Content Development - Reference Guide

Software Version: 9.31

Table of Contents

Overview

This document provides a detailed explanation of the syntax of source XML files that a content developer has to author in order to build a content pack using CDE.

For information on SHR BusinessObjects Universes shipped out of the box, see section "Validating your Installation" in the *HP Service Health Reporter Installation and Configuration Guide*.

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 SHR, 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) or
$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) or
$*CDE_HOME*/cplib/SystemManagement/CoreSystemManagement.ap/doc (Linux)

For information on SHR data warehouse bus matrix, see the SHR 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.

Prerequisites

This guide assumes that the reader has prior understanding of the following:

Prerequisite	Reference Documentation
SHR Content Development concepts	Content Development - Getting Started Guide.
Data Warehouse concepts	You can find resources related to data warehouse concepts on the world wide web.
XML concepts and how to create XML documents	You can find resources related to XML concepts and examples on the world wide web.
SQL	You can find resources related to SQL on the world wide web.
Understanding of HP Run-time Service Model (RTSM) and HP Configuration Management Database (CMDB)	HP BSM 9.2 Modeling guide, RTSM Administration Guide.

Definition of terms

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

1 Model XML definition

A model xml defines the data warehouse model for a particular domain.

1.1 Model XML Structure

```
<schema>
 <relational>
  <dimensionTable>
     <column/>
   </dimensionTable>

  <bridgeTable>
     <column/>
   </bridgeTable>

  <factTable>
     <column/>
   </factTable>
 </relational>
 <logical>
   <dimension>
     <memberGroup>
       <member/>
       <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/>
            <calculatedMeasure/>
            <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 another schema's relational and logical elements.

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.
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 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
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}
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 indentified 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 (Groupby 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: 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" 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).
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)

1.2 Model XML to BO Universe component mapping

The model xml serves as input for generating BO universe. CDE framework in SHR 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.

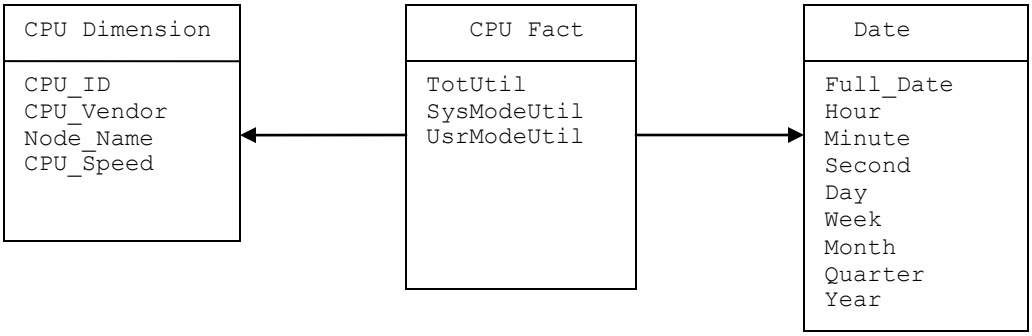
Model XML Element	BO Universe component
cube	Class containing measure objects
dimension	Class containing dimension objects
hierarchy	Hierarchy
levelCaption attribute in level element	Dimension Object
member elements defined under level element	Detailed objects under a Dimension object
memberGroup element 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.

1.3 Model definition Examples

1.3.1 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 definition 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

1.3.2 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>
```

1.3.3 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

2 RTSM Collection Policy XML Definition

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

2.1 XML Structure

```
<etldefinition>
 <views>
  <view>
     <citype>
          <aliassource>
             <aliastarget/>
          </aliassource>
          <ciattribute/>
          <ciref>
             <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 SHR) • datasource – CIs from this citype are considered only as datasource and not dimension (only entry is made in to data source dictionary table in SHR 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 SHR 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 SHR 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

2.2 RTSM collection policy Examples

2.2.1 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.

3 OM Collection Policy XML Definition

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

3.1 XML Structure

```
<etldefinition>
  <sncollection>
     <rule>
       <aliassource>
            <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
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

		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 SHR) • 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 SHR 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 SHR 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

3.2 OM Collection policy Examples

3.2.1 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.

4 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.

4.1 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_name	Yes	Domain for which the collection is defined e.g. System 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:* *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.

4.2 OA collection policy Examples

4.2.1 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

5 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.

5.1 XML Structure

```
<etldefinition>
<queries>
   <query>
      <tables>
      <table>
            <columns>
               <column/>
            </columns>
            <condition/>
      </table>
  </tables>
  <genericsql>
      <statement>
      </statement>
  </genericsql>
  <joinqueries>
      <joinquery>
        <aliassource>
          <aliastarget/>
        </aliassource>
        <content></content>
      </joinquery>
  </joinqueries>
  </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 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: SHR 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**
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, etc)

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 SHR 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 SHR database. In this case joinquery must be defined to dump the data from temporary table in SHR 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 SHR 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 SHR 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 initialhistory is set to 48 then the data will be collected from 48 hours behind the current time when the collection happens for the first time. Note: SHR 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 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 SHR 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, etc).
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.

5.2 DB Collection Policy Examples

5.2.1 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.

6 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.

6.1 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 recordset. 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.

Note:

- *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

6.2 Transformation Policy Examples

6.2.1 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 SHR 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

6.2.2 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

6.2.3 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,:)"/>
```

HOSTNAME, INSTANCE ⟹ Name
shr1.ind.hp.com, BAT92 ⟹ shr1.ind.hp.com:BAT92

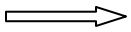
OPR_STRINGSPLIT

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)"/>
```

HOSTNAME,
shr1.ind.hp.com



DisplayName
shr1

7 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).

7.1 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	

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.
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>}"* *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="${<TARGET_COLUMN>}"* *Example:* *Instance* *<targetcolumn name="Instance" suffix="ORA_"/>* → Instance1 ORA_Instance1
suffix		The suffix string that needs to be suffixed to the value of the target column for building the registry lookup key. The suffix can be a string or a substitution parameter for a target column, which substitutes the value of the given column as suffix during registry building. The substitution parameter will be of form *suffix="${<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.

7.2 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.

1. 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

```
<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>
```

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

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

2. 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

Add the CIID column to the collected fact data

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_REPORT	4/18/2012 5:30	215	vm8	1	211408823a8cc17eca52174549f82f80
SHR2.IND.HP.COM	DBSPI_ORA_REPORT	DBSPI_ORA_REPORT	4/18/2012 5:35	209	vm8	1	211408823a8cc17eca52174549f82f81

8 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).

8.1 XML Structure

```
<stageRule>
  <targets>
      <target>
            <sources>
               <source/>
            </sources>
            <columnMap>
               <mapping/>
            </columnMap>
            <relational>
               <relation/>
            </relational>
            <keyColumns>
                  <key/>
            </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: **relational**

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.

8.2 Stage Rule Examples

8.2.1 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

8.2.2 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.

9 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. SHR currently uses only the "Control" functionality of ABC. A stream definition XML typically contains set of related step definitions and their relationships.

9.1 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. 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

9.2 ABC Stream Definition Examples

ABC streams in SHR can be broadly classified in to two types

9.2.1 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

9.2.2 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.

10 Type and category attributes in ETL policies

All SHR 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.

10.1 Type and category attributes in collection policies

SHR collection 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.

10.1.1 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*

10.1.2 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*

10.1.3 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*

10.1.4 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*

10.2 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)*

10.3 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.*

10.4 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*

10.5 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>
```

11 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 shouldn't 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 shouldn't 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.

11.1 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
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. 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.

11.2 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.

Appendix A: Filters and functions in SHR ETL policies

SHR 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 SHR

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