Oracle Financial Services Analytical Applications Infrastructure

Administration Guide

DOCUMENT CONTROL

Version Number	Revision Date	Changes Done
Draft	Created: July 2012	Captured the administration configurations for 7.3.2.0.0 Interim Release enhancements.
1.0	Updated: September 2012	Updated the suggested reviewed changes.
1.1	Updated February 2013	Removed the Filter Servlet section from this document and the same can be obtained by contacting support team.
1.2	Updated: June 2013	Included Forms Framework enhancement configuration details.
1.3	Updated: June 2013	Included the following sections: ▪ MLS Utility
1.4	Updated: August 2013	Included the following sections: ▪ Config Schema Upload/ Download Settings ▪ Database Password Reset/ Change
1.5	Updated: September 2013	Removed some sections and added those as FAQs in Installation Guide.
1.6	Updated November 2013	Updated *MLS Utility* section.
1.7	Updated December 2013	Added *Configure SSO Authentication* section.
1.8	Updated August 2014	Included the following sections: ▪ Configure Stylesheet ▪ Hierarchy Node Internationalization for 7.3.5.0.0 IR
1.9	Updated Jan 2018	Added *Performance Optimization Setting for RRF Module* section.
Created by: Anand / Gitcy	Reviewed by: Bharath / Lokesh/ Sue	Approved by: Jeevraj / Surag Ramachandran / Subhashini

Executive Summary

This document includes the necessary instructions for module specific configurations. We recommend you to download the latest copy of this document from OTN library which includes all the recent revisions (if any) done till date.

TABLE OF CONTENTS

1 OFSAAI Administration

This section consists of information related to module specific configurations. You can refer to the required sections appropriately.

- Dimension Configuration: Alphanumeric and Numeric Codes
- Hierarchy Node Internationalization
- T2T and PR2 Query Performance Optimization
- Configure Data Quality Rule Approval Parameters
- Run Rule Framework Configuration
- Configure Forms xml to execute Server Side Rule
- Data Element Filters Classification
- Configure Forms Framework Enhancements
- Multiple Language Support (MLS) Utility
- Config Schema Upload/ Download Settings
- Database Password Reset/ Change
- Configure SSO Authentication
- Configure Stylesheet
- Hierarchy Node Internationalization (For 7.3.5.0.0. IR)

Conventions and Acronyms

Conventions	Description
Actions are indicated in **Bold**	
AIX	Advanced Interactive eXecutive
EPM	Enterprise Performance Management
KBD	Key Business Dimensions
OEL	Oracle Enterprise Linux
OFSAAI	Oracle Financial Services Analytical Applications Infrastructure
RHEL	Red Hat Enterprise Linux
SQL	Structured Query Language
UDP	User Defined Properties

Conventions	Description
UMM	Unified Metadata Manager
T2T	Table to Table
RRF	Run Rule Framework

1.1 Dimension Configuration: Alphanumeric and Numeric Codes

This section explains the configuration changes for Alphanumeric Code enhancements introduced as part of 7.3.1.0.0 IR.

OFSAAI supports both numeric and alphanumeric dimensions. Both dimension types require a numeric member code. An alphanumeric dimension will additionally store an alphanumeric member code. A numeric dimension can optionally store an alphanumeric code as well, but it will be equivalent to the numeric code value.

After the successful installation of this IR, you may need to run SQL updates on the REV_DIMENSIONS_B table. This table stores the required dimension metadata including dimension member data type and the member column names for dimension member tables where the numeric and alphanumeric codes are stored.

In the REV_DIMENSIONS_B table:

- The column MEMBER_DATA_TYPE_CODE with value 'NUMBER' identifies a dimension as numeric and value 'VARCHAR2' identifies a dimension as alphanumeric.
- MEMBER_CODE_COLUMN specifies the member table column which holds the alphanumeric member code. This is optional for numeric dimensions, where alphanumeric and numeric member codes would be equivalent.
- MEMBER_COL specifies the numeric member code column.

NOTE: Any change done in REV_DIMENSIONS_B table requires web server restart since, the dimension definitions data in cache memory has to be refreshed.

A new installation by default will have the seeded key dimensions configured as numeric, although those dimension member tables include a column for alphanumeric member codes. You can configure any of these dimensions as alphanumeric. For more information refer to Configure Alphanumeric Dimensions.

You might also need to run some SQL updates for numeric dimensions. For more information refer to Configure Numeric Dimensions.

1.1.1 Configure Alphanumeric Dimensions

To configure an editable (numeric) dimension as alphanumeric and to remove the optional code attribute from prior releases you have to back up the affected dimension tables (like REV_DIMENSIONS_B, REV_DIM_ATTRIBUTES_B, REV_DIM_ATTRIBUTES_TL, and DIM_<DIMENSION>_ATTR) and perform the following steps on each applicable dimension.

1. Set the member type as alphanumeric (VARCHAR2) in REV_DIMENSIONS_B, and identify the member table's alphanumeric code column name if it is not populated already.

   ```
   Update REV_DIMENSIONS_B SET
   Member_Data_Type_Code = 'VARCHAR2' [, Member_Code_Column = '{Alphanumeric Column Name}'] Where Dimension_ID = {Dimension ID}
   ```

   Example:

   ```
   Update REV_DIMENSIONS_B SET
   Member_Data_Type_Code = 'VARCHAR2', Member_Code_Column = 'TP_PRODUCT_CODE' Where Dimension_ID = 5;
   ```

   NOTE: In OFSAAI 7.3, the seeded key dimensions have already populated MEMBER_CODE_COLUMN.

2. If there is an associated code attribute which is present from OFSAAI release 7.3 for capturing the alphanumeric member code, remove it manually as follows:

   If the attribute currently contains values:

   - Update the alphanumeric codes from the attribute table to the Member Code Column of the member table.

     I.e. Execute the **Update_Dimension_Code** procedure using steps detailed in the Data Model Utilities Guide. For example: Either run **fn_updateDimensionCode** from SQL Plus or use a Batch to run TRANSFORM DATA with the **Update_Dimension_Code** procedure for your Dimension ID.

   - Remove the attribute values from the dimension attribute table (DIM_<DIMENSION>_ATTR).

     For example, if you are updating the Common Chart of Accounts Dimension, where Attribute Name is 'COMMON COA CODE', Member_B_Table_Name is 'DIM_COMMON_COA_B', and Attribute Table Name is 'DIM_COMMON_COA_ATTR', execute the following statement:

     ```
     Delete from DIM_COMMON_COA_ATTR where Attribute_ID = (Select Attribute_ID from REV_DIM_ATTRIBUTES_TL where
     ```

```
Attribute_Name = 'COMMON COA CODE' and dimension_id =
(select dimension_id from REV_DIMENSIONS_B where
member_b_table_name = 'DIM_COMMON_COA_B'));
```

- Remove the attribute definition from REV_DIM_ATTRIBUTES_B & REV_DIM_ATTRIBUTES_TL tables. This is necessary to prevent the Dimension Member UI from showing the code as an optional attribute, since it already displays the alphanumeric and numeric codes based on the member table.
- Delete alphanumeric attribute metadata from ATTRIBUTE_B table.

  Example:

  ```
  Delete from REV_DIM_ATTRIBUTES_B where Attribute_ID = (Select
  Attribute_ID from REV_DIM_ATTRIBUTES_TL where Attribute_Name =
  'COMMON COA CODE' and dimension_id = (select dimension_id from
  REV_DIMENSIONS_B where member_b_table_name =
  'DIM_COMMON_COA_B'));
  ```

- Delete alphanumeric attribute metadata from ATTRIBUTE_TL table.

  Example:

  ```
  Delete from REV_DIM_ATTRIBUTES_TL where Attribute_ID = (Select
  Attribute_ID from REV_DIM_ATTRIBUTES_TL where Attribute_Name =
  'COMMON COA CODE' and dimension_id = (select dimension_id from
  REV_DIMENSIONS_B where member_b_table_name =
  'DIM_COMMON_COA_B'));
  ```

3. In case any rows in the Dimension member table contain a null alphanumeric code even after running the **fn_updateDimensionCode** procedure, you need to manually run an SQL update as illustrated in the example below:

   ```
   Update DIM_GENERAL_LEDGER_B set GL_Account_Code = GL_Account_ID
   Where GL_Account_Code is null;

   Commit;
   ```

4. Save your changes using `Commit;`

1.1.2 Configure Numeric Dimensions

If REV_DIMENSIONS_B.Member_Code_Column is populated for a dimension, any UI which displays an alphanumeric code will look in the specified column for the member's alphanumeric code. If REV_DIMENSIONS_B.Member_Code_Column is null, the UI will assume no alphanumeric code column exists in the member table and will display the alphanumeric code with the same value as the numeric code. Therefore, for numeric dimensions, you may want to update the metadata.

There are two options available to configure Numeric dimension.

- Option 1: When the dimension does not have <DIM>_CODE column in <DIM>_B table
- Option 2: When the dimension have <DIM>_CODE column in <DIM>_B table

NOTE: By default, there may not be configuration changes required in Rev_Dimensions_B for Numeric dimension, since the REV_DIMENSIONS_B.MEMBER_CODE_COLUMN column would have value either <Dim>_Code or null depending on the availability of <Dim>_Code column.

Option 1: When the dimension does not have <DIM>_CODE column in <DIM>_B table. In this case, the alphanumeric and numeric code value are stored in the same <DIM>_ID column.

- Back up the table REV_DIMENSIONS_B, if you have not done it already.
- Clear the Member Code Column entries for applicable dimensions.

  Example:

  - For specific numeric dimensions:

    ```
    Update REV_DIMENSIONS_B Set Member_Code_Column = null Where
    Dimension_ID in([values]);

    Commit;
    ```

  - For all editable numeric dimensions:

    ```
    Update REV_DIMENSIONS_B Set Member_Code_Column = null Where
    Member_Data_Type_Code = 'NUMBER' and DIMENSION_EDITABLE_FLAG =
    'Y';

    Commit;
    ```

NOTE: If the dimension have <Dim>_Code column and Option 1 is used (i.e. the REV_DIMENSIONS_B.MEMBER_CODE_COLUMN is set to null), this will cause the dimension loaders and seeded T2T extracts to fail.

Option 2: When the dimension have <DIM>_CODE column in <DIM>_B table. In this case, the alphanumeric and numeric code value are stored separately in <DIM>_CODE and <DIM>_ID column (though both the values are same).

- Back up the table REV_DIMENSIONS_B, if you have not done it already.
- Populate the Member Code Column entries for applicable dimensions.

  Example:

  - For specific numeric dimensions:

    ```
    Update REV_DIMENSIONS_B Set Member_Code_Column = <dim>_code
    Where Dimension_ID in([values]);

    Commit;
    ```

  - For all editable numeric dimensions:

    ```
    Update REV_DIMENSIONS_B Set Member_Code_Column = <dim>_code
    Where Member_Data_Type_Code = 'NUMBER' and
    DIMENSION_EDITABLE_FLAG = 'Y';

    Commit;
    ```

For upgrades from 7.3 release, if data already exists for numeric dimension members, copy the numeric code values to the alphanumeric code column.

Example:

- For the General Ledger Account dimension:

  ```
  Update DIM_GENERAL_LEDGER_B set GL_Account_Code = GL_Account_ID;

  Commit;
  ```

1.1.3 Configure Alphanumeric Code in Simple Dimension Tables

For some editable seeded and user-defined simple dimensions, the alphanumeric code column currently might not be present in the data model. To add this column to a user-defined simple dimension table, you can use Model Upload. You will also need to update the REV_DIMENSIONS_B table as indicated in Dimension Configuration section, to configure alphanumeric properties.

NOTE: You should not modify the structure of any seeded simple dimensions.

1.1.4 Create Index on Code Column

You need to create a unique index on the alphanumeric code column if such an index is does not exist. Also while creating, you need to ensure that this index uniqueness should be case insensitive.

Example:

```
Create unique index IDX1_DIM_PRODUCTS_B on DIM_PRODUCTS_B
Upper(PRODUCT_CODE)

Commit;
```

1.2 Hierarchy Node Internationalization

This enhancement is included as part of OFSAAI 7.3.2.0.0 IR Patch Update. See Hierarchy Node Internationalization for enhancements done in 7.3.5.0.0 IR.

Hierarchy Node Internationalization is an enhancement brought into the Business Hierarchy section of the Oracle Financial Services Analytical Applications Infrastructure. This feature is introduced to internationalize the node description of Regular Business Intelligence Enabled (BI) and Parent Child (PC) Hierarchies and to display them in Hierarchy Browser.

Previously, node descriptions were fetched from the Description column of the Dimension table (<DIM> table) to facilitate the node generation in REV_LOCALE_HIER table. Each Node has a description. Hierarchy node Internationalization enhancement changes the way in which the descriptions are stored in the REV_LOCALE_HIER. The locale specific node descriptions are fetched from Multi Language Support tables (<DIM>_MLS table). This table holds the node descriptions in all the installed locale, that is, in the locales in which OFSAAI is available.

1.2.1 Scope

The scope of this enhancement is limited to the Hierarchy Browser window. The hierarchies defined are displayed in Hierarchy Browser and the Hierarchy Browser is used in other modules/ screens such as Unified Metadata Manager, Rules Framework, Metadata Browser, Map Maintenance, Forms Framework, and Hierarchy Maintenance.

1.2.2 Prerequisites

The following are the prerequisites for creating a Hierarchy with Multi Language Support Descriptions:

- The Hierarchy under creation should be either Regular Business Intelligence Enabled (BI) or Parent Child (PC).
- The Multi Language Support table <DIM>_MLS should be created either through Data Model Upload or manually in atomic schema. For more information on <DIM>_MLS table and structure, refer to Multi Language Support (MLS) Table.

- The Description columns used for node generation should be of **Varchar** / **Varchar2** data type.

1.2.3 Multi Language Support (MLS) Table

The <DIM>_MLS tables are created by appending "_MLS" to the existing <DIM> table name.

NOTE: The insertion of data into <DIM>_MLS tables should be performed manually.

1.2.3.1 <DIM>_MLS Table Structure

The <DIM>_MLS table created for a <DIM> table entirely depends on the structure of the <DIM> table. The following points must be followed during <DIM>_MLS table creation.

- Description columns on which the Hierarchy definition is based should also be present in the <DIM>_MLS table.
- A column namely DESCLOCALE of data type **Varchar** / **Varchar2** should be present in the <DIM>_MLS table. This column should contain the information about the locale (such as **fr_FR**, **ko_KR**) and should be part of the composite primary key.
- The primary key of the <DIM>_MLS table is formed by the columns which are the primary keys of <DIM> table, along with an additional column namely DESCLOCALE. Ensure that the column names are same in both the tables.

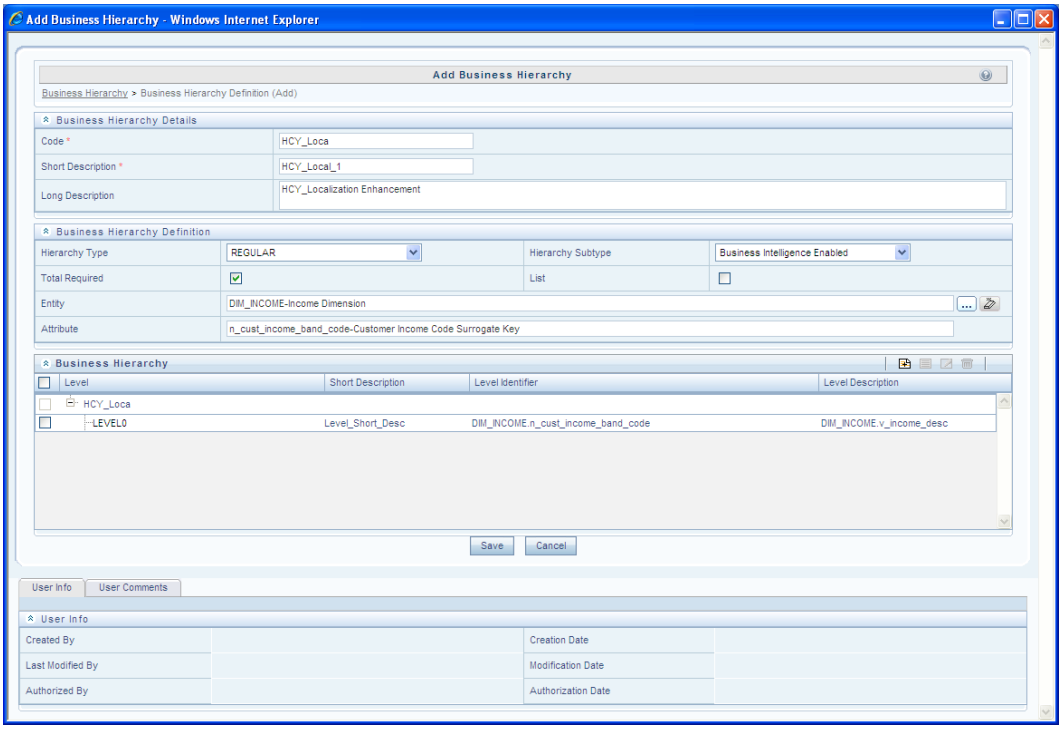
Example:

Consider a Hierarchy "**Income"** defined on table "DIM_INCOME". The table structure is as indicated:

Column Name	Primary Key	Datatype
N_CUST_INCOME_BAND_CODE	PK	Number(5,0)
FIC_MIS_DATE		Date
V_CUST_INCOME_SHORT_DESC		Varchar2(80)
V_INCOME_DESC		Varchar2(80)
N_D_INCOME_UPPER_VALUE		Number(22,3)
N_D_INCOME_LOWER_VALUE		Number(22,3)

The primary key of DIM_INCOME table is PK_DIM_INCOME and is created using N_CUST_INCOME_BAND_CODE.

The following figure represents the **Income** Hierarchy definition:



<DIM>_MLS table corresponding to the DIM_INCOME can be created as mentioned below:

The <DIM>_MLS table would be DIM_INCOME_MLS with columns:

- N_CUST_INCOME_BAND_CODE
- V_INCOME_DESC (both columns are used in DIM_INCOME table)
- DESCLOCALE

All the columns which act as the primary key in the DIM_INCOME table should also be created in DIM_INCOME_MLS table and the data type of these columns in both tables should be the same.

The structure of the DIM_INCOME_MLS table would be as follows:

Column Name	Primary key	Datatype
N_CUST_INCOME_BAND_CODE	PK	Number(5,0)
V_INCOME_DESC		Varchar2(80)
DESCLOCALE	PK	Varchar2(30)

The following table displays sample data which can be populated in DIM_INCOME_MLS table in a setup where there are 2 locales installed i.e. English (en_US) and Chinese (zh_CN).

N_CUST_INCOME_BAND_CODE	V_INCOME_DESC	DESCLOCALE
1	AAA	en_US
2	BBB	en_US
1	CCC	zh_CN
2	DDD	zh_CN

Note the following:

- The <DIM> table name **should not** exceed 26 characters as the corresponding <DIM>_MLS table name would need four more characters (_MLS) and the Oracle table nomenclature allows maximum of 30 characters. Any Dimension table names with more than 26 characters in length have to be reduced to 26 characters if this enhancement needs to be configured for that <DIM> table and the Hierarchy based on it. This change would require a Data model change and the possible impacts of this across AAI should be addressed.
- The expression created in Level Description field **should not** contain any CASE statements.
- In Regular BI enabled and PC Hierarchies, the Level Description expression **should not** contain columns with Number or Date data types. The inclusion of such a column in the Level Description expression would prevent the Business Hierarchy from generating nodes.
- There is no concept of **default** locale. Whenever a Hierarchy is saved, the translated node descriptions present in <DIM>_MLS table are saved in the corresponding columns of the REV_LOCALE_HIER table depending on the availability of translated values in the <DIM>_MLS table.
- The inclusion or exclusion of nodes from a Hierarchy will be reflected in Forms once the Hierarchy is resaved.

1.2.4 Node Generation Process

During Hierarchy definition, the nodes get generated depending on the structure of the Hierarchy. Node generation is possible in the following two scenarios:

- Node Generation when <DIM>_MLS Table is Present
- Node Generation when <DIM>_MLS Table is Not Present

1.2.4.1 Node Generation when <DIM>_MLS Table is Present

When <DIM>_MLS table is present, the nodes are generated by fetching the Description from the <DIM>_MLS table. Thus, entry in the Description column is mandatory.

1.2.4.2 Node Generation when <DIM>_MLS Table is Not Present

When <DIM>_MLS table is not present, by default the nodes are generated by fetching the Description from the <DIM> table.

1.2.5 Configure Mapper for Multiple Locales

This step is optional and is required if *Node Generation Process* explained in the above section is done.

To configure mapper for multiple locales:

1. Duplicate the data in REVELEUS_MASTER table with different locales in LOCALE_ID column.
2. Translate V_OBJECT_DESC column in REVELEUS_MASTER table to the desired locale.
3. Duplicate data in LOCALE_ID column in REV_MAST_MAP_ITEMS table for different LOCALE_ID.

Example:

An existing mapper namely **Mapper A** (created in any locale) can be translated into other locales as indicated in the following example:

1. Login to the configuration schema and duplicate the data in REVELEUS_MASTER table by changing the locale in LOCALE_ID column.
2. Change V_OBJECT_DESC for the corresponding locale in REVELEUS_MASTER table.
3. Duplicate the data in REV_MAST_MAP_ITEMS table by changing Locale in LOCALE_ID column.

NOTE: 2nd and 4th steps need to be performed for all the locales to which you wish to translate mapper A.

1.2.6 Update Nodes in Existing Regular BI and PC Hierarchies

Currently, the node description is generated only for one locale on which the Hierarchy is saved. With the introduction of Hierarchy Node Localization, the nodes will be generated in all the installed locales.

To generate the localized node descriptions for the existing Hierarchies, you need to edit and re-save the Hierarchies post <DIM>_MLS table creation. You can also mass update the existing Hierarchies from **Administration** > **Save Metadata** section. The node description data for all the installed locales will be populated in REV_LOCALE_HIER table.

NOTE: If an SCD (Slowly Changing Dimension) is configured on a <DIM> table, synchronize the new entries with the corresponding <DIM>_MLS table also.

1.2.7 Limitations

If the Hierarchies are accessed via Modeling Framework module, the node descriptions of the same will be displayed only in English, despite the locale you have logged in to the application.

1.3 T2T and PR2 Query Performance Optimization

This enhancement is included as a part of OFSAAI 7.3.2.0.0 IR Patch Update.

A configuration file, **OracleDB.conf** has been introduced to accommodate any configurable parameter related to operations for Oracle database. If you do not want to set a parameter to a specific value, then the respective parameter entry can be removed/commented from the **OracleDB.conf** file which resides in the path *$FIC_DB_HOME/conf*.

The following table details the configurable OFSAA parameters in **OracleDB.conf** file with its purpose, and the way it maps to Oracle Database Parallelism settings.

Parameters	Description
CNF_PARALLEL_DEGREE_POLICY	Sets the parallel degree policy. Possible values – **MANUAL**, **LIMITED**, or **AUTO**. Query fired on the database - *ALTER SESSION SET PARALLEL_DEGREE_POLICY=<<CNF_PARALLEL_DEGREE_POLICY>>*
CNF_PARALLEL_QUERY	Sets parallelism for queries. Possible values – **DISABLE**, **ENABLE**, or **FORCE**. Query fired on the database - *ALTER SESSION <<CNF_PARALLEL_QUERY>> PARALLEL QUERY*
CNF_PARALLEL_DML	Sets parallelism for DML operations. Possible values – **DISABLE**, **ENABLE**, or **FORCE**. Query fired on the database - *ALTER SESSION <<CNF_PARALLEL_QUERY>> PARALLEL DML*
CNF_DEGREE_OF_PARALLELISM	Sets the degree of parallelism. Possible values – Value can be any positive integer. The default mode of a session is *DISABLE PARALLEL DML*. If *CNF_DEGREE_OF_PARALLELISM* is not set, then the default degree, as decided by Oracle will be used. Queries fired on the database - *ALTER SESSION <<CNF_PARALLEL_QUERY>> PARALLEL QUERY PARALLEL <<CNF_DEGREE_OF_PARALLELISM>>* *ALTER SESSION <<CNF_PARALLEL_QUERY>> PARALLEL DML PARALLEL <<CNF_DEGREE_OF_PARALLELISM>>*

For more information, refer *Oracle Database VLDB and Partitioning Guide - 11g Release 2 (11.2)* Chapter 8 - **Using Parallel Execution** section.

1.4 Configure Data Quality Rule Approval Parameters

This enhancement is included as a part of OFSAAI 7.3.2.0.0 IR Patch Update.

To facilitate users to edit and delete an approved, not grouped, and non executed Data rule in the Data Quality framework, you need to set the required configuration parameter. In the config schema, set **ALLOW_CHNG_OF_NONEXEC_DQRULE** parameter value to "**Y**". This value by default is set to "Y" unless modified for a specific condition.

1.5 Run Rule Framework Configuration

The 7.3.2.0.0 IR release includes the enhanced RRF (Run Rule Framework). This framework is a redesigned version of the earlier PR2 Framework and includes UI and functional enhancements. This section consists of the following:

- Enable RRF/PR2 Links in OFSAAI
- Command Line Utilities
- Component Registration

1.5.1 Enable RRF/PR2 Links in OFSAAI

In case of OFSAAI 7.3 fresh installation, on successfully installing the 7.3.2.0.0 IR, RRF module will be enabled by default. For upgrade from older version to v7.3.2.0.0 IR, PR2 module will be retained if it has been used, else RRF would get enabled.

However, if you wish to enable RRF module in parallel to the existing PR2 module, you need to execute the below "enable_rrf.sql" script in Config schema of your installation. This enables RRF in the User Interface.

enable_rrf.sql

In case of a fresh installation, if you wish to enable the PR2 framework, you need to execute the below "enable_pr2.sql" script in Config schema of your installation. This enables PR2 in User Interface.

enable_pr2.sql

For using the PR2 metadata in RRF, you need to archive and restore metadata in the new RRF User Interface. For more information, refer to the *PR2 to RRF Migration Guide* available at OTN library.

Further, if you wish to execute the RRF engines you would be required to update the column F_IS_RRF value in configuration table in Configuration schema to “RRF”.

NOTE: Once this value is modified, the PR2 engines cannot be executed. In case you need to execute the PR2 engines, you would be required to switch back the value to “PR2”. *For more information, refer OFSAAI 7.3.2.0.0 User Manual > System Configuration > Configuration > Others tab*. In addition, the value for this parameter needs to be switched to either RRF/PR2 to use the respective modules.

1.5.2 Command Line Utilities

This section explains the various Command Line Utilities available in OFSAAI 7.3.2.0.0.

Utility Name	Usage
Migration	To Migrate object definitions across Information Domains / Setups. Currently this utility supports the following object types: ▪ **Form** - Forms framework definition. ▪ **Menu** - Forms Menu definition. ▪ **Map** – Map Maintenance definition. ▪ **Rule** – Rule definition of RRF. ▪ **Process** – Process definition of RRF. ▪ **Run** – Run definition of RRF. This utility is available under *$FIC_HOME/utility/Migration* of OFSAAI APP tier post v7.3.2.0.0 IR patch update. *For usage of this utility, refer **Command Line Utilities** section in OFSAAI 7.3.2.0.0 User Manual available at OTN library.*
Rule Execution	To execute RRF Rule definitions. This utility is available under *$FIC_HOME/utility/RuleExecution* of OFSAAI APP tier post v7.3.2.0.0 IR patch update. *For usage of this utility, refer **Command Line Utilities** section in OFSAAI 7.3.2.0.0 User Manual available at OTN library.*
Rule Execution through Web Service (*WSExecution*)	To execute RRF Rule definitions through Web Services. This utility is available under *$FIC_HOME/utility/WSExecution* of OFSAAI APP tier post v7.3.2.0.0 IR patch update. *For usage of this utility, refer **Command Line Utilities** section in OFSAAI 7.3.2.0.0 User Manual available at OTN library.*

1.5.3 Component Registration

A Component in the context of OFSAAI is an entity which can be executed individually in Operations module to carry out some definite job for which it has been formed. Components within OFSAAI and its application need to be registered so that it is configurable for different installations with very minimal change.

The component registration process helps you to make the components of Process and Run module configurable inside Run Rule Framework (RRF). With component registration, components can be added, modified and deleted from RRF by doing very minimal changes to the system. For registering a component in RRF, the same should be present in ICC also.

Steps to Register Component

Registering Component has been divided into the following steps respectively:

- Component Detailed Implementation Class
- Deployment
- Entry to PR2_COMPONENT_MASTER Table

1.5.3.1 Component Detailed Implementation Class

The component implementation class has to be made for all the components which are inserted to the PR2_COMPONENT_MASTER table.

This class has to extend **com.ofs.aai.pr2.comp.PR2ComponentProps**, in turn to implement the following methods.

- getComponentDescription
- getPorbableParamValues (optional)

Implementation of interface com.ofs.aai.pr2.comp.PR2Component is optional. This interface will be implemented for only the components which can be directly used in a Process or Run. By implementing this class file following methods has to be over written.

- getSummay
- getCompDescMap
- fillTaskParameter
- getUsedTables

Each method takes current username and locale by default.

1.5.3.1.1 getComponentDescription

This method is used to get the description for all the components which are show in the component tree.

The Input Parameters are:

- String username
- String locale

Return is:

- String

It returns the localized string that has to be displayed for the component in the component tree.

1.5.3.1.2 getPorbableParamValues

This method is used to identify if a parameter input should be a text box or a drop down field.

The Input Parameters are:

- String username
- String locale
- String infodom

Return is:

- Map<String, String>

It returns map containing entry key as the value which is shown to the user. The entry value is stored in database.

1.5.3.1.3 getSummary

This method is used to get all existing definition of the component type existing in the system.

The Input Parameters are:

- String username
- String locale
- String infodom

Return is:

- Hashtable<String, Vector<com.ofs.aai.pr2.comp.bean.TaskDefinition>>

It returns a Hashtable of <String, Vector<TaskDefinition>>. Where key denotes any specific sub-levels to be shown, which in turn contains a JSON object with compName, compDesc, isDinamic, levelImg properties for that sub-level and the Vector<TaskDefinition> contains all the data needed for using the component in a process or run.

1.5.3.1.4 getCompDescMap

This method is used to find all details about all specified definitions.

The Input Parameters are:

- String username
- String locale
- String infodom
- Map<String, String> descMap
- Boolean allData

Return is:

- Map<String, String>

Passed to the method in Map<String, String>, where key is the definition unique code. The value is a JSON object with defnDesc property with the value same as code. The same JSON has to be replaced with another JSON object containing defnDesc, defnSubType, defnRef1Name, defnRef1Value, defnRef2Name, defnRef2Value, defnRef3Name, defnRef3Value, defnRef4Name, defnRef4Value, defnOptParamName properties. The values populated for these properties as follows.

Property Name	Description
defnDesc	Populated with <name> for the <code> of the definition, if <name> exists. If <name> does not exist, then populated with <code>:SD. If definition does not exist, then populated with <code>:NA.
defnSubType	Sub-Type of the definition
defnRef1Name defnRef1Value defnRef2Name defnRef2Value defnRef3Name defnRef3Value defnRef4Name defnRef4Value	Any references which can be used to Identify the definition uniquely. There are four of them. So can be put as name and value pairs.
defnOptParamName	If any optional parameter exits and has to be taken as input from user, then only the name can be provided by this property.

There is another input called **allData**, which is a flag. If it is false then only **defnDesc** has to be passed and when true all the data has to be passed.

After putting the corresponding JSON Object to its <code> the same map is returned back.

1.5.3.1.5 fillTaskParameter

This method is used to get the parameters for the component which will be used to execute the component in Operations module.

The Input Parameters are:

- String username
- String locale
- String infodom
- String uniqueName
- String subtype
- Map<String, String> allParams

Return is:

- Map<String, String>

It takes uniqueName which is nothing but the <code> of the definition. It also takes subType of the definition and an allParams which is of data type Map<String, String>. This map contains all the probable parameters with it, where key is the parameter name and value is the parameter value. This map contains following params.

- Dollar variables ($RUNID, $RUNSK, $EXEID, $RUNEXECID, $MODE).
- All reference name and value.
- Optional parameter if any.

By using the map another LinkedHashMap will be created in this method with all the parameters needed to run the component in Operations module. Al the parameter in this map has to be put in correct order. This LinkedHashMap will be returned back to the calling method.

1.5.3.1.6 getUsedTables

This method is used to get the dependent tables for specified definition of the component type.

The Input Parameters are:

- String username
- String locale
- String infodom

- String uniqueName
- Map<String, String> allParams

Return is:

- Set<String>

It takes uniqueName which is <code> of the definition and the same allParam map which is used in fillTaskParameter method. By using these inputs a Set<String> will be formed with all the dependant table data. This data is used to identify a Rule Filter / Process Filter can be applied to this component. This Set will be returned to the calling method.

1.5.3.2 Deployment

Below steps should be followed for deployment of the component.

1. Place all the image files to the folders mentioned in **V_TREE_IMAGE** column of **PR2_COMPONENT_MASTER** table, relative to *<FIC_WEB_HOME>/webroot* folder of the application.
2. The jar containing the component implementation classes has to be placed into *<FIC_WEB_HOME>\webroot\WEB-INF\lib* folder.
3. Rebuild and redeploy the application.

1.5.3.3 Entry to PR2_COMPONENT_MASTER Table

PR2_COMPONENT_MASTER is the table for storing all components which are used in RRF. You can enter either through backend which is explained here or through UI which is explained in the *Component Registration* section under RRF module in the *OFSAAI User Guide* available at OTN Library.

An entry contains the following fields.

Column Name	Type	Description	Null
V_PR2_COMPONENT_ID	VARCHAR2(30)	Represents component type in a Process or Run.	N
V_PR2_COMPONENT_PARENT_ID	VARCHAR2(30)	Indicates parentage which refers to V_PR2_COMPONENT_ID.	Y
V_COMPONENT_ID	VARCHAR2(30)	Existing ICC Component Id.	Y
V_PR2_COMPONENT_CLASS	VARCHAR2(100)	Fully qualified class path of the implementation class for this component.	N

Column Name	Type	Description	Null
V_TREE_IMAGE	VARCHAR2(100)	Name with relative path (with respect to web context) of the image which will be displayed in the component tree.	N
N_TREE_ORDER	NUMBER(9)	Display order of the component in the tree. The order is done upon the peers.	N
V_SEEDED_BY	VARCHAR2(8)	Differentiates user created and system created. The system created will have this field filled with an application name which cannot be edited from the front-end utility. The components created from front-end utility will not populate any value in this field which can be edited or deleted from front-end.	Y
V_CREATED_BY	VARCHAR2(30)	Stores the creator username.	N
D_CREATED_DATE	TIMESTAMP(6)	Stores created date and time.	N
V_LAST_MODIFIED_BY	VARCHAR2(30)	Stores the modifier username.	Y
D_LAST_MODIFIED_DATE	TIMESTAMP(6)	Stores modified date and time.	Y

Example:

```
insert into PR2_COMPONENT_MASTER (V_PR2_COMPONENT_ID,
V_PR2_COMPONENT_PARENT_ID, V_COMPONENT_ID, V_PR2_COMPONENT_CLASS,
V_TREE_IMAGE, N_TREE_ORDER, V_SEEDED_BY, V_CREATED_BY) values ('COMPTYP',
null, 'Component Sample', 'com.sample.ComponentSample',
'sampleImages/sampleComp.gif', 0, 'SEEDEDBY', 'USER')
```

1.5.3.4 Sample Code

The below file contains the sample code of a created component.

ComponentSample.txt

1.6 Configure Forms xml to execute Server Side Rule

This enhancement is included as a part of OFSAAI 7.3.2.0.0 IR Patch Update.

You can execute database stored procedure and PR2 Run using the Forms Framework server side rule configuration. Post this release you can also include RRF Run to be executed as a server side rule.

In order to execute RRF Run using Forms xml, the Form where server side rule is being executed with Type as "**REVELEUS_RULE**" you need to manually update the Type as "**FIRERUN**".

For example, the **RiskRecalculate.xml** having server side rule is used to re-calculate the risk. Here the Type needs to be changed as suggested below.

Replace the following attribute **Type** value:

```
<RULESET ID="110" TYPE="REVELEUS_RULE">
```

With

```
<RULESET ID="110" TYPE="FIRERUN">
```

1.7 Data Element Filters Classification

This enhancement is included as a part of OFSAAI 7.3.2.0.0 IR Patch Update.

This section explains the option to categorize "Filter classification Types" as **Classified, UnClassified**, or **All** which can be used to define Data Element filters on Business Metadata Management objects.

To classify the tables available for a Filter in an existing information domain, perform a Model upload (Incremental / Sliced / Complete) to trigger object registration, which in turn will populate all the necessary entries to the registration tables. This is an optional one-time activity required to register all the tables, so that the tables without classification code are also made available in the Data Element filters.

During Model upload, Object Registration is done for all Tables and columns.

- Tables with the classification code will continue to have entry in REV_TABLE_CLASS_ASSIGNMENT with the appropriate classification code.
- Tables without classification code will also have entry in REV_TABLE_CLASS_ASSIGNMENT with the value as 1000 (UnClassified).

Once tables are registered successfully, user can go to the *Filter* screen to Define Data Element Filters on any tables and columns. Based on the Classification, the appropriate Classification type option has to be selected in the *Data Element Selection* screen to list the tables.

Note the following:

- If the field value in **CLASSIFICATION_FLG** column of **REV_TABLE_CLASSIFICATION_B** table is set to '**1**', then it is considered as a **Classified** table.

  By Default, the classification codes 20, 200, 210, 310, 370, 50, 300, and 500 will have the **CLASSIFICATION_FLG** set to "**1**".

- The **REV_TABLE_CLASSIFICATION_TL** table will have an entry TABLE_CLASSIFICATION_CD = "**1000**", TABLE_DESCRIPTION = "**UnClassified**" to identify UnClassified Tables (i.e. tables which are not classified in the Erwin through UDP).

- The category "**All**" option will select all the tables available in the infodom, irrespective of whether table is classified or not.

  Since the above option doesn't check the classification type, So even table which has CLASSIFICATION_FLG = **Blank**, in the **REV_TABLE_CLASSIFICATION_B** table will also be listed. These tables will not be displayed under Classified or Unclassified Category.

1.7.1 Limitations

Following are the limitations with Data Element Filters classification:

- While defining Data Element Filter/Group Filter, it is not recommended to use features like using an Expression in a Filter and Macro Columns, since the generated SQL query for these features is unresolved.

- Defining Hierarchy/Attribute Filter is not recommended using BMM objects since the underlying Dimension and Hierarchy data are more specific to EPM Apps, and data will be available only if EPM Apps are installed in same Information Domain.

- Dependency check is not available when any of the BMM objects uses Filters. To maintain dependency between parent and child objects, an appropriate entry has to be added in to the **REV_OBJECT_DEPENDENCIES** table. Since the BMM object definition details are stored in Config schema, and do not populate entry into the **FSI_M_OBJECT_DEPENDENCY_B/TL** tables, the dependency check will not happen especially while deleting a Filter.

1.8 Configure Forms Framework Enhancements

The following enhancements are introduced as a part of 7.3.2.2.0 ML release and the required configuration to enable these features in Forms Framework module are explained below.

- **Forms Framework now supports additional set of styles to configure the User Interface and screen elements. (Bug 16327172)**

  The required configuration can be done at an application level. In the application, the Container and Control specific styles can be configured using GroupStyles.

  Application level configuration

  1. The configurability is decided at the time of choosing the AAI LHS menu option.
     - Configuration changes are done in the LHS Menu (locale specific) xml file using the "cssFileName" request parameter.
     - If the cssFileName value is not defined, then the default stylesheet file is loaded.
     - CSS_OFSAAI is the new CSS value that gets passed for the parameter.
  2. Application users can navigate directly to the application landing page without having to choose the application link from AAI LHS menu using the "Make my Home Page" option in AAI home page.
     - Database values are modified. The request parameter "cssFileName" is appended to the field "START_PAGE_URL" in Table cssms_start_page_master.
     - If the cssFileName value is not defined, then the default stylesheet file is loaded.
     - CSS_OFSAAI is the new CSS value that gets passed for the parameter.

  Configuration of Application Elements

  Page level elements such as Containers or Controls can be configured using CssClassName or GroupStyle.

ATTRIBUTE	DESCRIPTION
CSSCLASSNAME	A CSSCLASSNAME is defined for a Container or a Control. It is a single style that is used to alter the look and feel of a defined element in the Container or the Control.
GROUP_STYLE	A GROUP_STYLE is defined for a Container. It comprises of a group of styles for every element of the Container.

  Example to configure a Container:

```
<CONTAINER CREATEDIV="Y" CSSCLASSNAME="formlegend" WIDTH="25"
GROUP_STYLE="lvflv2" ID="5" NAME="" NOOFCOLS="2" TYPE="1"
VIEWMODE="1" BORDERREQUIRED="N" COLLAPSEREQUIRED="N">
```

Example to configure a Control:

```
<CSSCLASSNAME TD_LABEL_STYLE=" className 1" TD_DATA_STYLE="
className 2">
```

- **The seeded Forms now have two additional columns introduced in "Forms_master" table. (Bug 16542057)**

  Two additional columns have been introduced in Forms_master table namely Module **(varchar2(200))** and SubModule **(varchar2(200))** to distinguish two or more solutions installed in same information domain.

- **Grid title can be parameterized based on request parameters. (Bug 16541973)**

  Existing grid control title can be changed dynamically. For example, on load of a page ruleset, you can define a rule configuration validation (front end, backend, page loading) with the proposed function (i.e. setGridTitle) to change the grid title dynamically.

```
<RULESET ID="111">
<RULE EXPRESSION="setGridTitle('<Grid Parent Form ID>','<Grid
ControlID>',getParamValWithDiffDelimiter([~queryString],'<Parameter
name passed in request String>'))"/>
<RULESET>
```

- **Grid control has been enhanced to load in collapsed mode, even if there is data in grid container. (Bug 16490089)**

  On page load, grids can now be displayed in collapsed/minimized state even if it contains data, in addition to the existing functionality where a grid is minimized only when it does not have data/rows.

  A new control specific tag (**Type 50**) **GRID_MINIMIZE_REQ** is introduced to display grid in minimized mode when set to "Y" even with data. By default, it is set to "N" (expanded state).

```
<GRID_MINIMIZE_REQ>Y</GRID_MINIMIZE_REQ>
```

  Additionally to enable the collapse/expand button, set "COLLAPSE_OPTION_REQ" to "Y" in GRID_TOOLBAR tag in control type 50.

```
<GRID_TOOLBAR>
<COLLAPSE_OPTION_REQ>Y</COLLAPSE_OPTION_REQ>
```

- **The Second level Header displaying header names can be configured for non-English locale support based on the translated descriptions. In addition, the second level header is displayed in the form by default even without grid data. (Bug 16526053)**

  Follow the below configuration to configure locale specific second level headers.

```
<SPAN ID="81" >

   <CONTROL_ID>1</CONTROL_ID>
   <CONTROL_ID>81</CONTROL_ID>
```

```
</SPAN>
<SPAN ID="105">
   <CONTROL_ID>18</CONTROL_ID>
   <CONTROL_ID>7</CONTROL_ID>
</SPAN>
```

Following is the sample entries of **FORMS_LOCALE_MASTER** table.

FORM_CODE	DSN_ID	CONTROL_ID	LOCALE_ID	CONTROL_NAME	CONTEXT HELP	TOOL TIP	FORM_VERSION	CONTROL_TYPE
		81	en_US	**2nd lvl Hdr One**			0	**4**
		105	en_US	**2nd lvl Hdr two**			0	**4**
		81	fr_FR	Un deuxième lvl Hdr			0	**4**
		105	fr_FR	2e lvl Hdr deux			0	**4**

- **New control Type attribute has been introduced with rich text formatting capabilities to modify text attributes like bold, italics, font style, and so on. (Bug 16489910)**

  The new control Type attribute (**Type 40**) has the rich text formatting capabilities to modify the required text attributes.

```
<!--type 40 is for Richtext -->
<CONTROL ID="1111" TYPE="40">
   <CONTROLPROPS>
      <RENDERMODE>1</RENDERMODE>
      <ISMANDATORY>Y</ISMANDATORY>
      <ISMASKINGREQUIRED>N</ISMASKINGREQUIRED>
      <ASSOCIATEDLABEL>RichText</ASSOCIATEDLABEL>
      <CSSCLASSNAME>formlegendScrl</CSSCLASSNAME>:
      <CONTROL_SPECIFIC_TAGS>
         <NOOFROWS></NOOFROWS>
      <CONTROL_SPECIFIC_TAGS>
   </CONTROLPROPS>
</CONTROL>
```

- **Hierarchy field is parameterized and is configurable to enable /disable the Text entry option and to display dynamic selection options during text entry. Similarly even date field can be entered manually. (Bug 16489789** - *This enhancement is for non-custom hierarchies only***)**

  For a **single select Hierarchy** (type 41) you can enter data manually into the hierarchy field using which, dynamic selection options are displayed. This also helps in the usability aspect and reduces the loading time of entire hierarchy list.

```
<CONTROL ID="131" TYPE="41">
   <CONTROLPROPS>
      <CONTROL_SPECIFIC_TAGS>
         <IS_SUGGEST_REQUIRED
         SUGGEST_SIZE="5">Y</IS_SUGGEST_REQUIRED>
      </CONTROL_SPECIFIC_TAGS>
   </CONTROLPROPS>
</CONTROL>
```

  For **KBD Hierarchies**, two Form level parameters “isSuggestDropReq” and “suggestSize” are added in the menu_items table, under the FORM_PARAMETERS column.

  To enable this option, you need to set **IS_SUGGEST_REQUIRED** as **True** and also specify in the <**SUGGEST_SIZE**> to limit the number of suggestions.

```
isSuggestDropReq=true and suggestSize=5 or any value.
```

  Below configuration can be used to enter Date values directly without calendar popup.

```
<CONTROL ID="1" TYPE="11">
   <CONTROLPROPS>
      <CONTROL_SPECIFIC_TAGS>
         <ENTER_DATE_REQUIRED>Y</ENTER_DATE_REQUIRED>
      </CONTROL_SPECIFIC_TAGS>
   </CONTROLPROPS>
</CONTROL>
```

- **Dynamic parameterized messages are enabled to display custom messages overwriting the default static messages. (Bug 16541992)**

  Specific configuration has been introduced to display more than one message in the given sequence and overwrite the default messages with custom message including messages on server-side operations. Each message will have an identifier and Message Type like

Warning, Confirmation, Information, Error, and so on. Those message where “Message Type” of the seeded AAI and custom messages are same are only overwritten.

For example, consider a Form with a text box. When you enter the required text and Save, a message is displayed “*Your case <<ID>> details are saved successfully*”. Here <<ID>> is a placeholder which is fetched dynamically based on logged in locale at run time.

Below configuration is required to be added at form level tags.

```
<MESSAGEID="123" IDENTIFIER="RENDERER.MESG_WARN" TYPE="Warning">
```

ID - Any unique message ID

IDENTIFIER - RENDERER.<message identifier defined in messages_locale tables>

TYPE - Message type like Warning, Confirm, Information, Error, and Message. Value should be case sensitive.

Also place holder variables are required to be added in message to evaluate and display the messages with dynamic values in the form.

E.g: `<PARAMETER NAME="placeholder1" VALUE="[~FrmKILibP_123_1]"/>`

Assuming that the identifier RENDERER.MESG_WARN's message value is "*Selected case ID <placeholder1> is already closed. Please select case with <placeholder2> status*"

Then the above message will get displayed in front end as, "*Selected case ID **3454** is already closed. Please select case with **OPEN** status*"

```
<FORMS_METADATA>
<CUSTOM_MESSAGE_SET>
      <MESSAGE ID="123" IDENTIFIER="RENDERER.MESG_WARN"
      TYPE="Warning">
      <PARAMETER NAME="placeholder1" VALUE="[~FrmKILibP_123_1]"/>
      <PARAMETER NAME="placeholder2" VALUE="[~FrmKILibP_124_2]"/>
   </MESSAGE>
   <MESSAGE ID="345" IDENTIFIER="RENDERER.MESG_CONFIR"
   TYPE="Confirm">
      <PARAMETER NAME="placeholder1" VALUE="[~FrmKILibP_123_1]"/>
      <PARAMETER NAME="placeholder2" VALUE="[~FrmKILibP_123_2]"/>
   </MESSAGE>
   <MESSAGE ID="676" IDENTIFIER="RENDERER.IS_LEAF"
   TYPE="Information">
      <PARAMETER NAME="placeholder1" VALUE="[~FrmKILibP_123_1]"/>
      <PARAMETER NAME="placeholder2" VALUE="[~queryString]"/>
```

```
   </MESSAGE>
</CUSTOM_MESSAGE_SET>
</FORMS_METADATA>
```

To overwrite default messages, below configuration is to be done in Form level tags.

```
</FORMS_METADATA>
   <REPLACE_PLATFORM_MESSAGES>
      <REPLACE FFW_MSG_CODE="11851" CUSTOM_MSG_CODE="123"/>
      <REPLACE FFW_MSG_CODE="11852" CUSTOM_MSG_CODE="345"/>
   </REPLACE_PLATFORM_MESSAGES>
</FORMS_METADATA>
```

FFW_MSG_CODE - Message code as given in the messages_locale tables.

CUSTOM_MSG_CODE - Use the custom messages code defined in the Form.

Parameter name "displayCustomMessages" can be passed while performing DB operation to overwrite the return messages with custom message.

<PARAMETER NAME="displayCustomMessages" VALUE="comma separated custom message codes "/>

New function to display custom messages can be used in the rule set.

Function:showCustomMessage('<comma separated custom message codes>')

- **URL control now supports onMouseOver and onMouseOut events to display pop-ups (messages, help text, etc) on MouseOver of the hyperlinks. (Bug 16490182)**

  The existing functionality on control type "25" has been enhanced, where a function is called on MouseOver and MouseOut to display the required messages/ text on mouseover of the hyperlinks.

```
<CONTROL ID="1" TYPE="25">
   <CONTROLPROPS>
      <FUNCTIONS>
         <PARAMETER NAME=" onMouseOver " VALUE="anyfunction(xx)"/>
         <PARAMETER NAME=" onMouseOut " VALUE="anyfunction(xx)"/>
      </FUNCTIONS>
   <CONTROLPROPS>
</CONTROL>
```

- **Masking evaluation feature has been introduced to dynamically evaluate the display mode of controls on export to excel, based on rights. (Bug 16483612)**

  The dynamically evaluation of the display mode of controls (i.e. View, Edit, and Hidden) during UI rendering, has been extended to export option.

- **Checkboxes in a grid can now be hidden and grid rows can be made editable as required. (Bug 16174096)**

  The existing option to select a checkbox in the grid to make row(s) editable has now been enhanced such that the checkboxes in a grid can be hidden and grid rows can be made editable as required. A function “editAllGridRows('formId_gridId')” can be used to make all grid rows editable when checkboxes are hidden.

  Below configuration can be used to hide the check box.

```
<CONTROL ID="1" TYPE="50">
   <CONTROLPROPS>
      <CONTROL_SPECIFIC_TAGS>
         <CHECKBOXREQUIRED DISPLAYREQ="N">Y</CHECKBOXREQUIRED>
      </CONTROL_SPECIFIC_TAGS>
   <CONTROLPROPS>
</CONTROL>
```

  Below function can be used in client validation rules to edit all the gird records.

  Function: editAllGridRows('formId_gridId') // to perform “select All” onclick event.

  Ensure to configure the below control specific tags along with new attribute.

  <MULTISELECTREQUIRED>Y</MULTISELECTREQUIRED>

  <ISEDITABLE>Y</ISEDITABLE>

- **Grid level pagination has been enhanced for direct page navigation using “Jump to Page” option. In addition, grid pagination has the option to specify the number of grid rows to be displayed per page. (Bug 16527110)**

  Refer to the following configuration:

```
<CONTROL ID="1" TYPE="50">
   <CONTROLPROPS>
      <CONTROL_SPECIFIC_TAGS>
         <JUMPTO_PAGEREQUIRED> Y/N </JUMPTO_PAGEREQUIRED> <!--'Y' to
         enable jump to page -->
```

```
<PAGINATION_OPTIONSREQ> Y/N </PAGINATION_OPTIONSREQ> <!--
'Y' to enable number of rows option -->
</CONTROL_SPECIFIC_TAGS>
<CONTROLPROPS>
</CONTROL>
```

In case of "**Three Header Configuration**", below tags is also required to be added:

```
<JUMPTOPAGE ONHEADER="1" ONPANEL="3" ORDER="2"/>
<PAGINATION_OPTIONS ONHEADER="2" ONPANEL="2" ORDER="3"/>
```

- **Currency control properties (currency and amount) are now grouped together as an entity. (Bug 16489936)**

  Currency control enhancement involves grouping of two controls as an entity, especially for Currency and Amount, which are displayed together in UI. This facilitates to specify the amount and select the type of currency wherein the Currency control will be either a text box control (type 7) or Hierarchy control (Type 41- single select), and Amount field will be a number control (type 10).

  Refer to the following configuration:

```
<CONTAINER>
<CONTAINER ID="10" TYPE="1">
<CONTROL_ID CURRENCY_GROUPID="1">122</CONTROL_ID> <!—amount
control (type 10) with groupID -->
<CONTROL_ID CURRENCY_GROUPID="1">1000</CONTROL_ID> <!—currency
control (type 41 or 7) with same groupID -->
<CONTROL_ID>125</CONTROL_ID>
<CONTROL_ID>121</CONTROL_ID>
</CONTAINER>
```

1.8.1 Performance Optimization in Forms Framework

The Forms framework application transactions are basically done through xml data. With concurrent users, the XPathAPI are not processing multiple request simultaneously and there is an impact on performance with high concurrency in terms of increased response time.

Hence, to optimize the performance, you can set the following parameters in Webserver startup:

```
-
Djavax.xml.xpath.XPathFactory=com.sun.org.apache.xpath.internal.jaxp.X
PathFactoryImpl
```

```
-
Dcom.sun.org.apache.xml.internal.dtm.DTMManager=com.sun.org.apache.xml
.internal.dtm.ref.DTMManagerDefault
-
Djavax.xml.parsers.DocumentBuilderFactory=com.sun.org.apache.xerces.in
ternal.jaxp.DocumentBuilderFactoryImpl
```

NOTE: The above performance optimization is verified in Weblogic server with Sun JDK and the same is expected to work for other WebServers also.

1.9 Multiple Language Support (MLS) Utility

Multiple Language Support (MLS) refers to the ability to run multiple languages in the same Applications instance. MLS provides multiple language architecture, while specific language packs provide the individual language translations.

With 7.3.3.0.0 IR, Multiple Language Support (MLS) has been introduced for the following objects:

- Unified Metadata Manager- All Objects.
- Run Rule Framework- Run, Process and Rule definitions.
- Financial Services Applications- Dimension Management:- Attributes, Members, Hierarchies; Filters, Expressions and Object Migration.

The MLS Utility can be invoked through the execution of the following steps with an appropriate parameter. The purpose and the parameters are listed below.

To execute the MLS utility, perform the following steps:

1. Navigate to *$FIC_HOME/MLS_ofsaai* directory of OFSAAI APP tier.
2. Execute the MLS utility. <Command> <parameter>

Available Parameters

MIG

To support MLS, the various data structures that hold metadata objects (as listed above for UMM and RRF) have been modified with 7.3.3.0.0 IR. You need to execute this utility with the parameter **MIG** (Migration) to migrate the metadata to new structures compatible with 7.3.3.0.0 IR, only when installing OFSAA applications on top of that release.

Scenario 1: Upgrade of the system from earlier OFSAAI versions (For example, from OFSAAI 7.3.2.1 to 7.3.3.0.0.). The utility will be executed automatically with the MIG parameter during the upgrade to 7.3.3.0.0 IR.

Scenario 2: If you install any OFSAA applications post installation of 7.3.3.0.0, you need to execute this utility with this parameter to migrate the metadata to new structures of objects compatible with 7.3.3.0.0 IR.

Command:

```
./MLS_ofsaai.sh MIG
```

MES

You need to invoke the utility with this parameter for population of seeded text such as menu labels and popup messages.

You need to execute this utility with this parameter only after you install an OFSAAI language pack, where the language pack has a version lower than the installed OFSAAI software version. For example, you are upgrading your OFSAAI 7.3.2.4 environment with 7.3.2.1.0 LP to 7.3.3.0.0 IR release.

There are additional labels and messages that have been added or modified as part of 7.3.3.0.0 IR. In order to update/ populate the messages_<locale> table with delta records, you need to run the utility with this parameter. Running this utility will copy the incremental set of text to the language-specific **messages_<locale>** tables as a placeholder, so you will see an American English message (default for base install) until the translation is available in language packs.

For example, if you are on OFSAAI 7.3.3.0.0 IR and have installed OFSAAI 7.3.2.1.0 language packs for French and Spanish (since the latest 7.3.3 language pack is not yet available), running the utility with the MES parameter will duplicate the incremental labels and messages from the **messages_en_US** table to the language specific tables for French and Spanish. Later when the 7.3.3.0.0 specific language packs are available, you will be able to update the incremental set of translated strings.

Command:

```
./MLS_ofsaai.sh MES
```

MLS

You need to execute the MLS utility with this parameter in order to pseudo-translate the translatable attributes of user-defined metadata objects. For example, this will copy Names and Descriptions as placeholders in rows for other installed languages.

Please refer to the above list of MLS-enabled OFSAAI object types. After installation of 7.3.3.0.0 IR, the base metadata and translatable data for these object types will have rows for US (American English) only. Executing the utility with the MLS (Multiple Language Support) parameter will duplicate the translatable attributes of the metadata objects for other installed locales.

Command:

```
./MLS_ofsaai.sh MLS
```

Multilingual Support (MLS) architecture has been enabled by segregation of the metadata definitions into non-translatable content (such as Codes), and translatable content (such as Names and Descriptions) for the en_US and other installed languages. The object information has been organized with a single row of base information (containing non-translatable attributes) and multiple associated language rows for holding translatable content (one for each language including a row for en_US.).

For example, you have a Hierarchy which has been defined in en_US (US English) language and then you install 7.3.2.1 language packs for 2 more languages, say fr-FR (French), and es-ES (Spanish). Post execution of the utility with the MLS parameter, the same Hierarchy rule will be available in the two additional languages that you have installed. You can then log into each locale (language) and edit the Hierarchy definition to enter translated text for the Hierarchy Name and Description.

Before you run the utility, you will have only one row for English, for example:

LANGUAGE=US, Description="Organization Hierarchy – Level 1", SOURCE_LANG=US

After you run the utility, you will have two more rows: One for French, and one for Spanish:

LANGUAGE=FR, Description="Organization Hierarchy – Level 1", SOURCE_LANG=US

LANGUAGE=ES, Description="Organization Hierarchy – Level 1", SOURCE_LANG=US

That is, the utility has created a copy of the source row for each target language. The source language in each row is American English (US), the Description data is American English, and the LANGUAGE column contains the target language code. The Hierarchy rule will be available when you log in with any of the above languages. For example, if you log in with French, you can select and edit the object definition, then update the Name and Description to a French translation of the text.

NOTE: As in the above example, running with MLS is necessary for objects (such as a Hierarchy rule) that exist in OFSAAI 7.3.3.0.0 (or later release) prior to applying a language pack for a new locale. If you create a Hierarchy after you apply the language pack, OFSAAI will automatically replicate text (such as Name and Description) into the new locale.

1.10 Config Schema Upload/ Download Settings

This feature is available from OFSAAI 7.3.3.0.0.

The Excel Upload module in DEFQ supports the download and upload of data into configuration schema, in addition to the existing Atomic Schema Upload.

The list of tables to be accessible for the user for Config schema Excel Download/Upload operation is set based on the following entry in CONFIGURATION Table. Against the PARAMNAME ('CONFIG_TABLES_DISPLAY'), PARAMVALUE value in CONFIGURATION table can be manipulated.

1.11 Database Password Reset/ Change

The database password for config schema and atomic schema should be changed periodically for security. The following configurations are required on changing the database passwords:

For Single Tier Installation:

- Delete Reveleus.sec from FIC_HOME/conf.
- Restart the Infrastructure server.
- Enter the latest config schema password when you are prompted.

For Multi Tier Installation:

- Delete Reveleus.sec from FIC_HOME/conf and FIC_HOME/ficdb/conf.
- Restart the Infrastructure server.
- Copy Reveleus.sec from FIC_HOME/conf to FIC_HOME/ficdb/conf.

WebServer Configuration

- For Tomcat Web Server.
  - Update the *Server.xml* file present in FIC_HOME/ficdb/conf in the deployed area with the latest config schema and atomic schema passwords.
- For other Web Servers like WebSphere/Weblogic
  - Update DataSources with the latest config schema and atomic schema passwords.

Unified Metadata Manager Configuration

- Navigate to **Unified Metadata Manager**- **Data Ingestion- Data Sources.** If you have defined any DataSource, update it with the latest passwords.

NOTE: The Data Management Tools and Data Ingestion were previously known as Data Integrator Framework and Warehouse Designer respectively. These new terminologies are applicable only for OFSAAI versions 7.3.2.3.0 and above.

1.12 Configure SSO Authentication

This feature is available from OFSAAI 7.3.3.0.0.

Before you configure SSO authentication, ensure that:

- You have configured OAM (Oracle Access Manager) or equivalent server for SSO user authentication.
- The configured SSO server is up and running and an SSO login page is displayed for users to provide the authentication details.
- The configuration fields are updated correctly before saving the details.
- /<context-name>/login.jsp should be the only resource that is protected.
- The following URLs are there in the excluded URL list in SSO server:

1. MAP_WSDL_LOCATION=$PROTOCOL$://$WEBSERVERHOST$:$WEBSERVERPORT$/$CONTEXT$/mdbObjAppMap?wsdl
2. MDBPUBLISH_EXECUTION_WSDL_LOCATION=$PROTOCOL$://$WEBSERVERHOST$:$WEBSERVERPORT$/$CONTEXT$/mdbPublishExecution?wsdl
3. MIGRATION_OFFLINE_WSDL_LOCATION=$PROTOCOL$://$WEBSERVERHOST$:$WEBSERVERPORT$/$CONTEXT$/offlineMigration?wsdl
4. RULE_EXECUTION_WSDL_LOCATION=$PROTOCOL$://$WEBSERVERHOST$:$WEBSERVERPORT$/$CONTEXT$/ruleExecution?wsdl
5. MRE_WSDL_LOCATION=$PROTOCOL$://$WEBSERVERHOST$:$WEBSERVERPORT$/$CONTEXT$/manageRunExecution?wsdl
6. MODEL_EXECUTION_WSDL_LOCATION=$PROTOCOL$://$WEBSERVERHOST$:$WEBSERVERPORT$/$CONTEXT$/modelExecution?wsdl
7. MIGRATION_OFFLINE_WSDL_LOCATION=$PROTOCOL$://$WEBSERVERHOST$:$WEBSERVERPORT$/$CONTEXT$/offlineMigration?wsdl
8. $PROTOCOL$://$WEBSERVERHOST$:$WEBSERVERPORT$/$CONTEXT$/servlet/com.iflex.fic.ficml.FICMaster

9. $PROTOCOL$://$WEBSERVERHOST$:$WEBSERVERPORT$/$CONTEXT$/servlet/com.iflex.fic.icc.iccwl.ICCComm
10. $PROTOCOL$://$WEBSERVERHOST$:$WEBSERVERPORT$/$CONTEXT$/help.jsp
11. $PROTOCOL$://$WEBSERVERHOST$:$WEBSERVERPORT$/$CONTEXT$/help/*

NOTE: The place holders such as $PROTOCOL$, $WEBSERVERHOST$, $WEBSERVERPORT$, and $CONTEXT$ in the URLs should be updated appropriately.

In case of any errors, the mapped users will not be able to login to the application and you may need to correct the details by logging to the system as **sysadmn**.

For System Users:

- You can access OFSAAI Application using <Protocol (http/https)>://<IP/HOSTNAME>:<SERVLET PORT>/<CONTEXT NAME>/direct_login.jsp.
- You have to select the appropriate user id from the drop-down list.

For Application Users:

- The login page will be their respective SSO Authentication page.
- After successful login, you can change your locale from the **Select Language** link in the application header of the landing page. Move the pointer over the link and select the appropriate language from the listed languages. Based on the locales installed in the application, languages will be displayed.
- The **Change Password** link will not be available in the application header.

To configure SSO authentication and SMS authorization:

1. From the LHS menu, expand **System Configuration** and click **Configuration**. The *Configuration- General Details* window is displayed.
2. Enter the configuration details as tabulated:

Field	Description
SSO Enabled	Select this check box to enable SSO Authentication & SMS Authorization.
Path for Application Packaging	Enter the Application Packaging path where the JSP's generated through DEFQ is saved.
Session Timeout Value (in seconds)	Enter the permitted duration of inactivity after which the session will be automatically timed out and the user will be requested to login again. Note the following:

Field	Description
	▪ The session time out depends on the specified **Session Timeout Value** and web server internal session maintenance. It may vary for different web servers. ▪ For SSO authentication, ensure you set the **Session Timeout Value** equivalent to the configured server session time to avoid improper application behavior after session expired.
Environment Details	Enter the system environment details such as Development, UAT, Production and so on, which are displayed in the application top banner as the “In Setup” info.
SSO Method	Select the required SSO method. These methods are to specify how the user id should be passed from the SSO engine. ▪ **HTTP Request Header** - Returns the value of the specified request header as a string from the server. If selected, you need to specify the header value in **SSO Header Value** field. For example, SM_USER and iv-user header values are supported in OAM. ▪ **HTTP Request Remote User** - Returns the login details of the user who is requesting access to the application remotely. ▪ **HTTP Request User Principal** - Returns a “java.security.Principal” object containing the name of the current authenticated user.
SSO Logout URL	Enter the URL of the page that is to be displayed when users exit the application.
SSO Redirect URL	Enter the URL of the page to which the user should be redirected if some error occurs.
Authentication Type	By default, **SSO Authentication & SMS Authorization** is selected if the **SSO Enabled** check box is selected. You cannot modify it.
Display login details in the header	Select the checkbox to display the login details such as Last Login Date and Last Failed Login Date on the application header. For SSO authentication, the Last Failed Login Date is displayed during the subsequent login for SYSADMN and SYSAUTH users only.
Allow user to login from multiple machines	Select the checkbox to allow concurrent user login.

Field	Description
Hierarchy Security Type	Select the hierarchy security node type from the drop down list. The available options are: ▪ **Group Based Hierarchy Security** ▪ **User Based Hierarchy Security** Depending on the selection, the user/ group details are displayed in the *Hierarchy Security* screen.

3. Click **Save**.

Note the following:

If SSO Authentication is enabled, the following menus will not be available within OFSAAI:

- Administration> Security Management> User Administrator> Profile Maintenance
- Administration> Security Management> System Administrator> Holiday Maintenance
- Administration> Security Management> System Administrator> Restricted Passwords

1.13 Configure Stylesheet

As part of the OFSAAI 7.3.5.0.0 release, user will have two stylesheets theme or UI skin to choose from during product installation:

- Default Existing Blue theme (stylesheetAAI)
- New White & Red Theme (stylesheetAAI2) - This is configurable.

By default, the existing Blue theme is selected. To configure the new white & red theme, set the paraname key 'DEFAULT_AAICSS_INFO' in the Configuration table to ‘stylesheetAAI2’. The paraname value for default theme is ‘stylesheetAAI’.

1.14 Hierarchy Node Internationalization

This enhancement is included as part of OFSAAI 7.3.5.0.0 IR Patch Update. For previous IRs, see Hierarchy Node Internationalization section.

Hierarchy Node Internationalization is an enhancement brought into the Business Hierarchy section of Oracle Financial Services Analytical Applications Infrastructure. This feature is introduced to internationalize the node description of Regular Business Intelligence Enabled (BI) and Parent Child (PC) Hierarchies and to display them in Hierarchy Browser.

Previously, the node descriptions were fetched from the Description column of the Dimension table to facilitate the node generation in REV_LOCALE_HIER table. Each Node has a description. Hierarchy node Internationalization enhancement changes the way in which the descriptions are stored in the REV_LOCALE_HIER. The locale specific node descriptions are fetched from Multi

Language Support table (MLS table). This table holds the node descriptions in all the installed locales, that is, in the locales in which OFSAAI is available.

1.14.1 Scope

The scope of this enhancement is limited to the Hierarchy Browser window. The hierarchies defined are displayed in Hierarchy Browser and the Hierarchy Browser is used in modules such as Unified Metadata Manager, Rules Framework, Metadata Browser, Map Maintenance, Forms Framework, and Hierarchy Maintenance.

1.14.2 Prerequisites

Following are the prerequisites for creating a Hierarchy with Multi Language Support Descriptions:

- The Hierarchy under creation should be either Regular Business Intelligence Enabled (BI) or Parent Child (PC).
- The Multi Language Support table MLS should be created either through Data Model Upload or manually in atomic schema. For more information on MLS table and structure, refer to Multi Language Support (MLS) Table.
- The Description columns used for node generation should be of **Varchar** / **Varchar2** data type.

1.14.3 Multi Language Support (MLS) Table

The MLS table which is meant to provide multi language support can have any name as per Oracle database nomenclature and details of this table need to be configured for further usage. More details about the configuration are explained below:

NOTE: The insertion of data into MLS tables should be performed manually.

1.14.3.1 MLS Table Structure

Following points must be taken care during MLS table creation:

- Description columns on which the Hierarchy definition is based should also be present in the MLS table.
- A column of data type **Varchar** / **Varchar2** should be present in the MLS table. This column should contain the information about the locale (such as **fr_FR**, **ko_KR**). Refer to the MLS Table Configuration section for more details.
- Going forward Dimension related information will be maintained in OFSAAI tables. Before proceeding with the configuration of Dimension and its MLS table, the following master tables need to have data.
  - CSSMS_SEGMENT_MAST

This table holds information about the segments present in OFSAAI and an entry needs to be present in this table for mapping a dimension to a segment/ folder. The Dimension data to be seeded into AAI tables can be mapped to the folder/segment 'DEFAULT'. So the entry for 'DEFAULT' folder needs to be included in this table.

- AAI_OBJ_TYPE_B

  This table holds information about various object types supported in OFSAAI such as Dataset, Business Measure, and so on. For Dimension management, the object type will be DIMENSION.

- AAI_OBJ_TYPE_TL

  This table holds locale specific information about various object types present in OFSAAI. Locale specific information about the object type 'DIMESNION' needs to be added here.

- AAI_OBJ_SUBTYPE_B

  This table holds information about different objects' sub types supported in OFSAAI. The different sub types associated with a 'DIMENSION' object will be mentioned in this table.

- AAI_OBJ_SUBTYPE_TL

  This tables hold locale specific information about various object sub types present in OFSAAI and information on the subtypes of 'DIMESNION' are maintained in this table.

NOTE: Refer to the attached file for more information on the sample data. The data provided in each of these tables is not exhaustive and has been provided as per requirements of Hierarchy Node Localization only.

1.14.3.2 MLS Table Configuration

Consider a Hierarchy "**Income"** defined on a dimension table "DIM_INCOME". The table structure is as indicated:

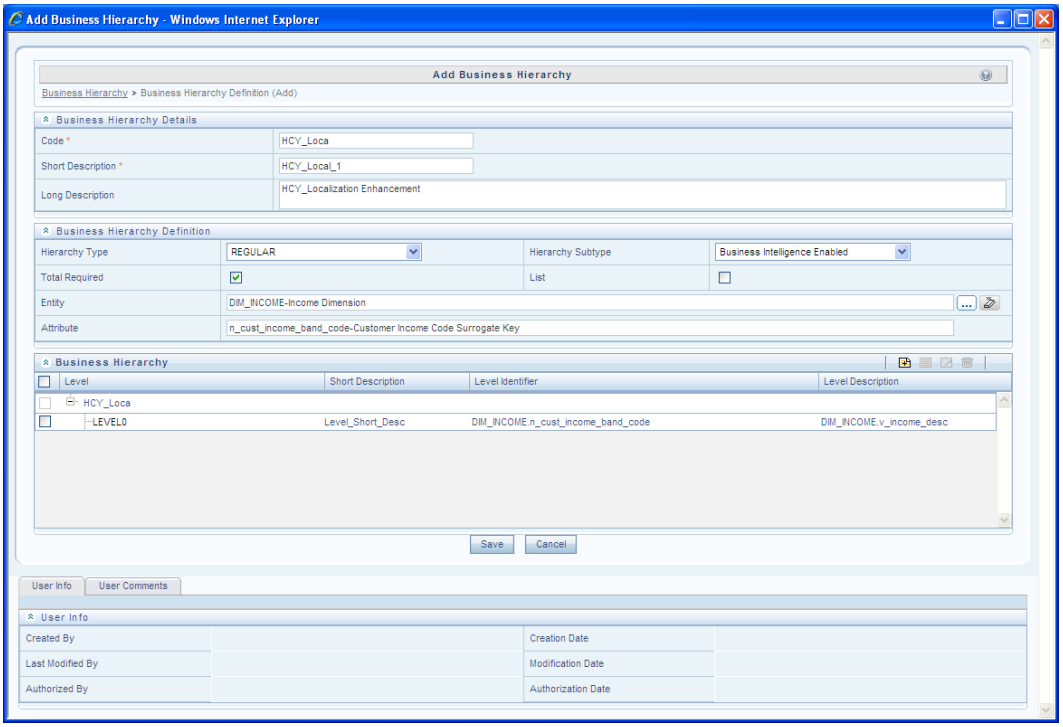
Column Name	Primary Key	Datatype
N_CUST_INCOME_BAND_CODE	PK	Number(5,0)
FIC_MIS_DATE		Date
V_CUST_INCOME_SHORT_DESC		Varchar2(80)
V_INCOME_DESC		Varchar2(80)
N_D_INCOME_UPPER_VALUE		Number(22,3)
N_D_INCOME_LOWER_VALUE		Number(22,3)

The primary key of DIM_INCOME table is PK_DIM_INCOME and is enforced on the column N_CUST_INCOME_BAND_CODE.

An MLS table with name, say “DIM_INCOME_LANG” can be created in the atomic schema to provide MLS support for DIM_INCOME. The structure of this table can be as provided below:

Column Name	Primary Key	Datatype
N_INCOME_BAND_CODE	PK	Number(5,0)
LOCALE_CD		Varchar2(10)
V_CUST_INCOME_SHORT_DESC		Varchar2(80)

The following figure represents the **Income** Hierarchy definition:



The MLS table corresponding to the Dimension DIM_INCOME can be created as mentioned below:

- Create a table to provide MLS support for the Dimension DIM_INCOME. Let’s say the name of the table is DIM_INCOME_LANG. This table to provide MLS related information for DIM_INCOME needs to be configured:
  - AAI_OBJECT_B

This table registers information about an AAI object. Since Dimension is considered as an AAI object, the data corresponding to the Dimension DIM_INCOME needs to be maintained in this table.

- AAI_OBJECT_TL

  This table holds locale specific information about an object in AAI. So locale specific information pertaining to the Dimension, DIM_INCOME, needs to be maintained in this table.

- AAI_DIMENSION

  This table will provide further information about the DIMENSION table. Information such as whether the data in dimension table is in PC structure, whether the members are acquired in the dimension, and so on are maintained in this table.

- AAI_DIM_META_TABLE

  This is the metadata table for a DIMESNION. Information about the table such as the MLS table meant for the Dimension, the hierarchy table, the attribute table, and so on will be maintained in this table.

- AAI_DIM_META_COLUMN

  This table provides information about various columns that will be used for a Dimension table. From Hierarchy Node Localization perspective, the name of the locale column which will hold locale information needs to be maintained here.

- AAI_DIM_META_JOIN

  This table holds information about the columns that will be used for joining the Dimension table with other tables such as the MLS table, Hierarchy table, Attribute table, and so on. Here multiple join conditions can be specified as well. Refer to the attached excel for further information on providing joining columns information with respect to Hierarchy Node Localization.


HNL_Data.xls

The following table displays sample data which can be populated in DIM_INCOME_MLS table in a setup where there are 2 locales installed say, English (en_US) and Chinese (zh_CN).

N_CUST_BAND_CODE	V_INCOME_DESC	LOCALE_CD
1	AAA	en_US
2	BBB	en_US
1	CCC	zh_CN

N_CUST_BAND_CODE	V_INCOME_DESC	LOCALE_CD
2	DDD	zh_CN

Note the following:

- In Regular BI enabled and PC Hierarchies, the Level Description expression **should not** contain columns with Number or Date data types. The inclusion of such a column in the Level Description expression would prevent the Business Hierarchy from generating nodes.
- There is no concept of **default** locale. Whenever a Hierarchy is saved, the translated node descriptions present in MLS table are saved in the corresponding columns of the REV_LOCALE_HIER table depending on the availability of translated values in the MLS table.
- The inclusion or exclusion of nodes from a Hierarchy will be reflected in Forms once the Hierarchy is resaved.

1.14.4 Node Generation Process

During Hierarchy definition, the nodes get generated depending on the structure of the Hierarchy. Node generation is possible in the following two scenarios:

- Node Generation when <DIM>_MLS Table is Present & Configured
- Node Generation when <DIM>_MLS Table is Not Present or Not Configured

1.14.4.1 Node Generation when MLS Table is Present and Configured

When MLS table is present, the nodes are generated by fetching the Description from the MLS table. Thus, entry in the Description columns of MLS table is mandatory.

1.14.4.2 Node Generation when MLS Table is Not Present or Not Configured

When MLS table is not present, by default the nodes are generated by fetching the Description from the Dimension table.

1.14.5 Configure Mapper for Multiple Locales

This step is optional and is required if *Node Generation Process* explained in the above section is done.

To configure mapper for multiple locales:

1. Duplicate the data in REVELEUS_MASTER table with different locales in LOCALE_ID column.
2. Translate V_OBJECT_DESC column in REVELEUS_MASTER table to the desired locale.
3. Duplicate data in LOCALE_ID column in REV_MAST_MAP_ITEMS table for different LOCALE_ID.

Example:

An existing mapper namely **Mapper A** (created in any locale) can be translated into other locales as indicated in the following example:

1. Login to the configuration schema and duplicate the data in REVELEUS_MASTER table by changing the locale in LOCALE_ID column.
2. Change V_OBJECT_DESC for the corresponding locale in REVELEUS_MASTER table.
3. Duplicate the data in REV_MAST_MAP_ITEMS table by changing locale in LOCALE_ID column.

NOTE: 2nd and 4th steps need to be performed for all the locales to which you wish to translate mapper A.

1.14.6 Update Nodes in Existing Regular BI and PC Hierarchies

Currently, the node description is generated only for one locale on which the Hierarchy is saved. With the introduction of Hierarchy Node Internationalization, the nodes will be generated in all the installed locales.

To generate the localized node descriptions for the existing Hierarchies, you need to edit and re-save the Hierarchies post MLS table creation and configuration. You can also mass update the existing Hierarchies from **Administration** > **Save Metadata** section. The node description data for all the installed locales will be populated in REV_LOCALE_HIER table.

NOTE: If an SCD (Slowly Changing Dimension) is configured on a Dimension table, synchronize the new entries with the corresponding MLS table also.

1.14.7 Limitations

If the Hierarchies are accessed via Modeling Framework module, the node descriptions of the same will be displayed only in English, despite the locale you have logged in to the application.

1.15 Performance Optimization Setting for RRF Module

This is an enhancement introduced in 7.3.5.2.0 ML release.

The Rule execution engine has been enhanced to support partition as a filter in the Rule Merge query. To achieve this, a new table called **AAI_OBJ_QUERY_OPTIMIZATION** is introduced. You need to configure this table as explained in the following table:

Column Name	Description	Value
V_OBJ_CODE	Rule	Rule(PR2_RULE_B.V_RULE_NAME)
V_INFODOM_CODE	Infodom Code	Infodom
V_OBJ_TYPE	Rule	Rule(RL)
V_EXECUTION_MODE	Type of query used while executing.	MERGE- Merge statement will be used
F_USE_PARTITION	If partition is used as a filter	Y/N This filter will be taken care by Rule execution engine.
F_USE_ROWID	If ROWID is used other than primary key in MERGE. This is used only for MERGE query execution.	Not Applicable
V_MERGE_HINT	Used for MERGE or INSERT hint.	Not Applicable
V_SELECT_HINT	Used for SELECT hint	Not Applicable
V_PRE_SCRIPT	Used for alter statements executed before rule execution	Not Applicable
V_POST_SCRIPT	Used for alter statements executed after rule execution.	Not Applicable

You also need to add the partition table name and column name in the V_TABLE_NAME and V_COLUMN_NAME respectively in the REV_TAB_PARTITIONS table.

OFSAAI
Administration Guide

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
www.oracle.com/us/industries/financial-services/