HPE Operations Bridge Reporter

Software Version: 10.10

Content Development Guide

Document Release Date: March 2017
Software Release Date: January 2017

Hewlett Packard Enterprise

Legal Notices

Warranty

The only warranties for Hewlett Packard Enterprise products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Hewlett Packard Enterprise shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from Hewlett Packard Enterprise required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notice

Trademark Notices

Adobe™ is a trademark of Adobe Systems Incorporated.

Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation.

UNIX® is a registered trademark of The Open Group.

This product includes an interface of the 'zlib' general purpose compression library, which is Copyright © 1995-2002 Jean-loup Gailly and Mark Adler.

Documentation Updates

To check for recent updates or to verify that you are using the most recent edition of a document, go to: https://softwaresupport.hpe.com/.

This site requires that you register for an HP Passport and to sign in. To register for an HP Passport ID, click **Register** on the HPE Software Support site or click **Create an Account** on the HP Passport login page.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HPE sales representative for details.

Support

Visit the HPE Software Support site at: https://softwaresupport.hpe.com/.

Most of the support areas require that you register as an HP Passport user and to sign in. Many also require a support contract. To register for an HP Passport ID, click **Register** on the HPE Support site or click **Create an Account** on the HP Passport login page.

To find more information about access levels, go to: https://softwaresupport.hpe.com/web/softwaresupport/access-levels.

HPE Software Solutions Now accesses the HPSW Solution and Integration Portal website. This site enables you to explore HPE Product Solutions to meet your business needs, includes a full list of Integrations between HPE Products, as well as a listing of ITIL Processes. The URL for this website is https://softwaresupport.hpe.com/km/KM01702731.

Contents

Part I: Content Development

This section introduces Content Development Environment (CDE) and the flow of content pack development, and provides detailed information and instructions on developing content pack using CDE on Windows and Linux environments where OBR is installed. This section also provides specific instructions to develop content on a system where OBR is not installed.

- **Preparing for content development**
- **Developing content using CDE on Windows**
- **Developing content using CDE on Linux**

Chapter 1: About this Guide

This guide provides an overview of content development using the HPEOperations Bridge Reporter Content Development Enthronement (CDE) and describes the process of creating a content pack.. You will use sample files available in your HPE OBR installation media and the instructions in this guide to create a content pack.

This guide contains the following sections:

1. **Introduction**: Provides an overview of OBR CDE, prerequisites for content development, and the architecture of content packs.
2. **Part I: Content Development**: Lists the prerequisites and provides step-by-step instructions to create a content pack on Windows and Linux environments, using the example of Retail Point of Sales (RetailPOS).
3. **Part II: Content Development - Simplified**: Describes a simplified method of creating the Domain, ETL, and Report components of a content pack.
4. **Part III: Content Development Reference**: Lists the XML, CSV, and properties files used to build a content pack using CDE, and explains the syntax of the source XML files.
5. **Glossary**: Describes the terms used in this guide.

Intended Audience

This guide is meant for developers who want to create content packs on HPE OBR or extend existing content packs. For information about creating content packs using the OBR Content Designer, see *Operations Bridge Reporter Content Designer Guide*.

Chapter 2: Introduction

The Operations Bridge Reporter (OBR) Content Development Environment (CDE) consists of a set of tools for content development. These tools use XML files authored by the content pack developer to generate the installable Content Pack component packages.

You can develop content using one of the following CDE tools:

- **Command-based CDE** - Step-by step commands to develop content. This document provides instructions and sample `.XML` files to develop content using commands.
- **Content Designer** - A user interface to generate content. For more information, see *OBR Content Designer Guide*.

Content Pack

A content pack is a domain or application-specific data mart deployed on the OBR performance management database platform. Content packs determine the metrics to be collected, how to process and store the metrics, and display the processed data on the reports.

A typical content pack consists of three components

- Domain (data model definition),
- Extraction Transformation Loading (ETL), and
- Reports .

Tools to Create Content Packs

You can develop Content Packs using one of the following tools available in OBR:

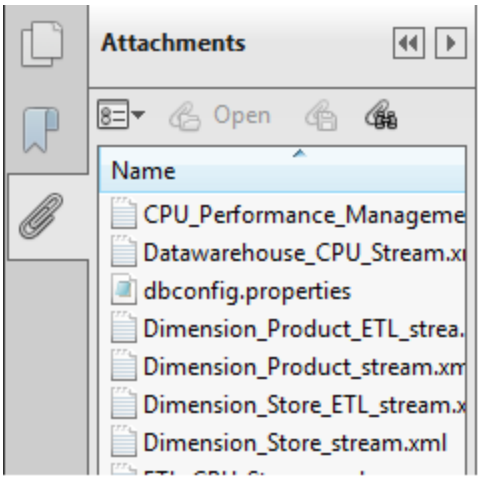
- **Advanced Content Development** - Use a step-by-step commands to create Domain, ETL, and Report components one after the other.
- **Content Development - Simplified-** Use CDE to create Domain, ETL, and Report components at the same time.
- **Content Designer** - Create Domain, ETL, and Report components quickly using the CDE design studio. For more information, see *Operations Bridge Reporter Content Designer Help*.

> **Note:** HPE does not provide support for Content Packs created or modified using the Content

> Development Environment (CDE) tool, or otherwise, via HPE's standard product support/service agreements. Any defects found in the underlying CDE tool, however, will be addressed by HPE.

CDE Guide Attachments

OBR Content Development Guide includes samples for XML, CSV, and database configuration files. Click the **Attachments: View File Attachments** icon and double-click a file to view the file on your browser.

> **Note:**
>
> 
>

Prerequisites and Reference Documentation

This guide assumes you have a prior understanding of the following:

Prerequisite	Reference Documentation
HPE Operations Bridge Reporter concepts and usage	Read the following documents available at `Start > Programs > HPE Software > Operations Bridge Reporter > Documentation` • **Concepts Guide:** This guide explains the key concepts, architecture, and typical workflow of HPE OBR. Read this guide to understand the concept and

Prerequisite	Reference Documentation
	working of content packs before you start developing any. • **Interactive Installation Guide:** Use this guide to learn the prerequisites and detailed steps for installing OBR in your environment. • ***Configuration Guide:*** Use this guide to plan your deployment scenarios, configure OBR in the supported deployment, and install content packs. • ***Online Help for Administrators*:** In this Help you find information about monitoring the installed content packs. • ***Online Help for Users:*** In this Help you find information about the out-of-the-box content packs provided by HPE OBR.
Data Warehouse concepts	You can find resources related to data warehouse concepts and examples on the internet.
SAP BusinessObjects reporting concepts	• ***SAP BusinessObjects BI LaunchPad:*** http://scn.sap.com/docs/DOC-19231 • ***SAP BusinessObjects Information Design Tool User Guide:*** In this Help you find information about creating, building, and managing Universes. For information and the latest help documents, see **http://help.sap.com/businessobject/product_guides/.**
XML concepts and creating XML documents	You can find resources related to XML concepts and examples on the internet. HPE OBR does not recommend any particular resource.

Definition of Terms

Term	Definition
CDE	Content Development Environment
CI	Configuration Item
CIUID	Configuration Item Unique Identifier
CMDB	Configuration Management Database

Term	Definition
DBMS	Database Management System
ETL	Extract, Transform, Load
OA	HP Operations Agent
OM	HP Operations Manager
PA	HP Performance Agent
RTSM	Run time Service Model
SPI	HP Smart Plug-ins
XML	Xtensible Markup Language

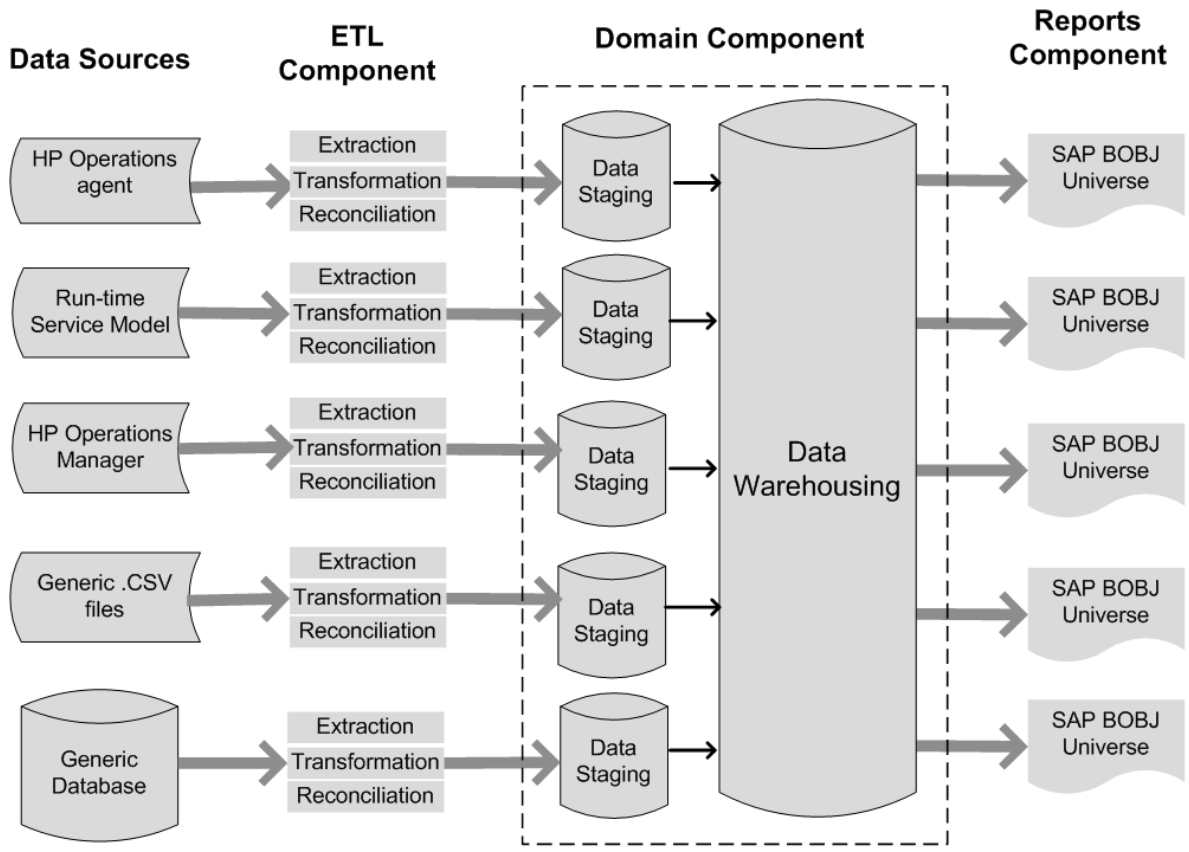
Content Pack Architecture

A content pack is a domain or application-specific data mart deployed on the HPE OBR performance management database platform. Content packs determine the metrics to be collected, how to process and store the metrics, and display the processed data on the reports.

Components of a Content Pack

A content pack consists of three components - the Domain, Extraction Transformation Loading (ETL), and Reports.

The following figure shows the data flow between the components.



Note: The Transformation and Reconciliation steps in an ETL component are optional and may not be applicable for all data sources.

HPE OBR enables you to create the following content on the performance management database platform:

- **Content pack:** You can create new content packs and extend the out-of-the-box content packs provided by HPE OBR. This guide uses an example (RetailPOS) to describe the steps to create a content pack.

- **Web Intelligence reports:** Using the SAP BusinessObjects Web Intelligence application via BI Launch Pad interface, you can create new reports and customize the out-of-the-box reports provided by HPE OBR.

Domain Component

The Domain component defines the data model of the domain you are reporting on along with the logic to perform processing on the data. It requires domain experts to model the data according to the business requirements. This component is independent of the data source. The Domain component includes:

- The data model consisting of the facts and dimensions of the domain you are reporting on and the relationship between them.
- Workflow streams that control and monitor the processing of the data. A stream is made up of steps that are related to one another in a sequential relationship. A content pack contains a set of workflow streams that define and control the flow of data from one step to another. In the Domain component of a content pack, workflow streams are defined in XML files to load the data into tables and perform offline aggregation of the data.
- Optionally defines the dimensions and cubes for the business view to be used by one or more Reports components.

Extraction Transformation Loading (ETL) Component

The ETL component is data source dependent; it defines the collection of data from the specified data source, followed by transformation and loading of the data into the data warehouse. Therefore, for a particular domain, each data source application has a separate ETL content pack component. Before you start creating the ETL component you must identify the data source that provides the metrics suitable to be fed into the domain data model. The ETL component includes the following:

- **Data Collection (Extraction) rules:** After you identify a data source you must create a collector program or use an existing collector program to collect the required facts and dimensions from the data source. A collection policy must be written in XML to define the metrics to be collected by the collector program from the data source. The collector program collects data as defined in the collection policy and places this data into `.csv` files.

  HPE OBR supports data collection from a set of known data sources and provides collector programs for each such data source.

  The data sources supported by OBR are:

- Run-time Service Model (RTSM)
- HP Operations Agent
- HP Operations Manager
- HP Business Service Management Profile Database
- HP Operations Manager i (Operations database)
- Generic `.csv` files
- Databases supporting JDBC

> **Note:** At present, HPE OBR does not support creating custom content using NRT ETL (NPS).

- **Data Transformation rules (optional):** Data transformation rules are required if the collected data, as `.csv` files, need to be transformed before loading the data into the data warehouse. For example, you can write a rule to remove the rows that contain an empty value in the 'host name' column. Transformation rules are written in XML files. HPE OBR provides a data transformation utility called the 'mapper' utility for the out-of-the-box transformation rules.
- **Data Reconciliation rules (optional):** Data reconciliation is the process of associating fact data to the corresponding dimension data. In HPE OBR data reconciliation rules are written to associate fact data from one source to the corresponding dimension data from another source using common business keys. For example, in the Service and Operations Bridge (SaOB) deployment, the dimension data is collected from RTSM and the fact data is collected from HP Operations agent. Reconciliation rules are written in XML to reconcile the fact data with the dimension data.
- **Data Staging rules:** After the data (in the form of `.csv` files) is collected, transformed, and reconciled, it is moved to staging tables. Data staging rules define how the data must be moved to staging tables including the process to merge columns and rows.
- **Workflow Stream definitions:** In the ETL component, workflow streams are defined in XML to control the data movement from collection to stage through transformation and reconciliation steps if required.

Reports Component

The Reports component contains the SAP BusinessObjects Web Intelligence reports and universes. A content pack universe provides a business-oriented meaningful mapping of the underlying complex database and simplifies the creation of reports. It is a logical view of the underlying data model that you define in the Domain component. The Reports component imports the dimensions and cubes defined in the corresponding Domain component.

Chapter 3: Preparing for Content Development

Study the Business Domain

Consider a large electronics retail chain with:

- a hundred stores spread across five locations,
- each store having about 10,000 individual products on its shelves, and
- the individual products identified by Stock Keeping Units (SKUs).

The retail chain is automated and each product has a scanner label attached to it. The collection of transaction data is done mainly at the Point-of-Sale (PoS) system at the front door of a store where the bar codes are scanned and directly entered into the system. The customer's takeaway is measured at this place.

After you understand the retail store chain business, you can proceed to determine the business user personas and their respective reporting requirements.

Identify User Personas and Reporting Requirements

In the Retail Point of Sales example, we consider the business management personnel as the users of the reports.

The management users are interested in a sales summary report that shows the product sales information for different product categories in stores located across different locations. The sales information must be available across time periods such as yearly, quarterly, monthly and daily.

It is recommended to create a design mock-up of the required reports at this stage. This initial mock-up can be created on paper or any design tool of choice. The actual development of the Web Intelligence reports by using SAP BusinessObjects can be done when you create the Reports component package.

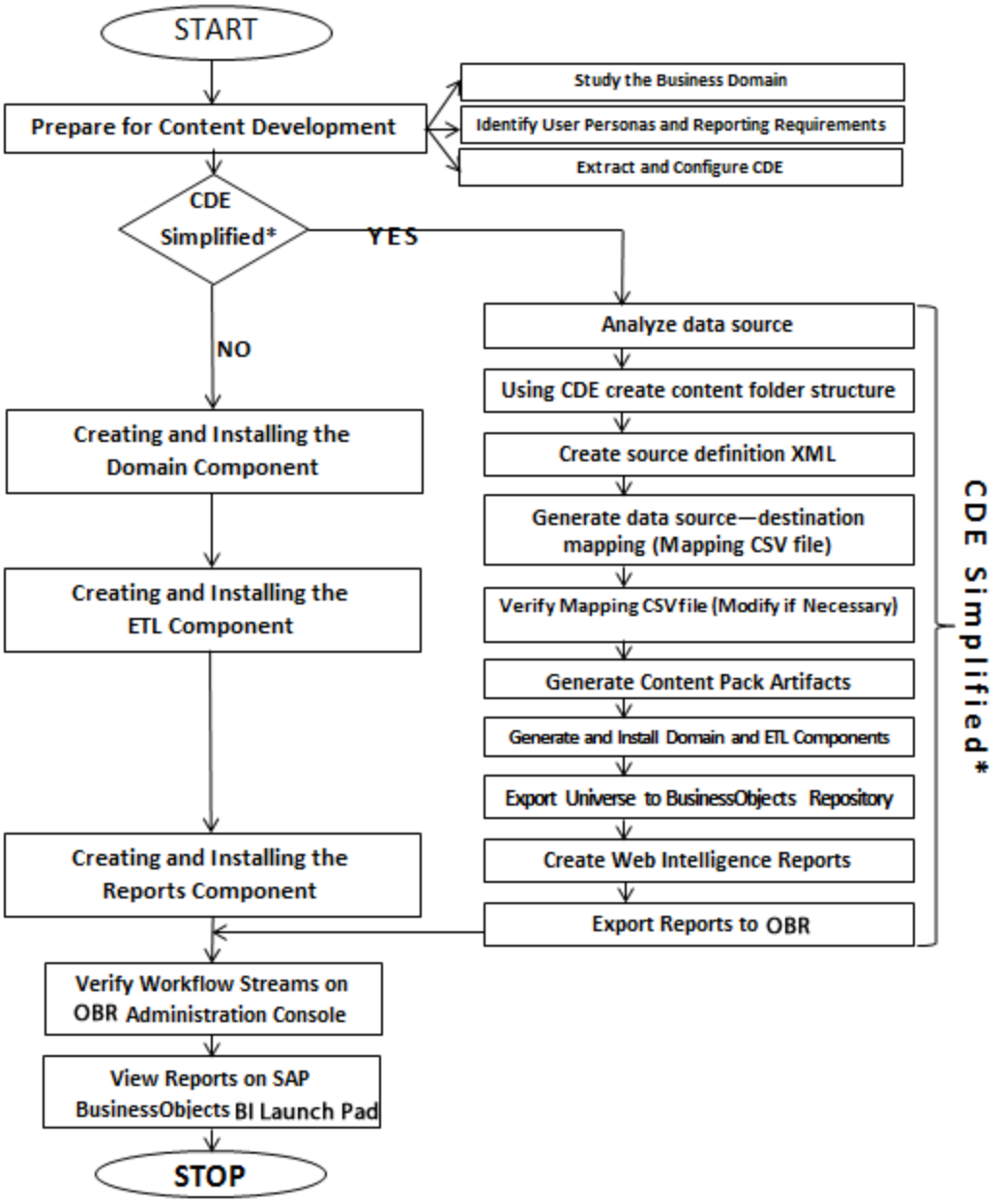
Content Development Environment

The Content Development Environment (CDE) consists of a set of tools that you use during the process of content development. These tools use XML files authored by the content pack developer to generate the installable content pack component packages.

The CDE tools are provided in the HPE OBR media as a self-extracting `CDE.exe` file in the *<installation_directory>*. The latest version of CDE is available in HPE Live Network.

Content Pack Development - Flow

This chapter describes the steps to create a sample content pack by using a real world example related to the retail sales industry, RetailPOS. The following flowchart shows the steps in CDE and Simplified-CDE to create a content pack. To evaluate if you want to use the simplified steps, see section "When to use CDE Simplified?" on the next page".



CDE Simplified* — See section "When to use CDE Simplified?" to determine if you can utilize this method.

When to use CDE Simplified?

The simplified method of creating a content pack can be used in the following cases:

1. When a fact table is associated with one or more dimensions (star schema) and the dimensions are not further normalized (dimensions without a parent table).

2. When all the facts are to be reconciled against host (when HP Operations agent is the data source).
3. When the generated reports do not require any roll up or drill down of data.

If you choose to follow the CDE-simplified steps, see "'Part II: Content Development - Simplified" on page 72". Otherwise, use the regular method for content development using section "'Content Pack Development - Flow" on page 19".

Chapter 4: Developing Content Using CDE on Linux

The following section provides step-by-step instructions to create content packs using CDE on Linux:

> **Note:** If you have created content pack on a Windows system and want to install the content pack on a Linux system, perform the conversion using the `dos2unix` command on all the generated `.xml` files before installing the content pack on the Linux system.
>
> For example, the following command converts all the generated .xml files under the content pack folder:
>
> ```
> find $PMDB_HOME/packages/<contentpack_folder_name> -type f -name "*.xml" -
> print0 | xargs -0 dos2unix
> ```

Extract and Configure CDE on Linux

1. On the Linux server where OBR is installed, navigate to the `$PMDB_HOME/..` (*/opt/HP/BSM*) directory.
2. Extracts the contents of the `CDE.zip` file.

   Contents of the `ZIP` file are extracted to `/opt/HP/BSM/CDE` (`$CDE_HOME`)directory.
3. Navigate to the *$CDE_HOME/bin* directory and provide execution rights to shell script files.
4. Run the following command to set execute permission for the `setenv.sh` and `createCPFolders.sh` files:

   ```
   chmod a+x setenv.sh createCPFolders.sh
   ```
5. Run the `source ./setenv.sh` command.

   The following environment variables are set:

   - `CDE_HOME`
   - `ANT_HOME`
   - `JRE_HOME`
   - `JRE_BIN`

> **Note:** Using the `chmod z+x <filename>` command, set execute permission for all the `.sh` files in the `$CDE_HOME/bin` directory to execute the files without any issues.

6. Navigate to the *$CDE_HOME/ant/bin* directory and run the following command to provide execution rights to `ant` file:

```
chmod a+x ant
```

Creating and Installing the Domain Component Package

Create the Directory Structure

1. Navigate to `$CDE_HOME/bin` directory.
2. Run the following command to create the directory structure for the Domain component source files:

   ```
   ./createCPFolders.sh -package RetailPOS -subpackage RetailPOSDomain -type domain
   ```

   where,

   - `RetailPOS` is the name of the content pack you are creating
   - `RetailPOSDomain` is the Domain component within RetailPOS content pack

   OBR provides sample source files for the Retail POS content pack at the following location:

   $CDE_HOME/samples/RetailPOS_Demo_Content_Pack/Source/

   You can use these files as reference to create your own content pack. The directory contains templates provided by OBR to create the Domain component source files — the model XML and the workflow streams XML files.

Identify the Grain, Dimensions, and Facts

The grain of the fact table is the most granular data. In the Retail Point of Sales Domain example, the grain is an individual line item on a Point of Sales transaction.

The dimensions are:

- Date
- Product
- Store

The facts collected by the Retail Point of Sales system are:

- Quantity of Sales
- Amount of Sales

Design the Data Model

A data model illustrates the relationship between entities (facts and dimensions tables) and their attributes (database table columns). In OBR, a data model is an XML file. To create a data model start by creating a schema diagram and then implement the same into an XML file.

The following figure shows the schema diagram that we will use to create the data model for the RetailPOS Content Pack.

Implement the Data Model in XML

The schema that you designed in the previous step must be implemented using XML to be used by the CDE to create the Domain component package. This XML file is called Model XML.

A typical Model XML file has the following sections:

- **Relational section** defines the fact tables and dimension tables, and the relationship between the facts and dimensions.
- **Logical section** defines the cubes, hierarchies, and levels. You define one cube for every fact table.
- **Aggregate section** defines the aggregation to be performed on the source tables.

After you create the Domain component directory structure for the RetailPOS example, you find a template model_template.xml in the following location: `$CDE_HOME/workspace/RetailPOS/RetailPOSDomain.ap/source/model`

You can edit this `.xml` file to author the Model XML.

Create the Workflow Streams

HPE OBR provides a workflow framework to control and monitor the workflow of the data processes. This framework is made up of workflow streams. A stream is made up of steps that are related to one another in a sequential relationship. Each content pack contains a set of streams that define and control the flow of data from one step to another.

For example, the Domain component may have the following steps in the workflow stream:

Loading data to rate table > Hourly aggregation > Daily aggregation

You must implement the following workflow streams in XML to be used by CDE to create the Domain component package:

- One workflow stream XML to load and aggregate the fact.
- One workflow stream XML to load the `Store` dimension.
- One workflow stream XML to load the `Product` dimension.

After you create the directory structure for the Domain component you find a workflow stream template file named `ABC_stream_template.xml` at the following path:

$CDE_HOME`/workspace/RetailPOS/RetailPOSDomain.ap/source/orchestration/stream_definitions`

You can edit this `ABC_stream_template.xml` file to author the workflow streams XML files.

You can find sample workflow stream XML files for the **RetailPOSDomain** at the following location:

$CDE_HOME`/samples/RetailPOS_Demo_Content_Pack/Source/`

To read the contents of the sample workflow stream XML files, on this PDF document click the **Attachments: View File Attachments** icon and select the following `.xml` files:

- `Implement the Data Model in XML`: Workflow stream XML to load and aggregate the fact.
- `Dimension_Store_stream.xml`: Workflow stream XML to load the `Store` dimension.
- `Dimension_Product_stream.xml`: Workflow stream XML to load the `Product` dimension.

Double-click to open the XML files on your browser window.

Use CDE to Generate Domain Component Package

To generate the Domain component package, follow these steps:

1. **Create Manifest XML file**

   The Manifest XML file contains the definitions of the source files to be used by CDE to generate the Domain component package.

   To create the Manifest XML file using CDE:

   a. Navigate to *$CDE_HOME*`/workspace/RetailPOS/RetailPOSDomain.ap` directory.

   b. Run the following command:

   ```
   ant createManifestTemplate
   ```

   The Manifest XML file named `RetailPOSDomain_manifest_template.xml` is created in the *$CDE_HOME*`/workspace/RetailPOS/RetailPOSDomain.ap/source` directory.

   You can see a sample Manifest XML file for the `RetailPOSDomain` at the following location:

   $CDE_HOME`$/samples/RetailPOS_Demo_Content_Pack/Source/RetailPOS/RetailPOSDomain.ap/source/RetailPOSDomain_manifest_template.xml`

   > To read the contents of the sample Manifest XML file, on this PDF document click the **Attachments: View File Attachments** icon and select `RetailPOSDomain_manifest_template.xml`.Double-click to open the XML file on your browser window.

2. **Enable Custom Content Licensing**

   a. Open the `<ContentPack_Name>_manifest_template.xml` file.

   b. Modify `value` and `type` attributes of the `licenseDef` element.

   You can define four types of licenses for custom content: STATIC, SQL, DEPENDS, and COMMAND.

   Example:

   ```
   <license>
   <licenseDef value="500" type="STATIC"/>
   </license>
   ```

   For more information about custom content licensing, see "Enable custom content licensing" on page 42.

3. **Build Domain component package**

   To build the Domain component package using CDE:

a. Navigate to `$CDE_HOME/workspace/RetailPOS/RetailPOSDomain.ap/dist` directory.

b. Run the following command:

   `ant`

   The installable Domain component package is created in *$CDE_HOME*`/workspace/RetailPOS/RetailPOSDomain.ap/dist`.

c. Navigate to the `$CDE_HOME/workspace/RetailPOS/RetailPOSDomain.ap` directory.

d. Copy `RetailPOS` to *$PMDB_HOME*`/packages` to make it available in the OBR Administration Console Deployment Manager for installation.

> **Note:** It is not necessary to have HPE OBR installed on the machine where you are creating the Domain component package. If you created the Domain component package on another machine, you must copy the package to the HPE OBR machine under *%PMDB_HOME%*`\packages`.

Install the Domain Component Package

HPE OBR provides the Deployment Manager utility on the OBR Administration Console to install the content pack component packages. For instructions on how to install the content pack components, see the Operations Bridge Reporter *Configuration Guide.*

Verifying Workflow Streams on Administration Console

After the `HPE_PMDB_Platform_Timer` service is started, log on to the Administration Console to check the status of the Domain component workflow streams. Follow these steps:

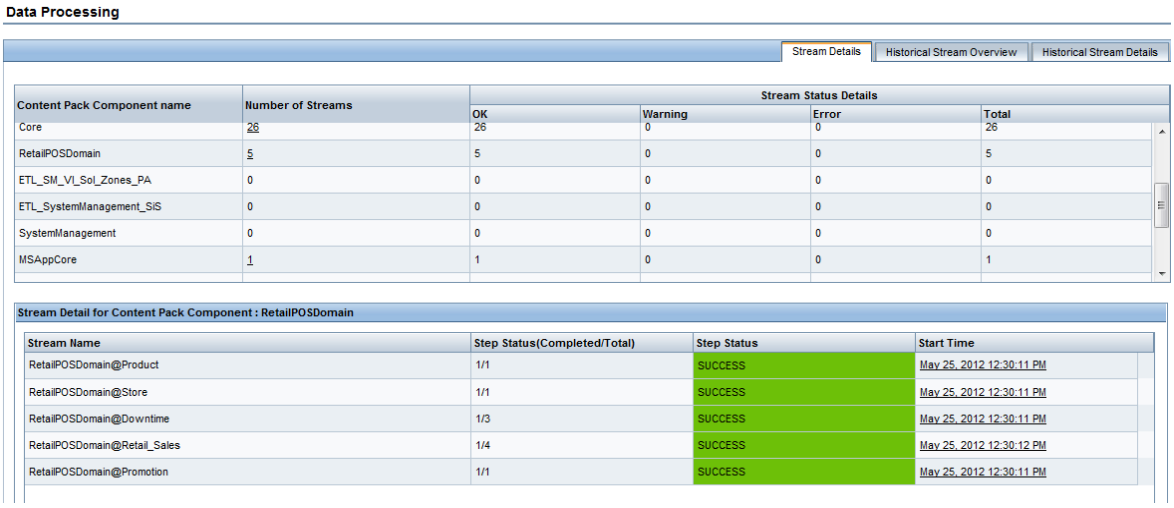
1. Log on to OBR Administration console.
2. Click **Internal Monitoring**-> **Data Processing**.
3. On the **Stream Details** tab, view the status of streams in the RetailPOSDomain content pack you have created. All streams must show a status of OK to indicate successful completion.

In the RetailPOS example, the Domain component has the following workflow streams with one or more steps within each stream.

- A workflow stream XML to load and aggregate the `Retail_Sales` fact.
- A workflow stream XML to load the `Store` dimension.

- A workflow stream XML to load the `Product` dimension.
- A workflow stream XML to load the `Promotion` dimension.

As shown in the following figure, successful completion of the streams is indicated using green color.



Creating and Installing the ETL Component Package

Prerequisite

Create and Install the Domain Component Package

Before you start creating the ETL component, you must create Domain content. See "Creating and Installing the Domain Component Package" on page 23 to create the Domain model and generate the Domain component package.

Analyze the Data Source

Before you start creating the ETL component, you must identify the data source containing metrics that are suitable to be fed into the Domain data model. As described in the chapter See "Content Pack Architecture " on page 14,

In this guide, we consider a simple database as the source of data. The database is created using PostgreSQL software and contains data suitable to be fed into the RetailPOS Domain model. Sample files and scripts are provided in the HPE OBR media to create the database and populate data into the tables. The "Creating a Data Source for ETL Component" describes how to create the PostgreSQL database, the database tables, and how to insert data into the tables.

Create the Directory Structure

Use the following steps to create the directory structure for the ETL component source files:

1. Navigate to `$CDE_HOME/bin` directory.
2. Run the `./createCPFolders.sh -package RetailPOS -subpackage RetailPOSETL -type etl` command.

where,

- `RetailPOS` is the name of the content pack you are creating
- `RetailPOSETL` is the name of the ETL component within `RetailPOS`

The directory contains templates provided by HPE OBR to create the ETL component source files.

HPE OBR provides sample source files for the Retail POS content pack at: *$CDE_HOME*`/samples/RetailPOS_Demo_Content_Pack/Source/`. You can use these files as reference to create your own content pack.

Define Collection Policy in XML

To collect the data from the `retailpos` database tables you must define a collection policy in XML. Use the collection policy template named `DB_collection_template.xml` available in the following folder:

$CDE_HOME`/workspace/RetailPOS/RetailPOSETL.ap/source/etl/collection/`

For reference see the sample `RetailPOS_DB_Collection_Policy.xml` in the following folder:

$CDE_HOME`/samples/RetailPOS_Demo_Content_Pack/Source/RetailPOS/RetailPOSETL.ap/source/etl/collection/`

> **Note**: To read the contents of the sample collection policy XML file, on this PDF document click the **Attachments: View File Attachments** icon and select RetailPOS_DB_Collection_Policy.xml. Double-click to open the XML file on your browser window.

Define Data Transformation Rules

In the RetailPOS example, the data transformation rule is used to append the name of the city, state, country, and the zip code to the address column; and use the whitespace character as a delimiter of the column values in the `.csv` files.

For reference see the sample `RetailPOS_transformation.xml` in *$CDE_HOME*`/samples/RetailPOS_Demo_Content_Pack/RetailPOS/RetailPOSETL.ap/doc`

To read the contents of the sample data transformation rule XML file, on this PDF document click the **Attachments: View File Attachments** icon and select `RetailPOS_transformation.xml`. Double click to open the XML file on your browser window.

Define Stage Rules

Stage rules defined in XML files are used to map the column names in the source .csv files to the column names in the target physical tables, called staging tables, in the database. Use the stage rules template available in the`RetailPOS/RetailPOSDomain.ap/source/stagerule_templates` to write the stage rules XML files for each of the dimensions – product, sales, and store.

For reference see the sample stage rule XML files in *$CDE_HOME*`/samples/RetailPOS_Demo_Content_Pack/RetailPOS/RetailPOSETL.ap/source/etl/stage_rules`

For all the OOTB domain content packs, stage rule templates are available in the folder `$CDE_HOME/cplib/<ContentPack>/<SubPackage.ap>/source/stagerule_templates`

To read the contents of the sample stage rule XML files, on this PDF document click the **Attachments: View File Attachments** icon and select the following files:

- `Stage_product_stagerule.xml`: Stage rule for Product
- `Stage_retail_sales_stagerule.xml`: Stage rule for Sales

- `Stage_store_stagerule.xml`: Stage rule for Store

Define Workflow Streams

Workflow streams defined in XML files are used to collect and stage the `.csv` files in the data warehouse.

When topology information is collected, HPE OBR Collector generates a `*relations*.csv` file for each topology view in addition to the CI type mentioned as part of the RTSM collection policy. You must author stage rules to stage this data and workflow streams to enable the process of loading the `*relations*.csv` files.

The `*relations*.csv` files contain topology data that must be loaded to the CI Bridge table. To create a stage rule definition for the CI Bridge, use the stage rule templates for the CI Bridge available in the folder `$CDE_HOME/cplib/Core/Core.ap/source/stagerule_templates[Core_0_Stage_K_CI_Bridge_0_stagerule.xml]`.

In addition to authoring stage rules, you must define a workflow stream to enable the data flow. For the syntax on creating ABC Streams, see *Operations Bridge Reporter Content Development - Reference Guide*.

> **Note:** In the COLLECT step, use the "View name" specified in the collection policy as the Category and "Relations" as the Type.

Use the workflow streams template available in the folder*$CDE_HOME*`/samples/RetailPOS_Demo_Content_Pack/source/RetailPOS/RetailPOSETL.ap/source\orchestration/stream_definitions` to write the workflow stream XML files, one file for each of the dimensions – product, sales, and store.

To read the contents of the sample ETL workflow stream XML files, on this PDF document click the **Attachments: View File Attachments** icon and select the following files:

- `Dimension_Product_ETL_stream.xml`: Workflow stream XML for Product dimension
- `Dimension_Store_ETL_stream.xml`: Workflow stream XML for Store dimension
- `Fact_Retail_Sales_ETL_stream.xml`: Workflow stream XML for the fact

Double click to open the XML files on your browser window.

Generate the ETL Component

To generate the ETL Component package using CDE:

1. Navigate to *$CDE_HOME*`/workspace/RetailPOS/RetailPOSETL.ap`
2. Run the `ant createManifestTemplate` command.
3. Run the `ant` command.

   The ETL component package is created in *$CDE_HOME*`/workspace/RetailPOS/RetailPOSETL.ap/dist` directory.
4. Navigate to the *$CDE_HOME*`/workspace/RetailPOS/RetailPOSETL.ap/dist` directory.
5. Copy `RetailPOS` to `$PMDB_HOME/packages` to make it available in the Administration Console Deployment Manager for installation.

Install the ETL Component Package

HPE OBR provides the Deployment Manager feature on the Administration Console to install the content pack component packages. Use the Deployment Manager to install the ETL component package; in the RetailPOS example, the name of the package is RetailPOSETL.

For instructions on how to install the content pack components using Deployment Manager, see the *Operations Bridge Reporter Online Help for Administrators.*

Verify the ETL Component Package

After you install the ETL component package and the `HPE_PMDB_Platform_Timer` service is started, logon to the OBR Administration Console user interface to check the status of the ETL component work-flow streams. Follow these steps:

1. On the Administration user interface, click **Internal Monitoring** -> **Data Processing**.
2. On the **Stream Details** tab, view the status of streams in the RetailPOSETL content pack. All streams must show a status of OK to indicate successful completion.

In the RetailPOS example, the ETL component has the following workflow streams with one or more steps within each stream.

- A workflow stream `RetailPOSETL@Retail_Sales_ETL` to move Sales fact to stage tables.
- A workflow stream `RetailPOSETL@Product_ETL` to move Product dimension to stage tables.
- A workflow stream `RetailPOSETL@Store_ETL` to move Store dimension to stage tables.
- A workflow stream `RetailPOSETL@Promotion_ETL` to move Promotion dimension to stage tables.

Creating and Installing the Application Component Package

If you have installed OBR on a Linux server, you must use a Microsoft Windows XP or later system for developing or customizing application content.

Prerequisites

A system with the following software:

1. Microsoft Windows XP or later operating system.
2. Java™ Platform, Standard Edition Development Kit (JDK™) 1.7.

   Download from http://www.oracle.com/.
3. SAP BusinessObjectsXI 4.1 Client.
   a. Copy the BusinessObjectsXI-4.1-Clienttools.zip from your HPE OBR media (OBR bits) folder.
   b. Extract the contents of the ZIP file to the Windows system.

      The contents of the ZIP file are extracted to the *BusinessObjectsXI-4.1-Clienttools* folder.
   c. Run the `setup.exe` file and follow the instructions on the installer wizard to complete the installation.

Creating the Application Component

1. On the Linux server where OBR is installed, navigate to the $PMDB_HOME$ (*/opt/HP/BSM*) directory.
2. Copy the `CDE.zip` file to the Windows system.
3. Extract the contents of the Zip file. The contents are extracted to the `CDE` folder(`%CDE_HOME%`).
4. From the *%CDE_HOME%\bin* folder, run the `setenv.bat` command with the complete directory path of CDE and Java Runtime Environment (JRE) as follows:

   ```
   setenv.bat -CDE_HOME C:\CDE -JRE_HOME C:\Java\jdk1.7.0_xx\jre
   ```

5. From the *%CDE_HOME%\bin* folder, run the `updateCDEProperty.bat` file.
   a. Enter values for the `bo.username` and `bo.password` parameters.
   b. Enter the short name of the OBR server for the `bo.server=` field.

      For example, `bo.server=obrdev`
6. Launch the **Command Prompt** and run the following command to create the application content pack directory structure:

   ```
   createCPFolders.bat -package cpname -subpackage subpackagename.ap -type
   application
   ```

7. Navigate to the *%CDE_HOME%\workspace \package\subpackagename.ap* folder and run the following commands:
   - `ant createManifestTemplate`
   - `ant`

   The universe is created at **Application**: `{CDE_HOME}\workspace\<ContentPack>\<ContentPack>.ap\dist\UnxFolder\UnxLayer`
8. Export the universe to the SAP BusinessObjects Server. For more information, see section "Export the Universe to SAP BusinessObjects Repository".

Chapter 5: Developing Content Using CDE on Windows

This section explains in details the steps to create Domain, ETL, and Report content packs on a Windows system, where OBR is installed. This section also provides instructions to create reports and integrate them with OBR if you are creating content using CDE on a system where OBR is not installed.

> **Note:** If you create content pack on a Windows system and want to install the content pack on a Linux system, perform the conversion using the `dos2unix` command on all the generated `.xml` files before installing the content pack on the Linux system.
>
> For example, the following command converts all the generated .xml files under the content pack folder:
>
> `find $PMDB_HOME/packages/<contentpack_folder_name> -type f -name "*.xml" -print0 | xargs -0 dos2unix`

Extract and Configure CDE on Windows

1. On you Windows system, open the directory where OBR is installed, *<OBR_installation_directory>*.
2. Run the file `CDE.exe` to extract the CDE tools.

   Contents of the `CDE.exe` are extracted to the `<OBR_installation_directory>\CDE` (`%CDE_HOME`) directory.
3. Open the command prompt and navigate to `%CDE_HOME%\bin` directory.
4. Run the `setenv.bat` command with CDE directory and JDK directory.

   Example: `setenv.bat -CDE_HOME C:\CDE -JRE_HOME C:\Java\jdk1.7.0_xx\jre`

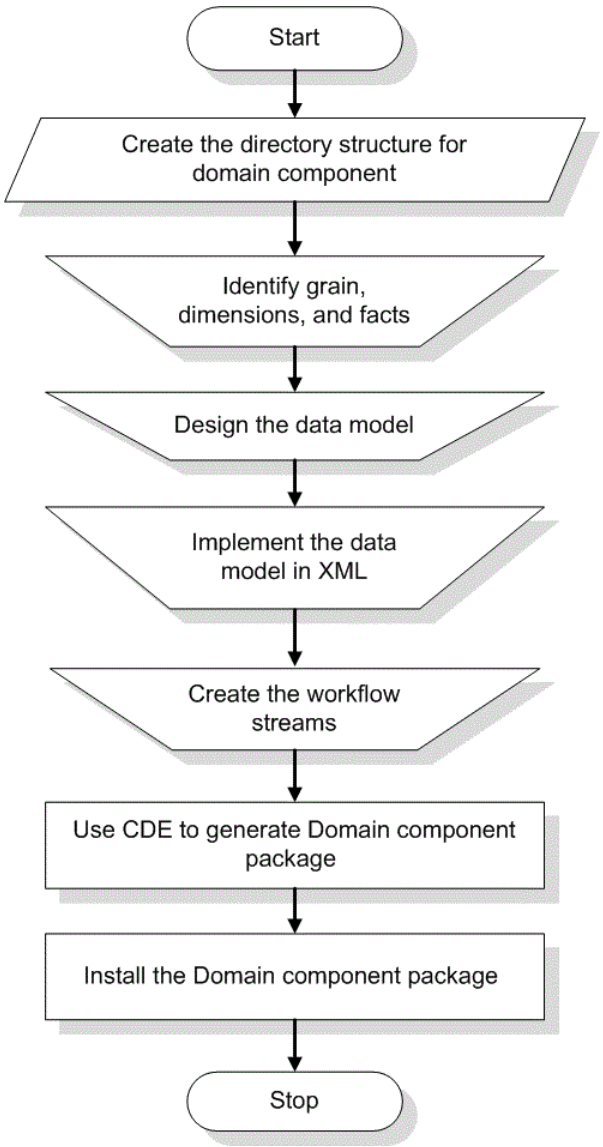
   Where, `C:\CDE` is the CDE extracted directory and `C:\Java\jdk1.7.0_xx\jre` is the JDK™ directory.

   The following environment variables are set in the path:

- CDE_HOME
- ANT_HOME
- JRE_HOME

Creating and Installing the Domain Component Package

The following flowchart shows the steps to create the Domain component of a content pack:



Create the Directory Structure

To create the directory structure for the Domain component source files, perform the following steps:

1. Open the command prompt.
2. Navigate to `%CDE_HOME%\bin` directory.
3. Run the `setenv.bat` command with CDE directory and JDK directory.

   Example: `setenv.bat -CDE_HOME C:\CDE -JRE_HOME C:\Java\jdk1.7.0_xx\jre`

   Where, `C:\CDE` is the CDE extracted directory and `C:\Java\jdk1.7.0_xx\jre` is the JDKTM directory.

   The following environment variables are set in the path:

   - CDE_HOME
   - ANT_HOME
   - JRE_HOME
4. Run the following command.:

   ```
   createCPFolders.bat -package RetailPOS -subpackage RetailPOSDomain -type domain
   ```

where,

- `RetailPOS` is the name of the content pack you are developing
- `RetailPOSDomain` is the name of the Domain component within `RetailPOS`

The newly-created directory structure contains templates provided by HPE OBR to create the Domain component source files — the model XML and the workflow streams XML files.

You can use the sample files provided by HPE OBR to create your own content pack. The sample files for RetailPOS are available at `%CDE_HOME%\samples\RetailPOS_Demo_Content_Pack\Source\`.

Identify the Grain, Dimensions, and Facts

The grain of the fact table is the most granular data. In the Retail Point of Sales Domain example, the grain is an individual line item on a Point of Sales transaction.

The dimensions are:

- Date
- Product
- Store

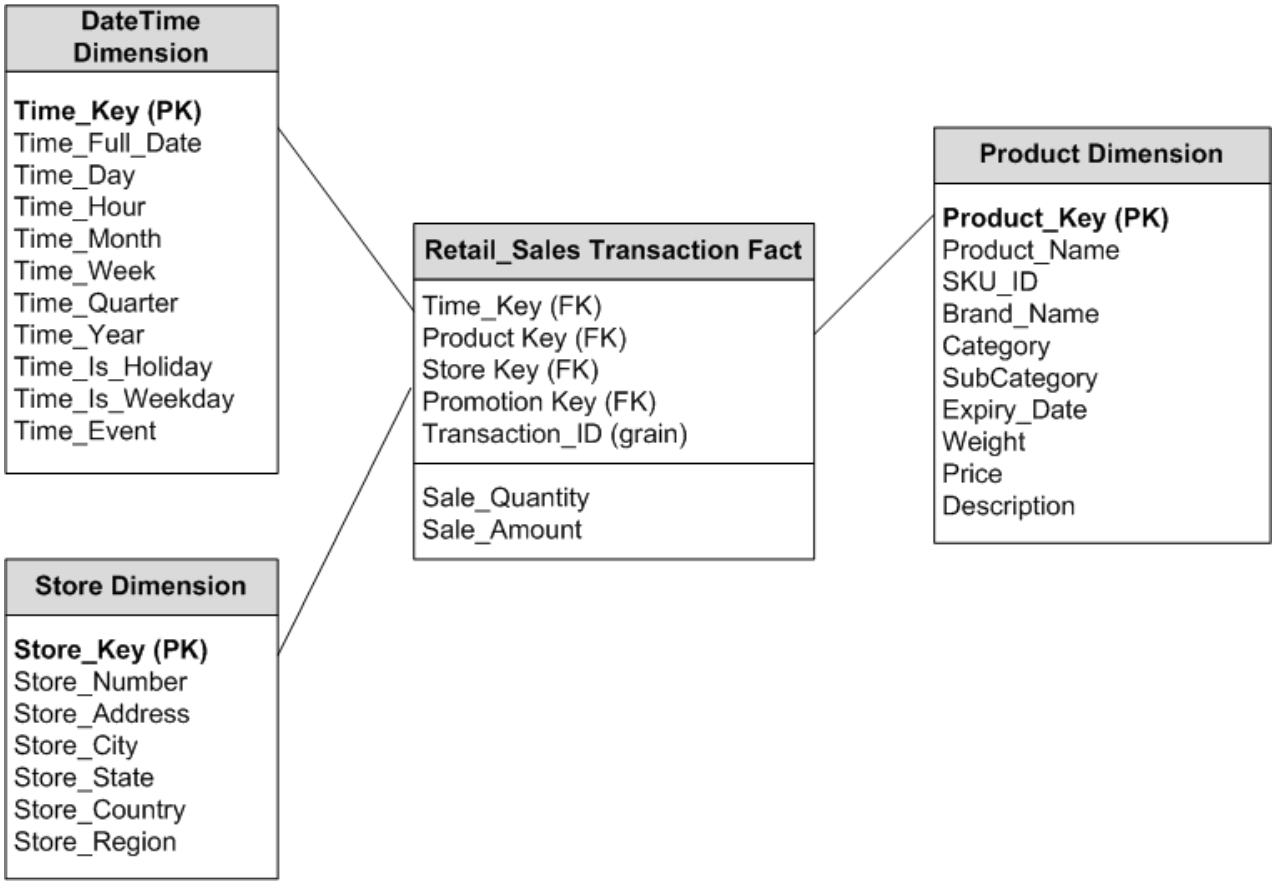
The facts collected by the Retail Point of Sales system are:

- Quantity of Sales
- Amount of Sales

Design the Data Model

A data model illustrates the relationship between entities (facts and dimensions tables) and their attributes (database table columns). In HPE OBR, a data model is an XML file. To create a data model start by creating a schema diagram and then implement the same into an XML file.

The following figure shows the schema diagram that we will use to create the data model for the `RetailPOS` Content Pack.



Implement the Data Model in XML

The schema that you designed in the previous step must be implemented using XML to be used by the CDE to create the Domain component package. This XML file is called Model XML.

A typical Model XML file has the following sections:

- **Relational section** defines the fact tables and dimension tables, and the relationship between the facts and dimensions.
- **Logical section** defines the cubes, hierarchies, and levels. You define one cube for every fact table.
- **Aggregate section** defines the aggregation to be performed on the source tables.

After you create the Domain component directory structure, you will find a template `model_template.xml` at the following path:

`%CDE_HOME%\workspace\RetailPOS\RetailPOSDomain.ap\source\model`

You can edit this `.xml` file to author the Model XML.

Sample

For your reference, you can find a sample `model.xml` file for the `RetailPOSDomain` content pack at the following location:

%CDE_HOME%\samples\RetailPOS_Demo_Content_Pack\Source\RetailPOS\RetailPOSDomain.ap\source\model

> **Note:** To view the contents of the sample Model XML file, on this PDF document click the **Attachments: View File Attachments** tab and select `RetailPOS_dimension_model.xml` file. Double-click the file to open the XML file on your browser window.

Create the Workflow Streams

HPE OBR provides a workflow framework to control and monitor the workflow of the data processes. This framework is made up of workflow streams. A stream is made up of steps that are related to one another in a sequential relationship. Each content pack contains a set of streams that define and control the flow of data from one step to another.

As an example, the Domain component may have the following steps in the workflow stream:

`Loading data to rate table > Hourly aggregation > Daily aggregation`

You must implement the workflow streams in XML to be used by CDE to create the Domain component package. The streams that you must create using XML are:

- One workflow stream XML to load and aggregate the fact.
- One workflow stream XML to load the `Store` dimension.
- One workflow stream XML to load the `Product` dimension.

After you create the directory structure for the Domain component, you will find a workflow stream template file named `ABC_stream_template.xml` at the following location:

%CDE_HOME%\workspace\RetailPOS\RetailPOSDomain.ap\source\orchestration\stream_definitions

You can edit this `ABC_stream_template.xml` file to author the workflow streams XML files.

Also, as reference you can find sample workflow streams XML files for the **RetailPOSDomain** at the following location:

%CDE_HOME%\samples\RetailPOS_Demo_Content_Pack\Source\

- "Implement the Data Model in XML" on page 39: Workflow stream XML to load and aggregate the fact.
- Dimension_Store_stream.xml: Workflow stream XML to load the Store dimension.
- Dimension_Product_stream.xml: Workflow stream XML to load the Product dimension.

> **Note:** To read the contents of the sample workflow stream XML files, click the **Attachments: View File Attachments icon** on this PDF document, and double-click the .xml files.

Generate Domain Component Package

To generate the Domain component package, follow these steps:

1. **Create Manifest XML file**

   The Manifest XML file contains the definitions of the source files to be used by CDE to generate the Domain component package.

   To create the Manifest XML file using CDE:

   a. Navigate to the %CDE_HOME%\workspace\RetailPOS\RetailPOSDomain.ap folder.

   b. Run the following command:

   ```
   ant createManifestTemplate
   ```

   The Manifest XML file named RetailPOSDomain_manifest_template.xml is created in %CDE_HOME%\workspace\RetailPOS\RetailPOSDomain.ap\source

   You can see a sample Manifest XML file for the RetailPOSDomain at the following location:

   ```
   %CDE_HOME%\samples\RetailPOS_Demo_Content_
   Pack\Source\RetailPOS\RetailPOSDomain.ap\source\RetailPOSDomain_manifest_
   template.xml
   ```

   > **Note:** To group custom content pack components in HPE OBR **Administration Console** > **Deployment Manager**:
   >
   > a. Open the <ContentPack>_manifest_template.xml in an XML editor.
   >
   > b. Under the **metadata** element, add your **datasource_application** and **content** names in the **value** attribute as shown in the sample below:

Enable custom content licensing

Custom content licensing helps you account for the usage of custom content packs you create using the OBR CDE. At present, custom content licensing is enabled based on the node count.

> **Note:** Licensing type has to be specified before running the ANT build for Content generation.

Below mentioned are the licensing types supported in OBR Custom Content:

- **STATIC** - When you have the node count for the new custom content, you can specify the node count in the value attribute and STATIC as the `license type` in the license section of the `<ContentPack_Name>_manifest_template.xml`file.
- **SQL** - If you know the information about the table that contains node data, you can specify SQL query in the value attribute and SQL as the license type in the License section in the `<ContentPack_Name>_manifest_template.xml`.
- **DEPENDS** - If the newly created custom content is dependent on any existing custom content and for which already licensing is calculated, specify the custom content name in the `value` attribute and DEPENDS as the `license type` in the `<ContentPack_Name>_manifest_template.xml`.
- **COMMAND** - If you want to calculate license usage based on a custom Perl script, perform the following steps:

1. Create the Perl script and suffix the script name with _license.pl. For example, if the Perl script name is `test`, the script name should be `test_license.pl`.
2. Place the script in the `%CDE_HOME%\workspace\<package>\<subpackage>.ap\source\scripts\perl\` folder.
3. Specify COMMAND as the license type and add the script name (Example, `test_license.pl`) in the value attribute in the License section in the `<ContentPack_Name>_manifest_template.xml` file.

   When you run the perl script for the license type COMMAND, the you will get the license count. For example `LICENSECOUNT:100.`

> **Note:** Only Perl scripts are supported as commands. Necessary entries are made in the `install.pkg` file after the package generation and licensing artifacts are copied to required directories during custom content pack installation. The OBR License manager performs the calculation and displays overall count in OBR Administration Console.

Steps to enable custom content licensing

1. Create Content pack folders for the Custom Content.
2. Create Manifest XML for the Custom Content after authoring all required Content Artifacts.
3. Modify the license in the `<ContentPack_Name>_manifest_template.xml` file. You need to modify value and type attributes of the `licenseDef` element.

You can define four types of licenses for custom content: STATIC, SQL, DEPENDS, and COMMAND.

Example:

```
<license>
<licenseDef value="500" type="STATIC"/>
</license>
```

Build Domain component package

To build the Domain component package using CDE:

1. Navigate to the `%CDE_HOME%\workspace\RetailPOS\RetailPOSDomain.ap` directory.
2. Run the following command:

   ```
   ant
   ```

   The installable Domain component package is created in *%CDE_HOME%*`\workspace\RetailPOS\RetailPOSDomain.ap\dist`
3. Navigate to `%CDE_HOME%\workspace\RetailPOS\RetailPOSDomain.ap\dist` directory.
4. Copy `RetailPOS` to *%PMDB_HOME%*`\packages\RetailPOS` directory. Copying the package makes it available in the Deployment Manager for installation.

> **Note:** It is not necessary to have HPE OBR installed on the machine where you are creating the Domain component package. If you created the Domain component package on another machine, you must copy the package to the HPE OBR machine under *%PMDB_HOME%*`\packages`.

Install the Domain Component Package

HPE OBR provides the Deployment Manager utility on the Administration Console to install the content pack component packages. For instructions on how to install the content pack components, see the *HPE Operations Bridge Reporter Configuration Guide.*

Verifying Workflow Streams on Administration Console

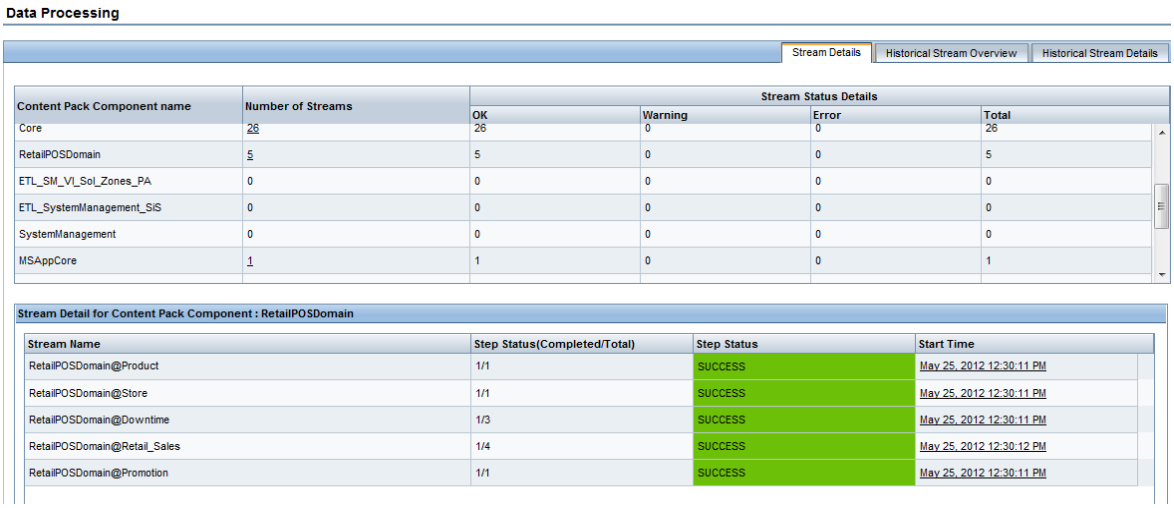
After the `HPE_PMDB_Platform_Timer` service is started, you logon to the Administration Console to check the status of the Domain component workflow streams. Follow these steps:

1. On the Administration user interface, click **Internal Monitoring > Data Processing**.
2. On the **Stream Details** tab, view the status of streams in the RetailPOSDomain content pack. All streams must show a status of OK to indicate successful completion.

In the RetailPOS example, the Domain component has the following workflow streams with one or more steps within each stream.

- A workflow stream XML to load and aggregate the `Retail_Sales` fact.
- A workflow stream XML to load the `Store` dimension.
- A workflow stream XML to load the `Product` dimension.
- A workflow stream XML to load the `Promotion` dimension.

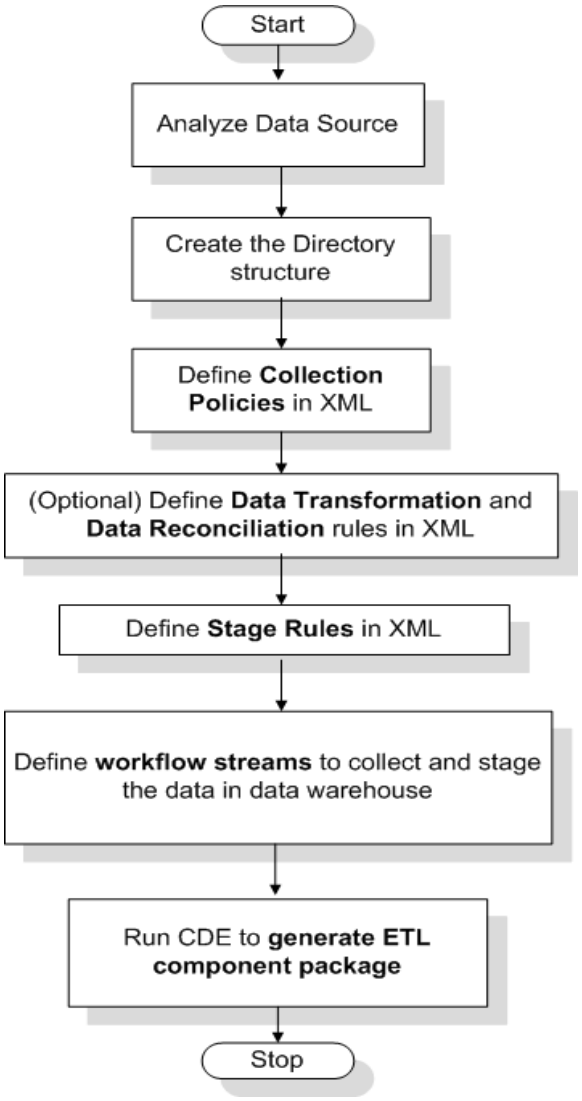
As shown in the following figure, successful completion of the streams is indicated using green color.



Creating and Installing the ETL Component Package

This chapter describes the steps to create an ETL component package for the existing RetailPOS Domain component package.

Creating an ETL Component package involves the following steps:



Note: This Chapter does not describe data reconciliation.

Prerequisite

Domain Component Package

Before you start creating ETL component package, you must create and install the Domain component package. Follow the steps described, see "Creating and Installing the Domain Component Package" on page 36 to create the Domain model, and to use CDE to generate and install the Domain component package.

Analyze the Data Source

Before you start creating the ETL component, you must identify the data source containing metrics that are suitable to be fed into the Domain data model. See "Content Pack Architecture " on page 14

In this guide we consider a simple database as the source of data. The database is created using PostgreSQL software and contains data suitable to be fed into the RetailPOS Domain model. Sample files and scripts are provided in the HPE OBR media to create the database and populate data into the tables. The "Creating a Data Source for ETL Component" describes how to create the PostgreSQL database, the database tables, and how to insert data into the tables.

Create the Directory Structure

To create the directory structure for the ETL component source files, at the command prompt run the following command:

```
<installation_directory>\CDE\bin>CreateCPFolders.bat -package RetailPOS -subpackage
RetailPOSETL -type etl
```

where,

- *<installation_directory>* is the directory where you have installed HPE OBR
- `RetailPOS` is the name of the content pack you are developing
- `RetailPOSETL` is the name of the ETL component within `RetailPOS`

The directory contains templates provided by HPE OBR to create the ETL component source files.

HPE OBR provides sample source files for the Retail POS content pack at the following location:

```
%CDE_HOME%\samples\RetailPOS_Demo_Content_Pack\Source\
```

Define Collection Policy in XML

To collect the data from the `retailpos` database tables you must define a collection policy in XML. Use the collection policy template named `DB_collection_template.xml` available in the following folder:

%CDE_HOME%`\workspace\RetailPOS\RetailPOSETL.ap\source\etl\collection\`

For reference see the sample `RetailPOS_DB_Collection_Policy.xml` in the following folder:

%CDE_HOME%`\samples\RetailPOS_Demo_Content_Pack\Source\RetailPOS\RetailPOSETL.ap\source\etl\collection\`

To read the contents of the sample collection policy XML file, on this PDF document click the **Attachments: View File Attachments** icon and select `RetailPOS_DB_Collection_Policy.xml`. Double click to open the XML file on your browser window.

Define Data Transformation Rules

In the RetailPOS example, the data transformation rule is used to append the name of the city, state, country, and the zip code to the address column; and use the whitespace character as a delimiter of the column values in the `.csv` files.

For reference, see the sample `RetailPOS_transformation.xml` in *%CDE_HOME%*`\samples\RetailPOS_Demo_Content_Pack\RetailPOS\RetailPOSETL.ap\doc`

To read the contents of the sample data transformation rule XML file, on this PDF document click the **Attachments: View File Attachments** icon and select `RetailPOS_transformation.xml`. Double-click to open the XML file on your browser window.

Define Stage Rules

Stage rules defined in XML files are used to map the column names in the source `.csv` files to the column names in the target physical tables, called staging tables, in the database. Use the stage rules template available in the folder `RetailPOS\RetailPOSDomain.ap\source\stagerule_templates` to write the stage rules XML files for each of the dimensions – product, sales, and store.

For reference, see the sample stage rule XML files in *%CDE_HOME%*`\samples\RetailPOS_Demo_Content_Pack\RetailPOS\RetailPOSETL.ap\source\etl\stage_rules`

For all the OOTB domain content packs, stage rule templates are available in the folder `%CDE_HOME%\cplib\<ContentPack>\<SubPackage.ap>\source\stagerule_templates`

Taking the System Management Content Pack for example, the syntax is `C:\HPE-OBR\CDE\cplib\SystemManagement\CoreSystemManagement.ap\source\stagerule_templates`

To read the contents of the sample stage rule XML files, on this PDF document click the **Attachments: View File Attachments** icon and select the following files:

- `Stage_product_stagerule.xml`: Stage rule for Product
- `Stage_retail_sales_stagerule.xml`: Stage rule for Sales
- `Stage_store_stagerule.xml`: Stage rule for Store

Double-click to open the XML files on your browser window.

Define Workflow Streams

Workflow streams defined in XML files are used to collect and stage the `.csv` files in the data warehouse.

When topology information is collected, HPE OBR Collector generates a `*relations*.csv` file for each topology view in addition to the CI type mentioned as part of the RTSM collection policy. You must author stage rules to stage this data and workflow streams to enable the process of loading the `*relations*.csv` files.

The `*relations*.csv` files contain topology data that must be loaded to the CI Bridge table. To create a stage rule definition for the CI Bridge, use the stage rule templates for the CI Bridge available in the folder `%CDE_HOME%\cplib\Core\Core.ap\source\stagerule_templates[Core_0_Stage_K_CI_Bridge_0_stagerule.xml]`.

In addition to authoring stage rules, you must define a workflow stream to enable the data flow. For the syntax on creating ABC Streams, see "Part III: Content Development Reference" on page 80.

> **Note:** In the COLLECT step, use the "View name" specified in the collection policy as the Category and "Relations" as the Type.

Use the workflow streams template available in the folder *%CDE_HOME%*`\samples\RetailPOS_Demo_Content_Pack\source\RetailPOS\RetailPOSETL.ap\source\orchestration\stream_definitions` to write the workflow stream XML files, one file for each of the dimensions – product, sales, and store.

For reference see the sample workflow stream XML files in *%CDE_HOME%*`\samples\RetailPOS_Demo_Content_Pack\RetailPOS\RetailPOSDomain.ap\source\orchestration\stream_definitions`

To read the contents of the sample ETL workflow stream XML files, on this PDF document click the **Attachments: View File Attachments** icon and select the following files:

- `Dimension_Product_ETL_stream.xml`: Workflow stream XML for Product dimension
- `Dimension_Store_ETL_stream.xml`: Workflow stream XML for Store dimension
- `Fact_Retail_Sales_ETL_stream.xml`: Workflow stream XML for the fact

Double-click to open the XML files on your browser window.

Generate the ETL Component

You must create Manifest XML to generate ETL component package. The Manifest XML file contains the definitions of the source files to be used by CDE to generate the ETL component package.

1. **To create the Manifest XML file using CDE:**
   a. Navigate to the **%CDE_HOME%\workspace\RetailPOS\RetailPOSETL.ap** folder.
   b. Run the `ant createManifestTemplate`command:

   The Manifest XML file named `RetailPOSETL_manifest_template.xml` is created in *%CDE_HOME%*`\workspace\RetailPOS\RetailPOSETL.ap\sourc`

   You can see a sample Manifest XML file for the `RetailPOSETL` at the following location:

   %CDE_HOME%`\samples\RetailPOS_Demo_Content_Pack\Source\RetailPOS\RetailPOSETL.ap\source\RetailPOSETL_manifest_template.xml`

   To read the contents of the sample Manifest XML file, on this PDF document click the **Attachments: View File Attachments** icon and select `RetailPOSETL_manifest_template.xml`.Double-click to open the XML file on your browser window.
2. **To create the ETL component package using CDE:**
   a. Navigate to the *%CDE_HOME%*`\workspace\RetailPOS\RetailPOSETL.ap` directory.
   b. Run the `ant` command.

      The installable ETL component package is created in

      %CDE_HOME%`\workspace\RetailPOS\RetailPOSETL.ap\dist`
3. Navigate to *%CDE_HOME%*`\workspace\RetailPOS\RetailPOSETL.ap\dist` directory.
4. Copy `RetailPOS` to *%PMDB_HOME%*`\packages\RetailPOS`. Copying the package makes it available in the Deployment Manager for installation.

> **Note:** It is not necessary to have HPE OBR installed on the machine where you are creating the ETL component package. If you created the ETL component package on another machine, you must copy the package to the HPE OBR machine under *%PMDB_HOME%*`\packages`.

Install the ETL Component Package

HPE OBR provides the Deployment Manager feature on the Administration Console to install the content pack component packages. Use the Deployment Manager to install the ETL component package; in the RetailPOS example, the name of the package is RetailPOSETL.

For instructions on how to install the content pack components by using Deployment Manager, see the *HPE Operations Bridge Reporter Online Help for Administrators.*

Verify the ETL Component Package

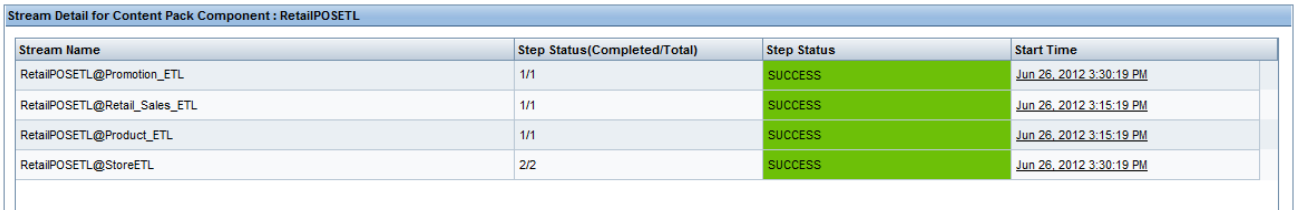
After you install the ETL component package and the `HPE_PMDB_Platform_Timer` service is started, you log on to the Administration user interface to check the status of the ETL component workflow streams. Follow these steps:

1. On the Administration user interface, click **Internal Monitoring** > **Data Processing**.
2. On the **Stream Details** tab, view the status of streams in the RetailPOSETL content pack. All streams must show a status of OK to indicate successful completion.

In the RetailPOS example, the ETL component has the following workflow streams with one or more steps within each stream.

- A workflow stream `RetailPOSETL@Retail_Sales_ETL` to move Sales fact to stage tables.
- A workflow stream `RetailPOSETL@Product_ETL` to move Product dimension to stage tables.
- A workflow stream `RetailPOSETL@Store_ETL` to move Store dimension to stage tables.
- A workflow stream `RetailPOSETL@Promotion_ETL` to move Promotion dimension to stage tables.

As shown in the following figure, successful completion of the streams is indicated using green color.



Stream Detail for Content Pack Component : RetailPOSETL

Stream Name	Step Status(Completed/Total)	Step Status	Start Time
RetailPOSETL@Promotion_ETL	1/1	SUCCESS	Jun 26, 2012 3:30:19 PM
RetailPOSETL@Retail_Sales_ETL	1/1	SUCCESS	Jun 26, 2012 3:15:19 PM
RetailPOSETL@Product_ETL	1/1	SUCCESS	Jun 26, 2012 3:15:19 PM
RetailPOSETL@StoreETL	2/2	SUCCESS	Jun 26, 2012 3:30:19 PM

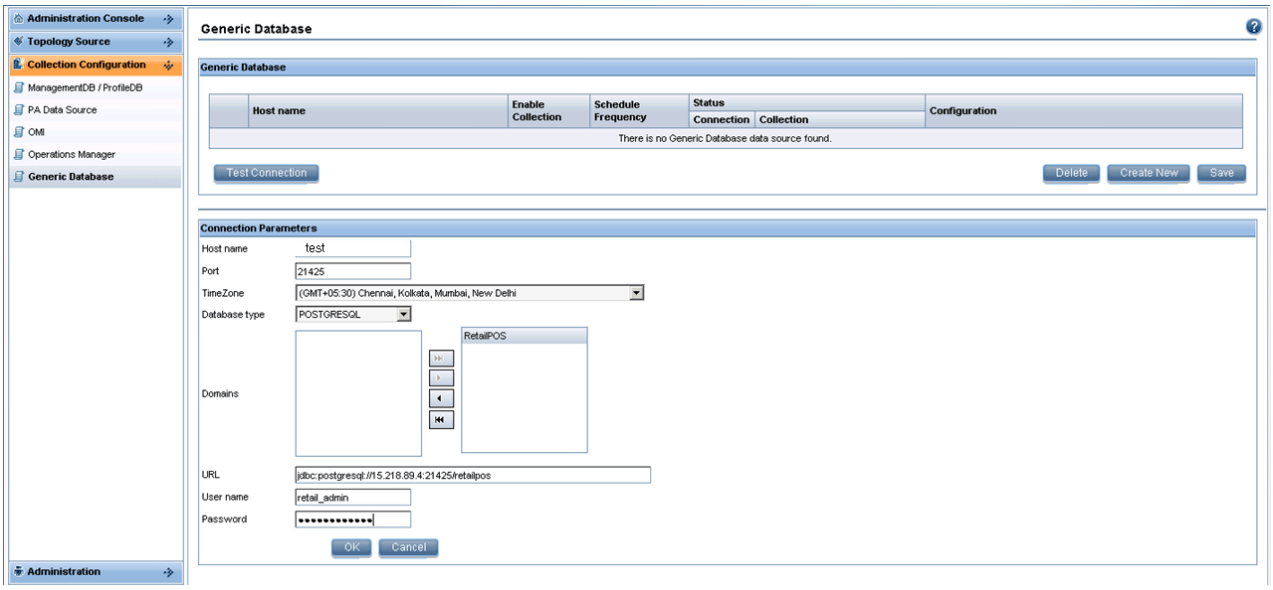
Working with the ETL Component

Configure a Generic Database

After you install the RetailPOS Domain and ETL components, you must configure a connection to the `retailpos` database to collect data based on the collection policy XML. You configure the connection on the Administration Console using the Generic Database page. Follow these steps:

1. In the Administrator Console, click **Collection Configuration** > **Generic Database**. The Generic Database page opens.
2. Click **Create New.** The Connection parameters dialog box appears.
3. Type or select the following values:

Field	Description
Host name	Type the IP address or FQDN of the server where you created the `retailpos` database.
Port	Type the port number to query the database server.
TimeZone	Select the time zone under which the database instance is configured.
Database type	Select POSTGRESQL.
Domain	Select RetailPOS.
URL	Type `jdbc:postgresql//<server>:<port>/retailpos`
User Name	Type the name of the generic database user. In this example, the user name is `retail_admin`.
Password	Type the password of the generic database user. In this example, the password is retail admin.



View Reports

Now that you installed the Domain and Reports component packages and the data is loaded into the data warehouse, you can view the report on the SAP BusinessObjects BI Launch Pad interface. See "Viewing Reports on SAP BusinessObjects BI Launch Pad" on page 71.

Custom Data Loading Using CSV Files

The ETL component consists of data collection, transformation, reconciliation, and staging rules. The creation of the complete ETL component using all the rules can be quite complex. Therefore, to enable you to create a sample content pack this chapter describes a simple alternative method to generate data in the form of `.csv` files and load into the data warehouse.

In this method you create a set of CSV files in the required format and place them in the following locations which are then loaded into the HPE OBR data warehouse tables:

%PMDB_HOME%`\stage`

> **Note:** For the CSV files to get processed, ensure that the files you create should contain the CRLF as newline character. If you are not able to create CSV file with CRLF as newline character, you introduce a transform step prior to staging the CSV files. The transformation step rewrites the files to the required format. For information on transform step, see Transformation policy section.

Perform these steps to create and load the `.csv` files.

1. **Install the Domain component package:** Before you begin creating the CSV files for loading, make sure you generate the Domain component package you created in "Install the Domain Component Package" on page 44. The Domain component creates a `stage interface` html file that contains the format in which the CSV files must be created. The stage interface html file is created in the following directory:

   %PMDB_HOME%`\packages\RetailPOS\RetailPOSDomain.ap\doc`

   For reference, see the example `RetailPOSDomain_INTERFACE.html` file available in *%CDE_HOME%*`\samples\RetailPOS_Demo_Content_Pack\RetailPOS\RetailPOSDomain.ap\doc`.

2. **Generate CSV files**: HPE OBR provides a simple CSV files generator program to create sample `.csv` files for the RetailPOS content pack. To create the CSV files in the given format provided in the template .html file by using the `.csv` file generation program follow these steps:

   a. Navigate to the folder *%CDE_HOME%*`\samples\RetailPOS_Demo_Content_Pack\Sample CSV Generator`.

   b. Copy the following files to the specified location as shown in the following table:

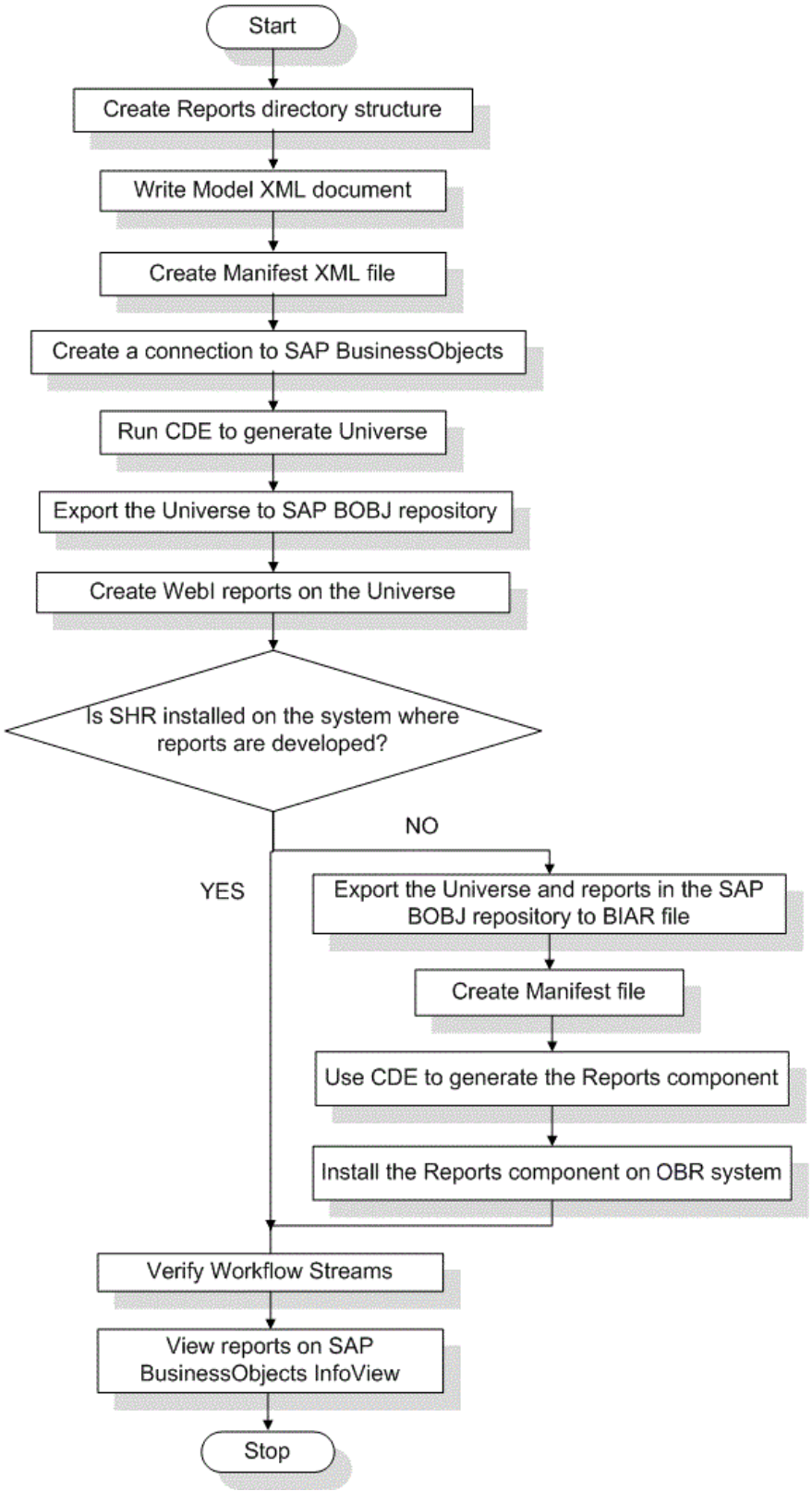
File	Copy to location
`retailpos_csvgen.exe`	*%PMDB_HOME%*`\bin`
`retailpos_csvgen.ini`	*%PMDB_HOME%*`\config\startup`
`retailposcsvgen.jar`	*%PMDB_HOME%*`\lib`

   c. Run the `retailpos_csvgen.exe.` in the command prompt.

For the sample RetailPOS content pack that you are creating, the `.csv` file generator program generates two months worth of `.csv` files and places these files in the `%PMDB_HOME%\stage` folder. The Domain component that you installed earlier loads the CSV files to the HPE OBR data warehouse tables.

Creating and Installing the Reports Component Package

The following flowchart shows the steps to create the Reports component.


Start
Create Reports directory structure
Write Model XML document
Create Manifest XML file
Create a connection to SAP BusinessObjects
Run CDE to generate Universe
Export the Universe to SAP BOBJ repository
Create Webl reports on the Universe
Is SHR installed on the system where reports are developed?
NO
YES
Export the Universe and reports in the SAP BOBJ repository to BIAR file
Create Manifest file
Use CDE to generate the Reports component
Install the Reports component on OBR system
Verify Workflow Streams
View reports on SAP BusinessObjects InfoView
Stop

Prerequisites

Before you create the Reports component, make sure the following are done:

- CDE is installed on the same machine where you have installed HPE OBR and SAP BusinessObjects.
- The Domain component that you created in "Install the Domain Component Package" on page 44 is installed. Use the Deployment Manager utility on the Administration Console to install the Domain component. For instructions, see the *HPE Operations Bridge Reporter Online Help for Administrators*.

Create the Directory Structure

To create the directory structure for the Reports component source files, at the command prompt run the following command:

<installation_directory>`\CDE\bin> CreateCPFolders.bat -package RetailPOS -subpackage RetailPOSApp -type application`

where,

- *<installation_directory>* is the directory where you have installed HPE OBR
- `RetailPOS` is the name of the content pack you are developing
- `RetailPOSReporting` is the name of the Reports component within `RetailPOS`

The directory will contain templates provided by HPE OBR that you will use to create the Reports component source files.

HPE OBR provides sample source files for the Retail POS content pack at the following location. You can use these files as reference to create your own content pack.

%CDE_HOME%`\samples\RetailPOS_Demo_Content_Pack\Source\`

Write the Model XML Document

The Model XML document for the Reports component must have a Logical section. In this section, you provide a reference to the cube that you defined in the Domain component Model XML document.

For reference find a sample Model XML file for the `RetailPOSReporting` available at the following location:

%CDE_HOME%\samples\RetailPOS_Demo_Content_
Pack\Source\RetailPOS\RetailPOSReporting.ap\source\model

> **Note**: To read the contents of the sample Model XML file, on this PDF document click the **Attachments: View File Attachments** tab and select `RetailPOS_App_model.xml`Double-click to open the XML file on your browser window.

Create a Connection to SAP BusinessObjects

To generate the SAP BusinessObjects Universe you must create a secured database connection to SAP BusinessObjects by using the Information Design Tool (IDT). HPE OBR provides a batch script to create the connection.

To create a connection by using the script:

1. Open the command prompt and navigate to the `%CDE_HOME%\bin` directory.
2. Run the following command:

   `setenv.bat`
3. Run the following batch scripts:

   `updateCDEProperty.bat`

   > **Note:** The `updateCDEProperty.bat` script prompts for the following details:
   >
   > - `Hostname` (fully qualified domain name) of the SAP BusinessObjects system
   > - `User name` of the SAP BusinessObjects designer studio administrator (default: Administrator)
   > - `Password` of the above user (default: blank)

   `createUniverseConnection.bat`

   A message `'Default BO Universe connection ("MA") was created successfully'` appears.

> **Note:** The connection name must be unique across all application content pack components.
>
> To ensure a unique connection name, use the OBR_<content_pack_name> convention. You can change the default connection name by modifying the `bo.connection` parameter in the `cde.properties` file available at *$CDE\config* directory.

Use CDE to Generate SAP BusinessObjects Universe

To generate the universe by using CDE, at the command prompt:

1. Browse to the `%CDE_HOME%\workspace\RetailPOS\RetailPOSReporting.ap` directory.
2. Run the `ant` command.

The universe is created with the following file name extensions:

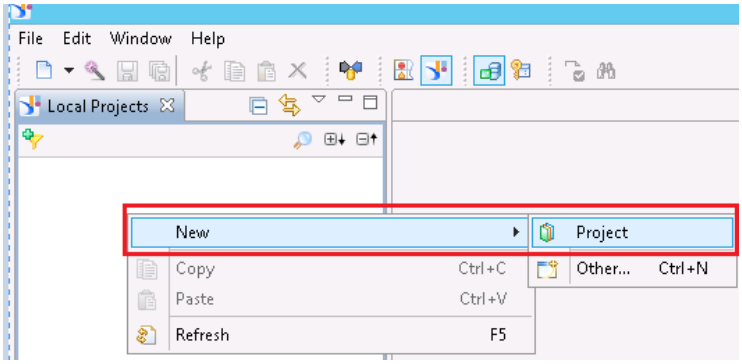
- `.blx`
- `.cnx`
- `.dfx`
- `.unx`

The files are available at `%CDE_HOME%\workspace\RetailPoS\RetailPoSReporting.ap\dist\RetailPOS\RetailPOS.ap\UnxFolder\UnxLayer`

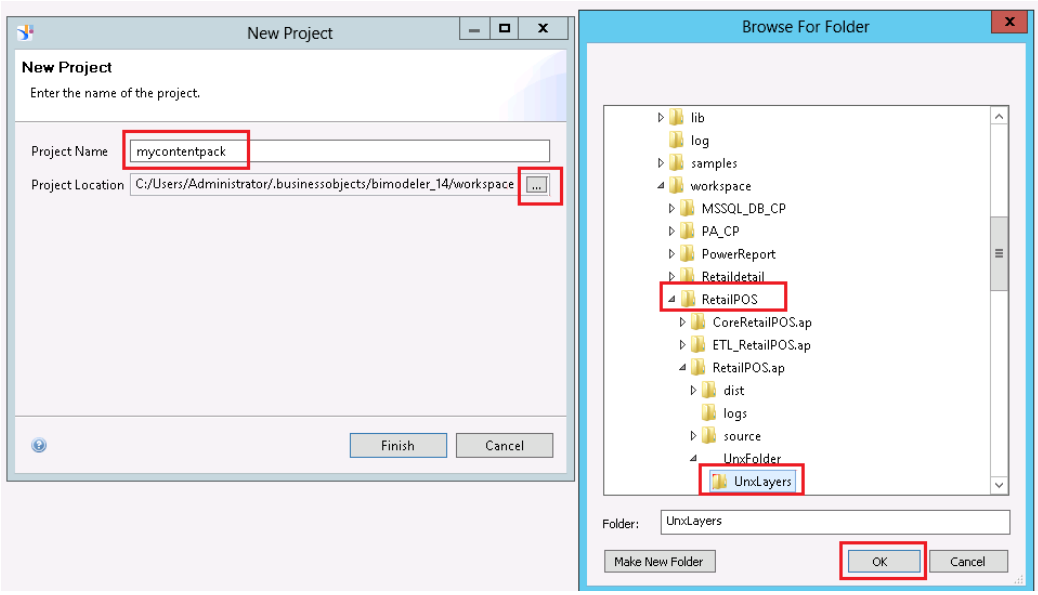
Export the Universe to SAP BusinessObjects Repository

Perform the following steps after the application content pack is generated using the OBR Content Development Environment (CDE) or OBR Content Designer:

1. Open Information Design Tool.
2. Right-click in the Local Projects area, and select **New** -> **Project** .



3. Specify the **Project Name** and **Project Location**. The project location is the path of unx folder (UnxFolder) generated by OBR CDE or O BR Content Designer.

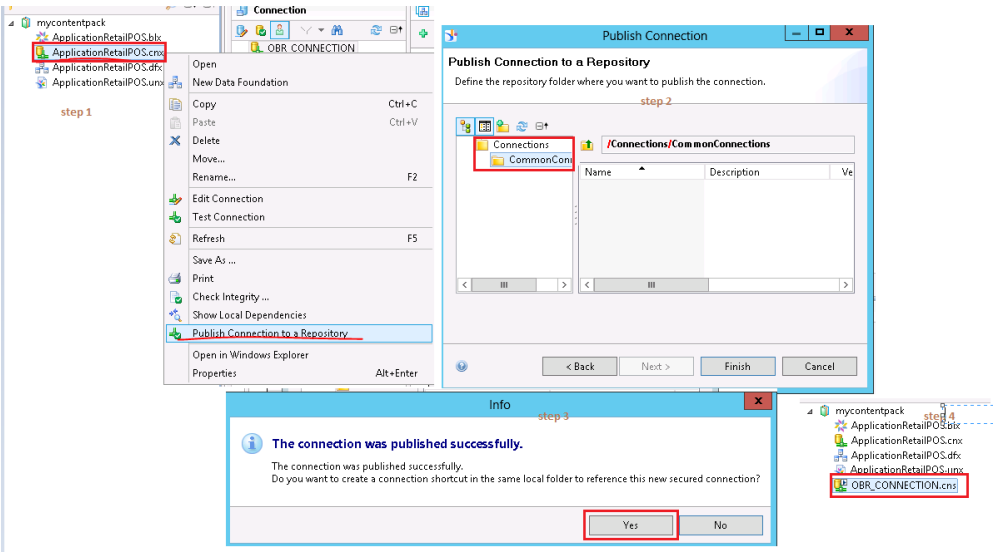


4. Click **OK**.

   Files are created with the following extensions:

   - `.blx`
   - `.cnx`
   - `.dfx`
   - `.unx`

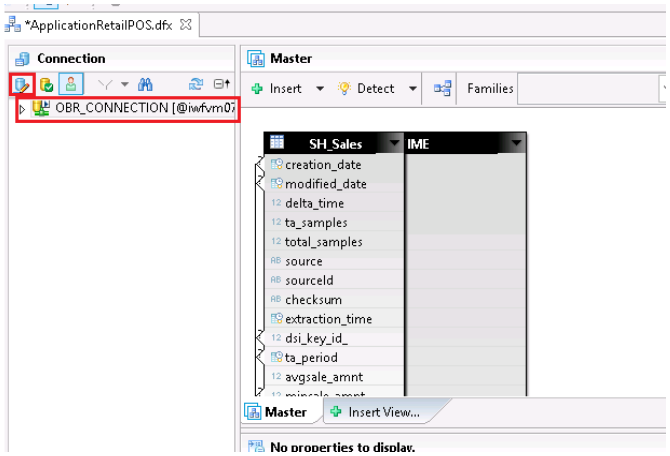
5. Right-click the **Application<*ContentPack*>.cnx** file and select **Publish Connection to a Repository**.



6. Specify the system details where you want upload the connection, and then save the connection into **CommonConnections** folder.

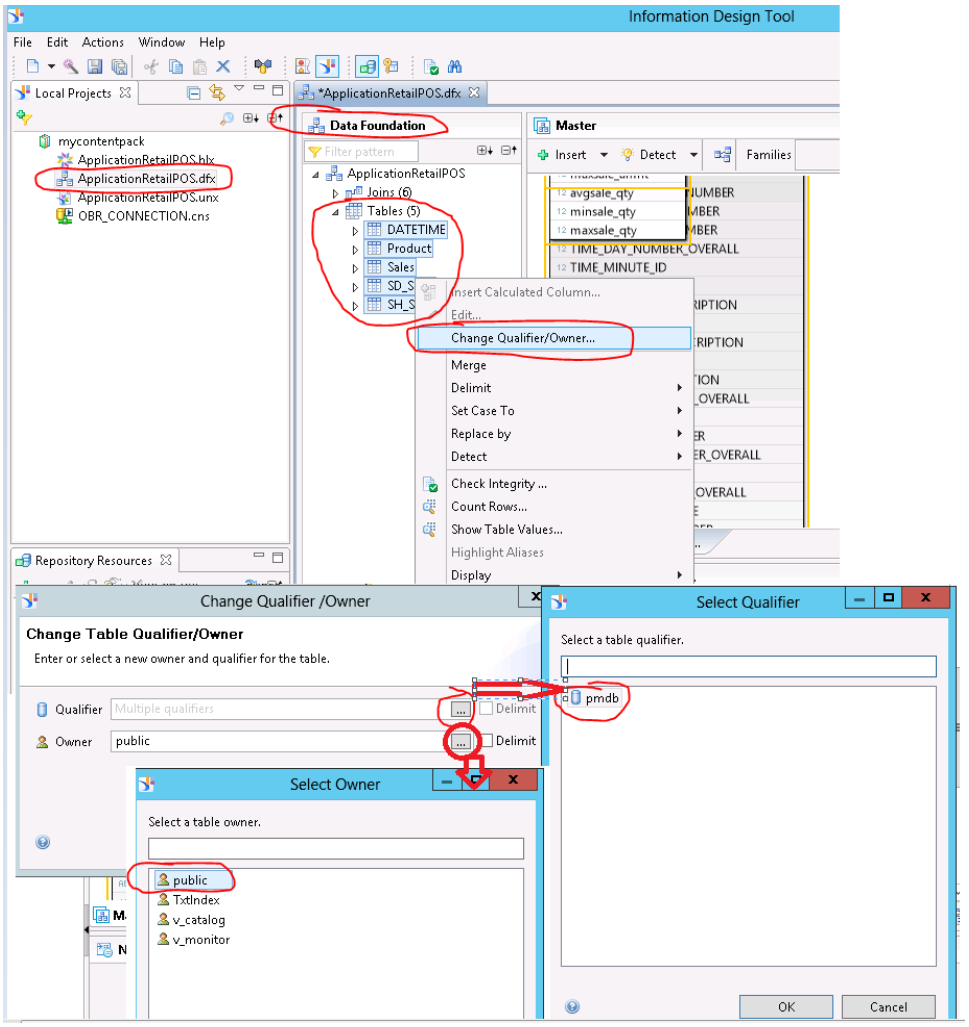
   Click **Yes** if you want to create a shortcut to the connection file, or else click No. Once done you will see `OBR_CONNECTION.cns` in your project

7. Delete the **Application<*ContentPack*>.cnx** file.
8. Double-click the **Application<*ContentPack*>.dfx** file; the file opens on the right.
9. Click the **Connection** -> **Change Connection** and select the new **OBR_CONNECTION.cns**.



10. Change the **Qualifier** and **Owner** as per your configuration during the on post-install configuration.

    By default, the Owner is `Public` and Qualifier is `pmdb`.



Note: If you are not able to change **Qualifier** and **Owner** successfully:

a. Go to `<BO installation Drive>:\Program Files (x86)\SAP BusinessObjects\SAP BusinessObjects Enterprise XI 4.0\dataAccess\connectionServer\jdbc\extensions\qt` directory.

b. Open the `vertica.prm` file.

c. Change the following parameters to `Y` instead of N:

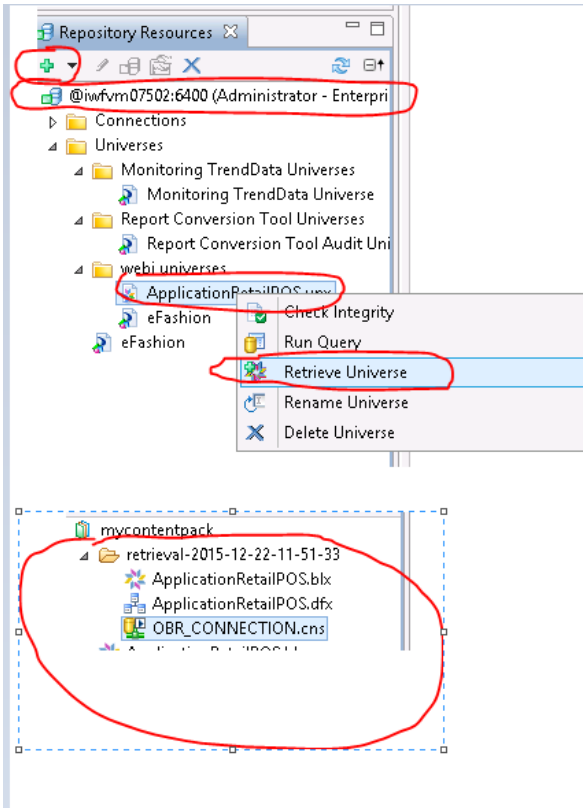
```
<Parameter Name="OWNER">Y</Parameter>
<Parameter Name="QUALIFIER">Y</Parameter>
```

11. Save the **Application<*ContentPack*>.dfx** file and close it.
12. Right-click the **Application<*ContentPack*>.blx** file and publish to the repository.

13. Select appropriate folder to export; if a folder does not exist, create a folder and export the **Application<*ContentPack*>.blx** file.

Points to remember :

1. The connection name must be `OBR_CONNECTION and` and the connection must be saved under the `CommonConnections` folder.
2. Qualifier and Owner must be appropriate as per your the custom change.
3. Please note that `.dfx` and `.cns` are connected with `.blx` layer. So, it is enough to export only the `.blx` layer export to repository.
4. If you want to import the universe from repository and make changes, right-click and retrieve the universe as shown below in IDT, so that you can view the `.dfx`and `.blx`. The `.cns` layer will extracted in a folder named with the date of retrial.

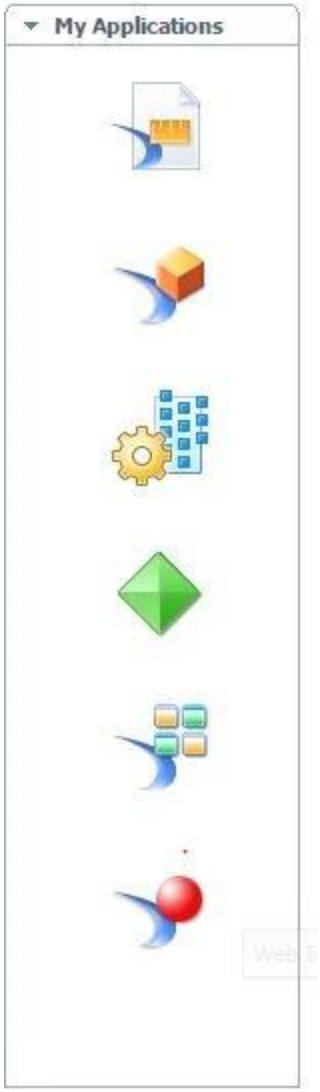


Create Web Intelligence Reports

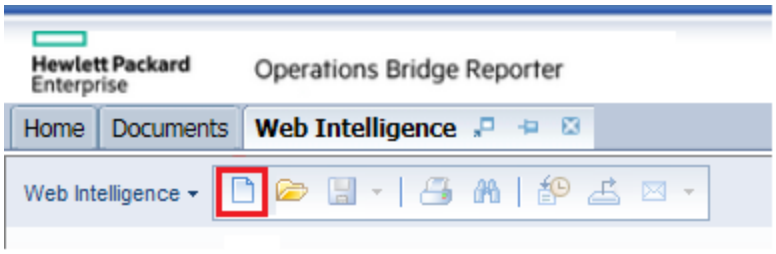
You can create Web Intelligence reports by selecting the universe in SAP BusinessObjects BI Launch Pad and building one or more queries to define the data content of the reports.

To create a simple sales report containing a table of sales quantity and sales amount per product category, follow these steps:

1. Logon to SAP BusinessObjects BI Launch Pad by using one of the following ways:
   - In the address bar of your web browser, type the URL of the SAP BusinessObjects system. The URL of the machine will be in the format: `https://<HostName>:8443/BI`.
   - In the Administration Console, click **Administration**>**SAP BOBJ** and then click **Launch BI Launch Pad**. The BI Launch Pad login page appears.
   - Log on to the SAP BusinessObjects BI Launch Pad with your System details, User Name and Password.
2. Under **My Applications**, click **Web Intelligence** icon.

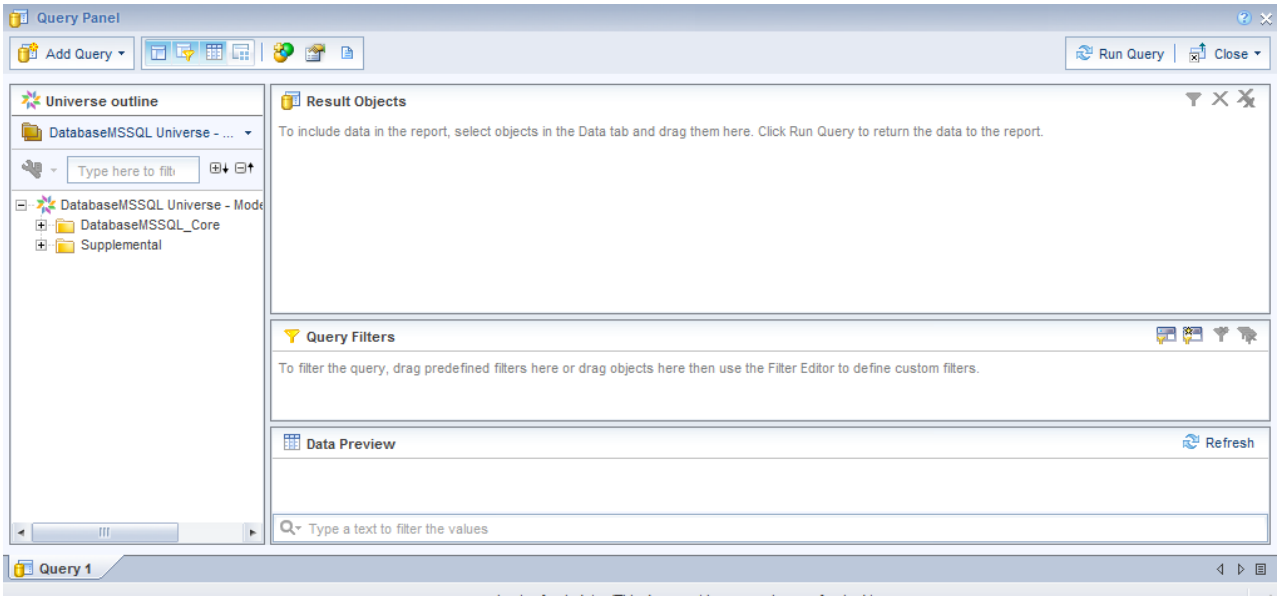


3. Click the **New** icon.



4. From the **Create a Document** page, click **Universe**.

The Query Panel window opens. The **Universe Outline** tab displays the objects – dimensions and measures – available in the universe as shown in the following figure:



5. Select **RetailPOSReporting Universe - Model Generator.**

6. To include data in the report, select the following in the data tab and drag them into the Result Objects window. Alternatively, you can double click the objects to place them in the Result Objects window.
   - Dimension: `Category` (under `Product(Retail Sales)`)
   - Measures:
     - `Sale Quantity` (under `Sales Measures`)
     - `Sale Amount` (under `Sales Measures`)
7. Click **Run Query** to return the data to the report.

   A table of Sale Quantity and Sale Amount by Product Category is created. You can rename the table to an appropriate title.

Enabling Time Drill Option in Reports

To enable time drill option in the report, click **Drill** on the BI Launch Pad toolbar. You can drill down and roll up by the product category dimension.

> **Note: Option 1:** If you developed the Web Intelligence reports on a different system where HPE

OBR is not installed, you must do the following:

- Export the BIAR file to the system where HPE OBR is installed
- Generate the Reports component and install it using the Deployment Manager.

See instructions for See "Exporting Reports to OBR" below

Option 2: If you developed the Web Intelligence reports on the same system where HPE OBR is installed, you can verify the workflow streams and view the reports on the SAP BusinessObjects BI Launch Pad as described in the following sections.

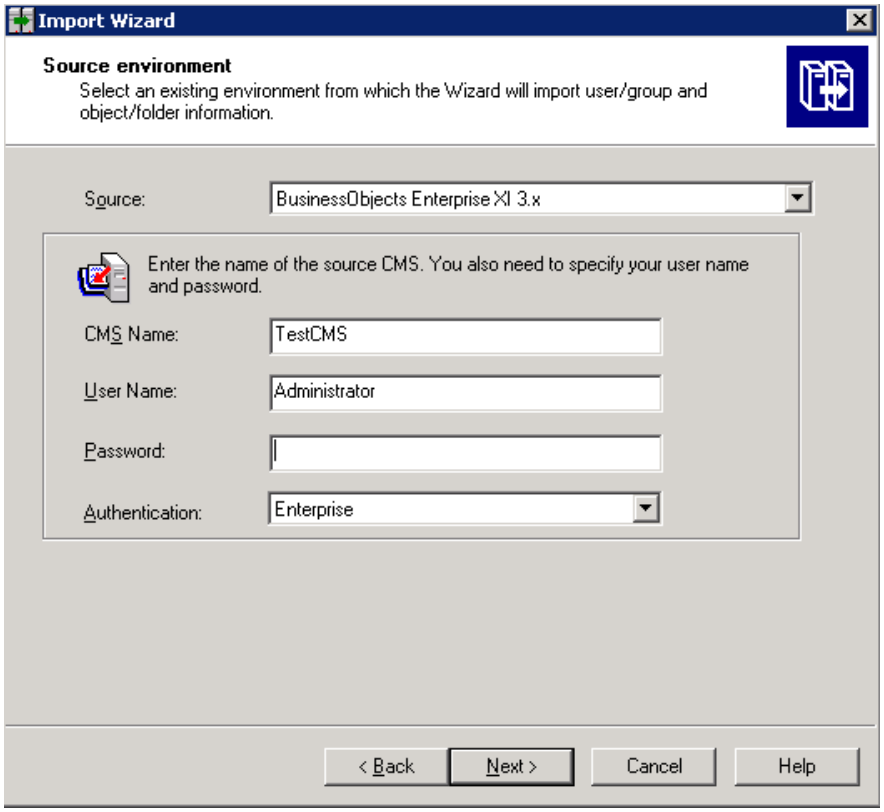
Exporting Reports to OBR

Export Business Intelligence Archive Resource (BIAR) File

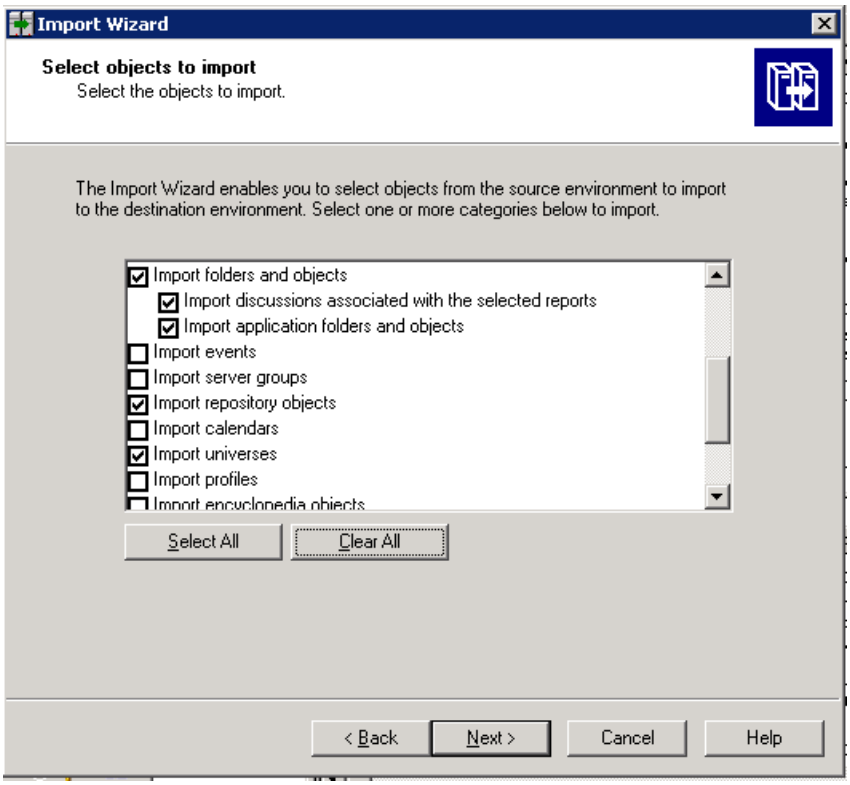
If you developed the reports on a system where HPE OBR is not installed, you must export the BIAR file and install the Reports component on the system where HPE OBR is installed. You select a source, a destination, and the objects that you want to import. Follow these steps:

1. Open the BusinessObjects Import Wizard.
2. On the **Source environment** page, enter the following:
    - CMS Name: The name of the machine on which BusinessObjects is installed.
    - User Name: The username for BusinessObjects user.
    - Password: The password for BusinessObjects user.

- Authentication: Select **Enterprise**.

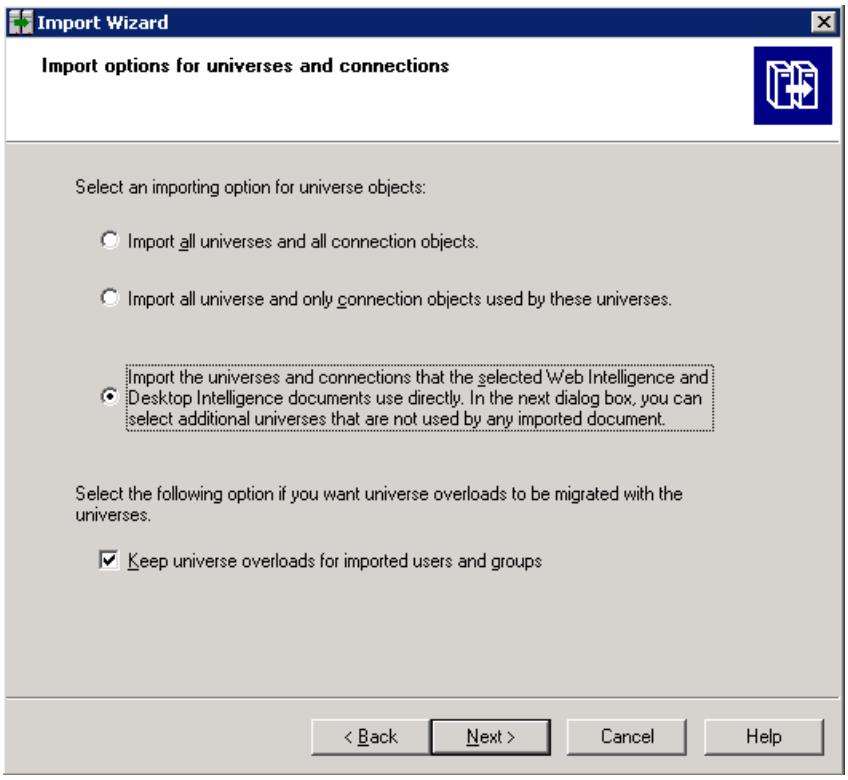


3. On the **Destination environment** page, select the following:
   - Destination: Business Intelligence Archive Resource (BIAR) File.
4. On the **Select objects to import** page, select the following:
   - Import folders and objects.
   - Import repository objects.
   - Import universes.

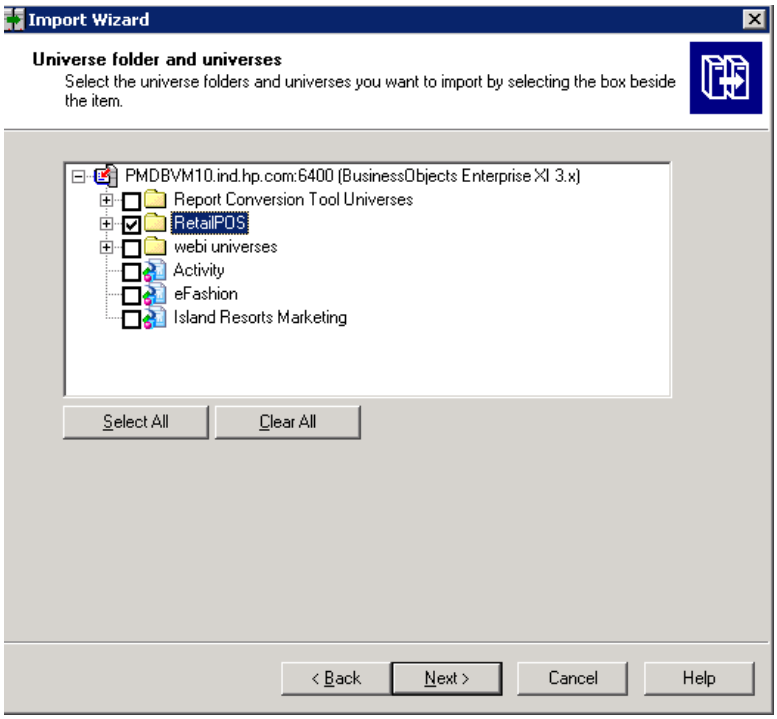


5. On the **Import options for universes and connections** page, select the following option:

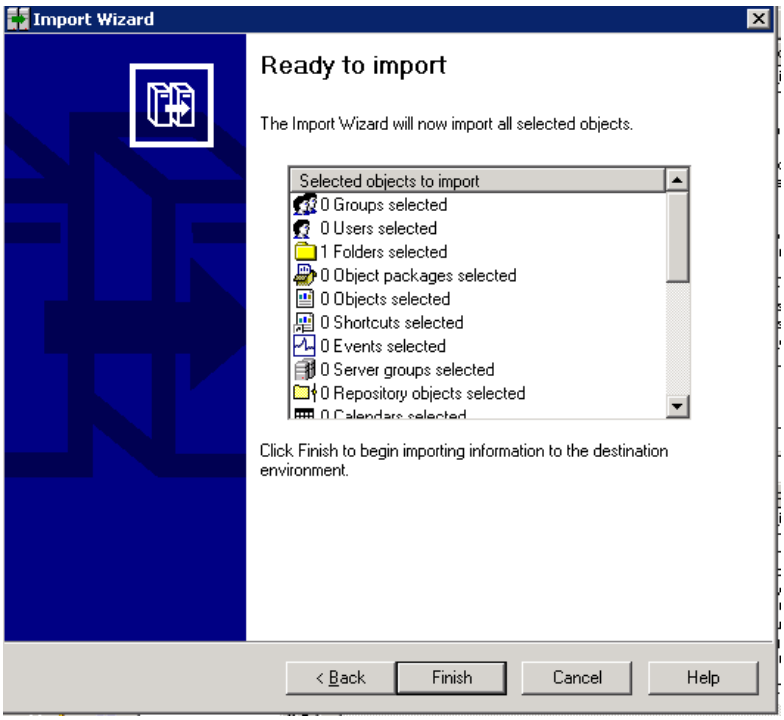
   `Import the universes and connections that the selected Web Intelligence and Desktop Intelligence documents use directly.`



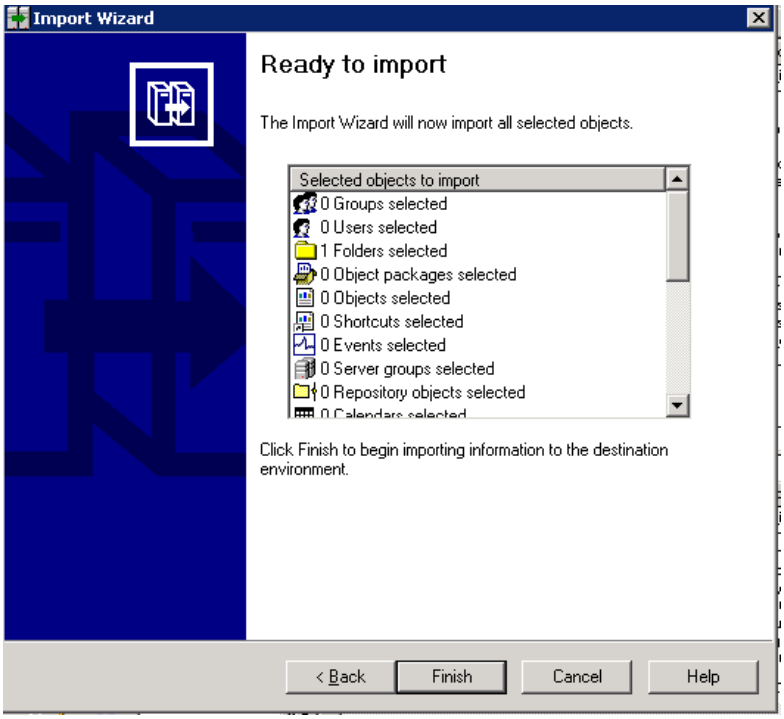
6. On the **Universe folder and universes** page, select the RetailPOS universe. Click **Next**.



7. On the **Import options for publications** page, select the **Do not import recipients** option.



The **Ready to import page** appears. Click **Finish** to import the universe.



Create the Manifest XML File

The Manifest XML file contains the definition of the BIAR file that you exported in the previous step to be used by CDE.

To create the Manifest XML file using CDE:

- Using `cd` command change directory to:

  %CDE_HOME%`\workspace\RetailPOS\RetailPOSReporting.ap`

- Run the following command:

  ```
  ant createManifestTemplate
  ```

The Manifest XML file, `RetailPOSReporting_manifest_template.xml` is created in *%CDE_HOME%*`\workspace\RetailPOS\RetailPOSReporting.ap\source`

Fore reference, see a sample Manifest XML file for the `RetailPOSReporting` at the following location:

%CDE_HOME%`\samples\`

> **Note:** To group custom content pack components in HPE OBR **Administration Console** > **Deployment Manager**:
>
> 1. Open the `<ContentPack>_manifest_template.xml` in an XML editor.
> 2. Under the **metadata** element, add your **datasource_application** and **content** names in the **value** attribute as shown in the sample below:

Use CDE to Generate Reports Component Package

To generate the Reports component package using CDE:

1. Using `cd` command change directory to:

   %CDE_HOME%`\workspace\RetailPOS\RetailPOSReporting.ap`

2. Run the following command

   ```
   ant createManifestTemplate
   ```

   The Reports component package `RetailPOSReporting.ap` is created in

   %CDE_HOME%`\workspace\RetailPOS\RetailPOSDomain.ap\dist`

3. Browse to *%CDE_HOME%*`\workspace\RetailPOS\RetailPOSDomain.ap\dist`.
4. Copy `RetailPOSReporting.ap` to *%PMDB_HOME%*`\packages\RetailPOS`

Install the Reports Component Package

HPE OBR provides the Deployment Manager utility on the OBR Administration Console to install the content pack component packages.

For instructions on how to install the content pack components, see the HPE Operations Bridge Reporter *Configuration Guide*.

After you install the Reports component package, you can verify the workflow streams on the Administration Console and view the reports on SAP BusinessObjects BI Launch Pad.

See "Verifying Workflow Streams on Administration Console" on page 44

See "Viewing Reports on SAP BusinessObjects BI Launch Pad" below.

Viewing Reports on SAP BusinessObjects BI Launch Pad

Now that you installed the Domain and Reports component packages and the data is loaded into the data warehouse, you can view the reports on the SAP BusinessObjects BI Launch Pad interface.

If you installed the sample `RetailPOS_Demo_Content_Pack` from *%CDE_HOME%*`\samples\`, you will see the `Retail Sales Report` in the `Document List` in IBI Launch Pad. For instructions on how to log on to the BI Launch Pad and view reports, see HPE Operations Bridge Reporter *Online Help for Users*.

The Retail Sales Report contains the Sales Summary document which displays the sales revenue information for each of the dimensions that you defined in the data model – location, time, and product category. You can drill down and roll up on the dimensions to view granular information.

Part II: Content Development - Simplified

The simplified method of creating a content pack allows you to create Domain, ETL, and Report components at the same time.

Use the simplified content development method in the following scenarios:

1. When a fact table is associated with one or more dimensions (star schema) and the dimensions are not further normalized (dimensions without a parent table).
2. When all the facts are to be reconciled against host (when HP Operations agent is the data source).
3. When the generated reports do not require any roll up or drill down of data.

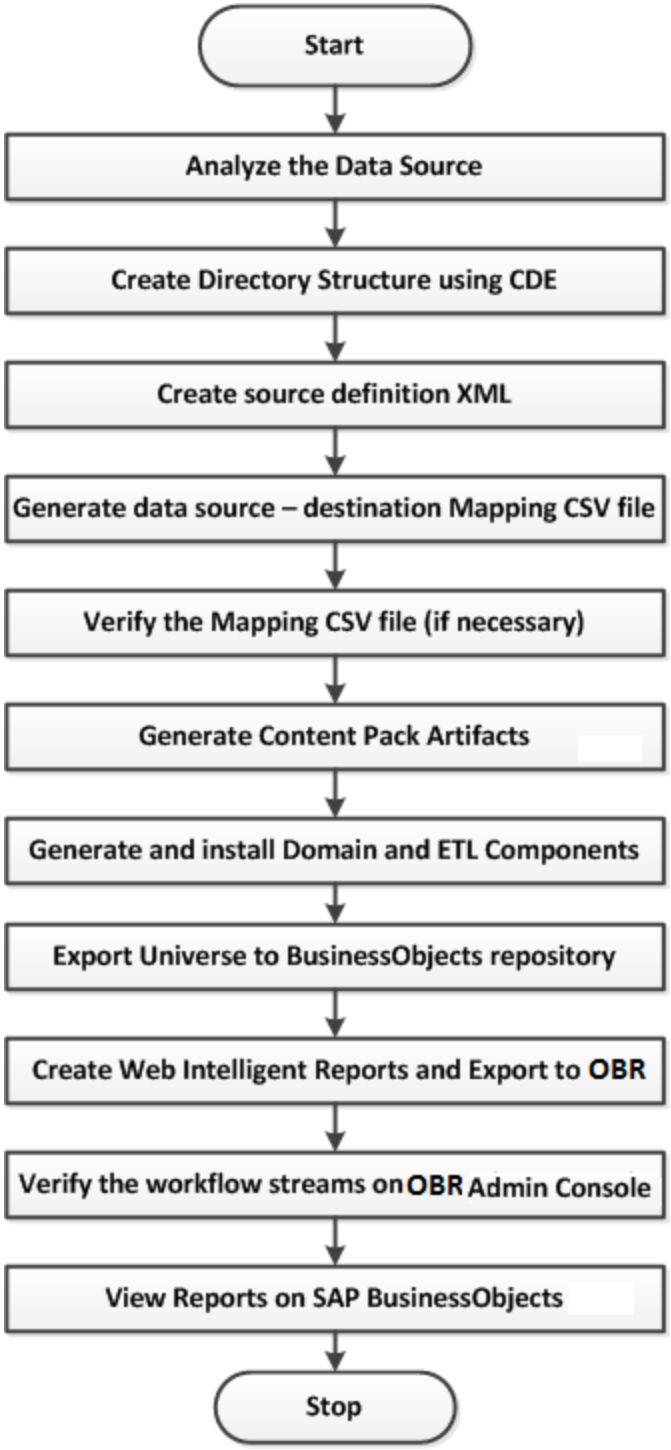
This section describes a simplified method of creating the Domain, ETL, and Report components of a content pack with the following data sources:

- **Generic database as source**
- **CSV file as source**
- **HP Operations Agent or HP Performance Agent as source**

> **Note:** CDE-Simplified is supported only on Windows.

CDE-Simplified Flow

The following chart shows the flow of CDE-simplified:


Start
Analyze the Data Source
Create Directory Structure using CDE
Create source definition XML
Generate data source – destination Mapping CSV file
Verify the Mapping CSV file (if necessary)
Generate Content Pack Artifacts
Generate and install Domain and ETL Components
Export Universe to BusinessObjects repository
Create Web Intelligent Reports and Export to OBR
Verify the workflow streams on OBR Admin Console
View Reports on SAP BusinessObjects
Stop

Chapter 6: Use CDE-Simplified to Create Content Pack

You must identify the data source containing metrics that are suitable to be fed into the Domain data model. To design the data model, see section ""Design the Data Model " on page 38".

Generic Database as Data Source

OBR collects data only from databases that support Java Database Connectivity (JDBC). When the HPE OBR data source is a generic database (Microsoft SQL, Oracle, Sybase IQ and so on), perform the following steps to generate the Domain, ETL, and Report components of the content pack:

1. Using the command prompt, change to the *%CDE_HOME%\bin* directory.
2. Run the following command:

   ```
   createCPFolders.bat -package RetailPOS -type all
   ```

   The folders for the content pack components (Domain, ETL, and Report) are created.
3. Create the following input files based on the attached samples for RetailPOS and save them in the *%CDE_HOME%\samples\RetailPOS_Demo_Content_Pack\Source* folder:

   > **Note:** To read the contents of the sample attachments in this PDF document, click the **Attachments: View File Attachments** icon and double click to open the XML file in your browser window.

   a. Database Connection Properties File (*dbconfig.properties*)

      This file contains log in credential details of the data source (generic database) from which the content pack fetches data. It also contains the OBR (Vertica) database credentials required for generating the SAP BusinessObjects Universe.

   b. Database Query XML File (*RetailPOS_Query.xml*)

      This file contains the SQL queries to fetch data from the data source (generic database). It also contains the table and column definitions in OBR to store the data.

OBR bundles JDBC drivers for Microsoft SQL, Oracle, PostgreSQL, and Sybase IQ databases in the `CDE.zip` file. If you have a different database, you must copy the relevant JDBC driver to the OBR system and the content development environment.

4. With the Database Connection Properties File and the Database Query XML File as input, run the following command to generate the mapping (ModelMapper.CSV) file:

   ```
   createMappingFile.bat -useDB -passwords "mssql.password=<password>" -dbQueryXml
    %CDE_HOME%\samples\RetailPOS_Demo_Content_Pack\Source\RetailPOS_Query.xml -
   config %CDE_HOME%\samples\RetailPOS_Demo_Content_
   Pack\Source\dbconfig.properties
   ```

   Verify the values in the mapping CSV file and, if required, modify it manually based on your business requirement.

5. Run the following command to generate the content pack:

   ```
   createendtoendcp.bat -mappingfile "%CDE_HOME%\workspace\RetailPOS\RetailPOS_
   MappingFile2394283040045872.csv" -collectionpolicyxml "%CDE_
   HOME%\workspace\RetailPOS\RetailPOS_CollectionPolicy.xml" -configfile "%CDE_
   HOME%\samples\RetailPOS_Demo_Content_Pack\Source\dbconfig.properties" -password
   "db.password=<password>"
   ```

   The Domain, ETL, and Report components of the content pack are now created.

CSV Files as Data Source

When the HPE OBR data source is a set of CSV files, perform the following steps to generate the Domain, ETL, and Report components of the content pack:

1. Using the command prompt, navigate to the *%CDE_HOME%\bin* directory.
2. Run the `setenv.bat` file to set the environment, `createUniverseConnection.bat` to create universe connection, and `updateCDEProperty.bat` to update CDE property.

   Enter inputs for the following when prompted:

   - **User Name**: Your BusinessObjects username
   - **Password**: Your BusinessObjects password
   - **Server Name**: Name of the server where SAP BusinessObjects is installed

3. Run the following command:
   ```
   createcpfolders.bat -package RetailPOS -type all
   ```

The folders for the content pack components (Domain, ETL, and Report) are created in the CDE workspace.

The sample source CSV files and configuration files are available at: `%CDE_HOME%/samples\CSV_as_source_RetailPOS_Demo_Content_Pack`:

- **Sample source CSV files**: Sample source files for RetailPOS is located at: `%CDE_HOME%\samples\CSV_as_source_RetailPOS_Demo_Content_Pack\sourcecsvs`
- **dbconfig.properties**: Database configuration (*dbconfig.properties*) file is located at `%CDE_HOME%\\samples\CSV_as_source_RetailPOS_Demo_Content_Pack\dbconfigfile`. This property file contains the OBR Vertica database details used during universe creation of Application content pack. For more information on the *dbconfig.properties* file, see"Properties Files" on page 171 .
- **RetailPOS.xml**: A sample policy XML file to be written by content developer is available at `%CDE_HOME%\samples\CSV_as_source_RetailPOS_Demo_Content_Pack\csvxml`. This file provides the metadata about input CSV files to categorize them into dimension CSVs, fact CSVs, business key columns, and primary dimension.

> To read the contents of the sample attachments in this PDF document, click the **Attachments**: **View File Attachments** icon and double-click an XML file to open in your browser.

4. With the Source CSV Files and the CSV Policy XML File as input, run the following command to generate the mapping (Model Mapper CSV) file:

   ```
   createMappingFile.bat -useCSV -csvXml %CDE_HOME%\samples\CSV_as_source_RetailPOS_Demo_Content_Pack\csvxml\RetailPOSCSV.xml -config %CDE_HOME%\samples\CSV_as_source_RetailPOS_Demo_Content_Pack\dbconfigfile\dbconfig.properties -inputLocation %CDE_HOME%\inputcsvs -outputLocation %CDE_HOME%\workspace\RetailPOS
   ```

   ModelMapper CSV file is generated and available in the output location. A Model Mapper CSV file is an intermediate document before generating the end-to-end content pack for Domain and ETL. Name of the file will be <*CSV policy's name attribute*>_MappingFile_<*timestamp*>.CSV. The content developer can edit the required column names in the CSV file.

5. Run the following command to generate the content pack:

   ```
   createendtoendcp.bat -mappingfile "%CDE_HOME%\workspace\RetailPOS\RetailPOS_MappingFile2394283040045872.csv" -configfile "%CDE_HOME%\samples\RetailPOS_Demo_Content_Pack\Source\dbconfig.properties" -password "<database>.password=<password>"
   ```

The Domain, ETL, and Report components of the content pack are now created.

HP Performance Agent as Data Source

When the HPE OBR data source is HP Performance Agent, perform the following steps to generate the Domain, ETL, and Report components of the content pack:

1. Using the command prompt, change to the *%CDE_HOME%/bin* directory.
2. Run the following command:

   ```
   createcpfolders.bat -package RetailPOS -type all
   ```

   The folders for the content pack components (Domain, ETL, and Report) are created.
3. Create the following input files based on the attached samples for RetailPOS and save them in the *%CDE_HOME%\samples\RetailPOS_Demo_Content_Pack\Source* folder:

   > **Note:** To read the contents of the sample attachments in this PDF document, click the **Attachments: View File Attachments** icon and double click to open the XML file in your browser window.

   a. Connection Properties File (*dbconfig.properties*)

      This file contains information on the RTSM host name, port, user name, and the HP Performance Agent host name from which the content needs to be developed. It also contains the HPE OBR database credentials that you will require for generating the SAP BusinessObjects Universe.

   b. Topology Policy XML File (*TopologyRTSM_OM.xml*)

      This file contains the view names, CI types and the attributes to be collected or excluded from CI types. It also contains the HPOM class names and data source details. In case of HPOM collection policy, the attribute names are generated for OOB content. For new CI types, the attributes must match with the attributes specified in the CMDB collection policy.

   c. HP Performance Agent Policy XML File (*Fact_PA.xml*)

      This file contains the HP Performance Agent data source and classes from which data needs to be collected.
4. With the earlier files as input, run the following command to generate the mapping (ModelMapper.csv) file:

```
createMappingFile.bat -usePA -paXml %CDE_HOME%\samples\RetailPOS_Demo_Content_
Pack\Source\Fact_PA.xml -topologyXml %CDE_HOME%\samples\RetailPOS_Demo_Content_
Pack\Source\TopologyRTSM_OM.xml-config %CDE_HOME%\samples\RetailPOS_Demo_
Content_Pack\Source\dbconfig.properties-outputLocation %CDE_
HOME%\workspace\RetailPOS -passwords <RTSM password>
```

Verify the values in the mapping CSV file and, if required, modify it manually based on your business requirement.

5. Run the following command to generate the content pack:

```
createendtoendcp.bat -mappingfile "%CDE_HOME%\workspace\RetailPOS\RetailPOS_
MappingFile2394283040045872.csv" -topologyxml %CDE_HOME%\samples\RetailPOS_
Demo_Content_Pack\Source\TopologyRTSM_OM.xml -paXml %CDE_
HOME%\samples\RetailPOS_Demo_Content_Pack\Source\Fact_PA.xml -configfile %CDE_
HOME%\samples\RetailPOS_Demo_Content_Pack\Source\dbconfig.properties
```

The Domain, ETL, and Report components of the content pack are now created.

Chapter 7: Generating, Installing, and Viewing Reports

Generating Domain and ETL Component Package

1. Browse to %CDE_HOME%\workspace\RetailPOS\RetailPOSDomain.ap\dist (for Domain) and %CDE_HOME%\workspace\RetailPOS\RetailPOSETL.ap\dist (for ETL)
2. Copy RetailPOS Domain and ETL to %PMDB_HOME%\packages.

   Copying the package makes it available in the Deployment Manager for installation.

> **Note:** It is not necessary to have HPE OBR installed on the machine where you are creating the Domain component package. If you created the Domain component package on another machine, you must copy the package to the HPE OBR machine under *%PMDB_HOME%\packages*.

Install the Domain and ETL Component Package

OBR provides the Deployment Manager utility on the Administration Console to install the content pack component packages. For instructions on how to install the content pack components, see the HPE Operations Bridge Reporter *Online Help for Administrators*.

Viewing OBR Reports

To view the reports based on the data you collected using the content pack you created, perform the steps in the following sections according to the order mentioned:

1. "Export the Universe to SAP BusinessObjects Repository" on page 57
2. "Create Web Intelligence Reports" on page 62
3. "Exporting Reports to OBR" on page 65
4. "Viewing Reports on SAP BusinessObjects BI Launch Pad" on page 71

Part III: Content Development Reference

- **XML Definitions**
- **CDE-Simplified Reference**

Overview

This section provides a detailed explanation of the syntax of source XML, CSV, and Properties files that a content developer has to author or modify in order to build a content pack using CDE.

For information on stage interface and model documentation for all out of the box domain components, see the stage interface and model document available as part of each domain component.

For out-of-the-box (OOTB) content packs of HPE OBR, the stage interface and model documentation is available in each content package. For example, in the System Management domain, the stage interface and model documents are located at:

`%PMDB_HOME%\packages\SystemManagement\CoreSystemManagement.ap\doc` (Windows)

`$PMDB_HOME/packages/SystemManagement/CoreSystemManagement.ap/doc` (Linux)

For newly developed custom content, the stage interface and model documentation is automatically created by the CDE. For example, in the System Management domain, the stage interface and model documents are located at:

`%CDE_HOME%\cplib\SystemManagement\CoreSystemManagement.ap\doc` (Windows)

`$CDE_HOME/cplib/SystemManagement/CoreSystemManagement.ap/doc` (Linux)

For information on HPE OBR data warehouse bus matrix, see the HPE OBR data warehouse bus matrix (HTML) document available at

`%CDE_HOME%\doc\bus_matrix` (Windows) or `$CDE_HOME\doc\bus_matrix` (Linux).

> **Note:** HP does not provide support for Content Packs created or modified with the Content Development Environment (CDE) tool, or otherwise, via HP's standard product support/service agreements. Any defects found in the underlying CDE tool, however, will be addressed by HP.

Chapter 8: XML File Definitions

This chapter provides information about the XML files used for content development and describes each element in the XML files.

Model XML definition

A model XML file defines the data warehouse model for a specific domain.

Model XML Structure

<schema>

< relational >

< dimensionTable >

< column />

</ dimensionTable >

< bridgeTable >

< column />

</ bridgeTable >

< factTable >

< column />

</ factTable >

</ relational >

< logical >

< dimension >

< memberGroup >

< me m b er />

< calculatedMember />

```
  </ memberGroup >
  < hierarchy >
   < level >
    < group />
     </ level >
  </ hierarchy >
</ dimension >
 < cube >
  < measures >
   < measure />
   < calculatedMeasure />
  </ measures >
  < dimensionRef />
 </ cube >
 < cubeRef />
 < cubeRegion >
  < derivedMeasures >
   < measureRef />
  </ derivedMeasures >
  < aggregRef />
 </ cubeRegion >
</ logical >
< aggregates >
< aggregate >
< aggMeasure />
  < calculated Measure />
 < aggLevel />
</ aggregate >
```

```
 < forecast >
  <aggMeasure />
  < calculatedMeasure />
 < aggLevel />
 </ forecast >
</ aggregates >
</ schema >
```

XML Element: **schema**

Root element for the model definition. Schema represents the database model of a domain. It includes relational data model (database tables), logical representation of relational data model and data aggregation definition. A schema can also refer to relational and logical elements of another schema.

Attribute	Mandatory	Description
name	Yes	Name of the schema
version	Yes	Version of the schema
caption	Yes	Label for the schema.

Child Elements	Multiplicity	Description
relational	0..1	Relational schema section
logical	0..1	Logical schema section
aggregation	0..1	Aggregation section

XML Element: **relational**

Relational schema definition

Child Elements	Multiplicity	Description
factTable	0..n	List of fact tables in the Relational Model
dimensionTable	0..n	List of dimension tables in the Relational Model
bridgeTable	0..n	List of bridge tables in the Relational Model

XML Element: factTable

A fact table element represents a physical data warehouse table containing numerical performance measures

Attribute	Mandatory	Default	Description
name	Yes		Unique name of the fact table
caption	Yes		Label for the fact table.
version	Yes		Table version in the model
description	No		Brief description about the fact table
type	Yes		Granularity of the fact table. The allowed values are • transactionGrain : Transaction grain fact table contains transaction data (one row per transaction event) • accumulatedSnapShot : Accumulated snapshot fact table contains life cycle data which gets updated whenever there is a state change(one row per life cycle) • periodicSnapShot : Periodic snapshot fact table contains performance data at regular intervals (one row per period)
subType	No		Subtype of fact table granularity. Example rate:5-minute event:as-polled **Note:** If subtype is set as event:as-polled then duplicate rows in fact table will not be deleted automatically
isExternal	No	false	External table (or View) handling: when true, the schema of the table is specified (attribute elements) but the table is not created and managed by the framework.

Child Elements	Multiplicity	Description
column	0..n	List of table columns

Note: Fact table definition should define the following two columns mandatorily

dsi_key_id_: Foreign key column to primary dimension table

ta_period: Time period column

XML Element: **dimensionTable**

A dimension table element represents a physical data warehouse table containing dimension data/attributes.

> **Note:** Primary Key (surrogate key) for the dimension table is implicitly created by the framework.

Attribute	Mandatory	Default	Description
name	Yes		Unique Name of the dimension table
caption	Yes		Label for the dimension table.
version	Yes		Table version in the model
description	No		Brief description about dimension table
type	Yes		Type of dimension table
conformedTo	No		Master dimension table that this dimension table conforms to. The dimension table always contains a subset of the rows from the master dimension table. The business key columns defined in this table should be same as the business key columns defined in the master dimension table. The master dimension table might be defined in the same schema or referenced from other dependent schema. If it is referenced from dependent schema then master dimension table should be prefixed with the dependent schema name using ":" as delimiter
isExternal	No	false	External table (or View) handling: when true, only the schema of the table is specified (attribute elements) but the table is not created and managed by the framework.

Child Elements	Multiplicity	Description
column	0..n	List of table columns

XML Element: **bridgeTable**

A bridge table element represents a physical data warehouse table that acts as a variable depth bridge table containing parent child relationship between Configuration Items (CI).

Attribute	Mandatory	Default	Description
name	Yes		Unique Name of the bridge table
caption	Yes		Label for the bridge table.

Attribute	Mandatory	Default	Description
version	Yes		Table version in the model
description	No		Brief description about the table
type	Yes		Type of the table(bridge)
isExternal	No	false	External table (or View) handling: when true, the schema of the table is specified (attribute elements) but the table is not created and managed by the framework.
subType	No	variable	Type of bridge table. Allowed values are variable - variable depth fixed - fixed depth (not supported currently)

Child Elements	Multiplicity	Description
column	0..n	List of table columns

Note:

- A variable depth bridge table must define two business key columns both referencing the same dimension table representing a parent child relationship. A fixed depth bridge table can have only one such relationship.
- A variable depth bridge table can have any number of business key columns apart from the two columns that are involved in a parent child relationship

XML Element: **column**

A column element defines physical columns of a table

Attribute	Mandatory	Default	Description
name	Yes		Name of the column (unique for the table)
caption	Yes		Label for the column.
dataType	Yes if not a reference column	Integer if reference column	Column data type (db independent). Allowed values are String Time Date Integer Float

Attribute	Mandatory	Default	Description
			Double varbinary
subType	No		Techno specific details on type
size	No	255 if String	Data type size
reference	No		Foreign key column. If the referred table is not defined locally to the schema, it should be prefixed with the schema name using ":" as delimiter
description	No		Column description
default	No		Default value for the column. Note: String values must be specified with single quotes Example: default="'default-value'"
notNull	No	false	Allow null value or not
businessKey	No	false	Is Business key (Unique)

Logical model

Logical model comprises logical cubes, measures, dimensions, dimension attributes, dimension hierarchies and hierarchy levels. The logical model definition is used to create BO universe for reporting.

XML Element: **logical**

Logical schema definition

Child Elements	Multiplicity	Description
dimension	0..n	List of global dimensions in the logical model
cube	0..n	List of cubes in the logical model
cubeRef	0..n	Defines reference to a cube defined in dependent schema
cubeRegion	0..n	Defines a cube derived from a source cube defined in dependent schema as a subset of the former.

XML Element: **dimension:**

A dimension element defines set of unique attributes that qualifies the fact data and dimension hierarchies.

Attribute	Mandatory	Description
name	Yes	Unique name of the Dimension
caption	Yes	Label for the dimension
description	No	Dimension description

Child Elements	Multiplicity	Description
hierarchy	1..n	Dimension hierarchies
memberGroup	0..n	Group of members

XML Element: **hierarchy**

A hierarchy element defines a dimension hierarchy. A dimension hierarchy defines how the data is organized at different levels of aggregation

Attribute	Mandatory	Default	Description
name	Yes		Unique name of the hierarchy
caption	Yes		Label for the hierarchy.
description	No		Hierarchy description
createLevels	No	true	When CreateLevels is set to false, the hierarchy won't be created in BO universe. This is extensively used for one level hierarchy.

Child Elements	Multiplicity	Description
level	1..n	Hierarchy level

XML Element: **level**

A level element defines a hierarchy level. A hierarchy level represents a position in hierarchy. Each level is more granular than its parent level. For example the Time dimension hierarchy can have levels like day, month, quarter and year.

Attribute	Mandatory	Description
name	Yes	Unique name of the hierarchy level
caption	Yes	Label for the hierarchy level

Attribute	Mandatory	Description
table	Yes	Dimension Table used in the level
levelCaption	Yes	Identifies level caption member. Level Caption is used to display the drill level in reports. It should be human readable and should uniquely identify members for the specific level. Note: In BO Universe, levelCaption is used for both online aggregation and level display in hierarchies.

Child Elements	Multiplicity	Description
member	0..n	Hierarchy level attribute (from the dimension table defined in the level)
calculatedMember	0..n	Calculated from other dimension table columns
group	0..n	Group of members

XML Element: **member**

A member element defines a level member. A level member maps logical member name to the physical column defined in the dimension table.

Attribute	Mandatory	Description
name	Yes	Member name.
column	Yes	Attribute name (as specified in the dimension table)
caption	Yes	Label for the member
description	No	Description of the member

XML Element: **calculatedMember**

A calculatedMember element defines a calculated member. A calculated member maps a logical member name to an SQL expression involving columns from the dimension table

Attribute	Mandatory	Description
name	Yes	Member name.
caption	Yes	Label for the member.
formula	Yes	Standard SQL expression involving columns from the dimension table Example: ${member1} + ${ member2} (${member1} + ${member2})/${member3}

Attribute	Mandatory	Description
description	No	Description of the calculated member
dataType	No	Data type of the calculated Member – defaults to String

XML Element: **group**

A group element references a member group identified by member group name.

Attribute	Mandatory	Description
name	Yes	Member group name.

XML Element: **memberGroup**

A memberGroup element groups members with a name. The members defined in the group will be inserted 'as is' into the level element.

Attribute	Mandatory	Description
name	Yes	Member group name.

Child Elements	Multiplicity	Description
member	0..n	Hierarchy level attribute (from the dimension table defined in the level)
calculatedMember	0..n	Calculated from other dimension table columns

XML Element: **cube**

A cube element defines a data cube. A cube is an abstract representation of multidimensional dataset. A cube holds measures of the business like sales qualified by the dimensions like product, store and city. A cube represents a star schema or snowflake schema in relational model. A star schema has a fact table at the center and one or more dimension tables that are referred by the fact table. A snowflake schema is an extension of a star schema such that the dimensions are normalized into multiple related dimension tables.

Attribute	Mandatory	Description
name	Yes	Unique Name of the cube
caption	Yes	Label for the cube.
description	No	Description for cube

Child Elements	Multiplicity	Description
measures	1	Measures for the cube (from fact table)
dimension	0..n	Locally defined dimensions used for the cube.
dimensionRef	0..n	Reference to a global dimension used for the cube.

XML Element: **dimensionRef**

dimensionRef element reference a global dimension element

Attribute	Mandatory	Description
name	Yes	Pseudo name of dimension (scope=cube)
reference	Yes	Name of dimension referenced. Note: If the dimension is referred from dependent schema the dimension name should be prefixed with the schema name using ":" as delimiter

XML Element: **measures**:

measures element defines list of measure elements

Attribute	Mandatory	Description
factTable	Yes	Fact table referenced from relational section
caption	No	caption for the measures

Child Elements	Multiplicity	Description
measure	1..n	Measures for the cube (from fact table)
calculatedMeasure	0..n	Calculated Measures for the cube (from measures)

XML Element: **measure**

A measure element maps a logical measure name to the fact table column name

Attribute	Mandatory	Description
name	Yes	Member name.
column	Yes	Column name from the fact table referenced
caption	Yes	Label for the member.
aggregation	Yes	Default aggregation rule used to drill-down/rollup in hierarchy. The allowed values are sum, min, max, average, count and none.
description	No	Description of the measure

XML Element: **calculatedMeasure**

A calculatedMeasure element maps a logical measure name to a SQL expression involving measures

Attribute	Mandatory	Description
name	Yes	Name of the measure
caption	Yes	Label for the calculated measure.
formula	Yes	Standard SQL expression involving measures. Example: ${meaure1} + ${measure2} (${meaure1} + ${measure2})/${meaure3}
description	No	Description of the calculated measure
dataType	No	Data type of the calculated measure

XML Element: **cubeRef**

A cubeRef element references a cube and corresponding aggregates defined in the dependent schema

Attribute	Mandatory	Description
name	Yes	Unique name of the cube
reference	Yes	cube referenced (scoped) in the parent/dependent schema

XML Element: **cubeRegion**

A cubeRegion element defines a cube derived from a source cube defined in a dependent schema as a subset of the former

Attribute	Mandatory	Description
name	Yes	Unique name of the cube
reference	Yes	cube referenced (scoped) in the parent schema

Child Elements	Multiplicity	Description
derivedMeasures	1..1	Measures for the cube derived from source cube
aggregRef	0..n	Reference to an aggregate used for the source cube to be inherited.

XML Element: **derivedMeasures**:

derivedMeasures element defines a set of measures derived from the source cube defined in dependent schema as a subset of the former.

Attribute	Mandatory	Description
caption	No	Label for the derived measures

Child Elements	Multiplicity	Description
measureRef	1..n	Measures for the cube derived from source cube

XML Element: **aggregRef**

An aggregRef element defines a set of aggregates derived from the source cube defined in dependent schema as a subset of the former.

Attribute	Mandatory	Description
name	Yes	Name of the aggregate
reference	Yes	aggregate referenced (scoped) in the parent schema

XML Element: **measureRef**

A measureRef element defines a set of measures/calculatedMeasures derived from the source cube defined in dependent schema as a subset of the former.

Attribute	Mandatory	Description
name	Yes	Name of the measure/calculatedMeasure derived from source cube
caption	No	Label for the measure

XML Element: **aggregates**

aggregates element contains a list of aggregate and forecast elements.

Child Elements	Multiplicity	Description
aggregate	0..n	List of aggregated tables
forecast	0..n	List of forecast aggregation tables

XML Element: **aggregate**

An aggregate element defines pre-aggregation rules for a fact table

Attribute	Mandatory	Default	Description
name	Yes		Name of the aggregated table.
caption	Yes		Label for the aggregated table.
cube	Yes		name of the (local) cube on which the aggregation is performed. (specified as cube element)
source	Yes		Source table for the aggregation
isExternal	No	false	When set to “true”, aggregate table won’t be generated
type	No	trend_ sum	Type of aggregation. Allowed values are • trend_sum - does pre-aggregation (aggregate table will be generated) • OLAP – does online aggregation (aggregate table won’t be generated)

Child Elements	Multiplicity	Description
aggMeasure	1..n	Aggregated measure
calculatedMeasure	1..n	Calculated Measures for aggregated table
aggLevel	0..n	Level on which the aggregate is performed Order is meaningful (Group by order)

XML Element: **aggMeasure**

An aggMeasure element specifies the aggregate measure column in the aggregate table and the type of aggregation performed on the source column

Attribute	Mandatory	Description
name	Yes	Name of the aggregated measure. The aggregated column will be generated by prefixing the aggregation function with the value of name attribute. For example if the name is “cpu_util” and aggregation function is “avg”. Note: If the value of name attribute is prefixed with “!” character then the generated aggregated column will be same as the value of name attribute.
source	Yes	Name of the column from the source table that is being aggregated.
caption	Yes	Aggregated measure caption
aggregation	Yes	Aggregation function. Allowed values are • If aggMeasure is part of aggregate element, supported functions are:

Attribute	Mandatory	Description
		avg, min, max, cnt, tot, med, std, slope, wav, perXX, lst, nlst. • If aggMeasure is part of forecast element, supported functions are: avg, min, max, cnt, tot, med, std, slope, wav, perXX, fXX, DTT [XX]. • If aggregate type is OLAP, supported functions are: min, max, avg, sum. For a list of descriptions of these functions, see Table ""Supported Functions in Aggregate and Forecast Elements" on page 177" in the Appendix.
description	No	Description of the aggregated measure

XML Element: **calculatedMeasure**

A calculatedMeasure element specifies the aggregate measure column in the aggregate table and the type of aggregation performed on the result of a SQL expression.

Attribute	Mandatory	Description
name	Yes	Name of the measure
caption	Yes	Label for the calculated measure.
aggregation	Yes	Aggregation function. Allowed values are • If aggregate type is trend_sum tot,min,max,avg,lst,per95,Per90,wav(),dtt[100],f30,f60,f90 • If aggregate type is OLAP min,max,avg,sum
formula	Yes	Standard SQL expression involving columns in fact table/dimension table
description	No	Description of the calculated measure
dataType	No	Data type of the aggregated measure – defaults to Double

XML Element: **aggLevel**

An aggLevel element represents the dimension level on which aggregation is performed

Attribute	Mandatory	Description
dimension	Yes	Name of the source dimension for the specified cube (specified in dimension element).

Attribute	Mandatory	Description
hierarchy	Yes	Name of the source hierarchy for the specified dimension (specified in hierarchy element).
level	Yes	Name of the aggregation level for the specified hierarchy (specified in level element).

XML Element: **forecast**

A forecast element defines forecast for a fact. Forecast is an extension of aggregate (for forecast aggregation) with an additional attribute baselinedays

Attribute	Mandatory	Description
baselinedays	Yes	No of days used for baseline. (e.g. 42-day baseline period for forecast)

Model XML to BO Universe component mapping

The model xml serves as input for generating BO universe. CDE framework in OBR processes the model xml and generates the BO universe. BO universe is a file that servers as a semantic layer between the underlying database and dependent reports. A BO universe file contains the following components

- Classes – A class is group of objects in universe. It provides structure to the layout of universe. Typically, a class contains a group measure objects or a group of related dimension and detailed objects. A class can in turn have sub-classes which enables grouping of objects into more granular subset.
- Objects – Objects refer to columns defined database tables or views
- Tables – Tables refer to physical data warehouse tables
- Joins – SQL expression on how tables relate
- Contexts – Groups of related tables with their related joins
- Hierarchy – Ordered sequence of dimension objects which allows users to analyze the data at one level and drill down to lower level for more granular data

The following table shows model XML elements to BO universe component mapping.

Model XML Element	BO Universe component
cube	Class containing measure objects

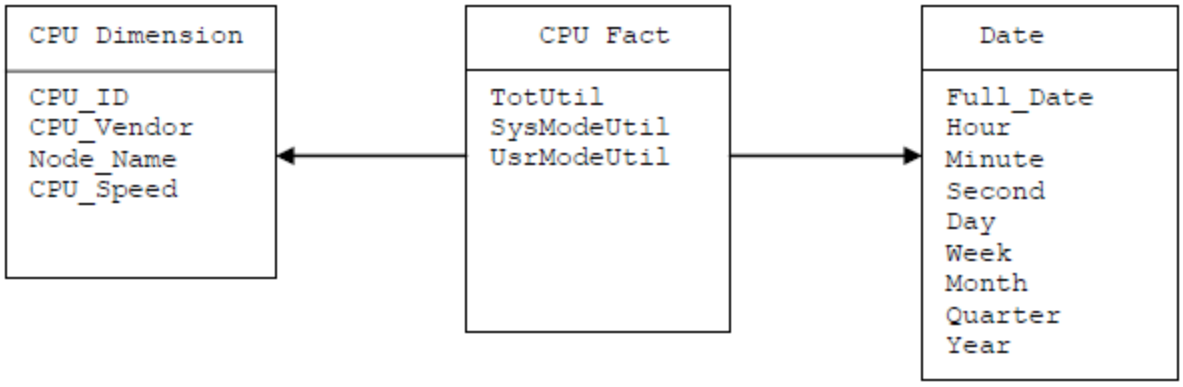
Model XML Element	BO Universe component
dimension	Class containing dimension objects
hierarchy	Hierarchy
levelCaption attribute in level element	Dimension Object
member elements defined under level element	Detailed objects under a Dimension object
memberGroup element referenced under level element	Detailed objects under a Dimension object
factTable, dimensionTable, bridgeTable, aggregate	Tables

BO universe joins and contexts are generated from foreign key references defined in relational section and aggregate level defined in aggregates section.

Model definition Examples

Sample model XML file

Let's consider a simple dimensional model for CPU performance management



This dimension model contains a CPU fact and three dimensions that qualify CPU fact.

The equivalent model definition XML will typically have the following

Relational Section

- Fact table definition for CPU fact
- Dimension table definition for CPU dimension
- Dimension table definition for Date dimension

Logical Section

- Dimension and hierarchy definition for CPU dimension
- Dimension and hierarchy definition for Date dimension
- Cube definition and dimension reference definition for CPU fact

Aggregate Section

- Hourly aggregate d efinition for CPU fact
- Daily aggregate definition for CPU fact
- Forecast aggregate definition for CPU fact

Please refer to the model definition file **CPU_Performance_Management.xml** in attachment section for the above dimensional model

Referring elements in dependent model XML file

A model xml can refer to elements defined in other dependent model xml by prefixing the schema name of the dependent model xml with element name. The elements that can be referred are dimensionTable, dimension and cube

Model1.xml

```
<schema name="schema1">
 <relational>
  <dimensionTable name="dimension_table1".....>
    .....
  </dimensionTable>
 <dimensionTable name="dimension_table2".....>
     .....
  </dimensionTable>
  <factTable name="fact_table1".....>
```

```
    .....
  </factTable>
 </relational>
<logical>
 <dimension caption="Dimension1" name="Dimension1">
   .....
   <hierarchy name="Dimension1Hierarchy".....>
     <level name="Dim1Level1".....>
       ......
    </level>
   </hierarchy>
  </dimension>
 <cube name="Fact1" caption="Fact1" description="Fact1 description">
   .....
  </cube>
 </logical>
</schema>
```

Model2.xml

```
<schema name="schema2">
 <relational>
  <dimensionTable name="dimension_table3".....>
    <! -- Referring a dimension table defined in dependent schema-->
    <column name="dim_attribute1" reference="schema1:dimension_table2"
                                                              ..... />
  </dimensionTable>
  <factTable name="fact_table2".....>
    .....
  </factTable>
```

```
 </relational>
<logical>
  <dimension caption="Dimension2" name="Dimension2">
   .....
   <hierarchy name="Dimension2Hierarchy”.....>
     .....
   </hierarchy>
  </dimension>
  <cube name="Fact2" caption="Fact2" description="Fact2 description">
   .....
   <! -- Referring a dimension defined in dependent schema-->
   <dimensionRef name="Dimension1" reference="schema1:Dimension1" />
</cube>
<! -- Referring a cube defined in dependent schema-->

<cubeRef name="Fact1" reference="schema1:Fact1"/>
</logical>
</schema>
```

Defining Hierarchies and levels

Let’s consider Date dimension. A Date dimension hierarchy that shows data at year, month, day and hour levels can be defined as follows

```
<dimension caption="DATETIME" name="DATETIME">
<hierarchy caption="DATETIME Hierarchy" name="DATETIMEH"
                          description="DATETIME hierarchy ">
 <level caption="Year" name="Year" table="DATETIME"
                                        levelCaption="Year">
```

```
<member name="Year" column="TIME_YEAR_NUMBER"
                        caption="Year" description="Year" />


         </level>
   <level caption="Month" name="Month" table="DATETIME"
                                        levelCaption="Month">
      <member name="Month" column="TIME_MONTH_NAME"
                     caption="Month" description="Month Name"/>
   </level>
         <level caption="Day" name="Day" table="DATETIME"
                                             levelCaption="Day">
      <member name="Day" column="TIME_DAY_MONTH_NUMBER"
                               caption="Day" description="Day" />
<member name="Day Name" column="TIME_DAY_NAME"
                     caption="Day Name" description="Day Name" />
   </level>
         <level caption="Hour" name="Hour" table="DATETIME"
                                             levelCaption="Hour">
<member name="Hour" column="TIME_HOUR_ID" caption="Hour"
   description="Hour" />
   </level>
</hierarchy>
</dimension>
```

ETL Collection Policies

RTSM Collection Policy XML Definition

RTSM collection policy xml defines CI types and its attributes of a view to be collected from RTSM. HPE OBR uses RTSM collection policy to extract topology and dimension data for a domain from a BSM machine.

XML Structure

< etldefinition >

< views >

< view >

< citype >

< aliassource >

  < aliastarget />

</ aliassource >

< ciattribute />

        < cire f >

  < ciattribute />

</ ciref >

</ citype >

</ view >

</ views >

</ etldefinition >

XML Element: **etldefinition**

etldefinition is the root element for CMDB collection policy definition. This element specifies type of the collection and domain for which the collection is defined

Attribute	Mandatory	Description
type	Yes	Specifies the ETL step. The value should be “collect” for a CMDB collection policy
collector	Yes	Type of the collector. The value should be “CMDB” for a CMDB collection policy
domain_name	Yes	Domain for which the collection is performed
contentpack_ name	No	Name of the content pack for the domain
description	No	Brief description about the collection policy

Child Elements	Multiplicity	Description
views	1..1	CMDB Views definition

XML Element: **views**

CMDB Views definition

Attribute	Mandatory	Description
description	No	Description about the view grouping

Child Elements	Multiplicity	Description
view	1..n	List of CMD views to be collected from

XML Element: **view**

A view element defines a CMDB view

Attribute	Mandatory	Description
name	Yes	Name of the CMDB view
dumplocation	No	dumplocation
description	No	Brief description about the view

Child Elements	Multiplicity	Description
citype	1..n	List of ci types to be collected

XML Element: **citype**

A citype element defines a CI type. A CI type is a named group of Configuration Items (CI- Managed component like node, software).

Attribute	Mandatory	Description
name	Yes	CI type name as defined in RTSM (BSM Modelling Studio > CI Type Manager).
alias	No	Alias for CI type
filter	No	Condition to filter rows. Please refer to Appendix A for complete list of supported filters and their syntax
type	No	Specifies whether the CI should be considered os dimension/datasource/both. The allowed values are • dimension – CIs from this citype are considered only as dimension and not data source (only dimension CSV file is created and no entry is made in to data source dictionary table in OBR) • datasource – CIs from this citype are considered only as datasource and not dimension (only entry is made in to data source dictionary table in HPE OBR and no dimension CSV file created) • both – CIs from this citype are considered both as dimension and data source dimension (entry is made into data source dictionary table in HPE OBR and dimension CSV file is created). The default value is “both”.
description	No	Brief description about CI type

Child Elements	Multiplicity	Description
aliassource	0..n	CSV file name aliases
ciattribute	1..n	CI attribute definition
ciref	1..n	CI reference definition

XML Element: **aliassource**

aliassource element is used to define the CSV file aliases for the data collected so that one have multiple copies of the same data in the form of multiple files. This provides the ability to performed different operations on the same data in an independent manner.

> **Note:** If aliassource is not defined the generated file name will have the pattern

* viewName _0_ citypeName _0_*.csv

Attribute	Mandatory	Description
description	No	Brief description about the alias source

Child Elements	Multiplicity	Description
aliastarget	1..n	Target CSV file name patterns

XML Element: **aliastarget**

Target CSV file aliases are defined using aliastarget element.

Attribute	Mandatory	Description
type	Yes	User defined alias for the citype
category	Yes	User defined alias for the view
description	No	Description about the alias

> **Note:** The value of “type” and “category” attribute will used in the generated CSV filename. The filename pattern will be *category_0_type_0_*.csv.
>
> If two citype elements define the same type and category in aliastarget then the data from both the citype will be merged and appended to one file.

XML Element: **ciref**

A ciref element is used to refer CI attributes of other CI type.

> **Note:** A CI type can only refer to CI attributes of CI types that are defined in same hierarchy as the current CI type.

Attribute	Mandatory	Description
name	Yes	Name of CI type being referred
relationName	No	Represents the type of relationship between the parent CI and the referenced CI. When relationName as defined in RTSM view (BSM Modelling Studio) is specified in the collection policy, only those CI instances that are associated to the parent CI with the specific relationName are collected by the HPE OBR RTSM collector.
description	No	Reference description

Child Elements	Multiplicity	Description
ciattribute	1..n	CI attribute definition

XML Element: **ciattribute**

A ciattribute element defines CI attribute.

Attribute	Mandatory	Description
name	Yes	Name of the attribute as defined in RTSM (BSM Modelling Studio > CI Type Manager).
datatype	Yes	Data type of the attribute Note: Not used currently
csvColumnName	No	Target CSV column name for the CI attribute. If not specified attribute name is used as CSV column name
description	No	Attribute description

RTSM collection policy Examples

Sample RTSM collection policy XML file

Let's consider defining CMDB collection policy for Oracle database deployment in an enterprise. The CMDB collection policy can be defined using the CI types in "ORA_Deployment" CMDB view provided by ORACLE DBSPI. Please refer to the sample CMDB collection policy file uCMDB_Collection_ Policy.xml in attachment section for Oracle Database deployment. The collection policy defines nt, unix and oracle CI types with oracle CI type referring to attributes in nt and unix CI type.

SiteScope API Collection Policy XML Structure

< etldefinition >

< category >

< monitor >

< aliassource >

  < aliastarget />

</ aliassource >

<counter>

<csvcolumn>

</counter>

</ etldefinition >

XML Element: **etldefinition**

etldefinition is the root element for SiteScope API collection policy definition. This element specifies type of the collection and domain for which the collection is defined

Attribute	Mandatory	Description
type	Yes	Specifies the ETL step. The value should be “SIS” for a SiteScope collection policy
collector	Yes	Type of the collector. The value should be “SIS” for a SiteScope collection policy

XML Element: **Category**

Child Elements	Description
Category	Logical name given for the group of monitors to be collected.

XML Element: **Monitor**

Attribute	Mandatory	Description
type	Yes	Type of the SiteScope monitor that needs to be collected. This name should be as defined in SiteScope.
summarizeinterval	Yes	Summarization interval that is to be used for any aggregation that is to be performed on individual measures.

XML Element: **aliassource**

aliassource element is used to define the CSV file aliases for the data collected so that one have multiple copies of the same data in the form of multiple files. This provides the ability to performed different operations on the same data in an independent manner.

Attribute	Mandatory	Description
aliasSourceType	Yes	Source type and category for which alias is to be defined

XML Element: **aliastarget**

Target CSV file aliases are defined using aliastarget element.

Attribute	Mandatory	Description
aliastargettype	Yes	Individual Target alias type and category for source

XML Element: **counter**

Counter element of the SiteScope xml

Attribute	Mandatory	Description
filter	yes	Java regular expression applied on source attribute
source	yes	Name of monitor attribute on which regular expression will be applied. Default attribute value is name.

Child Elements	Description
csvcolumn	Element to define csv column name. Includes the following attributes: • name: Target Column name • source: Source column name

OM Collection Policy XML Definition

OM collection policy xml defines rules involving PA Data source and PA class from which dimension data is collected. HPE OBR connects to Operations Manager configured in Admin UI to get the managed nodes and perform dimension discovery on those nodes.

XML Structure

< etldef inition >

< sncollection >

< rule >

< aliasso urce >

< aliastarget />

</ aliassource >

< mapping />

</ rule >

</ sncollection >

</ etldefinition >

XML Element: **etldefinition**

etldefinition is the root element for OM collection policy definition. This element specifies type of the collection and domain for which the collection is defined

Attribute	Mandatory	Description
type	Yes	Specifies the ETL step. The value should be “collect” for a OM collection policy
collector	Yes	Type of the collector. The value should be “OM” for a SN/OM collection policy
domain_name	Yes	Domain for which the collection is performed
contentpack_ name	No	Name of the content pack for the domain

Attribute	Mandatory	Description
description	No	Brief description about the collection policy

Child Elements	Multiplicity	Description
sncollection	1..1	Grouping of SN collection rules

XML Element: **sncollection**

Views definition

Attribute	Mandatory	Description
name	Yes	Name of SN collection
mappedby	No	PA data source name
description	No	SN collection description

Child Elements	Multiplicity	Description
rule	1..n	List of SN collection rules

XML Element: **rule**

SN collection rule definition

Attribute	Mandatory	Description
citype	Yes	User defined CI type name. The CI type name is analogous to CMDB CI type name.
datasource	Yes	PA data source name. If data source name is not constant (such as in SAP SPI), you can define a pattern (regular expression) to match the data source name. Syntax PATTERN:<Java_regex> Example PATTERN:ABC.* Limitation: If the pattern matches multiple data sources, the classes and metrics must be same across all data sources.
class	Yes	PA class name
type	No	Specifies whether the CI should be considered os

Attribute	Mandatory	Description
		dimension/datasource/both. The allowed values are • dimension – CIs from this citype are considered only as dimension and not data source (only dimension CSV file is created and no entry is made in to data source dictionary table in HPE OBR) • datasource – CIs from this citype are considered only as data source and not dimension (only entry is made in to data source dictionary table in HPE OBR and no dimension CSV file created) • both – CIs from this citype are considered both as dimension and data source dimension (entry is made into data source dictionary table in HPE OBR and dimension CSV file is created).The default value is "both"
filter	No	Condition to filter rows. Please refer to Appendix A for complete list of supported filters and their syntax
description	No	SN rule description

Child Elements	Multiplicity	Description
aliassource	0..n	File name aliases for the collected data
mapping	1..n	List of mappings between CSV column and PA metric

XML Element: **aliassource**

aliassource element is used to define the CSV file aliases for the data collected so that one have multiple copies of the same data in the form of multiple files. This provides the ability to performed different operations on the same data in an independent manner.

Note: If aliassource is not defined the generated file name will have the pattern

* collectionName _0_ citypeName _0_*.csv

Attribute	Mandatory	Description
description	No	Description about the source alias

Child Elements	Multiplicity	Description
aliastarget	1..n	Target CSV file name patterns

XML Element: **aliastarget**

Target CSV file aliases are defined using aliastarget element.

Attribute	Mandatory	Description
type	Yes	User defined alias for the citype
category	Yes	User defined alias for the view
description	No	Description about the alias

> **Note:** The value of “type” and “category” attribute will used in the generated CSV filename. The filename pattern will be *category_0_type_0_*.csv.
>
> If two rule elements define the same type and category in aliastarget then the data from both the class will be merged and appended to only one file.

XML Element: **mapping**

Attribute	Mandatory	Description
source	Yes	CSV column name
metric	Yes	PA metric name
defaultvalue	No	Default value. Following are the substitution variables for default value • $agentname • $currenttime
description	No	Mapping description

OM Collection policy Examples

Sample OM collection policy XML file

Let’s consider defining SN collection policy for Oracle database deployment in an enterprise. The OM collection policy can be defined using PA data sources “DBSPI_ORA_GRAPH”, “CODA”. Please refer to the sample OM collection policy OM_Collection_Policy.xml in attachment section for Oracle Database deployment. The collection policy defines PA collection rules for nt, unix and oracle CI types.

OA Collection Policy XML Definition

OA/PA collection policy xml defines PA data source and PA class from which the metrics are to be collected.

XML Structure

```
< etldefinition >
< domain >
  < datasource >
    < class >
     < aliassource >
     < aliastarget />
     </ aliassource >
     < metric />
    </ class >
  </ datasource >
</ domain >
</ etldefinition >
```

XML Element: etldefinition

etldefinition is the root element for OA/PA collection policy definition. This element specifies type of the collection and domain for which the collection is defined

Attribute	Mandatory	Description
type	Yes	Specifies the ETL step. The value should be “collect” for a PA collection policy
collector	Yes	Type of the collector. The value should be “PA” for a PA/OA collection policy
domain_	Yes	Domain for which the collection is defined e.g. System

Attribute	Mandatory	Description
name		Management, Network Performance, etc. Note: The value is used to map nodes discovered for a domain (through CMDB/OM collection policy) to PA collection policy defined for a domain
contentpack_name	No	Name of the content pack for the domain
description	No	Brief description about the PA collection policy

Child Elements	Multiplicity	Description
domain	1..n	Collection domain

XML Element: domain

Domain definition

Attribute	Mandatory	Description
description	No	Description about the domain

Child Elements	Multiplicity	Description
datasource	1..n	List of PA data sources from which the data is to be collected

XML Element: **datasource**

A datasource element defines a PA data source from which the data is to be collected.

Attribute	Mandatory	Description

name	Yes	Name of the PA data source. Example CODA, SCOPE, SPI (Oracle, MSSQL, Microsoft Exchange, and so on).
		If data source name is not constant (such as in SAP SPI), you can define a pattern (regular expression) to match the data source name.
		Syntax
		`PATTERN:<Java_regex>`
		Example
		`PATTERN:ABC.*`
		Limitation:

Attribute	Mandatory	Description
		If the pattern matches multiple data sources, the classes and metrics must be same across all data sources.
summarized	No	Boolean indicating the type of data (raw or summarized) to be collected. If set to true summarized data is collected. The default value is true.
description	No	Brief description about the PA data source

Child Elements	Multiplicity	Description
class	1..n	List of classes in PA data source to be collected

XML Element: **class**

A class element defines a PA class under the specified data source from which the data is to be collected.

Attribute	Mandatory	Description
name	Yes	Name of the PA class
summarized	No	Boolean indicating the type of data (raw or summarized) to be collected. If set to true summarized data is collected. The default value is true. Note: This value overrides the “summarized” attribute defined at data source level
description	No	Brief description about the class

Child Elements	Multiplicity	Description
metric	1..n	List of metrics to be collected from a PA class
aliassource	0..1	File name aliases for the collected data

XML Element: **metric**

A class element defines a metric under a PA class to be collected

Attribute	Mandatory	Description
name	Yes	Name of the metric defined under PA class
datatype	Yes	Data type of the metric. Note :The value is not used currently
identity	Yes	Is key column Note :The value is not used currently

XML Element: **aliassource**

aliassource element is used to define the CSV file aliases for the data collected so that one have multiple copies of the same data in the form of multiple files. This provides the ability to performed different operations on the same data in an independent manner.

> **Note:** If aliassource is not defined the generated file name will have the pattern
>
> *PAData source _0_PAClass_0_*.csv*

Attribute	Mandatory	Description
type	No	Name of the PA class
category	No	Name of the PA data source
description	No	Description about the alias

Child Elements	Multiplicity	Description
aliastarget	1..n	Target CSV file name patterns

XML Element: **aliastarget**

Target CSV file aliases are defined using aliastarget element.

Attribute	Mandatory	Description
type	Yes	User defined alias for the PA class
category	Yes	User defined alias for the PA data source
description	No	Description about the alias

> **Note:** The value of “type” and “category” attribute will used in the generated CSV filename. The filename pattern will be *category_0_type_0_*.csv.
>
> If two class elements define the same type and category in aliastarget then the data from both the class will be merged and appended to only one file.

OA collection policy Examples

Sample OA collection policy XML file

Let's assume we have to collect CPU metrics from Performance Agent. CPU metrics are available from

- CPU class in SCOPE data source
- CPU class in CODA data source

Please refer to the sample PA collection policy SystemManagement_OA_Collection_Policy.xml in attachment section that has definitions to collect CPU metrics from Performance Agent

DB Collection Policy XML Definition

A DB collection policy defines tables and queries to collect data from various DB data sources like OM, OMi, ManagementDB, ProfileDB and other JDBC supported databases.

XML Structure

```
<etldefinition >
< queries >
  < q u ery >
  < tables >
< table >
< columns >
  < column />
</ columns >
< condition />
</ table >
</ tables >
< genericsql >
< statement >
</ statement >
</ genericsql >
< joinquer i es >
< joinquery >
 < aliassource >
     < aliastarget />
    </ aliassource >
```

< co n tent ></ content >

</ joinquery >

</ joinquer i es >

</ query >

</ queries >

</ etldefinition >

XML Element: **etldefinition**

etldefinition is the root element for DB collection policy definition. This element specifies type of the collection and domain for which the collection is defined.

Attribute	Mandatory	Description
type	Yes	Specifies the ETL step. The value should be “collect” for a DB collection policy
collector	Yes	Type of the collector. The value should be “DB” for a DB collection policy
domain_name	Yes	Domain for which the collection is performed
contentpack_ name	No	Name of the content pack for the domain
description	No	Brief description about the collection policy

Child Elements	Multiplicity	Description
queries	1..1	Group of query specifications

XML Element: **queries**

Queries definition

Attribute	Mandatory	Description
name	Yes	Name of query grouping
description	No	Description about the query grouping

Child Elements	Multiplicity	Description
query	1..n	List of queries

XML Element: **query**

Query definition

Attribute	Mandatory	Description
name	Yes	Name of the query
type	Yes	Query type. The value indicates the DB source on which the query will be executed. The allowed values are OM - query will be executed against OM database configured in AdminUI OMI - query will be executed against OMI databases configured in AdminUI PROFILE_DATABASE - query will be executed against profile databases configured in AdminUI generic - query will be executed against generic databases configured in AdminUI
dbtype	Yes	DB type. The allowed values are ORACLE SYBASEIQ MSSQL **Note:** If type attribute is set to “PROFILE_DATABASE” then dbtype can be either “ORACLE” or “MSSQL”
description	No	Brief description about the query

Child Elements	Multiplicity	Description
tables	1..1	Grouping of tables to be queried
joinqueries	1..1	Grouping of join queries
genericsql	1..n	Generic SQL statements to be executed

XML Element: **genericsql**

The generic SQL statement element defines the SQL query that must be run on the data source (OM, OMI, Generic database, and so on.)

Attribute	Mandatory	Description
name	Yes	Name of the generic query to uniquely identify the same for incremental collection use-cases
timelag	No	Number of hours to go back in the past (from last collected maximum time) to collect data that has arrived late at the source. For example, if timelag is set as 1 hour and the maximum time recorded during last collection is 21:00:00, at 22:00:00 hours, data

Attribute	Mandatory	Description
		will be collected from 20:00:00 to 22:00:00 hours.
initialhistory	No	The amount of historical data to be collected when the collection occurs for the first time. For example, if initialhistory is set to 48, data for the last 48 hours before the current time is collected for the first time. **Note:** OBR records the maximum time for which the data is collected during every collection and uses it as the starting point for the subsequent collection.
description	No	Description of the query.
category	No	User defined name. Typically indicates the source (OMi, OM, and so on).
lastcollectioncol	No	Indicates that the column mentioned as value of this variable should be used for tracking last collection time. **Note:** Only one column should be used as last collection column. Also, the query actually mentioned should have the where clause updated with the expression <column_name> > $FETCH_START_TIME where <column_name> is the value of this particular variable.
timecoltypes	No	Contains a list of time period columns and their corresponding time period type (UTC/datetime). The format of specifying the list is <time_col1>,<time_col_type>::<time_col2>,<time_col_type> A sample entry could be start_time,UTC::end_time,datetime **Note:** To create customized reports for the data collected from HPOM, OMi, or Profile DB, make sure that data in the date/time columns is converted to UTC.
alias	No	Alias for the query. The SQL query defined under the joinquery element must use the alias name instead of query name.
persisttype	Yes	Contains the mode to persist the data collected from the query. It can take two values – File or Table. File mode indicates that data returned by the query execution will be saved as a CSV file directly using the type and category specified. Table mode is similar to the table section of querying, where the data collected is loaded to alias tables that can be used in join queries.
type	No	User defined name. Typically indicates the type of data (fact/dimension).

Child Elements	Multiplicity	Description
statement	1..1	The actual SQL statements to be run.

XML Element: **statement**

The statement element contains user-defined SQL queries to be run.

XML Element: **tables**

Attribute	Mandatory	Description
description	No	Brief description about the tables grouping

Child Elements	Multiplicity	Description
table	1..n	List of tables to be queried

XML Element: **table**

A table element defines a physical database table from source (OM, OMI, Generic database, and so on)

Attribute	Mandatory	Description
name	Yes	Name of the physical table in source database
alias	No	Alias for the table. The SQL query defined under joinquery element should use alias name instead of physical name since alias is used as temporary table name in OBR database. If not defined name attribute is used as temporary table name.
type	Yes	Allowed values are • persist – query the table and persist the result to a temporary table in OBR database. In this case joinquery must be defined to dump the data from temporary table in OBR to the CSV file. This option is generally used to reduce the load on source DB running simple query on the source DB and dump the data to a temporary table in OBR and then run complex join queries on these temporary tables to dump the desired data to CSV file for further processing • dumpcsv – query the table and save the result to a csv file. No temporary tables will be created in OBR database. This option is generally used when source DB can handle performance intensive query.
initialhistory	No	Number of hours of history data to be collected when the collection happens for the first time. For example if initial history is set to 48 then the data will be collected from 48 hours behind the current time when the

Attribute	Mandatory	Description
		collection happens for the first time. Note: OBR records the maximum time for which the data is collected during every collection and uses it as the starting point for the subsequent collection.
timelag	No	Number of hours to go behind (from last collected maximum time) to collect data that has arrived late at the source. For example if timelag is set as 1 hour and maximum time recorded during last collection is 21:00:00 then at 22:00:00 hours then data will be collected from 20:00:00 to 22:00:00 hours.
dstype	No	Allowed value is “BAC_MANAGEMENT”. This value indicates that the query should be executed against management DB instead of “PROFILE_DATABASE”. This attribute should be defined only if query type is set as “PROFILE_DATABASE”.
description	No	Table description

Child Elements	Multiplicity	Description
columns	0..1	Group of columns
condition	0..1	SQL conditional expression for filtering data

XML Element: **columns**

Attribute	Mandatory	Description
description	No	Description about the column grouping

Child Elements	Multiplicity	Description
column	1..n	List of columns in the table

XML Element: **column**

Table column definition

Attribute	Mandatory	Description
name	Yes	Name of the column
alias	No	Alias for the table. The SQL query defined under joinquery element should use alias name instead of physical column name

Attribute	Mandatory	Description
		since alias is used as column names in temporary table. If not defined name attribute is used as column names in temporary table.
lastColColumnType	No	Indicates that the column is a time period column. Allowed values are • utc • datetime
lastcollectiontime	No	Indicates that the column should be used for tracking last collection time. When database collector runs hourly collection, it queries for data from the last collection time until the current time. Allowed values are TRUE and FALSE, but only one column should be marked as lastcollectiontime="TRUE". The last collected record from the earlier data collection run might be repeated in the current run, resulting in duplicate data in the CSV files. However, the duplicates are discarded when the CSV files are loaded to the OBR data warehouse. You can prevent the duplicate collection by using generic SQL in the collection policies.
description	No	Column description

XML Element: **condition**

condition element defines the SQL conditional expression for filtering data from source table.

> **Note:**
>
> - If a column is marked as lastcollectiontime ="true" then condition involving this time column will be added automatically beginning with “where” keyword in the generated SQL. Hence “where” keyword should not be specified while defining the filter condition for other columns.
> - If no column is marked as lastcollectiontime ="true" then filter condition if any should be defined starting with “where” keyword

Attribute	Mandatory	Description
expression	Yes	Where clause condition for filtering data(SQL syntax)
description	No	Brief description about condition

XML Element: **joinqueries**

Attribute	Mandatory	Description
description	No	Description for join query grouping

Child Elements	Multiplicity	Description
joinquery	1..n	List of join queries

XML Element: **joinquery**

The joinquery element defines a query that will be executed against the temporary tables containing the data fetched from database sources like (OM, OMI, and Generic DB). All the values in the temporary tables are treated as “varchar” data type.

Attribute	Mandatory	Description
type	Yes	User defined name. Typically indicates the type of data (fact/dimension).
category	Yes	User defined name. Typically indicates the source (Omi, OM, and so on).
description	No	Description about the query

> **Note:** The value of “type” and “category” attribute will used in the generated CSV filename. The filename pattern will be *category_0_type_0_*.csv

Child Elements	Multiplicity	Description
aliassource	0..n	CSV file name aliases
content	1..1	SQL query

XML Element: **aliassource**

aliassource element is used to define the CSV file aliases for the data collected so that one have multiple copies of the same data in the form of multiple files. This provides the ability to performed different operations on the same data in an independent manner.

> **Note:** If aliassource is not defined the generated file name will have the pattern joinqueryCategory _0_ joinqueryType _0_*.csv

Attribute	Mandatory	Description
description	No	Alias description

Child Elements	Multiplicity	Description
aliastarget	1..n	Target CSV file name patterns

XML Element: **aliastarget**

Target CSV file aliases are defined using aliastarget element.

Attribute	Mandatory	Description
type	Yes	User defined alias for the joinquery type
category	Yes	User defined alias for the joinquery category
description	No	Description about the alias

XML Element: **content**

The content element contains user defined SQL query containing join conditions. You can perform only string operations as part of the SQL queries; arithmetic operations are not permitted.

DB Collection Policy Examples

Sample Generic DB collection policy XML file

Please refer to the sample generic DB collection policy Generic_DB_Collection_Policy.xml in attachment section that defines queries to collect system CPU data from Microsoft SCOM database.

Transformation Policy XML Definition

A transformation policy specifies a set of record sets with filter conditions. Each record set defined in a transformation XML will result in an output CSV file after the transformation process and the content of the CSV is determined by the records defined in the record set.

XML Structure

< etldefinition >

< recordSet >

< record />

</ recordSet >

</ etldefinition >

XML Element: etldefinition

etldefinition is the root element for transformation policy definition. This element specifies set of record sets with transformation rules.

Attribute	Mandatory	Description
type	Yes	Specifies the ETL step. The value should be “TRANSFORM” for a transformation policy
contentpack_ name	No	Name of the content pack
description	No	Brief description about the transformation policy

Child Elements	Multiplicity	Description
recordSet	1..n	List of record sets

XML Element: recordSet

recordSet element defines transformation rules for a source CSV pattern.

Attribute	Mandatory	Description
name	Yes	Name of the record set. The name should be unique.
source_ type	Yes	Source type. It can be PA class/citype/user defined alias for PA class/ user defined alias for citype. Source type is used to identify the source/input CSV file for the defined recordset.
source_ category	Yes	Source category. It can be PA data source/RTSM view/ user defined alias for PA data source/ user defined alias for RTSM view. Source category is used to identify the source/input CSV file for the defined recordset.
target_ type	Yes	Target type. Can be same as source_type value or can be user defined value. Target type will be used in output CSV file name.
target_ category	Yes	Target category. Can be same as source_category value or can be user defined value. Target category will be used in output CSV file name.
description	No	Record set description.
condition	No	Condition to filter rows. Please refer to Appendix A for complete list of supported filters and their syntax.
doPivot	No	A Boolean value that indicates whether the transformation involves a pivot transform or not. The default value is false.

- The values of source_type and source_category attributes are used to identify the source/input file for the recordSet definition. For example if a recordSet element defines source_type as “CPU” and source_category as “CODA” then source CSV files with filename pattern *CODA_0_CPU_0_*.csv will be picked up for transformation.
- The values of target_type and target_category attributes are used to form the output/target CSV filename. For example if a recordSet element defines target_type as “CPU1” and target_category as “CODA1” then output CSV filename will have the pattern *CODA1_0_CPU1_0_*.csv.

Child Elements	Multiplicity	Description
record	0..n	List of column mappings between input and output CSV.

XML Element: **record**

record element specifies column mapping between input CSV and output CSV columns with filter conditions.

Attribute	Mandatory	Description
name	Yes	Name of the column in the output CSV. The name has to be unique within the recordset.
source	Yes	The source column that corresponds to the output column. The source can just be a column name from the source/input CSV or can include one or more transform functions in which case, the functions will be executed and the final result will be placed in the output CSV. Please refer to Appendix A for complete list of supported filters and their syntax.
condition	No	Condition to filter rows. Please refer to Appendix A for complete list of supported filters and their syntax. Note: If the condition is not satisfied then the output CSV column value will be “NotFound”.
description	No	Output column description.

Transformation Policy Examples

Pivot transformation:

Pivot transform involves merging multiples rows in source CSV file to a single row in output CSV file based on id columns. The transformation module keeps track of rows with similar id column values and merges those rows in to a single row by merging the non id columns. Pivot transformation is configured for a recordSet in transformation xml by setting the doPivot attribute to “true”. Pivot transformation is generally done on data collected from Performance Agent where CODA logs metrics as key value pairs in case of HP SPI (DBSPI_MSS_REPORT(MSSQL), DBSPI_ORA_REPORT(Oracle), etc) data source.

For example consider Oracle instance space utilization data from Oracle SPI. The CSV data collected will look like below:

DBSPI_ORA_REPORT_0_InstanceSpaceutilization_0_723914438884967.csv							
HOSTNAME	DATASOURCE	CLASSNAME	AGENTTIMESTAMP	INSTANCE	METRICID	VALUEID	VALUE
shr1.ind.hp.com	DBSPI_ORA_REPORT	DBSPI_ORA_REPORT	4/5/2012 9:30	BAT92	212	1	1890
shr1.ind.hp.com	DBSPI_ORA_REPORT	DBSPI_ORA_REPORT	4/5/2012 9:35	BAT92	212	1	1890
shr1.ind.hp.com	DBSPI_ORA_REPORT	DBSPI_ORA_REPORT	4/5/2012 9:40	BAT92	212	1	1890
shr1.ind.hp.com	DBSPI_ORA_REPORT	DBSPI_ORA_REPORT	4/5/2012 9:45	BAT92	212	1	1890
shr1.ind.hp.com	DBSPI_ORA_REPORT	DBSPI_ORA_REPORT	4/5/2012 9:30	BAT92	212	2	225.48
shr1.ind.hp.com	DBSPI_ORA_REPORT	DBSPI_ORA_REPORT	4/5/2012 9:35	BAT92	212	2	225.56
shr1.ind.hp.com	DBSPI_ORA_REPORT	DBSPI_ORA_REPORT	4/5/2012 9:40	BAT92	212	2	225.56
shr1.ind.hp.com	DBSPI_ORA_REPORT	DBSPI_ORA_REPORT	4/5/2012 9:45	BAT92	212	2	225.56
shr1.ind.hp.com	DBSPI_ORA_REPORT	DBSPI_ORA_REPORT	4/5/2012 9:30	USERS	213	1	1.22
shr1.ind.hp.com	DBSPI_ORA_REPORT	DBSPI_ORA_REPORT	4/5/2012 9:35	USERS	213	1	1.22
shr1.ind.hp.com	DBSPI_ORA_REPORT	DBSPI_ORA_REPORT	4/5/2012 9:40	USERS	213	1	1.22
shr1.ind.hp.com	DBSPI_ORA_REPORT	DBSPI_ORA_REPORT	4/5/2012 9:45	USERS	213	1	1.22

The CSV data clearly shows that the metrics are logged as key value pairs in separate rows. In the CSV the “METRICID” column with the value “212” corresponds to instance space utilization and “VALUEID” column with value “1” and “2” corresponds to total instance space and free instance space respectively. These two columns in different rows needs to be merged in to one for each unique combination of id columns “HOSTNAME”, “INSTANCE” and “AGENTTIMESTAMP” before loading in to HPE OBR data warehouse.

The transformation policy to do pivot transform for the above CSV data can be specified as below

```
<?xml version="1.0" encoding="UTF-8"?>

<etldefinition type="TRANSFORM" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="SHRTransformNamespace schema_transform.xsd"
xmlns="SHRTransformNamespace">

<!-- specify a condition on METRICID column at recordSet level to pick only
instance space utilization metrics -->

 <recordSet name="InstanceSpaceutilization"

            condition="METRICID IN (212.0)"

            source_type="InstanceSpaceutilization"

            source_category="DBSPI_ORA_REPORT"

            target_type="InstanceSpaceUtilization"

            target_category="DBSPI_ORA_REPORT" doPivot="true">

    <record name="HostName" source="HOSTNAME" id="true" />
```

```
    <record name="InstanceName" source="INSTANCE" id="true" />
    <record name="AGENTTIMESTAMP" source="AGENTTIMESTAMP" id="true" />
    <!-- specify a condition on VALUEID column to pick the value of
          allocated  instance space  -->
    <record name="InstanceSpaceAllocatedSize" source="VALUE"
            condition="VALUEID=1.0" />
    <!-- specify a condition on VALUEID column to pick the value of
         free instance space  -->
    <record name="InstanceSpaceFreeSize" source="VALUE"
            condition="VALUEID=2.0" />
 </recordSet>
</etldefinition>
```

The output CSV will look like below

DBSPI_ORA_REPORT_0_InstanceSpaceutilization_0_723914438884975.csv				
HOSTNAME	INSTANCENAME	AGENTTIMESTAMP	InstanceSpaceAllocatedSize	InstanceSpaceFreeSize
shr1.ind.hp.com	BAT92	4/5/2012 9:30	1890	225.48
shr1.ind.hp.com	BAT92	4/5/2012 9:35	1890	225.56
shr1.ind.hp.com	BAT92	4/5/2012 9:40	1890	225.56
shr1.ind.hp.com	BAT92	4/5/2012 9:45	1890	225.56

Using conditions to filter data

Transformation policy XML supports various filter conditions that can be used to filter the rows in input/source CSV file. Conditions can be specified in recordSet element (to filter rows) and record element (to filter columns). Below are few examples for specifying conditions

```
<!--  The condition below is defined to select the rows whose CLASSNAME column
value is “MEMORY” and HOSTNAME column value is not ”NULL”   -->
<recordSet name=" Memory"
           condition="CLASSNAME IN (MEMORY) AND HOSTNAME NOT IN (NULL)"
```

```
            source_type=" MEMORY"
            source_category="SiS"
            target_type=" MEMORY"
            target_category="SiS" >
<record name="AGENTTIMESTAMP" source="AGENTTIMESTAMP" id="true"/>
<record name="HOSTNAME" source="HOSTNAME" id="true"/>
<record name="MEMORY_MB_FREE" source="MEMORY_MB_FREE"
              <!-- Filter out negative values-->
              condition="MEMORY_MB_FREE != -1" />
<record name="MEMORY_MB_TOTAL" source="MEMORY_MB_TOTAL"
        condition="MEMORY_MB_TOTAL != -1" />
</recordSet>
```

For example consider the source CSV

SiS_0_MEMORY_0_723914438884967.csv				
HOSTNAME	**CLASSNAME**	**AGENTTIMESTAMP**	**MEMORY_MB_ FREE**	**MEMORY_MB_ TOTAL**
shr1.ind.hp.com	MEMORY	4/5/2012 9:30	4096	8192
shr1.ind.hp.com	MEMORY	4/5/2012 9:35	4096	8192
shr1.ind.hp.com	MEMORY	4/5/2012 9:40	4096	8192
shr1.ind.hp.com	MEMORY	4/5/2012 9:45	3584	8192
shr1.ind.hp.com	MEMORY	4/5/2012 9:50	3584	8192
shr1.ind.hp.com	MEMORY	4/5/2012 9:55	3584	8192
Null	Null	4/5/2012 10:00	3584	8192
shr1.ind.hp.com	MEMORY	4/5/2012 10:00	-1	-1
shr1.ind.hp.com	MEMORY	4/5/2012 10:00	3584	8192

Applying above transformation policy on this source CSV will result in following output CSV

SiS_0_MEMORY_0_723914438884975.csv				
HOSTNAME	**CLASSNAME**	**AGENTTIMESTAMP**	**MEMORY_MB_FREE**	**MEMORY_MB_TOTAL**
shr1.ind.hp.com	MEMORY	4/5/2012 9:30	4096	8192
shr1.ind.hp.com	MEMORY	4/5/2012 9:35	4096	8192
shr1.ind.hp.com	MEMORY	4/5/2012 9:40	4096	8192
shr1.ind.hp.com	MEMORY	4/5/2012 9:45	3584	8192
shr1.ind.hp.com	MEMORY	4/5/2012 9:50	3584	8192
shr1.ind.hp.com	MEMORY	4/5/2012 9:55	3584	8192
shr1.ind.hp.com	MEMORY	4/5/2012 10:05	3584	8192

Using functions

Transformation policy supports a set of standard functions that can be used on source CSV column. Functions can also be used in conditions. This section will cover most widely used functions. For complete list of supported functions, please refer to Appendix A.

OPR_TOLOWER, OPR_TOUPPER

OPR_TOLOWER and OPR_TOUPPER functions convert the string value of the specified column from the source CSV to lower and upper case respectively

Usage

OPR_TOLOWER(source_column)

OPR_TOUPPER(source_column)

```
<record name="InstanceName" source="OPR_TOLOWER(INSTANCE)"/>

<record name="SegmentName" source=" OPR_TOUPPER(SEGMENT)"

   condition="OPR_TOLOWER(HOSTNAME)=shr1.ind.hp.com" />
```

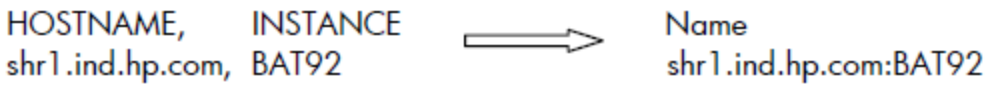
OPR_APPEND

This function appends the values of two columns specified separated by the delimiter.

Usage

OPR_APPEND(source_column1, source_column2,delimiter)

```
<record name="Name" source="OPR_APPEND(HOSTNAME, INSTANCE,:)"/>
```



OPR_STRINGSPLIT

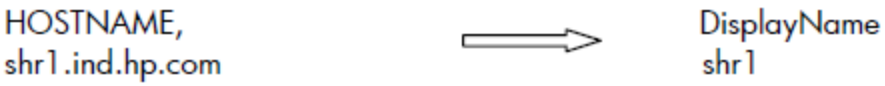
This function uses the string expression specified second to split the value for the column name specified and then returns the value from the array of strings after the split at the specified index. Please note that this index starts from 0.

Usage

OPR_STRINGSPLIT(<column name>,<String expression to be used as match for split>,<index of split to be returned>)

```
<record name="DisplayName"

source="OPR_ STRINGSPLIT(HOSTNAME, .,0)"/>
```



OPR_CHR_INDEX

The `OPR_CHR_INDEX` function gets the position/index of the first occurrence of a search string in the column value starting from the first letter of the string or the <`start index`>, if specified.

Usage

OPR_CHR_INDEX(<column>,<search string>)

OPR_CHR_INDEX(<column>,<search string>,<start index>)

OPR_TOINT

This function gets the integer value for the data present in the column. If the value is a decimal value then the decimal value is converted to the nearest integer.

Usage

OPR_TOINT(<column name>)

OPR_GENERATE_SHR_CIUID

This function is used to generate an unique ID for each CI

Usage

OPR_GENERATE_SHR_CIUID(<class>,<hostname>,<ciuid>)

OPR_STR_NOTCONTAINS

This function checks whether the value of the specified column does not contain the given expression and returns values based on the result.

Usage

OPR_STR_CONTAINS(<column name>,<string expression to be checked >)

OPR_DATEFORMAT

This function converts the date from <column> to HPE OBR expected format and configured time zone.

Usage

OPR_DATEFORMAT(<column name>)

- The destination format can be altered if required, by passing it as an argument to the OPR function:

  Example: `OPR_DATEFORMAT(<column name>,MMddyyyyHHmmss)`

- The destination format can also be rounded-off time to the nearest n^{th} minute boundary. This can be clubbed with any of the above usages.

  Example: `OPR_DATEFORMAT(<column name>,-5)`

  This will round-off the time to the nearest 5^{th} minute boundary (11:32 becomes 11:30)

- `OPR_DATEFORMAT(<column name>, MMddyyyyHHmmss,1 )` will seal the value to the next minute interval.

The source time formats that are supported in Mapper are:

- "MM/dd/yyyy HH:mm:ss Z"
- "MM/dd/yyyy HH:mm:ss"
- "yyyy-MM-dd HH:mm:ss"
- "yyyyMMdd HH:mm:ss"
- "yyyyMMdd-HHmmss"
- "HH:mm yyyyMMdd"
- "yyyyMMdd HHmm"
- "yyyyddMM-HHmm"
- "yyyyMMdd HH:mm"
- "MM/dd/yy HH:mm"
- "yyyyMMdd-HH:mm"
- "yyyyMMdd"
- "MM/dd/yy"
- "MM/dd/yyyy"
- "yyyy/MM/dd"
- "dd/MM/yy"

Reconciliation Rule XML Definition

A reconciliation policy XML defines set of rules to reconcile (process of attaching CIUID to the collected fact data) the collected fact data from data source like PA with CI type registry built during RTSM topology collection. Each rule defines a set of conditions involving columns from topology/dimension CSV (sourcecolumn – used to build registry) and fact CSV (targetcolumn – used to build keys to lookup registry).

XML Structure

```
< etldefinition >
< rule >
  < condition >
 < sourcecolumn />
  < operator ></ operator >
  < targetcolumn />
   </ condition >
    < relation ></ relation >
</ rule >
</ etldefinition >
```

XML Element: etldefinition

etldefinition is the root element for transformation policy definition.

Attribute	Mandatory	Description
type	Yes	Specifies the ETL step. The value should be “RECONCILE” for a reconciliation policy
contentpack_ name	No	Name of the content pack
description	No	Brief description about the transformation policy

Child Elements	Multiplicity	Description
rule	1..n	List of reconciliation rules

XML Element: rule

Attribute	Mandatory	Description
name	Yes	User understandable name for the rule
paClass	Yes	The PA class for which the reconcile rule is to be applied. The value can also be alias name for the PA class. This value is used to pick the right csv file for applying the rule. For example if the value is “CPU” then the rule will be applied to CSV file with the following file name pattern *_0_CPU_0_*.csv
ciType	Yes	The CI type against which registry lookup is to be done
idColumnName	No	The identity column to be used for the registry building from the dimension data set. By default this is the CIID column.
targetColumnName	No	The column name of the id column in the output CSV for fact data. The default value is CIID
defaultValue	No	The value to be used as default id column value if the id the column value is null. This attribute also takes substitution parameter as input where, the value of given column is substituted. Example: defaultValue=”${<SOURCE_COLUMN>}”
deploymentScenario	No	The scenario where the reconciliation rules are to be applied. The allowed values are • RTSM • OM If this attribute is not specified then the rule will be applied in all the scenarios.
category	No	The category against which the reconciliation has to happen
ignorecase	No	Boolean indicating whether to ignore the case of the value during reconciliation. By default this is set as true.

Attribute	Mandatory	Description
description	No	Brief description about reconciliation rule

Child Elements	Multiplicity	Description
condition	1..n	List of conditions
relation	0..n	Logical relation between conditions

XML Element: **condition**

Attribute	Mandatory	Description
description	No	Condition description

Child Elements	Multiplicity	Description
sourcecolumn	1..1	Source column to be used for registry building
operator	1..1	Logical operator
targetcolumn	1..1	Fact CSV column to be used for reconciliation against the registry

XML Element: **sourcecolumn**

sourceColumn element defines column name from the dimension CSV to be used for registry building.

Attribute	Mandatory	Description
name	Yes	Name of CSV column in dimension CSV
prefix	No	Prefix string that needs to be prefixed to the value of the source column for building the registry key/business key. The prefix can be a string or a substitution parameter for a source column, which substitutes the value of the given column as prefix during registry building. The substitution parameter will be of form prefix=”${<SOURCE_COLUMN>}” Example: Instance <sourcecolumn name="Instance" suffix=”ORA_”/> → Instance1 ORA_Instance1
suffix	No	The suffix string that needs to be suffixed to the value of the source column for building the registry key/business key. The suffix can be a string or a substitution parameter for a source column, which substitutes the value of the given column as suffix during registry building. The substitution parameter will be of form suffix=”${<SOURCE_COLUMN>}”

Attribute	Mandatory	Description
		Host <sourcecolumn name="Host" suffix=".ind.hp.com"/> → shr1 shr1.ind.hp.com
function	No	Function to be applied on the source column. Note: Only substring function is supported currently. This function returns the substring of the column at a given index when split over a given pattern. Example “Oracle_Instance1” → function=”SUBSTRING(“_”,1)” → “Instance1”
description	No	Source column description

XML Element: **operator**

operator element specifies the comparison operator between source column and target column. Following are the operators supported currently

- EQUALS – checks if the source and target column values are equal
- LIKE – checks if the target column value matches the pattern of the source column

XML Element: **targetcolumn**

targetColumn element defines column name from the fact CSV column to be used for reconciling the fact data against the registry.

Attribute	Mandatory	Description
name		Name of CSV column in fact CSV
prefix		Prefix string that needs to be prefixed to the value of the target column for building the registry lookup key. The prefix can be a string or a substitution parameter for a target column, which substitutes the value of the given column as prefix during registry building. The substitution parameter will be of form prefix=”${<TARGET_COLUMN>}” Example: Instance <targetcolumn name="Instance" suffix=”ORA_”/> → Instance1 ORA_Instance1

Attribute	Mandatory	Description
suffix		The suffix string that needs to be suffixed to the value of the target column for building the registry lookup key. The suffix can be a string or a substitution parameter for a target column, which substitutes the value of the given column as suffix during registry building. The substitution parameter will be of form suffix="${<TARGETE_COLUMN>}" Host <targetcolumn name="host" suffix=".ind.hp.com"/> → shr1 shr1.ind.hp.com
function		Function to be applied on the target column. Note: Only substring function is supported currently. This function returns the substring of the column at a given index when split over a given pattern. Example "Oracle_Instance1" → function="SUBSTRING("_",1)" → "Instance1"
description		Target column description

XML Element: **relation**

relation element defines logical relation two conditions. Only "AND" and "OR" operators are supported currently.

Reconciliation Rule Examples

Let's consider Oracle DBMS deployment in RTSM scenario. The CI types and topology information for Oracles DBMS are collected from RTSM and the metrics for Oracle DBMS performance are collected from HP Oracle Database Smart Plug-ins (SPI). These two data are tied together through reconciliation process. Data reconciliation is a two step process

1. Build reconcile registry of CIs using the data collected from RTSM
2. Attach the CI UID to the fact data collected from HP Operation Agent by looking up the reconcile registry.

Build reconcile registry of CIs using the data collected from RTSM

Data collected from RTSM			
ORA_Deployment_0_oracle_0_722990140479794.csv			
CiType	**CiId**	**database_dbsid**	**host_dnsname**
oracle	5935697711c6c268ca53888e5d30ecc7	BAT92	SHR1.IND.HP.COM
oracle	211408823a8cc17eca52174549f82f79	ORCL11G	SHR2.IND.HP.COM
oracle	1472234cfc2ee5bc2b3f2f79249ebc0f	VM8	SHR3.IND.HP.COM

Build Reconcile Registry based on reconciliation rule defined for source columns collected from RTSM

Reconcile Registry for "oracle" CI type	
business_key ciid	**business_key ciid**
_BAT92_SHR1.IND.HP.COM	5935697711c6c268ca53888e5d30ecc7
_ORCL11G_SHR2.IND.HP.COM	211408823a8cc17eca52174549f82f79
_VM8_SHR3.IND.HP.COM	1472234cfc2ee5bc2b3f2f79249ebc0f

Attach the CI UID to the fact data collected from HP Operation Agent by looking up the reconcile registry

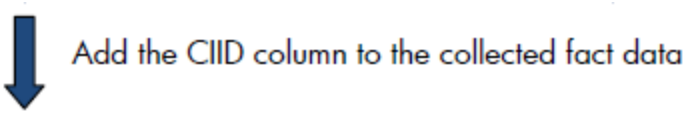
Fact Data (Oracle Instance space Utilization) collected from HP Operations Agent						
DBSPI_ORA_REPORT_0_InstanceSpaceutilization_0_723914434447063.csv						
HOSTNAME	DATASOURCE	CLASSNAME	AGENTTIMESTAMP	METRICID	INSTANCENAME	VALUEID
SHR1.IND.HP.COM	DBSPI_ORA_REPORT	DBSPI_ORA_REPORT	4/18/2012 5:25	215	BAT92	1
SHR1.IND.HP.COM	DBSPI_ORA_REPORT	DBSPI_ORA_REPORT	4/18/2012 5:30	215	BAT92	1
SHR1.IND.HP.COM	DBSPI_ORA_REPORT	DBSPI_ORA_REPORT	4/18/2012 5:35	215	BAT92	1
SHR2.IND.HP.COM	DBSPI_ORA_REPORT	DBSPI_ORA_REPORT	4/18/2012 5:25	173.1	vm8	1
SHR2.IND.HP.COM	DBSPI_ORA_REPORT	DBSPI_ORA_REPORT	4/18/2012 5:30	215	vm8	1
SHR2.IND.HP.COM	DBSPI_ORA_REPORT	DBSPI_ORA_REPORT	4/18/2012 5:35	209	vm8	1

Build reconcile registry lookup key based on the reconciliation rule defined for target columns collected from HP Operations Agent

```
<rule name="Reconciliation rule for InstanceSpaceutilization" ciType="oracle" paClass="InstanceSpaceutilization">
    <condition>
        <sourcecolumn name="database_dbsid"/>
        <operator>EQUALS</operator>
        <targetcolumn name="InstanceName"/>
    </condition>
    <relation>AND</relation>
    <condition>
        <sourcecolumn name="host_dnsname"/>
        <operator>EQUALS</operator>
        <targetcolumn name="HostName"/>
    </condition>
</rule>
```

Lookup reconcile registry with lookup key to get the corresponding CIID
e.g. InstanceName = BAT92, Hostname=SHR1.ind.hp.com
Lookup key = _BAT92_SHR1.IND.HP.COM

Reconcile Registry for "oracle" CI type	
business_key	**ciid**
_BAT92_SHR1.IND.HP.COM	5935697711c6c268ca53888e5d30ecc7
_ORCL11G_SHR2.IND.HP.COM	211408823a8cc17eca52174549f82f79
_VM8_SHR3.IND.HP.COM	1472234cfc2ee5bc2b3f2f79249ebc0f



Reconciled Fact Data (Oracle Instance space Utilization)							
DBSPI_ORA_REPORT_0_InstanceSpaceutilization_0_723914434447065.csv							
HOSTNAME	**DATASOURCE**	**CLASSNAME**	**AGENTTIMESTAMP**	**METRICID**	**INSTANCENAME**	**VALUEID**	**CIID**
SHR1.IND.HP.COM	DBSPI_ORA_REPORT	DBSPI_ORA_REPORT	4/18/2012 5:25	215	BAT92	1	5935697711c6c268ca53888e5d30ecc7
SHR1.IND.HP.COM	DBSPI_ORA_REPORT	DBSPI_ORA_REPORT	4/18/2012 5:30	215	BAT92	1	5935697711c6c268ca53888e5d30ecc8
SHR1.IND.HP.COM	DBSPI_ORA_REPORT	DBSPI_ORA_REPORT	4/18/2012 5:35	215	BAT92	1	5935697711c6c268ca53888e5d30ecc9
SHR2.IND.HP.COM	DBSPI_ORA_REPORT	DBSPI_ORA_REPORT	4/18/2012 5:25	173.1	vm8	1	211408823a8cc17eca52174549f82f79
SHR2.IND.HP.COM	DBSPI_ORA_REPORT	DBSPI_ORA_	4/18/2012 5:30	215	vm8	1	211408823a8cc17eca52174549f82f80

Reconciled Fact Data (Oracle Instance space Utilization)							
DBSPI_ORA_REPORT_0_InstanceSpaceutilization_0_723914434447065.csv							
HOSTNAME	**DATASOURCE**	**CLASSNAME**	**AGENTTIMESTAMP**	**METRICID**	**INSTANCENAME**	**VALUEID**	**CIID**
		REPORT					
SHR2.IND.HP.COM	DBSPI_ORA_REPORT	DBSPI_ORA_REPORT	4/18/2012 5:35	209	vm8	1	211408823a8cc17eca52174549f82f81

Stage Rule XML Definition

A stage rule xml defines rules and column mappings to load transformed data from CSV file to the corresponding stage table. A stage rule xml is typically developed referring the stage interface document which acts as an interface to the underlying stage area (physical stage tables).

XML Structure

< stageRule >

< targets >

< target >

< sources >

  < source />

</ sources >

< columnMap >

  < mapping />

</ columnMap >

< relational >

  < relation />

</ relational >

< keyColumns >

< ke y />

</ keyColumns >

</ target >

</ targets >

</ stageRule >

XML Element: stagerule

stagerule element is the root element for stage rule definition.

Attribute	Mandatory	Description
name	Yes	Name of the stage rule

Child Elements	Multiplicity	Description
targets	1..1	Grouping of stage area targets

XML Element: targets

targets element groups target element

Child Elements	Multiplicity	Description
target	1..n	List of stage area targets

XML Element: **target**

A target element links CSV file pattern to a stage table and defines mapping between the CSV and stage table columns.

Attribute	Mandatory	Description
name	Yes	Name of the stage table in stage interface document

Child Elements	Multiplicity	Description
sources	1..1	Group of source CSV file patterns
columnMap	1..1	Group of CSV and stage column mappings
relational	0..1	Join condition for merging columns from two source CSV files
keyColumns	0..1	Key columns in a CSV file (To delete duplicate rows)

XML Element: **sources**

Child Elements	Multiplicity	Description
source	1..n	List of source CSV file patterns

XML Element: **source**

Defines source CSV file patterns from which the data is moved to the stage table.

Attribute	Mandatory	Description
category	Yes	Source category. If the source CSV filename is Mapper_DISK_0_Reconcile_DISK_0_SCOPE_0_DISK_0_2506038836739754.csv then category is SCOPE. **Note:** Typically category in a CSV file name will be PA data source name or CMDB view name
type	Yes	Source type. If the source CSV filename is Mapper_DISK_0_Reconcile_DISK_0_SCOPE_0_DISK_0_2506038836739754.csv then type is DISK. **Note:** Typically type in a CSV file name will be PA class name or CMDB citype name
alias	Yes	User defined alias for type and category combination
schemaName	Yes	Schema name of the stage interface document that has the stage definition for the target
retentionDays	No	Retention period in days for the data in stage table. The default value is 3

XML Element: **columnMap**

Child Elements	Multiplicity	Description
mapping	1..n	List of CSV and stage column mappings

XML Element: **mapping**

mapping element defines column mapping between a CSV column and a stage column

Attribute	Mandatory	Description
srcColumn	Yes	Name of the source CSV column prefixed with alias defined for source Example: source_alias.csvColumnName
tgtColumn	Yes	Stage table column name defined stage interface document

XML Element: **relationa**l

Child Elements	Multiplicity	Description
relation	1..n	List of CSV and stage column mappings

XML Element: **relation**

Column mapping between CSV column and stage column

Attribute	Mandatory	Description
key	Yes	Join condition for merging two CSVs

XML Element: **keyColumns**

Child Elements	Multiplicity	Description
key	1..n	List of key columns in CSV file

XML Element: **key**

Defines key columns in source CSV file

Attribute	Mandatory	Description
alias	Yes	Source CSV alias
srcColumn	Yes	Source CSV column name
sortKey	No	Source CSV column which should be considered for deleting duplicate value. The default value is “INSERTTIME” column and duplicates will be deleted by retaining the row with latest insert time and discard the rest of the rows.

Stage Rule Examples

Sample stage rule XML file

Please refer to the sample stage rule `Stagerule_CPU_SCOPE.xml` in attachment section for loading CSV data to stage table

CSV column merge example

In some case a source CSV file (data/metrics from a PA class) might not have all the columns required by the stage table. However the missing columns might available as part of other source CSV files (data/metrics from a PA class). In such cases the columns from the CSV files needs to be merged based on a condition and then loaded to the stage table. Please refer to the sample stage rule sample

stage rule Stagerule_column_merge_example.xml in attachment section that defines column merge between two CSV files.

ABC Stream XML Definition

ABC (Audit, Balance, and Control) is a framework that enables to model and execute work flows. It provides the ability to set parent child relationship between tasks to be executed and group a set of related tasks called stream. Each task in a stream is called as step. OBR currently uses only the “Control” functionality of ABC. A stream definition XML typically contains set of related step definitions and their relationships.

XML Structure

```
< JobStream >
< JobStreamMetaInfo >
  <  JobStreamMetaData />
</ JobStreamMetaInfo >
< JobStreamSteps >
< JobStreamStep >
      < JobStreamStepMetaInfo >
        <  JobStreamStepMetaData />
      </ JobStreamStepMetaInfo >
    </ JobStreamStep >
</ JobStreamSteps >
< JobStreamLinks >
< JobStreamLink />
</ JobStreamLinks >
</ JobStream >
```

XML Element: JobStream

JobStream element is the root element in the stream file.

Attribute	Mandatory	Description
dwid	Yes	Unique id for the stream. The value should start with content pack name followed by '@' symbol followed by a unique stream name
businessname	Yes	Label for stream
scheduleloadstarttime	No	Specifies the start time from when the ABC stream should be executed. This start time will only be used for the very first execution of the stream. Once the stream has executed and data on the same is available in the runtime tables, this attribute is not considered any further. Note: The value for this attribute must be in 5 minutes granularity and must be given only in 24hr format of hour and minutes separated by colon(:). For example 22:05 means stream will be executed at 10:05 PM. If this attribute is not specified then stream will be loaded on the first abcLoadNRun that launches after the stream import.
scheduleloadfrequency	No	Specifies the frequency (in minutes) stream load is to be repeated. For example if the frequency is set to 30 minutes, the stream will be loaded every 30 minutes. The default value is 5 minutes

Child Elements	Multiplicity	Description
JobStreamMetaInfo	0..1	Metadata for stream
JobStreamSteps	1..1	Group of stream steps
JobStreamLinks	0..1	Group of step relationships

XML Element: JobStreamMetaInfo

Child Elements	Multiplicity	Description
JobStreamMetaData	1..n	Stream metadata in the form of name-value pairs

XML Element: **JobStreamMetaData**

Attribute	Mandatory	Description
name	Yes	Metadata name
value	Yes	Metadata value

XML Element: JobStreamSteps

Child Elements	Multiplicity	Description
JobStreamStep	1..n	List of steps

XML Element: **JobStreamStep**

The JobStreamStep element represents a task in a stream.

Attribute	Mandatory	Description
dwid	Yes	Unique id of the step
businessname	Yes	Label for step
catalog	Yes	ABC catalog from which the executables are referred. The value is always set to "platform" catalog which has a set of mnemonics.
executableidentifier	Yes	The value represents a mnemonic which maps to an underlying platform binary. The possible values are COLLECT - Consolidates and moves the collected data from %PMDB_HOME\collect folder to %PMDB_HOME\collect. Takes type and category as argument. The type and category given as argument should be associated with a policy/rule in the collection policy XML so that the collected CSV file is processed by this step. The syntax is type: category TRANSFORM - Performs transformation on the collected data based on the transformation policy defined. Takes type and category as argument. The type and category combination mentioned as argument should match the type and category combination in the filename of output CSV created by parent step (COLLECT/ RECONCILE) so that the output CSV file created by the parent step is picked up by this TRANSFORM step for data transformation. The syntax is type: category RECONCILE - Performs reconciliation of data based on reconciliation rule defined. Takes dim type and dim category as argument. The type and category combination mentioned as argument should match the type and category combination in the filename of output CSV created by parent step (COLLECT/ TRANSFORM) so that the CSV file created by the parent step is picked up by this RECONCILE step for data reconciliation.

Attribute	Mandatory	Description
		The syntax is type: category STAGE - Loads the data from csv file to the stage table(s) based on stage rule. Takes stage rule file name as argument LOAD - Loads data from stage table to data warehouse table. Takes data warehouse table name as argument AGGREGATE - Performs data aggregation based the configuration defined in model xml and inserts summarized data to the aggregate table. Takes aggregate table name prefixed with schema name as argument (i.e., schema_name: aggregate_table_name) EXEC_PROC - Executes a SQL procedure. Takes procedure name as argument
arguments	No	Argument for the step with respect to executableidentifier specified.
maxexectime	Yes	Maximum execution time in minutes
maxretries	Yes	Maximum number of retries on failure

Child Elements	Multiplicity	Description
JobStreamStepMetaInfo	0..1	Metadata for step

XML Element: **JobStreamStepMetaInfo**

Child Elements	Multiplicity	Description
JobStreamStepMetaData	1..n	Stream metadata in the form of name-value pairs

XML Element: **JobStreamStepMetaData**

> **Note:** JobStreamStepMetaData will be added automatically by CDE for all steps except for step having "EXEC_PROC" as executableidentifier

Attribute	Mandatory	Description
name	Yes	Metadata name
value	Yes	Metadata value

XML Element: JobStreamLinks

Child Elements	Multiplicity	Description
JobStreamLink	1..n	Relationship definition between steps

XML Element: **JobStreamLink**

JobStreamLink element specifies the parent-child relationship between steps.

> **Note:** The steps without parent stepidentifier will be executed in parallel

Attribute	Mandatory	Description
stepidentifier	Yes	dwid of the step
parentstepidentifier	No	dwid of the parent step

ABC Stream Definition Examples

ABC streams in HPE OBR can be broadly classified in to two types

ETL (Extract, Transform, Load) stream

ETL stream contains steps related ETL process (Collect, transform, reconcile and stage). Please refer to the sample stream file ETL_CPU_Stream.xml in attachment section that defines ETL steps for CPU performance data from performance agent

Data warehouse stream

Data warehouse stream contains steps to move data from stage area to data warehouse tables and summarize the data further. Please refer to the sample stream file Datawarehouse_CPU_Stream.xml in attachment section that defines steps for moving CPU performance data from stage area to data warehouse tables and perform data summarization.

Strategy XML Definition

A strategy XML defines configuration for generating various artifacts from a model XML. It defines the following

- Strategy for generating the stage area interface and schema for stage area from model XML. The strategy can be defined at the schema level, fact table level, dimension table level or can be defined for a particular fact/dimension table. The strategy defined at a level overrides the strategy defined at a level higher than this level.
- List of tables for which stage tables should not be created.
- Downtime configuration for the fact tables (enable/disable downtime enrichment, dimension to be considered for downtime enrichment)
- List of dimension tables for which downtime tables should not be created.

> **Note:** A strategy XML should contain configuration for only on model XML. If there more than one model XMLs then strategy xml should be defined separately for each model XML.

XML Structure

```
< schema >
< factTables >
   < table />
</ factTables >
< dimensionTables >
   < table />
</ dimensionTables >
< stageExclusions >
< excludeTable />
</ stageExclusions >
< downtimeExclusions >
```

< dimensionTable />

</ downtimeExclusions >

</ schema >

XML Element: schema

schema element is the root element in strategy definition XML.

Attribute	Mandatory	Description
name	Yes	Name of the schema. The name should match with the schema name mentioned in the corresponding model XML
strategy	No	Global strategy for generating stage table schema for a given model. Allowed values are • normalize Generated stage table DDL for each fact/ dimension table will have a column for each column in the fact/dimension table and a column for each business key column from all the parent dimension tables. • Denormalize This strategy is not supported currently The default value is normalize
referenceLevel	No	Table reference level that should be considered for generating the schema for stage table. For example consider the following table reference hierarchy Table1 → DimTable1→DimTable2 I n the above example DimTable1 is at reference level 1 and DimTable2 is at reference level 2 from the table Table1. If the referenceLevel is set as “1” for Table1 then the generated stage table DDL for the table “Table1” will have a column for each column in table “Table1” and a column for each business key column from “DimTable1” and will not have a column for each business key column from “DimTable2” (“DimTable2” is ignored).

Child Elements	Multiplicity	Description
factTables	0..1	Strategy for fact tables
dimensionTables	0..1	Strategy for dimension tables

Child Elements	Multiplicity	Description
stageExclusions	0..1	List of fact/dimension tables to for which stage table shouldn't be created
downtimeExclusions	0..1	List dimension tables to for which downtime table shouldn't be created

XML Element: factTables

Attribute	Mandatory	Description
strategy	Yes	Global strategy for generating stage table schema for fact tables. Allowed values are • normalize Generated stage table DDL for each fact table will have a column for each column in the fact table and a column for each business key column from all the parent dimension tables. • Denormalize This strategy is not supported currently The default value is normalize
referenceLevel	No	Table reference level that should be considered for generating the schema for stage table for all fact tables. For example consider the following table reference hierarchy FactTable1 → DimTable1→DimTable2 I n the above example DimTable1 is at reference level 1 and DimTable2 is at reference level 2 from the fact table Fact Table1. If the referenceLevel is set as "1" for FactTable1 then the generated stage table DDL for the fact table "FactTable1" will have a column for each column in fact table "FactTable1" and a column for each business key column from "DimTable1" and will not have a column for each business key column from "DimTable2" ("DimTable2" is ignored).
enrichDowntime	No	Boolean value indicating whether downtime enrichment should be enabled/disabled for all the fact tables. By default downtime enrichment is enabled for all the fact tables.

Child Elements	Multiplicity	Description
table	0..n	List of fact tables and its strategy configuration.

XML Element: **table**

Attribute	Mandatory	Description
name	Yes	Name of the fact table. The name should match with the fact table name defined in model XML. Note: Name is case sensitive
strategy	No	Strategy for generating stage table schema for the fact table. Allowed values are • normalize Generated stage table DDL for the fact table will have a column for each column in the fact table and a column for each business key column from all the parent dimension tables. • Denormalize This strategy is not supported currently The default value is normalize
referenceLevel	No	Table reference level that should be considered for generating the schema for stage table for the fact table. For example consider the following table reference hierarchy FactTable1 → DimTable1→DimTable2 I n the above example DimTable1 is at reference level 1 and DimTable2 is at reference level 2 from the fact table Fact Table1. If the referenceLevel is set as “1” for FactTable1 then the generated stage table DDL for the fact table “FactTable1” will have a column for each column in fact table “FactTable1” and a column for each business key column from “DimTable1” and will not have a column for each business key column from “DimTable2” (“DimTable2” is ignored).
enrichDowntime	No	Boolean value indicating whether downtime enrichment should be enabled/disabled for the fact table. The default value is true. Note: This value overrides the enrichDowntime value defined by its parent element factTables
downtimeDimension	No	Dimension table whose downtime should be considered for enriching fact table with downtime information. The default value is the primary dimension table (dimension table that dsi_key_id_ column in fact table refers to) for the fact table.

Attribute	Mandatory	Description
		Note: Fact table must reference the dimension table

XML Element: dimensionTables

Attribute	Mandatory	Description
strategy	Yes	Global strategy for generating stage table schema for dimension tables. Allowed values are • normalize Generated stage table DDL for each dimension table will have a column for each column in the dimension table and a column for each business key column from all the parent dimension tables. • Denormalize This strategy is not supported currently The default value is normalize
referenceLevel	No	Table reference level that should be considered for generating the schema for stage table for all dimension tables. For example consider the following table reference hierarchy DimTable1 → DimTable2→DimTable3 I n the above example DimTable2 is at reference level 1 and DimTable3 is at reference level 2 from the dimension table DimTable1. If the referenceLevel is set as “1” then the generated stage table DDL for the dimension table “DimTable1” will have a column for each column in dimension table “DimTable1” and a column for each business key column from “DimTable2” and will not have a column for each business key column from “DimTable3” (“DimTable3” is ignored).

Child Elements	Multiplicity	Description
table	0..n	List of dimension tables and its strategy configuration.

XML Element: **table**

Attribute	Mandatory	Description
name	Yes	Name of the dimension table. The name should match with the fact table name defined in model XML. Note: Name is case sensitive
strategy	No	Global strategy for generating stage table schema for dimension tables. Allowed values are • normalize Generated stage table DDL for the dimension table will have a column for each column in the dimension table and a column for each business key column from all the parent dimension tables. • Denormalize This strategy is not supported currently The default value is normalize
referenceLevel	No	Table reference level that should be considered for generating the schema for stage table for the dimension table. For example consider the following table reference hierarchy DimTable1 → DimTable2→DimTable3 I n the above example DimTable2 is at reference level 1 and DimTable3 is at reference level 2 from the dimension table DimTable1. If the referenceLevel is set as “1” for DimTable1 then the generated stage table DDL for the dimension table “DimTable1” will have a column for each column in dimension table “DimTable1” and a column for each business key column from “DimTable2” and will not have a column for each business key column from “DimTable3” (“DimTable3” is ignored).
groupBridge	No	Boolean value indicating that the dimension table is a group bridge table. The default value is “false”
locationBridge	No	Boolean value indicating that the dimension table is a location bridge table. The default value is “false”

XML Element: **stageExclusions**

Child Elements	Multiplicity	Description
excludeTable	1..n	List of dimension/fact tables to be excluded from stage table creation

XML Element: **excludeTable**

Attribute	Mandatory	Description
name	Yes	Name of the of dimension/fact table. The name should match with the fact/dimension table name defined in model XML.

XML Element: **downtimeExclusions**

Child Elements	Multiplicity	Description
dimensionTable	1..n	List of dimension tables that shouldn't be considered for downtime table creation.

XML Element: **dimensionTable**

Attribute	Mandatory	Description
name	Yes	Name of the dimension table that shouldn't be considered for downtime table creation. The name should match with the dimension table name defined in model XML.

Strategy Definition Example

Sample strategy XML file

Please refer to the sample strategy xml strategy.xml and its corresponding model xml model.xml in attachment section that defines strategy for fact tables and dimension tables, stage exclusions and downtime exclusions.

Chapter 9: Type and category attributes in ETL policies

All OBR ETL policies define two attributes namely type and category to link a rule/policy to a CSV file which contains these two attributes as part of its name.

Type and category attributes in collection policies

OBRcollection policies define two unique attributes for each collected CSV file. These two attributes are used in the file name of the collected CSV and are used to identify the file for further processing like transformation, reconciliation and staging. The rules/policy for transformation, reconciliation and stage should define the appropriate type and category so that the rules are applied to the right CSV files based on type and category attribute. The collected CSV file will have the following file name pattern

**category _0_type_0_*.csv*

> **Note:** If type and category attributes are not defined explicitly in collection policy, default values will be assumed based on the type of the collection policy.

Type and category in RTSM collection policy

In RTSM collection policy type and category is defined for each citype definition as each citype definition will result in a CSV file. If type and category attributes are not defined for a citype definition then default values are assumed. The default value for type is citype name and default value for category is view name. For example consider oracle CI type in ORA_Deployment view

```
<view name="ORA_Deployment">
    <citype name="oracle">
        .....
    </citype>
</view>
```

In the above example no type and category attribute is defined and hence the default value for type is “oracle” and default value for category is “ORA_Deployment”. The collected CSV file will have the name like

**ORA_Deployment_0_oracle_0_*.csv*

citype definition can have more than one user defined type and category combination(aliases) like below

```
<view name="ORA_Deployment">
    <citype name="oracle">
      <aliassource>
          <aliastarget  type="oracle" category="ORA_Deployment"/>
          <aliastarget  type="database" category="ORA_Deployment"/>
       </aliassource>
       .....
    </citype>
</view>
```

In the above example two type and category combinations are defined and hence there will be two CSV files created with the names like

**ORA_Deployment_0_oracle_0_*.csv*

**ORA_Deployment_0_database_0_*.csv*

Type and category in OM collection policy

In OM collection policy type and category is defined for each OM collection rule involving a PA class and data source as each rule definition will result in a CSV file. If type and category attributes are not defined for a rule definition then default values are assumed. The default value for type is the value of citype attribute and default value for category is OM collection name. For example consider OM collection rule for Oracle database

```
<sncollection name="ORA_Deployment" mappedby="DBSPI_ORA_GRAPH">
    <rule citype="oracle" class="DBSPI_ORA_GRAPH" datasource="DBSPI_ORA_GRAPH">
     .....
    </rule>
</sncollection>
```

In the above example no type and category attribute is defined and hence the default value for type is “oracle” and default value for category is “ORA_Deployment”. The collected CSV file will have the name like

**ORA_Deployment_0_oracle_0_*.csv*

Rule definition can have more than one user defined type and category combination (aliases) like below

```
<sncollection name="ORA_Deployment" mappedby="DBSPI_ORA_GRAPH">
    <rule citype="oracle" class="DBSPI_ORA_GRAPH" datasource="DBSPI_ORA_GRAPH">
       <aliassource>
          <aliastarget  type="oracle" category="ORA_Deployment"/>
          <aliastarget  type="database" category="ORA_Deployment"/>
       </aliassource>
       .....
    </rule>
  </sncollection>
```

In the above example two type and category combinations are defined and hence there will be two CSV files created with the names like

**ORA_Deployment_0_oracle_0_*.csv*

**ORA_Deployment_0_database_0_*.csv*

Type and category in OA collection policy

In OA collection policy type and category is defined for each OA Class definition as each Class definition will result in a CSV file. If type and category attributes are not defined for a Class definition then default values are assumed. The default value for type is Class name and default value for category is data source name. For example consider Oracle SPI data source

```
<datasource name="DBSPI_ORA_REPORT">
    <class name="DBSPI_ORA_REPORT" summarized="true">
        .....
    </class>
</datasource>
```

In the above example no type and category attribute is defined and hence the default value for type is “DBSPI_ORA_REPORT” and default value for category is “DBSPI_ORA_REPORT”. The collected CSV file will have the name like

*DBSPI_ORA_REPORT_0_ DBSPI_ORA_REPORT_0_*.csv

Class definition can have more than one user defined type and category combination (aliases) like below

```
<datasource name="DBSPI_ORA_REPORT">
    <class name="DBSPI_ORA_REPORT" summarized="true">
        <aliassource>
            <aliastarget  type="InstanceAvailability" category="DBSPI_ORA_REPORT"/>
            <aliastarget  type="InstanceSpaceutilization" category="DBSPI_ORA_REPORT"/>
        </aliassource>
         .......
    </class>
</datasource>
```

In the above example two type and category combinations are defined and hence there will be two CSV files created with the names like

*DBSPI_ORA_REPORT_0_ InstanceAvailability_0_*.csv

*DBSPI_ORA_REPORT_0_ InstanceSpaceutilization_0_*.csv

Type and category in DB collection policy

In DB collection policy type and category is defined for each join query definition as each join query definition will result in a CSV file. For example consider Oracle SPI data source

```
<joinquery type="Interface_Fact" category="Network">
    <content>SELECT * FROM nps_f_hour_InterfaceMetrics</content>
</joinquery>
```

For the above example the collected CSV file will have the name like

**Network_0_ Interface_Fact_0_*.csv*

Type and category in Transformation policy

In transformation policy two sets of type and category namely source type, source category and target type, target category are mentioned for each record set. Source type and source category is used to identify the source/input CSV file for applying the transformation rules whereas target type and target category will be used in output CSV file name.

The source type and source category should be defined based on the type and category of output CSV file generated by the ETL process (collection/reconciliation) that runs before the transformation process.

For example consider the transformation rule below

```
<recordSet name="InstanceSpaceutilization" condition="METRICID IN (212.0)"
           source_type="InstanceSpaceutilization" source_category="DBSPI_ORA_REPORT"
           target_type="InstanceSpaceUtilization" target_category="DBSPI_ORA_REPORT"
           doPivot="true">
    ......
</recordSet>
```

In this example the source type is “InstanceSpaceutilization” and source category is “DBSPI_ORA_REPORT”. Hence the rule applies to source CSV file (generated by collection/reconciliation process) whose name has this type and category

(DBSPI_ORA_REPORT_0_ InstanceSpaceutilization_0_*.csv)*

Type and category in Reconciliation rule

A reconciliation policy doesn’t define type attribute instead it defines “citype” attribute to identify the CSV file for building reconcile registry and “paclass” attribute and “category” attribute to identify the fact CSV for reconciliation.

The “paclass” attribute and “category” attribute should be defined based on the type and category of output CSV file generated by the ETL process (collection/transformation) that runs before the reconciliation process. The “citype” attribute should be defined based on the citype definition in RTSM collection policy. Also, the output CSV will have the same names as the source CSV

For example consider the reconciliation rule below

```
<rule name="Reconciliation rule for InstanceSpaceutilization"
      ciType="oracle" paClass="InstanceSpaceutilization" category="DBSPI_ORA_REPORT">
 ....
</rule>
```

This rule is applied to CSV file with the filename pattern like *_0_oracle_0_*.csv (generated by RTSM collection process) for registry building and CSV file with the filename pattern like

**DBSPI_ORA_REPORT_0_InstanceSpaceutilization_0_*.csv* (generated by collection/transformation process) for data reconciliation.

Type and category in Stage rule

In stage rule file one or more type and category sets are defined for each target indicating the source CSV files from which the data is loaded to the corresponding stage table.

For example consider the stage rule below

```
<stageRule xmlns="http://www.hp.com/SHR/Stage/v1.0" name="DB Oracle Instance Space Utilization">
    <targets>
        <target name="DB Oracle Instance Space Utilization">
            <sources>
                <source alias="ISU" type="InstanceSpaceUtilization" category="DBSPI_ORA_REPORT"
                        schemaName="CoreDatabaseOracle"/>
            </sources>
            .........
        </target>
  </targets>
</stageRule>
```

This rule is applied to CSV file with the filename pattern like **DBSPI_ORA_REPORT_0_ InstanceSpaceutilization_0_*.csv* (generated by collection/transformation/reconciliation process) to load the data from CSV file to the stage table

CSV File Flow

OA collection policy

```
<datasource name="DBSPI_ORA_REPORT">
    <class name="DBSPI_ORA_REPORT" summarized="true">
        .....
    </class>
</datasource>
```

**DBSPI_ORA_REPORT_0_ InstanceSpaceutilization_0_*.csv*

Transformation rule

```
<recordSet name="InstanceSpaceutilization" condition="METRICID IN (212.0)"
           source_type="InstanceSpaceutilization" source_category="DBSPI_ORA_REPORT"
           target_type="InstanceSpaceUtilization" target_category="DBSPI_ORA_REPORT"
           doPivot="true">
    ......
</recordSet>
```

**DBSPI_ORA_REPORT_0_ InstanceSpaceutilization_0_*.csv*

Reconciliation rule

```
<rule name="Reconciliation rule for InstanceSpaceutilization"
      ciType="oracle" paClass="InstanceSpaceutilization" category="DBSPI_ORA_REPORT">
 ....
</rule>
```

DBSPI_ORA_REPORT_0_ InstanceSpaceutilization_0_*.csv

Stage rule

```
<stageRule xmlns="http://www.hp.com/SHR/Stage/v1.0" name="DB Oracle Instance Space Utilization">
    <targets>
        <target name="DB Oracle Instance Space Utilization">
            <sources>
                <source alias="ISU" type="InstanceSpaceUtilization" category="DBSPI_ORA_REPORT"
                        schemaName="CoreDatabaseOracle"/>
            </sources>
            .........
        </target>
  </targets>
</stageRule>
```

Chapter 10: CDE-Simplified Reference

The following section lists the CSV and properties files for content development using the CDE-simplified method.

CSV Files

This section provides detailed information about the CSV files used for content development.

Model Mapper CSV File

Model Mapper CSV file is an intermediate document for the content developer to edit the required column names before generating the end-to-end content pack for Domain and ETL. Name of the file will be <*CSV policy's name attribute*>_MappingFile_<*timestamp*>.csv.

The following table lists the fields in the Model Mapper CSV file:

Field	Description
Source File	Name of the source CSV file
Source column name	Name of the source column
Destination Table	Name of the data warehouse table
Destination Column	Name of the data warehouse column
Caption	Caption for the column
Business Key	Specifies if this column is a business key or not (TRUE/FALSE)
data type	Type of the data in the column
Size	Size of the column
primary dimension	Specifies whether or on to specify this column as a primary dimension (TRUE/FALSE)
aggregation	Type of aggregation

Properties Files

dbconfig.properties

You can modify the following items in the dbconfig.properties file:

- `shrdb.db.enable` - Set the value to True to enable the database
- `shrdb.hostname` - Name of the system where Vertica database is hosted
- `shrdb.portnumber` - Port number =21424
- `shrdb.enginename`- Database engine name
- `shrdb.username` - Your Vertica database user name
- `shrdb.password` - Database password
- `shrdb.dbfilename`- Database file name
- `column.space.replace`- Set the value to True if you want the spaces in the source column names to be replaced by an underscore (_)

Part IV: Appendix

Appendix A: Filters and functions in OBR ETL policies

OBR provides a set of filters and functions that can be used in ETL policies like RTSM collection policy, OM collection policy, Reconciliation policy and transformation policy to cleanse/transform the collected data. Here's the list of filters and functions supported in OBR.

Filters

The filter conditions currently supported are

<SOURCE_COLUMN> NOT IN (VALUE1, VALUE2 …) – This condition checks if the value of the SOURCE_COLUMN is contained by the list of values that are provided. If yes, the condition returns false and the row is rejected from the output dataset else accepted.

<SOURCE_COLUMN> IN (VALUE1, VALUE2 …) - This condition checks if the value of the SOURCE_COLUMN is contained by the list of values that are provided. If yes, the condition returns true and the row is accepted into the output dataset else rejected.

<SOURCE_COLUMN> LIKE <VALUE_PATTERN> - This condition checks if the value of the SOURCE_COLUMN matches the VALUE_PATTERN that is specified. If yes, the condition returns true and the row is accepted into the output dataset else rejected.

<SOURCE_COLUMN> = <VALUE > - This condition checks if the value of the SOURCE_COLUMN is equal to the VALUE specified. If yes, the condition returns true and the row is accepted into the output dataset else rejected.

<SOURCE_COLUMN> != <VALUE > - This condition checks if the value of the SOURCE_ COLUMN is not equal to the VALUE specified. If yes, the condition returns true and the row is accepted into the output dataset else rejected.

<SOURCE_COLUMN> GREATER THAN <VALUE > - This condition checks if the value of the SOURCE_COLUMN is greater than the VALUE specified (supports both Numeric and String comparison). If yes, the condition returns true and the row is accepted into the output dataset else rejected

<SOURCE_COLUMN> GREATER THAN EQUAL TO <VALUE > - This condition checks if the value of the SOURCE_COLUMN is greater than or equal to the VALUE specified (supports both

Numeric and String comparison). If yes, the condition returns true and the row is accepted into the output dataset else rejected

<SOURCE_COLUMN> LESSER THAN <VALUE > - This condition checks if the value of the SOURCE_COLUMN is lesser than the VALUE specified (supports both Numeric and String comparison). If yes, the condition returns true and the row is accepted into the output dataset else rejected

<SOURCE_COLUMN> LESSER THAN EQUAL TO <VALUE > - This condition checks if the value of the SOURCE_COLUMN is lesser than or equal to the VALUE specified (supports both Numeric and String comparison). If yes, the condition returns true and the row is accepted into the output dataset else rejected

Filter conditions can be combined using ***AND*** and/or ***OR*** operators. Multiple filter conditions can be grouped by enclosing them within the square braces ***'[] '***.

A sample complex filter condition is provided below –

[[BYLS_LS_ROLE=RESPOOL AND BYLS_LS_HOST_HOSTNAME NOT IN (NA)] OR BYCPU_ID_UTIL GREATER THAN EQUAL TO 100]

Functions

The following functions/operators can be used with the conditions.

OPR_TOLOWER – Usage : OPR_TOLOWER(<column name>)

This function converts the value of the specified column from the csv to lower case

OPR_TOUPPER – Usage : OPR_TOUPPER(<column name>)

This function converts the value of the specified column from the csv to upper case

OPR_STRINGLENGTH – Usage : OPR_STRINGLENGTH(<column name>)

This function returns the length of the string value of the specified column from the csv

OPR_SUBSTRING – Usage : OPR_SUBSTRING(<column name>,<start index>) or OPR_SUBSTRING(<column name>,<start index>,<end index>)

This function returns a substring of the value of the specified column from the csv. The function takes either two or three arguments. The first argument is the column name, second is the starting index for the substring and the third one if specified gives the ending index for substring. The substring function then returns a substring for the given column name from the starting index (including it) and till the ending index if specified, else, the entire string till the end is returned.

OPR_APPEND – Usage : OPR_APPEND(<column1 name>,<column2 name>,<delimiter>)

This function appends the values of the columns specified separated by the delimiter – returns <value of column1><delimiter><value of column2>. Also, if the delimiter for the strings appended has to be "," (COMMA) or " " (SPACE) then the <delimiter> must be specified as OPR_FIXEDVALUE(COMMA) or OPR_FIXEDVALUE(SPACE) respectively.

OPR_SUM – Usage : OPR_SUM(<column1 name>,<column2 name>,<column3 name>)

The function finds the sum of the values from the csv for the specified columns – returns <value of column1> + <value of column2> + <value of column3>

OPR_DIFFERENCE – Usage : OPR_DIFFERENCE(<column1 name>,<column2 name>,<column3 name>)

The function finds the difference between the values for the specified columns from the csv – returns <value of column1> - <value of column2> - <value of column3>

OPR_PRODUCT – Usage : OPR_PRODUCT(<column1 name>,<column2 name>,<column3 name>)

The function calculates the product of the values of the specified columns from the csv – returns <value of column1>*<value of column2>*<value of column3>

OPR_DIVIDE – Usage: OPR_DIVIDE(<column1 name>,<column2 name>,<column3 name>)

The function divides values of the columns specified and then returns the result – returns <value of column1>/<value of column2>/<value of column3>

OPR_PERCENTILE – Usage OPR_PERCENTILE(<column1 name>,<column2 name>)

The function finds the percentage for the values of the given two column names – returns <value of column1>/<value of column2>*100

OPR_AVERAGE – Usage OPR_AVERAGE(<column1 name>,<column2 name>,<column3 name>)

The function returns the average of the values specified by the columns in the function – returns (<value of column1>+<value of column2>+<value of column3>)/<count of columns>

OPR_MATHFLOOR – Usage OPR_MATHFLOOR(<column name>)

The function floors the value of the column specified – returns Mathematical floor value for <value of column>

OPR_MATHCEIL – Usage OPR_MATHCEIL(<column name>)

The function ceils the value of the column specified – returns the Mathematical ceil value for <value of column>

OPR_FIXEDVALUE – Usage OPR_FIXEDVALUE(<string>)

The function returns the string specified in the function as is and does not treat it as a column in the csv – return <string>

OPR_MAXVALUE – Usage OPR_MAXVALUE(<column1 name>,<column2 name>,<column3 name>)

The function returns the maximum of the value of the columns specified in the function – returns maximum of <value of column1>, <value of column2> and <value of column3> and so on

OPR_MINVALUE – Usage OPR_MINVALUE(<column1 name>,<column2 name>,<column3 name>…)

The function returns the minimum of the value of the columns specified in the function – returns minimum of <value of column1> and <value of column2> and <value of column3> and so on

OPR_STRINGSPLIT – Usage OPR_STRINGSPLIT(<column name>,<String expression to be used as match for split>,<index of split to be returned>)

The function uses the string expression specified second to split the value for the column name specified and then returns the value from the array of strings after the split at the specified index. Please note that this index starts from 0. Also, if the split has to happen based on "," (COMMA) or " " (SPACE) then the expression must be specified as OPR_FIXEDVALUE(COMMA) or OPR_ FIXEDVALUE(SPACE) respectively.

The functions specified above can be nested. For example the nested function below is valid

OPR_SUBSTRING(OPR_TOLOWER(BYLS_LS_HOSTNAME),0,OPR_DIFFERENCE(OPR_ STRINGLENGTH(HOST_HOSTNAME),1

In addition to this, if-then-else expression can also be used as a condition string for individual records but not at the recordset level. The conditional is of the form <condition string>?<operator if true>:<operator if false> . Also note that only one conditional is valid for a record. The condition string itself of the conditional can be a complex including operators. Sample expression string –

[OPR_TOUPPER(BYLS_LS_HOSTNAME) LIKE *IND.HP.COM]?OPR_FIXEDVALUE (INDIA):OPR_TOUPPER(HOST_DNSNAME)

The above expression evaluates to INDIA if the value for BYLS_LS_HOSTNAME matches pattern *.IND.HP.COM else evaluates to value of HOST_DNSNAME in upper case

Supported Functions in Aggregate and Forecast Elements

The following table lists the supported functions and provides the description of each:

Function Name	Description	Supported by Aggregate	Supported by Forecast
avg	Average	Yes	Yes
min	Minimum	Yes	Yes
max	Maximum	Yes	Yes
cnt	Count	Yes	Yes
tot	Total	Yes	Yes
med	Median	Yes	Yes
std	Standard deviation	Yes	Yes
slope	Slope	Yes	Yes
wav	Weighted Average	Yes	Yes
perXX	Percentile (where XX is any value below 100)	Yes	Yes
lst	Last value	Yes	No
nlst	Last value not null	Yes	No
fXX	Forecast (where XX is any integer)	No	Yes
DTT[XX]	Days to threshold (where XX is any integer)	No	Yes

Limitations

The following are some of the limitations on specifying filters

1. The functions specified in this document cannot be used for comparator values in case of using conditions IN , NOT IN and LIKE
2. Only one if expression string allowed per condition string

Appendix B: Creating a Data Source for ETL Component

To create a sample PostgreSQL database, HPE OBR provides the following files and scripts:

Files / Scripts Provided by OBR	Location of File / Script

The following `.csv` files that are copied into the database tables by using scripts:
`RetailPOS_Product.csv`
`RetailPOS_Promotion.csv`
`RetailPOS_Sales.csv`
`RetailPOS_Store.csv`
	%CDE_HOME%\samples\RetailPOS_Demo_Content_Pack\RetailPOS_DB_Creation_Scripts\RetailPOS_CSV

`RetailPOS_CreateDatabase.sql` This SQL script creates a PostgreSQL database named RetailPOS for the user named retail_admin.	*%CDE_HOME%*\samples\RetailPOS_Demo_Content_Pack\RetailPOS_DB_Creation_Scripts
`RetailPOS_CreateTables.sql` This SQL script creates tables in the RetailPOS database.	*%CDE_HOME%*\samples\RetailPOS_Demo_Content_Pack\RetailPOS_DB_Creation_Scripts
`RetailPOS_PopulateTables.sql` This SQL script copies the `.csv` files from the location `%CDE_`	*%CDE_HOME%*\samples\RetailPOS_Demo_Content_Pack\RetailPOS_DB_Creation_Scripts

Files / Scripts Provided by OBR	Location of File / Script
HOME%\samples\RetailPOS_Demo_Content_Pack\RetailPOS_DB_Creation_Scripts\RetailPOS_CSV to the database tables.	

Prerequisites: Perform the following tasks before you start creating the PostgreSQL database:

Download and install PostgreSQL software from **http://www.postgresql.org/.** You can install PostgreSQL on any system which can be different than the system on which HPE OBR is installed.

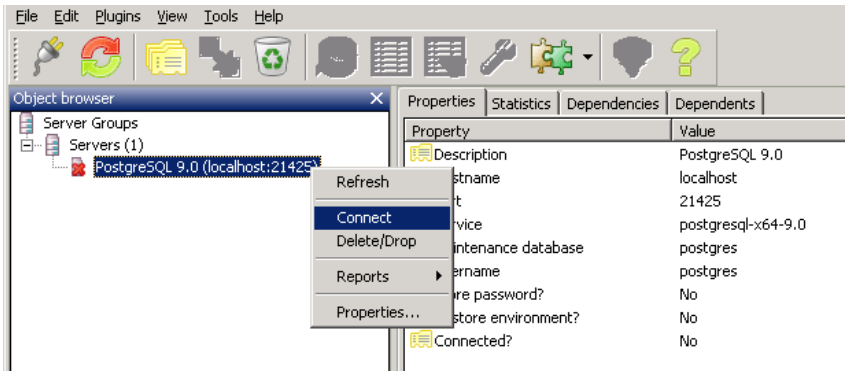
Copy the following files to the C:\ drive of the system where you installed PostgreSQL.

- RetailPOS_CSV
- RetailPOS_CreateDatabase.sql
- RetailPOS_CreateTables.sql
- RetailPOS_PopulateTables.sql

Create a PostgreSQL Database

To create the database named RetailPOS, follow these steps:

1. Logon to the system where you installed PostgreSQL as Administrator.
2. Start the PostgreSQL **pgAdmin III** program.
3. Connect to default user **postgre** with the password you configured.



The Object browser pane shows the databases available for the postgre user.

4. On the SQL Query Editor window click **File** ->**Open**. Browse to the location on C:\ drive where you copied the script `RetailPOS_CreateDatabase.sql` and click **Open**.
5. Click **Execute pgScript** to run the `RetailPOS_CreateDatabase.sql` script.

   The script creates the retailpos database and retail_admin user as the database owner.
6. Close the SQL Query Editor and click **Refresh**.

   The retailpos database with user retail_admin appear on the Object browser window.

Create Database Tables

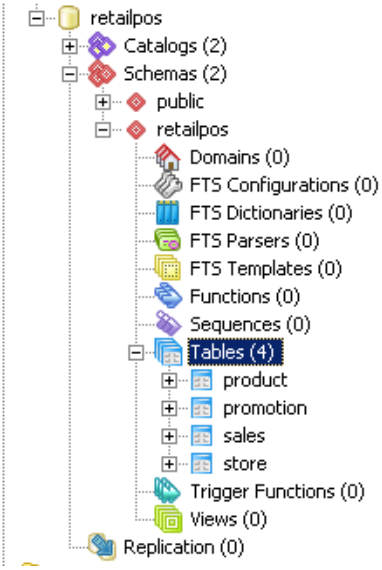
To create tables within the retailpos database, follow these steps:

1. On the Object browser window, select **retailpos** from the list of databases and open the SQL Query Editor.
2. On the SQL Query Editor window click **File** ->**Open**. Browse to the location on C:\ drive where you copied the script `RetailPOS_CreateTables.sql` and click **Open**.
3. Click **Execute pgScript** to run the `RetailsPOS_CreateTables.sql` script.

   The following tables are created as shown in the following figure:

   - retailpos.product
   - retailpos.store
   - retailpos.promotion

- retailpos.sales



Insert Data into the Database

To insert data from the `.csv` files into the tables, follow these steps:

1. On the SQL Query Editor window click **File** ->**Open**. Browse to the location on C:\ drive where you copied the script `RetailPOS_PopulateTables.sql` and click **Open**.
2. Click **Execute pgScript** to run the `RetailPOS_PopulateTables.sql` script.

   The script inserts the `.csv` files from `RetailPOS_CSV` to the database tables.

Frequently Asked Questions

1. **What can I do if I get the errorL *'FIND' is not recognized as an internal or external command, operable program or batch file.* while running the `setenv.bat`?**

   a. Open the `setenv.bat` file.

   b. Search for the following line:

      FOR /F "tokens=3" %%A IN ('JAVA -version 2^>^&1 ^| **FIND** /I "java version"')

   c. Replace the FIND in upper case to lower case (find) as shown here:

      FOR /F "tokens=3" %%A IN ('JAVA -version 2^>^&1 ^| **find** /I "java version"')

Glossary

C

CDE
CDE is a set of tools provided by OBR for development of content packs.

Collection policy
A Collection policy is written in XML to define the metrics to be collected by a collector program from the specified data source.

Content pack
Content packs are data marts deployed on the OBR performance management databaseplatform. Content packs enable the platform to collect, store, process, and report the data. Acontent pack has three components – Domain, ETL, and Reports.

D

Data Model
Data model is a schema diagram that illustrates the relationship between dimension tables (that have attributes) and fact tables (that have measures).

Domain Component
Domain component of a content pack defines the data model of the Domain you are reporting on along with the logic to perform processing on the data. It is independent of the data source you collect data from.

E

ETL Component
ETL component of a content pack is data source dependant; it defines the collection of data from the specified data source

L

Loading
The process of loading data from the stage tables to the data warehouse tables.

R

Reconciliation
Data reconciliation is the technique of associating fact data to corresponding dimension data.

Reports Component
The Reports component contains the SAP BusinessObjects Web Intelligence reports and universes.

S

Staging
Data staging is the process of moving the collected, transformed, and reconciled data into the staging tables.

T

Transformation
Data transformation is the optional step of cleaning the collected data according to business requirements.

W

Workflow Streams
Workflow streams in content packs are used to define and control the movement of data from one step to another.

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 Content Development Guide (Operations Bridge Reporter 10.10)

Just 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 docfeedback@hpe.com.

We appreciate your feedback!