslAuthentication="false"
encryptConnection="false" trustServerCertificate="false" requireSsl="false"
verifyServerCertificate="false" tlsAuthentication="false" tnsFilePath=""
serviceName="" tnsAlias="" tnsConnectIdentifier="" sid="ORA12C01"
connectionType="BASIC_SID" keyStoreAlias="K" trustStoreAlias="Tr"/>
```

```
useSsl: true/false
encryptData: true/false
sslAuthentication: true/false
encryptConnection: true/false
trustServerCertificate: true/false
requireSsl: true/false
verifyServerCertificate: true/false
keyStoreAlias: Keystore Alias
trustStoreAlias: Truststore Alias
```

```
For DB2: this option can ignore loading tree errors if you have not enough permissions
```

```
ignoreErrors: true/false
```

Connect to Target command

Describes Target Database Connection.

```
<ConnectCommand password="$passwordSourse" connectionType="$sourceConnectionType"
vendor="$targetVendor" origin="$typeSource" serverName="$connectionSourse"
serverPort="$portSourse" userName="$userNameSourse" sid="$vendorSourse"
processMode="$treeNodeLoadMode" recursionDepth="$treeNodeLoadDepth"></ConnectCommand>
```

Allowed values:

```
For Oracle:
        connectionType: BASIC_SERVICE_NAME, BASIC_SID, TNS_ALIAS,
TNS_CONNECT_IDENTIFIER
        NOTE: For TNS_ALIAS need to specify tnsAlias="" and tnsFilePath="". For
TNS_CONNECT_IDENTIFIER need to specify tnsConnectIdentifier=""

For Other vendors:

        connectionType: BASIC_SID
```

Connect command with SSL mode

Sample of connect command usage with SSL.

```
<ConnectCommand _I_D="2dc585c9-e723-4015-9a6d-de12cd29c1c4" origin="TARGET"
serverName="52.36.56.194" password="min_privs" serverPort="1522" userName="min_privs"
vendor="ORACLE" useSsl="true" encryptData="false" sslAuthentication="false"
encryptConnection="false" trustServerCertificate="false" requireSsl="false"
verifyServerCertificate="false" tlsAuthentication="false" tnsFilePath=""
serviceName="" tnsAlias="" tnsConnectIdentifier="" sid="ORA12C01"
connectionType="BASIC_SID" keyStoreAlias="K" trustStoreAlias="Tr"/>
```

```
useSsl: true/false
encryptData: true/false
sslAuthentication: true/false
encryptConnection: true/false
trustServerCertificate: true/false
requireSsl: true/false
verifyServerCertificate: true/false
keyStoreAlias: Keystore Alias
trustStoreAlias: Truststore Alias
```

Load Tree command

This commad allows to load tree after connection such as at ConnectCommand. This comman load only Schemas level objects, if you want load Schema tree, please use `LoadTreeNodeCommand` according to recursion depth and process mode. It is not mandatory, but could be used for loading metadata tree and then save project with some metadata there.

NOTE: To run this command connect commands should be described before. Connection should be estabilished

```
<LoadTreeCommand origin="$typeSource" processMode
="treeNodeLoadMode"></LoadTreeCommand>
```

To load specific object or category you can use LoadTreeNode command:

```
<LoadTreeNodeCommand recursionDepth="1" origin="SOURCE" processMode ="MAIN"
fullName="Schemas.TD_RULE_TEST" ></LoadTreeCommand>
```

Allowed values:

recursionDepth – value that defines how deep need to load tree. Only Integer values allowed. For Example if selected to load all schemas with recurstion Depth 1, it will load only Schema names + 1 level down to table names.

processMode – value that defines how detailed should be loading done. Possible values: MAIN, EXTENDED

- MAIN – load only object and main properties of it
- EXTENDED – fully load object and all properties of it.

fullName – only for LoadTreeNode command. Defines object address in tree. For Example: Schemas.SchemaName, Shames.SchemaName.Tables.TableName etc.

Create report command

This command runs create report. SCT will analyze selected schema and generate conversion assessment report. To Save Report need to use another commands.

```
<CreateReportCommand sourceFullName="$objectFullPath"></ CreateReportCommand >
```

Allowed values:

sourceFullName: Schemas.SchemaName, Schemas.SchemaName.Tables.TableName etc.

Using several nodes:

```
< CreateReportCommand sourceFullName="$objectFullPath">
       <node sourceFullName="$objectFullPath"/>
       <node sourceFullName="$objectFullPath"/>
</ CreateReportCommand >
```

Create server level objects report command

This command runs create report for server level objects. This command does not have any parameters.

```
< CreateServerLevelObjectsReportCommand/>
```

Save report to PDF command

This command allows to save conversion report in PDF.

NOTE: To use this command on LINUX UI components should be installed

```
<ReportToPdfFileCommand filePath="$filePath" fullName="$objectSourceFullPath " -></
ReportToPdfFileCommand >
```

Allowed values:

filePath – path on file system. Depends on OS version

fullName – defines object address in tree.

For Example: Schemas.SchemaName, Schemas.SchemaName.Tables.TableName etc.

Save server level objects report to PDF command

This command allows to save server level objects report in PDF.

```
<ServerLevelObjectsReportToPdfFileCommand filePath="$filePath"/>

Allowed values:
        filePath – path on file system. Depends on OS version
```

Save report to CSV command

This command allows to save conversion report in CSV.

```
<ReportToCSVFileCommand filePath="$filePath" fullName="$objectSourceFullPath" ></
ReportToCSVFileCommand >

Allowed values:
       filePath – path on file system. Depends on OS version
        fullName – defines object address in tree.
     For Example: Schemas.SchemaName, Schemas.SchemaName.Tables.TableName etc.
```

Save server level objects report to CSV command

This command allows to save server level objects report in CSV.

```
<ServerLevelObjectsReportToCsvFileCommand filePath="$filePath"/>

Allowed values:
        filePath – path on file system. Depends on OS version
```

Convert command

Describes convert operation.

```
<TransformTreeNodeCommand sourceFullName="$objectFullPath"></TransformTreeNodeCommand>

Allowed values:
       sourceFullName – Defines object address in tree.
       For Example: Schemas.SchemaName, Schemas.SchemaName.Tables.TableName etc.
Several nodes conversion example:
<TransformTreeNodeCommand>
   <node sourceFullName="$objectFullPath"/>
   <node sourceFullName="$objectFullPath"/>
</TransformTreeNodeCommand>
```

Apply to target command

This command will apply database to target server.

NOTE: To run this command connect commands should be described before. Connection should be estabilished

```
<WriteToDatabaseCommand fullName="$objectFullPathWriteDB"
overwriteExtensionPack="true" ></WriteToDatabaseCommand>
```

```
      fullName –Defines object address in tree.
      For Example: Schemas.SchemaName, Schemas.SchemaName.Tables.TableName etc.
      overwriteExtensionPack – Extention Pack Overwrite Mode. This attribute is
optional. If no attribute is pointed so application will create Extention Pack if it
isn't exist or will update Extension Pack to the current AWS SCT version. If attribute
overwriteExtensionPack=false so Extention Pack will created if it doesn't exist but if
it exists so no new version will not written. If attribute overwriteExtensionPack=true
so Extention Pack will created if it doesn't exist and if it exists so latest version
will written.
```

Several nodes applying example:

```
<WriteToDatabaseCommand overwriteExtensionPack="true">
  <node fullName="Schemas.pubs_dbo.Tables.authors"/>
  <node fullName="Schemas.pubs_dbo.Tables.discounts"/>
  <node fullName="Schemas.pubs_dbo.Tables.jobs"/>
</WriteToDatabaseCommand>
```

Save as SQL command

This command will save selected objects to SQL file (*.sql)

```
<SaveToSqlCommand fullName="$objectFullPathWriteDB" filePath="$filePath"
origin ="$origin" ></SaveToSqlCommand>
```

```
      fullName –Defines object address in tree.
      For Example: Schemas.SchemaName, Schemas.SchemaName.Tables.TableName etc.

      filePath – path on file system. Depends on OS version.

      origin: SOURCE, TARGET. Default value TARGET.
```

Several nodes saving example:

```
<SaveToSqlCommand filePath="C:\Users\mezentsev.y\Desktop\Test\test.sql">
  <node fullName="Schemas.pubs_dbo.Tables.authors"/>
  <node fullName="Schemas.pubs_dbo.Tables.discounts"/>
  <node fullName="Schemas.pubs_dbo.Tables.jobs"/>
</SaveToSqlCommand>
```

Save as SQL by stage command

This command will save selected objects to several SQL files splitted by objects.

```
<SaveToSqlByStageCommand filePath="C:\Users\mezentsev.y\Desktop\Test\"
fullName="Schemas.pubs_dbo" origin ="$origin"></SaveToSqlByStageCommand>
      fullName –Defines object address in tree.
      For Example: Schemas.SchemaName, Schemas.SchemaName.Tables.TableName etc.

      filePath – directory path on file system. Depends on OS version.

      origin: SOURCE, TARGET. Default value TARGET.


Several nodes saving example:
<SaveToSqlByStageCommand filePath="C:\Users\mezentsev.y\Desktop\Test\" >
  <node fullName="Schemas.pubs_dbo.Tables.authors"/>
  <node fullName="Schemas.pubs_dbo.Tables.discounts"/>
  <node fullName="Schemas.pubs_dbo.Tables.jobs"/>
</SaveToSqlByStageCommand>
```

Refresh database command

This command will reload selected objects from Database.

NOTE: To run this command connect commands should be described before. Connection should be estabilished

```
<ReloadTreeNodeCommand origin="SOURCE" fullName="Schemas.TD_RULE_TEST"
recursionDepth="1" processMode="MAIN"></ReloadTreeNodeCommand>
<ReloadTreeNodeCommand origin="TARGET" fullName="Schemas.TD_RULE_TEST"
recursionDepth="1" processMode="MAIN"></ReloadTreeNodeCommand>

Allowed values:
      fullName –Defines object address in tree.
      For Example: Schemas.SchemaName, Schemas.SchemaName.Tables.TableName etc.

      recursionDepth – value that defines how deep need to load tree.

      processMode – value that defines how detailed should be loading done. Possible
values: MAIN, EXTENDED

      -     MAIN – load only object and main properties of it
      -     EXTENDED – fully load object and all properties of it.
```

Application Conversion

This charpted describes working with Application Conversion feature.

New AppConversion project command

```
<NewProjectAppConversionCommand language="$langName"
appProjectName="$appConversionProjectName" sourceDir="$appConversionProjectFolder"
schemaFullName="$schemaName" knownParameterStyle="$parametersStyle"/>
```

Allowed values:

`language` – Defines source file language. Possible values: JAVA, C++, C#, ANY

`appProjectName` – Defines Application Conversion project name.

`sourceDir` – Defines Application Conversion saving path.

`schemaFullName` – Defines object address in tree.

`knownParamenterStyle` – Defines style of parameters exist at analyzed files. Possible values: Same as in source, Positional (?), Indexed (:1), Indexed ($1), Named (@name), Named (&name), Named ($name)

Convert AppConversion command

This command describes application conversion.

```
<ConvertAppConversionCommand appProjectName="$appConversionProjectName"
convertFullFileNodeName="$analyzedFilePath"/>
```

Allowed values:

`appProjectName` – Defines Application Conversion project name.

`convertFullFileNodeName` – Defines path to analyzed file.

Convert One Statement Command

This command allows to convert statements from file.

```
<ConvertOneStatementCommand knownParameterStyle="Indexed ($1)"
saveFilePath="D:\goldbug-testing-tool9\goldbug-testing-
tool\run\TargetPro\statements\oracle\Redshift_DDL_TestCase.sql"
schemaFullName="Schemas.ORA_DWH" separator="/*NAME" sqlFilePath="D:\goldbug-testing-
tool9\goldbug-testing-
tool\run\references\statement_test_references\oracledwh_redshift\source\ORACLE_DDL.sql
"/>
```

Allowed values:

`sqlFilePath` – path to source SQL script file or folder with SQL script files.

`saveFilePath` - path to target SQL script file or folder to save SQL script files

`separator` – string value for separaiting script in file.

knownParameterStyle – SQL parameter style for target database.

VALUES
Same as in source
Positional (?)
Indexed (:1)
Indexed ($1)
Named (@name)
Named (:name)
Named (&name)
Named ($name)

Apply AppConversion command

This command will apply converted code to application sources.

```
<ConvertAppConversionCommand appProjectName="$appConversionProjectName"
applyFullFileNodeName="$appliedFilePath"/>
```

Allowed values:

appProjectName – Defines Application Conversion project name.

applyFullFileNodeName – Defines path to applied conversion changes file.

Save AppConversion command

This command will save changed sources on file system.

```
<ConvertAppConversionCommand appProjectName="$appConversionProjectName"
saveFullFileNodeName="$appliedFilePath"/>
```

Allowed values:

appProjectName – Defines Application Conversion project name.

saveFullFileNodeName – Defines path to save changes file.

Collect statistics command (for DWH only)

Command to collect statistic for Datawarehouse. This command will load statistic from database

```
<CollectionCommand origin="SOURCE" fullName="$schemaName"
clearFlag="false"></CollectionCommand>
```

Allowed values:

fullName – full path to schema to use it context to search elements. For example: Schemas.SchemaName

clearFlag – defined should we clear old statistic or not. Possible values true/false

Upload statistics command (for DWH only)

Command to upload statistic for Datawarehouse. This command will load statistic from file.

```
<UploadCommand clearFlag="true" filePath="$filePath" fullName="$schemaName"
origin="SOURCE"></UploadCommand>
```

Allowed values:

fullName – full path to schema to use it context to search elements. For example: Schemas.SchemaName

clearFlag – defined should we clear old statistic or not. Possible values true/false

Change setting command

This command is needed to set any setting in project (conversion settings, log settings etc.)

```
<SetProjectSetting settingsType=”global” projectSettingName="converting_settings"
projectSettingValue="true" save=”true” />
```

Allowed values:

settingsType – defines settings type where setting could be found:

- Global – type to change global settings
- Default - type to change default settings
- User – type to change project settings

Setting Name	Possible VALUE
LICENSE_ACCEPTED_FOR_BUILD	"license_accepted_for_build" : string App.readVersion()
LOG_FOLDER_UI	"log_folder" : string path
LOG_FOLDER_CONSOLE	"console_log_folder" : string path
LOG_MAX_FILE_SIZE_MB	"log.max_file_size_mb" : number
LOG_MAX_FILES_COUNT	"log.max_files_count" : number
MESSAGE_LEVEL_GENERAL	"message_level.GENERAL": enum MessageType : TRACE DEBUG INFO WARNING ERROR CRITICAL MANDATORY
MESSAGE_LEVEL_LOADER	"message_level.LOADER": enum MessageType : TRACE DEBUG INFO WARNING ERROR CRITICAL MANDATORY

MESSAGE_LEVEL_PARSER	"message_level.PARSER": enum MessageType : TRACE DEBUG INFO WARNING ERROR CRITICAL MANDATORY
MESSAGE_LEVEL_PRINTER	"message_level.PRINTER": enum MessageType : TRACE DEBUG INFO WARNING ERROR CRITICAL MANDATORY
MESSAGE_LEVEL_RESOLVER	"message_level.RESOLVER": enum MessageType : TRACE DEBUG INFO WARNING ERROR CRITICAL MANDATORY
MESSAGE_LEVEL_TELEMETRY	"message_level.TELEMETRY": enum MessageType : TRACE DEBUG INFO WARNING ERROR CRITICAL MANDATORY
MESSAGE_LEVEL_TRANSFORMER	"message_level.TRANSFORMER": enum MessageType : TRACE DEBUG INFO WARNING ERROR CRITICAL MANDATORY
MESSAGE_LEVEL_TYPEMAPPING	"message_level.TYPEMAPPING": enum MessageType : TRACE DEBUG INFO WARNING ERROR CRITICAL MANDATORY
MESSAGE_LEVEL_UI	"message_level.UI": enum MessageType : TRACE DEBUG INFO WARNING ERROR CRITICAL MANDATORY
DEFAULT_PROJECT_PATH	"project.default_project_path": string path
INSTALL_DIR	"install_dir": string path
APP_FOLDER	"app_folder": string path
PROJECT_FOLDER	"project_folder": string path
DEBUG_MODE	"debug_mode": true false
ORACLE_DRIVER_FILE	"oracle_driver_file": string
MSSQL_DRIVER_FILE	"mssql_driver_file": string
MYSQL_DRIVER_FILE	"mysql_driver_file": string
POSTGRESQL_DRIVER_FILE	"postgresql_driver_file": string
TERADATA_DRIVER_FILE	"teradata_driver_file": string
REDSHIFT_DRIVER_FILE	"redshift_driver_file": string
SHOW_SEVERITY_LEVEL_IN_SQL	"show_severity_level_in_sql": enum SeverityType : CRITICAL HIGH MEDIUM LOW
PARSER_CLEANUP_MODE	"parser.cleanup_mode": enum ParserCleanupMode : AFTER_PARSING ON_LOW_MEMORY NEVER
REPORTING_MODE	"reporting.mode": enum SAVE_REPORT_TO_PDF_MODE : ALL, FIVE
STRATEGY	"strategy": strategy_1 strategy_2 strategy_3
CONVERTING_SETTINGS	"converting_settings": true false
GENERATE_ROW_ID	"generate_row_id": true false
CONVERT_WITHOUT_STATISTIC_REMINDER	"convert_without_statistic_reminder": true false
TREE_SETTINGS_FOR_SHOW	"tree_settings_for_show": `{ "treeSettingsModels":{ "ORACLEDWH":{"hideEmpty":true,"hideEmptyDb":true, "systemSchemas":["ANONYMOUS","ADAMS"], "userDefinedSchemas":[]}, "MSSQLDW":{"hideEmpty":true,"hideEmptyDb":true, "systemSchemas":[],"userDefinedSchemas":[]}, "DB2LUW":{"hideEmpty":true,"hideEmptyDb":true, "systemSchemas":[],"userDefinedSchemas":[]}, "MSSQL":{"hideEmpty":true,"hideEmptyDb":true,

	"systemSchemas":[],"userDefinedSchemas":[]}, "ORACLE":{"hideEmpty":true,"hideEmptyDb":true, "systemSchemas":[],"userDefinedSchemas":[]}} }'

Change Rule command (for DWH only)

This command will change Rules for DWH database conversion.

```
<ChangeRuleCommand rule="rule0" value="$rule0State" weightSK="$rule0weightSK"
weightDK="$rule0weightDK"/>
<ChangeRuleCommand rule="rule1" value="$rule1State" weightSK="$rule1weightSK"
weightDK="$rule1weightDK"/>
<ChangeRuleCommand rule="rule2" value="$rule2State" weightSK="$rule2weightSK"
weightDK="$rule2weightDK"/>
<ChangeRuleCommand rule="rule3" value="$rule3State" weightSK="$rule3weightSK"
weightDK="$rule3weightDK"/>
```

Allowed values:

value – defines if this rule ON or OFF. Possible values true/false

weightSK – integer value of Sort Key weigth

weightDK – integer value of Destribution Key weigth

Working with AWS Profiles

In this charpted describes how to work with AWS Profile inside SCT.

Add AWS Profile Command

This command add or update AWS Profile.

```
<OLTPDMSAddProfileCommand profileName="NewProfile 1"
newProfileName="NewProfileRenamed1" accessKey="sss" secretKey="sec"
region="us-gov-west-1"/>
```

Allowed values:

profileName – default profile name

newProfileName – new name

accessKey – access key

secretKey – secrete key

s3bucket – s3 bucket name

region – region. Allowed values:

Region Name	VALUE
GovCloud	"us-gov-west-1"
US_EAST_1	"us-east-1"
US_EAST_2	"us-east-2"
US_WEST_1	"us-west-1"
US_WEST_2	"us-west-2"
EU_WEST_1	"eu-west-1"
EU_CENTRAL_1	"eu-central-1"
AP_SOUTH_1	"ap-south-1"
AP_SOUTHEAST_1	"ap-southeast-1"
AP_SOUTHEAST_2	"ap-southeast-2"
AP_NORTHEAST_1	"ap-northeast-1"
AP_NORTHEAST_2	"ap-northeast-2"
SA_EAST_1	"sa-east-1"
CN_NORTH_1	"cn-north-1"
CA_CENTRAL_1	"ca-central-1"

Delete AWS Profile Command

This command will remove AWS Profile from SCT

```
<OLTPDMSDeleteProfileCommand profileName="Profile Name"/>
```

Set Project AWS Profile Command

This command defines which profile SCT should use in current project.

```
<OLTPDMSSetProjectProfileCommand profileName="alexk_json"/>
```

Working with Snowball Jobs

In this charpted describes how to work with snowball jobs.

Delete Job Command

This command deletes job.

```
<DeleteJobCommand profileName="profile-1" jobId="id"/>
```

Import Job Command

This command imports job from server to locale profile.

```
<ImportJobCommand profileName="profile-1" jobId="id" ip="127.0.0.1" port="8080"
secretKey="secretKey" accessKey="accessKey"/>
```

Update Job Command

This command updates job.

```
<UpdateJobCommand profileName="profile-1" jobId="id" ip="127.0.0.1" port="8080"
secretKey="secretKey" accessKey="accessKey"/>

All parameters are necessary for this command
```

Working with SQL Plus Scripts

This chapter describes how to work with SQL Plus Scripts.

Load Script Command

This command will load scripts from directory.

```
<LoadScriptCommand directory="D:\scripts"/>

Allowed values:
        directory – input directory path
```

Apply Settings Command

This command will apply settings to scripts.

```
<ApplySettingsCommand script="D:\scripts" currentSchema="ALUMNI" dateFormat="DD-RR-
MM">
        <TnsAlias name="name1" connection="connection1"/>
        <TnsAlias name="name2" connection="connection2"/>
        <TnsAlias name="name3" connection="connection3"/>
</ApplySettingsCommand>

Allowed values:
        script – directory or file path

        currentSchema – name of schema what will be used in transformation

        dateFormat – date format

        Also you can add Tns aliases as sub nodes:
```

```
        name – name of Alias

        connection – connection string
```

Convert Script Command

This command will convert scripts.

```
<ConvertScriptCommand script="D:\scripts"/>

Allowed values:
        script – directory or file path
```

Create Report Script Command

This command will create report for scripts.

```
<CreateReportScriptCommand script="D:\scripts"/>

Allowed values:
        script – directory or file path
```

Save Script Command

This command will save scripts to directory.

```
<SaveScriptCommand script="D:\scripts" rewrite="true" directory="D:\testOutput"/>

Allowed values:
        script – directory or file path

        rewrite – rewrite existed files : true or false

        directory – output directory path
```

Delete Script Command

This command will clear scripts loaded tree, and remove all children nodes.

```
<DeleteScriptCommand script="D:\scripts" origin="TARGET"/>

Allowed values:
        script – directory for deleting in tree

        origin – SOURCE, TARGET
```

Refresh Script Command

This command refresh source scripts tree.

```
<RefreshScriptCommand script="D:\scripts"/>
```

Allowed values:

```
        script – directory for loaded root folder
```

Data migration using DMS for OLAP databases

This charpter describes commands to work with DataMigration feature using DMS. This is available only for OLAP databases.

OLTP DMS. OLTP DMS Create EndPoint Command

This command will create new Endpoint on DMS

NOTE: AWS Profile is required, Connection to DMS required

```
<OLTPDMSCreateEndPointCommand origin="SOURCE" name="EndPoint Name" databaseName="
SALESDB" replicationInstance=" test-replication-instance"/>
```

Allowed values:

```
        origin – SOURCE, TARGET

        name – end point name

        databaseName – name of current DB
```

OLTP DMS. OLTP DMS Create Task Command

This command will create new DMS Task.

NOTE: AWS Profile is required, Connection to DMS required

```
<OLTPDMSCreateTaskCommand taskName="taskNameExampleCons"
                          fullName="Schemas.DMS_MIN_PRIVS"
                          replicationInstance="test-replication-instance"
                          sourceEndPoint="sourceep"
                          targetEndPoint="targetep"
                          includeLOB="Don't include LOB columns"
                          enableLogging="true"
                          prepMode="TRUNCATE_BEFORE_LOAD"
                          maxLogSize="64"
                          migrationType="FullLoad"
                          cdcStartTime="Jul/24/2017/23:44:02"
                          stopTaskAfterFullLoadCompletes="DontStop"/>
```

Allowed values:

taskName –nask name

fullName – node full name for task need to create

replicationInstance – instance name

sourceEndPoint – source end point

targetEndPoint – target end point

enableLogging – enable task true/false

includeLOB – Allowed values:

VALUES
"Don't include LOB columns"
"Full LOB mode"
"Limited LOB mode"

prepMode – Allowed values:

VALUES
DO_NOTHING
DROP_AND_CREATE
TRUNCATE_BEFORE_LOAD

migrationType – Allowed values:

VALUES	NAME
"FullLoad"	Migrate existing data
"FullLoadAndCdc"	Migrate existing data and replicate ongoing changes
"Cdc"	Replicate data changes only

cdcStartTime – Date format MMM/dd/yyyy/HH:mm:ss

stopTaskAfterFullLoadCompletes - Allowed values:

VALUES	NAME
"DontStop"	Don't stop
"StopBeforeApplyingCachedChanges"	Stop Before Applying Cached Changes

"StopAfterApplyingCachedChanges"	Stop After Applying Cached Changes

OLTP DMS. OLTP DMS Delete Task Command

This command will delete DMS Task

NOTE: AWS Profile is required, Connection to DMS required

```
<OLTPDMSDeleteTaskCommand taskName="taskNameExampleCons"/>
```

OLTP DMS. OLTP DMS Start Task Command

This command will start DMS Task

NOTE: AWS Profile is required, Connection to DMS required

```
<OLTPDMSStartTaskCommand taskName="taskNameExampleCons"/>
```

Data migration using Locale & DMS for OLTP databases

This charpter describes commands to work with Data Migration feature using Locale & DMS. This is available only for OLTP databases.

OLTP. OLTP Register Agent Command

This command will create new replicate agent.

```
<OLTPRegisterAgentCommand
      description="agentDescription"
      hostName="127.0.0.1"
      port="5556"
      password="password"
      useSsl="true"
      certificateKeyName="$certificateKeyName"
      certificateTrustName="$certificateTrustName"/>
```

OLTP. OLTP Refresh Agent Command

This command will refresh an agent by description.

```
<OLTPRefreshAgentCommand description="desc" />
```

OLTP. OLTP Refresh All Agents Command

This command will refresh all agents.

```
<OLTPRefreshAllAgentsCommand/>
```

OLTP. OLTP Unregister Agent Command

This command will unregister an agent.

```
<OLTPUnregisterAgentCommand description="agentDescription" />
```

OLTP. OLTP Wait Agent Command

This command will be waiting for equality of agent status and required status.

```
<OLTPWaitAgentCommand status="active" description="agentDescription" timeout="20"/>
```

description – agent will be choosen by this description

status allowed values :

Status	VALUE
ACTIVE	"active"
RUNNING	"running"
STOPPED	"stopped"
FAILED	"failed"
UNKNOWN	"unknown"
RECOVERING	"recovering"
WAITING_FOR_RECOVERY	"waiting_for_recovery"
FAILED_TO_RECOVER	"failed_to_recover"

timeout – maximal waiting time in seconds

OLTP. OLTP Create Task Command

This command will create new Locale & DMS task.

```
<OLTPCreateTaskCommand
         taskName="migrationTask"
         agent="Agent"
         replicationInstance="migrationhubinstance"
         description="teskDescription"
         loggingEnabled="true"
         role="Admin"
         targetTablePrepMode="DO_NOTHING"
         migrationMode="FullLoad"
         fullName="Schemas.MigrationSchema.Tables.SampleTable"
         useSnowBall="false"
         jobName="SnowballJobName"
```

```
          jobS3Bucket="jobS3Bucket"
          S3Bucket="s3/:url"/>
```

Allowed values:

taskName – task name

agent – agent description

replicationInstance – instance name

description – task description

loggingEnabled – enable task true/false

role – DMS access IAM role to opetate on S3 data

targetTablePrepMode – Allowed values:

VALUES
DO_NOTHING
DROP_AND_CREATE
TRUNCATE_BEFORE_LOAD

migrationMode – Allowed values:

VALUES	NAME
"FullLoad"	Migrate existing data
"FullLoadAndCdc"	Migrate existing data and replicate ongoing changes

fullName – node full name for task need to create

useSnowBall – enable snowball true/false

jobName – set job from existing by name

jobS3Bucket – job's s-3-bucket

S3Bucket – s-3-bucket(If not sent – get s-3 from current profile)

```
<OLTPCreateTaskCommand
          taskName="migrationTask"
          agent="Agent"
          replicationInstance="migrationhubinstance"
          description="teskDescription"
          loggingEnabled="true"
          role="Admin"
          targetTablePrepMode="DO_NOTHING"
          migrationMode="FullLoad"
```

```
          useSnowBall="false"
          jobName="SnowballJobName"
          jobS3Bucket="jobS3Bucket"
          S3Bucket="s3/:url">
        <node fullName ="Schemas.DBA_DEV.Tables.TABLE1"
        <node fullName ="Schemas.DBA_DEV.Tables.TABLE2"
</ OLTPCreateTaskCommand >
```

OLTP. OLTP Delete Task Command

This command will delete Locale & DMS task.

```
<OLTPDeleteTaskCommand taskName="taskName" />
```

Allowed values:

```
          taskName – task name
```

OLTP. OLTP Refresh All Tasks Command

This command will refresh all Locale & DMS tasks.

```
<OLTPRefreshAllTasksCommand/>
```

OLTP. OLTP Refresh Task Command

This command will refresh Locale & DMS task.

```
<OLTPRefreshTaskCommand taskName="taskName" />
```

Allowed values:

```
          taskName – task name
```

OLTP. OLTP Restart Task Command

This command will restart Locale & DMS task.

```
<OLTPRestartTaskCommand taskName="taskName" />
```

Allowed values:

```
          taskName – task name
```

OLTP. OLTP Resume Task Command

This command will resume Locale & DMS task.

```
<OLTPResumeTaskCommand taskName="taskName" />
```

Allowed values:

```
        taskName – task name
```

OLTP. OLTP Show Log Task Command

This command will save log for Locale & DMS task and its agent.

```
<OLTPShowLogTaskCommand taskName="taskName" folderPath="folderPath"/>
```

Allowed values:

```
        taskName – task name

        folderPath – directory path for saving log files in csv format
        Example: C:\MyFolder
```

OLTP. OLTP Start Task Command

This command will save log for Locale & DMS task and its agent.

```
<OLTPStartTaskCommand taskName="taskName"/>
```

Allowed values:

```
        taskName – task name
```

Warning: After the task is deleted, it will be launched endpoints cleaner. Make sure you have enough time to this action. If application will be closed earlier some endpoints can remain unremoved (use WaitCommand for resolve this problem).

OLTP. OLTP Stop Task Command

This command will pause Locale & DMS task.

```
<OLTPStopTaskCommand taskName="taskName" />
```

Allowed values:

```
        taskName – task name
```

OLTP. OLTP Test Task Command

This command will test connection for Locale & DMS task.

```
<OLTPTestTaskCommand taskName="taskName" />
```

Allowed values:

```
        taskName – task name
```

OLTP. OLTP Wait Task Command

This command will will be waiting for equality of Locale & DMS task status and required status.

```
<OLTPWaitTaskCommand taskName="taskName" status="ready" timeout="180" taskType="local"
/>
```

Allowed values:

taskName – task name

status – waiting for task status value

Status	VALUE
CREATING	"creating"
RUNNING	"running"
STOPPED	"stopped"
STOPPING	"stopping"
DELETING	"deleting"
FAILED	"failed"
STARTING	"starting"
READY	" "ready"
MODIFYING	"modifying"
TESTING	" "testing"
UNKNOWN	" "unknown"
PENDING	"pending"
PAUSED	"paused"
ERROR	"error"
NOT_EXIST	"not_exist"
RECOVERY	"recovery"

timeout – maximal waiting time in seconds

taskType – type of task

Task type	VALUE
MAIN	"main"
LOCAL	"local"
REMOTE	"remote"

Working with Mapping Rules

This chapter describes how to work with mapping rules.

Create RuleMap Command

This command creates conversion rules. It will affect conversion.

```
<CreateRuleMapCommand>
  <rule objectType="objectType1"
        action="CONVERT_LOWERCASE"
        ruleName="ruleName2"
        targetName="targetName3"
        active="true"
        databaseName="databaseName4"
        schemaName="schemaName5"
        tableName="tableName6"
        columnName="columnName7"
        sourceName="sourceName8"
        newDataType="TIMESTAMP WITHOUT TIME ZONE"
        newDataTypeLength="newDataTypeLength11"
        newDataTypePrecision="newDataTypePrecision12"
        newDataTypeScale="newDataTypeScale13"/>
        oldDataType="TIMESTAM"
        oldDataTypeLength="oldDataTypeLength14"
        oldDataTypePrecision="oldDataTypePrecision15"
        oldDataTypeScale="oldDataTypeScale16"/>
         <rule … />
         <rule … />
         <rule … />
</CreateRuleMapCommand>
```

Allowed values:

objectType – Allowed values:

VALUES
"table"
"schema"
"column"

action – rule action

Name	VALUE
rename	"RENAME"
add prefix	"ADD_PREFIX"
add suffix	"ADD_SUFFIX"
remove prefix	"REMOVE_PREFIX"
remove suffix	"REMOVE_SUFFIX"
change data type	"CHANGE_DATA_TYPE"
replace suffix	"REPLACE_SUFFIX"
replace prefix	"REPLACE_PREFIX"
convert uppercase	"CONVERT_UPPERCASE"
convert lowercase	"CONVERT_LOWERCASE"
move to	"MOVE_TO"

ruleName – rule name

targetName – target name (suffix, prefix, name)

active – on/off rule. Values true/false

databaseName – database name like …

schemaName - schema name like …

tableName – table name like …

columnName - column name like …

sourceName – source name (from).

newDataType – new data type (see documentation DataTypes.xlsx)

newDataTypeLength – new data type length value

newDataTypePrecision – precision for new data type (depends of newDataType)

newDataTypeScale – scale for new data type (depends of newDataType)

oldDataType – old data type

oldDataTypeLength – old data type length value

oldDataTypePrecision – precision for old data type (depends of oldDataType)

```
        oldDataTypeScale - scale for old data type (depends of oldDataType)
```

Export Script for DMS Command

This command exports mapping rules to JSON.

```
<ExportScriptForDmsCommand filePath="C:\outputfile"/>
```

Allowed values:

```
        filePath - path on file system. Depends on OS version.
```

Import Script into SCT Command

This command imports mapping rules into SCT.

```
<ImportScriptIntoSctCommand filePath="C:\inputfile.json"/>
```

Allowed values:

```
        filePath - path on file system. Depends on OS version.
```

Data migration using AWS Data Extractors. Available for DWH

This charpter describes Data Extraction using AWS Data Extractor.

OLAP. OLAP Register Data Extractor Command

This command will register new Data Extractor agent.

```
<OLAPRegisterDataExtractorCommand description="desc" hostName="127.0.0.1" port="5556"
useSsl="false" certificateKeyName="name" certificateTrustName="name" timeout="200"/>
        certificateKeyName - set key certificate from existing by name.

        certificateTrustName - set trust certificate from existing by name.

        Timeout - set the timeout in msec. Default timeout=20sec. (if attribute is not
present).
```

OLAP. OLAP Modify Agent Command

This command will modify data extractor agent

```
<OLAPModifyAgentCommand oldDescription="desc1" description="desc2"
hostName="127.0.0.1" port="5556" useSsl="false" certificateKeyName="name"
certificateTrustName="name" timeout="200"/>
        certificateKeyName - set key certificate from existing by name.

        certificateTrustName - set trust certificate from existing by name.
```

```
        Timeout - set the timeout in msec. Default timeout=20sec. (if attribute is not
present).
```

OLAP. OLAP Refresh Agent Command

This command will ping registered agent

```
<OLAPRefreshAgentCommand description="desc"/>
```

OLAP. OLAP Refresh All Agents Command

This command will ping all registered agents

```
<OLAPRefreshAllAgentsCommand/>
```

OLAP. OLAP Unregister Agent Command

This command will unregister agent

```
<OLAPUnregisterAgentCommand description="desc"/>
```

OLAP. OLAP Create Task Command

This command will create new local datamigration task.

```
<OLAPCreateTaskCommand fullPath="Schemas.DBA_DEV.Tables.DATATYPE_ALL" taskName="Tname"
sourceDbPassword="*******" timeout="200" useSnowball="true" awsJob="JobName"
migrationMode="EXTRACT_ONLY" s3bucket="bucketName" s3fodler="folder/Path"
loggingEnabled="true" loggingLevel="TRACE"/>
        Optional:
        Timeout - set the timeout in sec. Default timeout=30sec (if attribute is not
present).
        LoggingLevel  - set the logging level
        LoggingEnabled  - enables logging


Allowed values:
        migrationMode - EXTRACT_ONLY, EXTRACT_UPLOAD, EXTRACT_UPLOAD_COPY

<OLAPCreateTaskCommand fullPath="Schemas.DBA_DEV.Tables.DATATYPE_ALL" taskName="Tname"
timeout="200" useSnowball="true" awsJob="JobName" migrationMode="EXTRACT_ONLY"
s3bucket="bucketName" s3fodler="folder/Path">
      <node fullPath="Schemas.DBA_DEV.Tables.TABLE1"/>
      <node fullPath="Schemas.DBA_DEV.Tables.TABLE2"/>
</OLAPCreateTaskCommand>
```

OLAP. OLAP Start Task Command

This command will start Extractors task.

```
<OLAPDMSStartTaskCommand taskName = "Tname" timeout="200" waitForComplete="false"/>
        Optional:
        Timeout - set the timeout in sec. Default timeout=30sec (if attribute is not
present).
```

OLAP. OLAP Refresh All Tasks Command

This command will refresh all Extractors tasks.

```
<OLAPDMSRefreshAllTasksCommand/>
```

OLAP. OLAP Stop Task Command

This command will stop specified extractor task.

```
<OLAPDMSStopTaskCommand taskName = "Tname"/>
```

OLAP. OLAP Resume Task Command

This command will resume specified extractor task.

```
<OLAPDMSResumeTaskCommand taskName = "Tname"/>
```

OLAP. OLAP Restart Task Command

This command will restart specified extractor task.

```
<OLAPDMSRestartTaskCommand taskName = "Tname"/>
```

OLAP. OLAP Refresh Task Command

This command will refresh all Extractors task.

```
<OLAPDMSRefreshTaskCommand/>
```

OLAP. OLAP Delete Task Command

This command will delete Extractors task.

```
<OLAPDMSDeleteTaskCommand taskName = "Tname"/>
```

OLAP. OLAP Wait Task Command

This command will wait for specified stage status on extractor task.

```
<OLAPDMSWaitTaskCommand taskName = "Tname" timeout = "5400"
```

```
stage = "EXTRACT" status = "STOPPED"/>
Allowed values:
        stage - EXTRACT, UPLOAD, TRANSFER, COPY
        status - RUNNNING, FAILED, PENDING, CREATED, STOPPED, COMPLETED, STOPPING,
      CREATING, DELETING, STARTING, RESUMING, DELETED, RESTARTING
```

OLAP. OLAP Finalize Agent Session Command

This command will finalize Extractors Agent Session

```
<OLAPFinalizeAgentSessionCommand />
```

OLAP. Virtual Partitioning

Virtual partitions are used in case it is necessary to split a large table into separate parts according to a given criterion for further parallel loading into the target database. Virtual partitioning should work only with non-partitioning tables.

Note - table should be previously loaded to create virtual partition.

OLAP. Create Partition By List Command

This command will create virtual partition. This type of partitioning assigns rows to partitions based on column values. You can load information of values from .csv files.

```
<OLAPCreatePartitionByListCommand fullPath="Schemas.DBA_DEV.Tables.DATATYPE_ALL"
columnName="Cname" timeout="200" values="1,500,1000" loadFromFile="C:\list.csv"
includeOtherValues="true"/>
Allowed values:
        loadFromFile – path to CSV file.
```

OLAP. Create Partition By Range Command

This command will create virtual partition. This type of partitioning assigns rows to partitions based on column values falling within a given range. You can load information of values from .csv files or specify it in “values” attribute.

```
<OLAPCreatePartitionByRangeCommand fullPath="Schemas.DBA_DEV.Tables.DATATYPE_ALL"
columnName="Cname" timeout="200" values="1,500,1000" loadFromFile="C:\list.csv"/>
Allowed values:
        loadFromFile – path to CSV file.
```

OLAP. Create Partition By Auto Split Command

This command will create virtual partition. The interval partitioning can be represented by two particular cases:

- division by date interval
- division by integer interval

It depends on type of specified column. For date columns use “yyyy-MM-dd” date format.

```
<OLAPCreatePartitionByAutoSplitCommand fullPath="Schemas.DBA_DEV.Tables.DATATYPE_ALL" columnName="Cname"
timeout="200" startValue="1" endValue="1000"  interval="100"  dateIntervalType=""/>
```

Allowed values:

dateIntervalType – Day , Week , Month , Year (Day is default)

OLAP. Filtering Rules.

Commands pack for filtering rules management.

Common attributes for add and edit rule.

ruleName=”$rullName” – name of rule.

ruleDatabasePattern=”$ruleDatabasePattern” – rule database pattern (use “%” for all db).

ruleSchemaPattern=”$ruleSchemaPattern” – rule schema pattern (use “%” for all schemas).

ruleTablePattern=”$ruleTablePattern” – rule table pattern (use “%” for all tables).

ruleClause=”$clause“ – clause of rule.

ruleActive=”$ruleActive” – state of rule (true – enabled, false - disable).

OLAP. Add Filtering Rules Command

This command creates filtering rules.

```
<AddFilteringRulesCommand>
        <Rule ruleName="$rullName1" ruleDatabasePattern="$ruleDatabasePattern1" ruleSchemaPattern=
                "$ruleSchemaPattern1" ruleTablePattern="$ruleTablePattern1" ruleClause="$clause1 "
                ruleActive="$ruleActive1"/>
        <Rule ruleName="$rullName2" ruleDatabasePattern="$ruleDatabasePattern2"
                ruleSchemaPattern="$ruleSchemaPattern2" ruleTablePattern="$ruleTablePattern2"
                ruleClause="$clause2" ruleActive="$ruleActive2"/>
</AddFilteringRulesCommand>
```

OLAP. Remove Filtering Rules Command

This command removes filtering rules by rule name.

```
<RemoveFilteringRulesCommand>
        <Rule ruleName="$rullName1"/>
        <Rule ruleName="$rullName2"/>
</RemoveFilteringRulesCommand>
```

OLAP. Switch Filtering Rules Command

This command switches states of rules.

```
<SwitchFilteringRulesCommand>
        <Rule ruleName="$rullName1"/>
        <Rule ruleName="$rullName2" ruleActive="$ruleActive2"/>
</SwitchFilteringRulesCommand>
```

If you do not specify new filtering active state - the state **will be reversed**.

OLAP. Edit Filtering Rules Command

This command creates filtering rules.

```
<EditFilteringRulesCommand>
        <Rule ruleName="$rullName1" ruleNewName="$rullNewName1"
                ruleDatabasePattern="$ruleDatabasePattern1" ruleSchemaPattern="$ruleSchemaPattern1"
                ruleTablePattern="$ruleTablePattern1" ruleClause="$clause1" ruleActive="$ruleActive1"/>
        <Rule ruleName="$rullName2" ruleNewName="$rullNewName2"
                ruleDatabasePattern="$ruleDatabasePattern2"
                ruleSchemaPattern="$ruleSchemaPattern2" ruleTablePattern="$ruleTablePattern2"
                ruleClause="$clause2" ruleActive="$ruleActive2"/>
</EditFilteringRulesCommand>
```

Additional attribute for rules:

ruleNewName=”$rullNewName” – new rule name

Print Tree Node Children Command

This command will print all children of selected object

```
<PrintTreeNodeChildrenCommand origin="SOURCE"
fullName="Schemas.GOLD_CHECK_ORA.Packages"/>
```

```
Allowed values:
       fullName – Defines object address in tree.
       For Example: Schemas.SchemaName, Schemas.SchemaName.Tables.TableName etc.
        origin – SOURCE, TARGET
```

Security Settings

This charpted describes security settings. This settings needed for Working with SSL on any level (connection, working with Extractors)

Select Existing Key Store Command

This command will select exising Key Store

```
<SelectExistingKeyStoreCommand name="name" password="******"
store="C:\Agent\Vault\Key" />
```

Select Existing Trust Store Command

This command will select exising Trust Store

```
<SelectExistingTrustStoreCommand name="name" password="******"
store="C:\Agent\Vault\Key" />
```

Forget Existing Key Store Command

This command will forget exising Key Store

```
<ForgetExistingKeyStoreCommand name="name"/>
```

Forget Existing Trust Store Command

This command will forget exising Trust Store

```
<ForgetExistingTrustStoreCommand name="name"/>
```

Generate Trust Key Store Command

This command will generate new Trust and Key store.

```
<GenerateTrustKeyStoreCommand keyName="name" keyPassword="******"
trustName="trustName" trustPassword="******" location="D:\Trash\folder"/>
```

Allowed values:

`location – path to folder.`

Extension Pack Commands

This charpter describes all commands related to Extension Pack apply and usage.

OLTP/DWH: Extension Pack Schema Version Command

This command will check and Print Extension Pack version

```
<ExtPackSchemaVersionCommand/>
```

DWH: Extension Pack Python Version Command

This command will check and Print Python Extension Pack version

```
<ExtPackPythonVersionCommand/>
```

OLTP/DWH: Extension Pack Configuration Command

This command will configure Extension pack

```
<ExtPackConfigCommand cli="true"/>
```

Allowed values:

`cli – AWS Command Line Interface installed (true/false).`

If cli=false so Default profile setting will be used and <OLTPDMSAddProfileCommand/> is required.

OLTP: Create Email Service Command

This command will create Email Service emulation usin AWS Lambda.

NOTE: AWS Profile needed

```
<CreateEmailServiceCommand
arn="arn:aws:lambda:uswest2:468131460318:function:sct_mail_"/>
```

If attribute arn isn't set so the Operation Create AWS Lambda Function will be called.
Pre-condition: <OLTPDMSAddProfileCommand/>, <ExtPackConfigCommand/> are required.

OLTP: Delete Lambda Function Command

This command will remove lambda function

NOTE: AWS Profile needed

```
<DeleteLambdaFunctionCommand
arn="arn:aws:lambda:uswest2:468131460318:function:sct_mail_"/>
```

Pre-condition: <OLTPDMSAddProfileCommand/>, <ExtPackConfigCommand/> are required.

OLTP: Job Emulation Command

This command will configure Job emulation.

NOTE: AWS Profile needed

```
<JobEmulationCommand login="login" password="psswd" pythonDriverFolder="C:\folder"/>
```

Allowed values:
- **login – database login**
- **password – database password**
- **pythonDriverFolder – folder with Python driver**

If already is a Lambda function no attributes is required.

Pre-condition: <OLTPDMSAddProfileCommand/>, <ExtPackConfigCommand/> are required.

DWH: Python Library Upload Command

This command will upload Python Library and create it on Redshift

NOTE: AWS Profile needed. S3 bucket needed.

```
<PythonLibUploadCommand s3BucketFilePath=" s3://extractors-
test/TestUpload/aws_oracle_ext.zip" upload="true">
```

Allowed values:
- **s3BucketFilePath – s3 bucket file path**
- **upload – true if upload is required**

Pre-condition: <OLTPDMSAddProfileCommand/>, <ExtPackConfigCommand/> are required.

OLTP/DWH: Extension Pack Creation Command

This command will create Extension Pack.

```
<FunctionsEmulationCommand/>
```

Compare Schema Command

This command compare two schemas.

```
<DDLCompareCommand compareFullNodeName="$Schemas.SchemaName"/>
```

Allowed values:
compareFullNodeName – full path to schema to use it context to search elements.

Example: Schemas.SchemaName

Apply Compared Schema Command

This command applys compared schema to targed.

```
< DDLApplyCommand sourceFullNodeName="$Schemas.SchemaName"/>
```

Allowed values:
sourceFullNodeName – full path to source schema to use it context to search elements.

Example: Schemas.SchemaName

Wait Command

This command pause application for time in seconds.

```
< WaitCommand timeout="$time"/>
```

Allowed values:
time – time of waiting in seconds.

Example: 15

Commands for WQF

Specific CLI commands. Do not affect the current scenario and do not depend on it.

Create WQF Report command

Creates WQF Report for Oracle or Microsoft SQL Server. Command produces 1 zip archive along the specified path with 2 xml files (Assessment report – report for converting to MySQL and PostgreSQL, Inventory report – report for converting to same vendor).

```
<CreateWqfReportCommand password="$passwordSourse"
wqfDirectoryPath="$wqfDirectoryPath" connectionType="$sourceConnectionType"
vendor="$sourceVendor" serverName="$connectionSourse" serverPort="$portSourse"
userName="$userNameSourse" sid="$vendorSourse" >
        <Node sourceFullName=”$sourceFullName1/>
        <Node sourceFullName=”$sourceFullName2/>
</CreateWqfReportCommand>
```

Allowed values:

For Oracle:

connectionType: BASIC_SERVICE_NAME, BASIC_SID, TNS_ALIAS, TNS_CONNECT_IDENTIFIER

NOTE: For TNS_ALIAS need to specify tnsAlias=”” and tnsFilePath=””. For TNS_CONNECT_IDENTIFIER need to specify tnsConnectIdentifier=””

For Microsoft SQL server:

connectionType: BASIC_SID

<Node> sourceFullName: Node full name for converting.
You can indicate any quantity of nodes. If you indicate 0 nodes the assessment report will not be created.

wqfDirectoryPath: Folder path for saving.

Create WQF Report with SSL mode

Create WQF Report command usage with SSL.

```
<CreateWqfReportCommand serverName="52.36.56.194" password="min_privs"
serverPort="1522" userName="min_privs" vendor="ORACLE" useSsl="true"
encryptData="false" sslAuthentication="false" encryptConnection="false"
trustServerCertificate="false" requireSsl="false" verifyServerCertificate="false"
tlsAuthentication="false" tnsFilePath="" serviceName="" tnsAlias=""
tnsConnectIdentifier="" sid="ORA12C01" connectionType="BASIC_SID" keyStoreAlias="K"
trustStoreAlias="Tr" wqfDirectoryPath=”C:\wqf” >
<Node sourceFullName=”Schemas.TEST_ORA_PG”/>
</CreateWqfReportCommand>
```

useSsl: true/false
encryptData: true/false
sslAuthentication: true/false
encryptConnection: true/false
trustServerCertificate: true/false
requireSsl: true/false
verifyServerCertificate: true/false
keyStoreAlias: Keystore Alias

trustStoreAlias: Truststore Alias

Create Application Conversion WQF Report command

Creates application WQF Report for Oracle or Microsoft SQL Server. Command produces 1 zip archive along the specified path with 1 xml file.

```
<CreateAppConversionWqfReportCommand password="$passwordSourse"
wqfDirectoryPath="$wqfDirectoryPath" connectionType="$sourceConnectionType"
vendor="$sourceVendor" serverName="$connectionSourse" serverPort="$portSourse"
userName="$userNameSourse" sid="$vendorSourse" sourceDir="$appConversionProjectFolder"
schemaFullName ="$SchemaName" knownParameterStyle ="$parameterStyle"
language ="$ProgrammingLanguageName" convertFullFileNodeName ="$fullFileNodeName" >
</CreateAppConversionWqfReportCommand >
```

Allowed values:

For Oracle:

connectionType: BASIC_SERVICE_NAME, BASIC_SID, TNS_ALIAS, TNS_CONNECT_IDENTIFIER

NOTE: For TNS_ALIAS need to specify tnsAlias="" and tnsFilePath="". For TNS_CONNECT_IDENTIFIER need to specify tnsConnectIdentifier=""

For Microsoft SQL server:

connectionType: BASIC_SID

language – Defines source file language. Possible values: JAVA, C++, C#, ANY

sourceDir – Defines Application Conversion loading path.

schemaFullName – Defines object address in tree.

knownParamenterStyle – Defines style of parameters exist at analyzed files. Possible values: Same as in source, Positional (?), Indexed (:1), Indexed ($1), Named (@name), Named (&name), Named ($name)

convertFullFileNodeName – Defines path to analyzed file. Can be same as "sourceDir" or descendant of "sourceDir".

wqfDirectoryPath: Folder path for saving.

Create application conversion WQF Report with SSL mode

Create application conversion WQF Report command usage with SSL.

```
<CreateWqfReportCommand serverName="52.36.56.194" password="min_privs"
serverPort="1522" userName="min_privs" vendor="ORACLE" useSsl="true"
encryptData="false" sslAuthentication="false" encryptConnection="false"
trustServerCertificate="false" requireSsl="false" verifyServerCertificate="false"
tlsAuthentication="false" tnsFilePath="" serviceName="" tnsAlias=""
tnsConnectIdentifier="" sid="ORA12C01" connectionType="BASIC_SID" keyStoreAlias="K"
trustStoreAlias="Tr" wqfDirectoryPath="C:\wqf" knownParameterStyle="Positional (?)"
language="C#" schemaFullName="Schemas.TEST_ORA_PG" sourceDir="C:\app
convertFullFileNodeName="C:\app" />
```

```
useSsl: true/false
encryptData: true/false
sslAuthentication: true/false
encryptConnection: true/false
trustServerCertificate: true/false
requireSsl: true/false
verifyServerCertificate: true/false
keyStoreAlias: Keystore Alias
```

Sample of XML file to run cosole

```
<?xml version="1.0" encoding="UTF-8"?>
<tree>
  <instances>
    <BatchJob>
      <NewProjectCommand targetVendor="$targetVendor" projectName="$projectName"
directory="$projectFolder" sourceVendor="$sourceVendor"></NewProjectCommand>
      <ConnectCommand password="$passwordSourse" vendor="$sourceVendor"
origin="$typeSourse" serverName="$connectionSourse" serverPort="$portSourse"
userName="$userNameSourse" sid="$vendorSourse" processMode="$treeNodeLoadMode"
recursionDepth="$treeNodeLoadDepth"></ConnectCommand>
      <ConnectCommand password="$passwordTarget" vendor="$targetVendor"
```

```
origin="$typeTarget" serverName="$connectionTarget" serverPort="$portTarget"
userName="$userNameTarget" sid="$vendorTarget" processMode="$treeNodeLoadMode"
recursionDepth="$treeNodeLoadDepth"></ConnectCommand>
      <TransformTreeNodeCommand sourceFullName="$objectFullPath"/>
      <ReportToPdfFileCommand fullName="$objectFullPath" filePath="$outputFilePath"/>
      <ReportToCSVFileCommand filePath="$outputFilePath"/>
      <WriteToDatabaseCommand fullName="$objectFullPathWriteDB"/>
      <SaveToSqlCommand fullName="$objectFullPathWriteDB"
filePath="$filePath"></SaveToSqlCommand>
      <SaveProjectCommand></SaveProjectCommand>
    </BatchJob>
  </instances>
</tree>
```