Informatica® Data Archive
6.4 HotFix 2

Enterprise Data Manager Guide

Informatica Data Archive Enterprise Data Manager Guide
6.4 HotFix 2
April 2016

Publication Date: 2018-06-09

Table of Contents

Preface

The *Informatica Data Archive Enterprise Data Manager* is written for Database Administrators (DBA) that perform key tasks involving data backup, restore, and retrieval using the Informatica Data Archive user interface. This guide assumes you have knowledge of your operating systems, relational database concepts, and the database engines, flat files, or mainframe systems in your environment.

Informatica Resources

Informatica Network

Informatica Network hosts Informatica Global Customer Support, the Informatica Knowledge Base, and other product resources. To access Informatica Network, visit https://network.informatica.com.

As a member, you can:

- Access all of your Informatica resources in one place.
- Search the Knowledge Base for product resources, including documentation, FAQs, and best practices.
- View product availability information.
- Review your support cases.
- Find your local Informatica User Group Network and collaborate with your peers.

As a member, you can:

- Access all of your Informatica resources in one place.
- Search the Knowledge Base for product resources, including documentation, FAQs, and best practices.
- View product availability information.
- Find your local Informatica User Group Network and collaborate with your peers.

Informatica Knowledge Base

Use the Informatica Knowledge Base to search Informatica Network for product resources such as documentation, how-to articles, best practices, and PAMs.

To access the Knowledge Base, visit https://kb.informatica.com. If you have questions, comments, or ideas about the Knowledge Base, contact the Informatica Knowledge Base team at KB_Feedback@informatica.com.

Informatica Documentation

To get the latest documentation for your product, browse the Informatica Knowledge Base at https://kb.informatica.com/_layouts/ProductDocumentation/Page/ProductDocumentSearch.aspx.

If you have questions, comments, or ideas about this documentation, contact the Informatica Documentation team through email at infa_documentation@informatica.com.

Informatica Product Availability Matrixes

Product Availability Matrixes (PAMs) indicate the versions of operating systems, databases, and other types of data sources and targets that a product release supports. If you are an Informatica Network member, you can access PAMs at https://network.informatica.com/community/informatica-network/product-availability-matrices.

Informatica Velocity

Informatica Velocity is a collection of tips and best practices developed by Informatica Professional Services. Developed from the real-world experience of hundreds of data management projects, Informatica Velocity represents the collective knowledge of our consultants who have worked with organizations from around the world to plan, develop, deploy, and maintain successful data management solutions.

If you are an Informatica Network member, you can access Informatica Velocity resources at http://velocity.informatica.com.

If you have questions, comments, or ideas about Informatica Velocity, contact Informatica Professional Services at ips@informatica.com.

Informatica Marketplace

The Informatica Marketplace is a forum where you can find solutions that augment, extend, or enhance your Informatica implementations. By leveraging any of the hundreds of solutions from Informatica developers and partners, you can improve your productivity and speed up time to implementation on your projects. You can access Informatica Marketplace at https://marketplace.informatica.com.

Informatica Global Customer Support

You can contact a Global Support Center by telephone or through Online Support on Informatica Network.

To find your local Informatica Global Customer Support telephone number, visit the Informatica website at the following link:
http://www.informatica.com/us/services-and-training/support-services/global-support-centers.

If you are an Informatica Network member, you can use Online Support at http://network.informatica.com.

CHAPTER 1

Introduction

This chapter includes the following topics:

Enterprise Data Manager Overview

Enterprise Data Manager provides centralized management of all application data throughout an organization. Application archiving, records management, database subsectioning, sensitive data masking, and application retirement are all managed from a unified, integrated platform.

Informatica Enterprise Data Manager Use Cases

Enterprise Data Manager is an integrated tool which enables the DBA to build metadata for Data Archive. Enterprise Data Manager adapts dynamically and extends related functionality for Metadata definitions.

The rest of this document discusses how metadata can be developed for Data Archive using the Enterprise Data Manager interface.

Data Archive-Specific Functions

Enterprise Data Manager allows the following metadata definitions for installation:

- Defining Accelerators for Transaction data in the form of nodes in the hierarchical representation for an Application like its Application Version, Application Modules, Entities, and Interims or Tables.
- Defining Reporting Statements, Entity Parameters or List of Values, and Business rules for Entities.
- Defining Indexes and Constraints on Tables.

CHAPTER 2

Working with Enterprise Data Manager

This chapter includes the following topics:

Working with Enterprise Data Manager Overview

To access Enterprise Data Manager, double-click the Enterprise Data Manager.bat file in the Enterprise Data Manager installation directory. By default, Enterprise Data Manager is installed in the following location: `C:\Informatica\Enterprise Data Manager`.

Note: You must have execute permissions on javaws.exe to launch Enterprise Data Manager. You can find javaws.exe in your Java installation directory.

Authenticating and Authorizing Enterprise Data Manager

At startup, the user is prompted with a dialog box to specify information about the location where the Home Schema was installed. The following information needs to be specified here:

Parameter	Possible Values
Connection Name	A drop down list containing valid Connection Strings specified so far. Selecting an item from this list automatically populates information in the text boxes described below.
Database Type	A list containing supported Database drivers is available.
Database Host	IP Address or Host name of the machine where the Home Schema was installed.

Parameter	Possible Values
Database Port	Port number for connecting to the database.
Database Name	Unique name or SID for database where the Home Schema is installed.
User name	Login name for Home Schema.
Password	Password for accessing information in the Home Schema.

Enterprise Data Manager Authorization

Enterprise Data Manager authorizes the user with a valid login profile for accessing the Enterprise Data Manager Server according to the Roles granted by the Administrator from the Data Archive user interface.

Note: User Accounts can only be created from the Data Archive Web user interface.

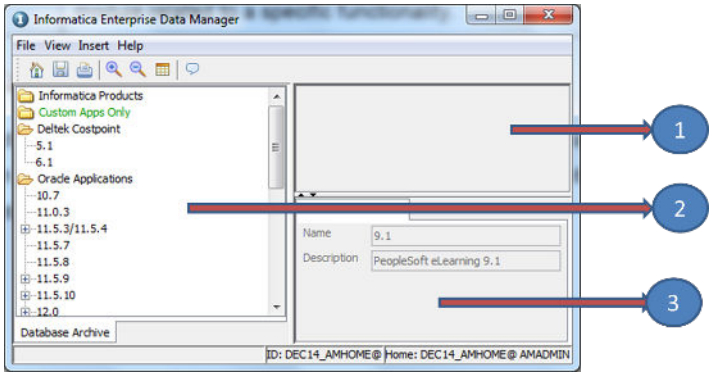
User Authorization information for each (User) Role is discussed in the following table:

User Role	Privileges
Server Viewer	Allows one to view Metadata in the Enterprise Data Manager.
Server Developer	Allows one to develop (and view) Metadata for the Enterprise Data Manager.

Application Interface

The Enterprise Data Manager Application interface is defined in such a way that it enables a user to perform a majority of the metadata development tasks in a single screen.

The interface consists of three panes as shown in the following figure. They are:



1. Explorer Pane
2. Canvas Pane
3. Details Pane

Each component of the application interface is explained in the following sections.

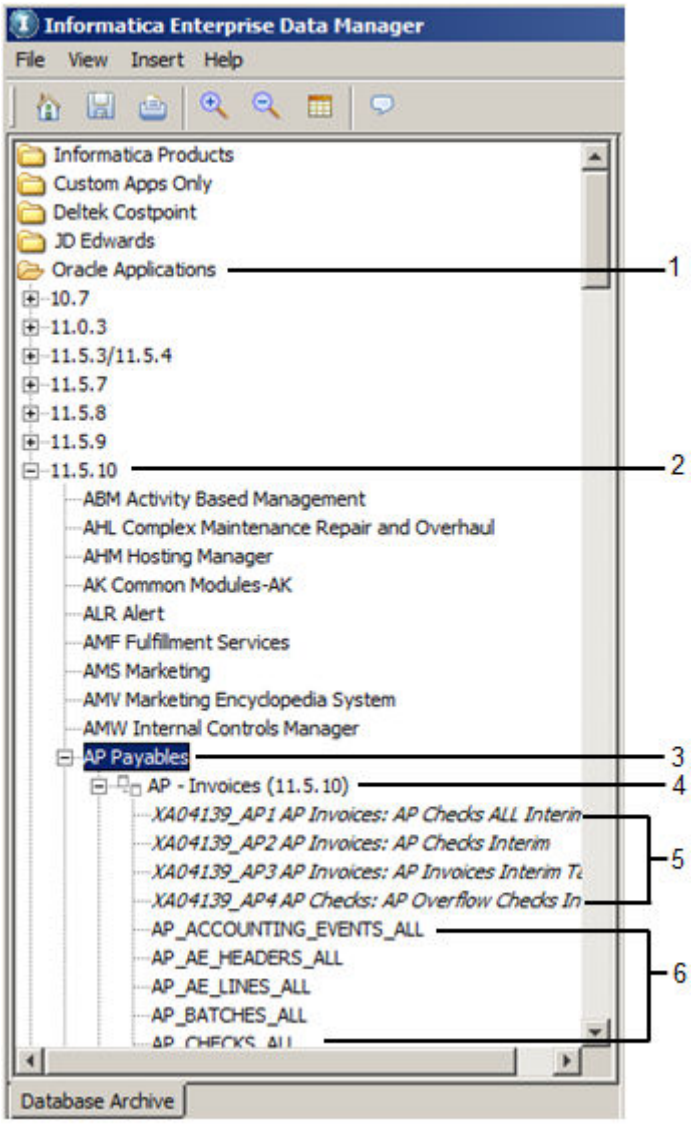
Explorer Pane

Data Archive internally employs a hierarchical structure for its components, and carries out all operations based on it. The **Explorer** pane shows the tree view of the components.

Nodes in the Application Hierarchy

Enterprise Data Manager organizes nodes in an hierarchical structure on the **Explorer** pane.

The following image shows the different levels in the hierarchical structure:



1. Application
2. Application Version
3. Application Module
4. Entity
5. Interim Tables
6. Business Tables

The **Explorer** pane displays the following nodes:

Application

Custom or brand name ERP applications, such as Oracle E-Business Suite and PeopleSoft.

Application Version

Supported versions for a particular application.

Application Module

Schemas for an application version.

Entity

Group of tables whose data collectively comprise a set of business transactions. Each table in the entity connects to one or more other tables through a primary and foreign key relationship.

Interim Tables

Temporary tables generated for an entity during an archive.

Business Tables

All related tables created to define an entity. Business tables appear at the same level of the hierarchy as the interim table.

Canvas Pane

The **Canvas** pane shows the diagram view of the relationships between tables within an entity.

The following list explains what the colors on the **Canvas** pane indicate:

- A blue background for a table name indicates a business table.
- A gray arrow indicates a relationship between tables.
- An orange arrow within connecting tables indicates a referential integrity constraint.
- A green background for a table name indicates a interim table.
- A green line indicates a relationship that includes interim tables.

Details Pane

The **Details** pane contains detailed information related a node or component.

You can select a node, such as an application version, entity, or interim table from the **Explorer** pane and view the details in the **Details** pane.

You can select a component such as a table or relationship from the **Canvas** pane and view the details in the **Details** pane.

Menu Structure

Menu Sequence	Function
File > Connect to Repository	Create a new connection profile or use an existing one with information such as database Driver, Host, Port, SID, User name and Password (for Schema). This option also allows you to connect to a different Data Archive repository. When you connect to EDM from the Accelerators menu, this option is disabled.
File > Login	Provide User name and Password for logging into Enterprise Data Manager.

Menu Sequence	Function
File > Connect to ID Repository	Specify connection profile for database that stores sequences for unique IDs generated while creating nodes and sub-nodes in the application tree.
File > Save	Save changes to the Data Archive repository.
File > Import Metadata from Database	Integrate one or more tables from a schema into the existing Metadata.
File > Discover Keys and Relationships	Discover key columns and relationships for an application.
File > Import	Import Data Model accelerators or Data Archive accelerators for standard or custom applications. This option also allows bulk import of (Primary and Referential) integrity constraints as well as Translation Columns.
File > Export	Export Data Model accelerators or Data Archive accelerators for standard or custom applications.
File > Print	Print “Diagram View” (Canvas Pane) of currently selected Entity.
File > Exit	Exit from Enterprise Data Manager.
View > Refresh	Refresh the information on the Enterprise Data Manager pane.
View > Zoom In	Magnify Entity illustration in the Canvas Pane.
View > Zoom Out	Shrink Entity illustration in the Canvas Pane.
View > Columns > All	View Entity tables with all Columns and Relationships.
View > Columns > Related	View Entity tables only with respective Columns that participate in a Relationship.
View > Columns > None	View only Table names, without any Columns.
View > Default Layout	Switch back to the default layout.
Help > About	Retrieve information about current (Enterprise Data Manager) version and build.

Enterprise Data Manager Toolbar

A toolbar is displayed at the top left corner of the application, with common functions:

Toolbar Button (and tool tips)	Function
Connection to Repository	Connect to an existing Connection Profile or specify a new one. This is synonymous with File >Connection to Repository. This option also allows you to connect to a different Data Archive repository. When you connect to EDM from the Accelerators menu, this option is disabled.
(Save)	Commit Metadata changes to the Data Archive repository. This is synonymous with File > Save.
(Print)	Print “Diagram View” for currently selected Entity. This is synonymous with File > Print.
(Zoom In)	Enlarge Entity illustration in the Canvas Pane by a zoom factor. This is synonymous with View > Zoom In.
(Zoom Out)	Minimize Entity illustration in the Canvas Pane by a zoom factor. This is synonymous with View > Zoom Out.
(Columns toggle)	Toggle between different views for displaying columns in Entity tables (in Canvas Pane), i.e. all columns, related columns and no columns. This is synonymous with View > Columns > All, View > Columns > Related and View > Columns > None respectively.
(Annotate)	Create an annotation for a node in the application tree.

Tabbed Navigation

When a view is selected from the View menu, a tab appears at the bottom left corner of the window.

Error Messages

An error message is displayed as a pop-up, whenever an operation initiated by the user fails to complete. A notification for the same is also displayed in the Status Bar. The Application usually does not allow the user to proceed further unless the error is resolved, so as to ensure Metadata consistency.

Status Bar

A Status Bar at the bottom of the window displays the following information:

- Status Messages

  An error that occurs on account of a user operation (for example, invalid values for a particular WHERE clause), or the status of an operation initiated by a user (for example, import or export of data) displays as a notification to the extreme left in the status bar.

- Connection to ID Repository

  Next to the Status Messages, the Schema which stores information on ID sequences for nodes created in the hierarchical structure of Enterprise Data Manager Metadata is displayed. This is generally negative for Custom Components (user-defined Application or Entity) and positive for Standard Components (offered by ERP).

- Home Schema

  The Home Schema which stores the actual Metadata that can be altered using the Enterprise Data Manager displays on the extreme right.

Temporary Save Mechanism

Any changes made to the Metadata are not directly committed to the Home Schema, unless the user explicitly saves it by clicking on the Save button (Enterprise Data Manager Toolbar) or uses the menu File > Save. Any node added anywhere to a application hierarchy is initially saved with a * before its name (in the Explorer Pane).

Database or Product Support

The Information Lifecycle Management Platform develops accelerators for the following third-party application servers:

- Amdocs
- JD Edwards
- Oracle E-Business
- PeopleSoft
- Siebel
- Custom applications

Data Archive is customized for applications on the following databases:

- Oracle
- IBM DB2 LUW
- IBM DB2 zOS
- Microsoft SQL Server
- MySQL
- Sybase ASE

CHAPTER 3

Enterprise Data Manager

This chapter includes the following topics:

Customizing Data Archive Metadata Example

A practical example of Data Archive Metadata definition is as follows:

1. Import the metadata from the source database. Create a node for respective Application under the intended Application Version. For example, add “BOM Bills of Material” as Application to Oracle Version 11.5.10.
2. Add an Entity to created Application module. For example: “BOM - Standard Cost Update (11.5.10)” to the Application module “BOM Bills of Material.”
3. Define one or more parameters for the entity defined above. Examples of parameters are indicated in the following table:

Order	Column	Name	Data type	Required	List of Values	Security group
1	p_cost_org_id	Cost Organization	Number		INV Organization	
2	p_update_date	Update Date	Time	true	INV Accounting Period (PF-8)	Organization Unit

The final SQL query for List of Values “INV Organization” and “INV Accounting Period (PF-8)” reads as follows:

List of Value Name	SQL Test Query
INV Organization	SELECT A.organization_name “Organization”, A.organization_code “Code”, TO_CHAR(A.organization_id) FROM ORG_ORGANIZATION_DEFINITIONS A WHERE A.inventory_enabled_flag = 'Y' ORDER BY A.organization_name, A.organization_name
INV Accounting Period (PF-8)	SELECT A.period_name “Period Name “, T.organization_name “Organization Name”, T.organization_code “Organization Code”, A.period_name “Period Name “, T.organization_code “Organization Code”, T.organization_name “Organization Name”, TO_CHAR(A.schedule_close_date, 'DD-MON-YYYY') FROM ORG_ACCT_PERIODS A, ORG_ORGANIZATION_DEFINITIONS T WHERE A.open_flag = 'N' AND A.organization_id = T.organization_id AND T.organization_id = NVL(:params.param1,T.organization_id)

Note: It is assumed that these List of Values have been created and available under the “List of Value tab” of the Explorer Pane.

4. Insert default Steps for Archive Definition under the Entity “BOM - Standard Cost Update (11.5.10).”
5. Add required Interim Tables. For example, for the Entity “BOM - Standard Cost Update (11.5.10)”, add an Interim Table with the following FROM and WHERE clauses:

Clause Type	Statement
FROM	bom.Cst_Cost_Updates C, inv.Mtl_Parameters Mp
WHERE	C.Organization_Id =decode(:p_cost_org_id,null,c.organization_id,:p_cost_org_id) And C.Organization_Id = Mp.Organization_Id And trunc(Update_Date) <= :P_Update_Date and Mp.Primary_Cost_Method=1

6. Add Tables (to the “Tables” tab of this Interim Table) that will participate in the archive job execution, and define their respective INSERT, DELETE and UPDATE statements. For this example, specify the following information:

Order	Name	Insert Statement	Delete Statement
1	CST_STD_COST_ADJ_VALUES	a.cost_update_id in (select cost_update_id from XA04156_BOM1 x where x.master_org_flag = 'Y' and x.purgeable_flag = 'Y')	a.cost_update_id in (select cost_update_id from XA04156_BOM1 x where x.master_org_flag = 'Y' and x.purgeable_flag = 'Y')
2	CST_STANDARD_COSTS	A.cost_update_id IN (Select cost_update_id From XA04156_BOM1 x where x.purgeable_flag = 'Y')	A.cost_update_id IN (Select cost_update_id From XA04156_BOM1 x where x.purgeable_flag = 'Y')

Order	Name	Insert Statement	Delete Statement
3	CST_ELEMENTAL_COSTS	A.cost_update_id IN (Select cost_update_id From XA04156_BOM1 x where x.purgeable_flag = 'Y')	A.cost_update_id IN (Select cost_update_id From XA04156_BOM1 x where x.purgeable_flag = 'Y')
4	CST_COST_UPDATES	A.cost_update_id IN (Select cost_update_id From XA04156_BOM1 x where x.purgeable_flag = 'Y')	A.cost_update_id IN (Select cost_update_id From XA04156_BOM1 x where x.purgeable_flag = 'Y')

7. Add the Default Columns. For example, for the Interim table created above, define the following Default Columns:

Order	Name	Type	Length	PK	Select Clause
1	cost_update_id	Number		true	C.Cost_Update_Id
2	cost_update_date	Date		false	C.Update_Date
3	org_id	Number		false	mp.Organization_Id
4	master_org_flag	Char	1	false	decode(Mp.Organization_Id, mp.cost_organization_id,'Y','N')
5	purgeable_flag	Char	1	false	'Y'
6	stats_date	Date		false	

8. Define Business Rules. For example, on the Interim Table generated for “BOM - Standard Cost Update,” define the Business Rule “HAS_NONPURGEABLE_STD_COSTS” with the following information:
   - Description. Cost Update has standard costs that are not being purged in this cycle.
   - Business rule order. 1
   - Condition. A.cost_update_id in (select cost_update_id From bom.cst_standard_costs b where not exists (select 1 from XA04156_BOM1 x where x.purgeable_flag = 'Y' and b.cost_update_id = x.cost_update_id and x.org_id=b.organization_id))
9. Generate default Indexes.
10. Specify reporting Statements. For example:

Order	Statement Label	Literals	Group
1	COST_UPDT_DT	A	Primary

A Parameter i.e. “cost_update_date” is specified here for the Statement which relies on “INTERIM_VALUE.”

Basic Operations

Basic operations for Data Archive Metadata definition are included in the subsections below.

Metadata Import for SAP Applications

The Import Metadata job runs steps that are specific to SAP applications when you indicate that the source is an SAP application. The Import Metadata job reads metadata for all SAP tables in the database, imports the data to the ILM repository, and adds SAP-specific table types.

When you import metadata from SAP applications, the Import Metadata job uses the parameters that you configured for the job to connect to the database. The job reads metadata from the database and the SAP catalog tables for all pool, cluster, and transparent tables, except for the tables that are included in the pre-packaged entities for SAP. Metadata for the transparent HR PCL1-PCL5 cluster tables, the transparent STXL table, and attachment tables are included in the pre-packaged entities. The job also reads metadata for ADK files.

The job adds an SAP-specific table type to the imported metadata in the ILM repository. The table type is required to differentiate SAP tables. When you run the Retirement Auto Entity Creation job, the job uses the table types to determine the tables to add to entities and the entity step configuration.

Table Types

When you import metadata from SAP applications, the Import Metadata job adds table types to the corresponding metadata tables in the ILM repository. The Retirement Auto Entity Creation job uses the table types to determine the tables to add to entities and the procedures the job adds to the entity steps.

Table types are required to determine the data source of the tables in the entity. Entities can only include one source for the Insert into Archive Tables step. Some SAP tables store data in both the database and in an external file system. A table is included in two separate entities if the table has data in the database and archived data in ADK files in an external file system. One entity includes the database as the source. The other entity includes the external file system as the source for the archived data.

The Import Metadata job adds one of the following table types to the imported metadata in the ILM repository:

Converted Transparent Table with Archived Data

Logical tables that store archived data in ADK files in an external file system. This table type is for archived data from transparent HR PCL1-PCL5 cluster tables and the STXL text table.

Pool Cluster

Pool and cluster logical tables that store data in the database in an SAP proprietary format.

Pool Cluster with Archived Data

Pool and cluster logical tables that store archived data in ADK files in an external file system.

Transparent Table

Physical tables that store data in the database.

Transparent Table with Archived Data

Physical tables that store archived data in ADK files in an external file system.

The pre-packaged entities for SAP include tables with the **Converted Transparent Tables** and **Attachments** table types. The job does not add these table types as the table types are already included in the pre-packaged entities.

Importing Metadata from the Database

Import metadata from the source database to the ILM repository. You must import metadata before you can run an archive or retirement job.

For SAP application retirement, you must initiate the metadata import from a customer-defined application. Run the Copy Application Version for Retirement job to copy the pre-packaged SAP application to a customer-defined application.

1. Select an application version and click **File** > **Import Metadata from Database**.

   The **Connect to Import Metadata from Database** dialog box appears.

2. Enter the connection details.

   To create a connection for Informix sources, use the following syntax for the service name:

   ```
   <server_name>;databasename=<dbname>
   ```

3. Click **OK**.

   The **Import Metadata from Database Wizard** appears.

4. Select the schemas to import metadata from.

   Click the down arrow button to insert the tables in to the Selected box. Use the control keyboard options to select multiple tables or all tables.

   Note: Tables that have special characters in the table name, do not appear in the list. Special characters include: `~`, `!`, `@`, `#`, `$`, `?`, `%`, `^`, `&`, `*`, `_`, `+`, `–`, and `=`.

5. Click **Next**.

6. Choose one of the following options to run the job:

   - **Submit Import Metadata as a Background Job**. Runs the job in the background. If you run the job in the background, you can continue to perform other tasks. Run the job in the background if you have a large volume of metadata to import and to avoid memory errors.

     Required to import metadata from SAP applications.

   - **Continue Import Metadata through EDM**. Runs the job in the foreground. If you run the job in the foreground, you must wait until the job completes to perform an other task. Additionally, if you run the job in the foreground and you have a large volume of metadata to import, you may receive memory errors. You may want to run the job in the foreground if you have a low volume of metadata to import.

7. Click **Next**.

8. Enter the mining parameters.

   To import metadata from databases for SAP applications, select **SAP Application Metadata**. The Import Metadata job runs steps that are specific to SAP applications. If you import from SAP applications and do not select the parameter, the job will not import metadata for pool tables, cluster tables, and tables that store data in ADK files.

9. Click **Finish**.

10. Click **OK**.

Adding or Deleting an Application

An application can be added from **Insert** > **New Application**.

Right-click **Delete Application** to delete an application.

Adding or Deleting Application Version

An application version can be added from **Insert** > **Application Version**. An option for the same is also available in the context menu for the corresponding application.

Right-click **Delete Application Version** to delete an application version.

Adding or Deleting an Application Module

An application module can be added from **Insert** > **New Application Module**. A context menu option for the corresponding application version, called Add Application Module is also available for similar purposes.

Right-click **Delete an Application Module** to delete an application module.

Adding or Deleting an Entity

An entity can be added from **Insert** > **New Entity**. A context menu option for the corresponding application module, called Add New Entity is also available for the same.

To delete an entity from the context menu option, select **Delete Entity**.

Copying and Pasting Modules

To copy all the modules for a particular Application Version, right-click the corresponding Application Version and select **Copy Archive Modules**.

To paste the modules into another Application Version, select **Paste Archive Modules**.

Creating or Deleting Reporting Statements

Reporting Statements are created for Interim Tables in an Entity Definition, in order to generate Detail reports when the same is requested during an Data Archive Definition (specified from Data Archive user interface).

To define a reporting statement:

1. Select the parent Application Version.
2. Click **Insert > Statement**. The New Statement dialog box appears.
3. Specify a Label for the Statement.
4. Specify whether values in a report must be generated using an SQL query or taken from the Interim table.
5. Specify SQL query in case SQL is selected.
6. Click **Finish** to save the Statement. The Statement will appear under the “Statements” tab in the Explorer Pane.
7. Click **Save** to commit changes to Home Schema.

To delete a statement, navigate to the **Statements** tab in the Explorer Pane. Select **Delete Statement**) from the context menu.

Creating a List of Values

Note: List of Values are created in order to associate them with a Parameter for an Entity.

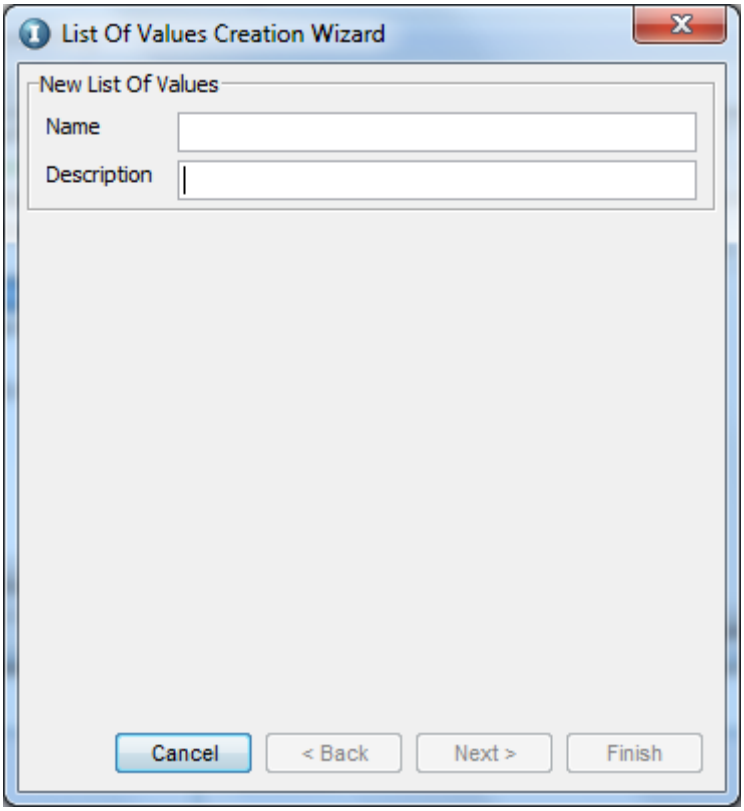
During Archive Definition, the user is prompted to supply values for these Parameters from a List of Values. This list is populated internally by building an SQL query according to an List of Value definition, as discussed in this Section.

An List of Value can be defined for an Application Module from Insert > List of Values.

To create an List of Value:

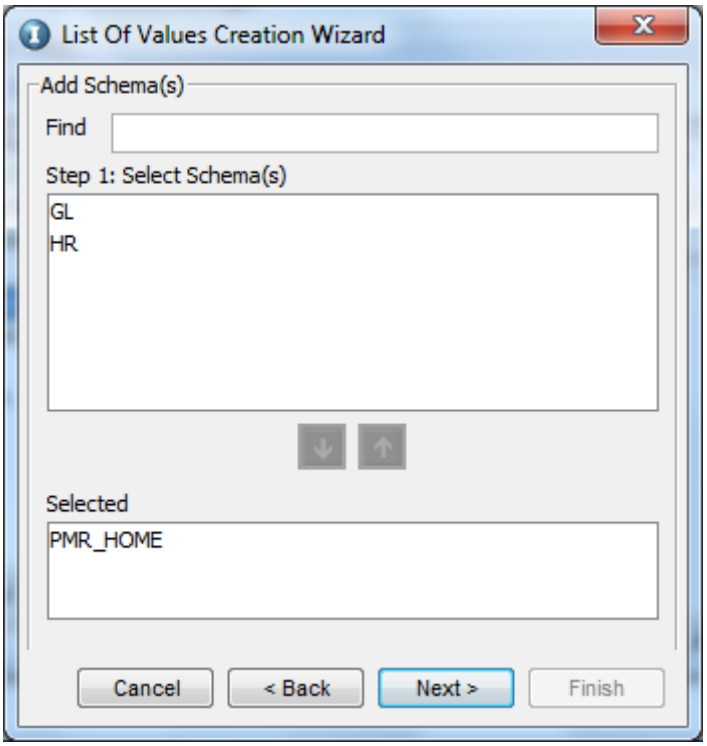
1. Select the required Application Module and click **Insert** > **List of Values** to invoke the List of Values Creation Wizard.

   The first page of the wizard appears as shown in the following figure:



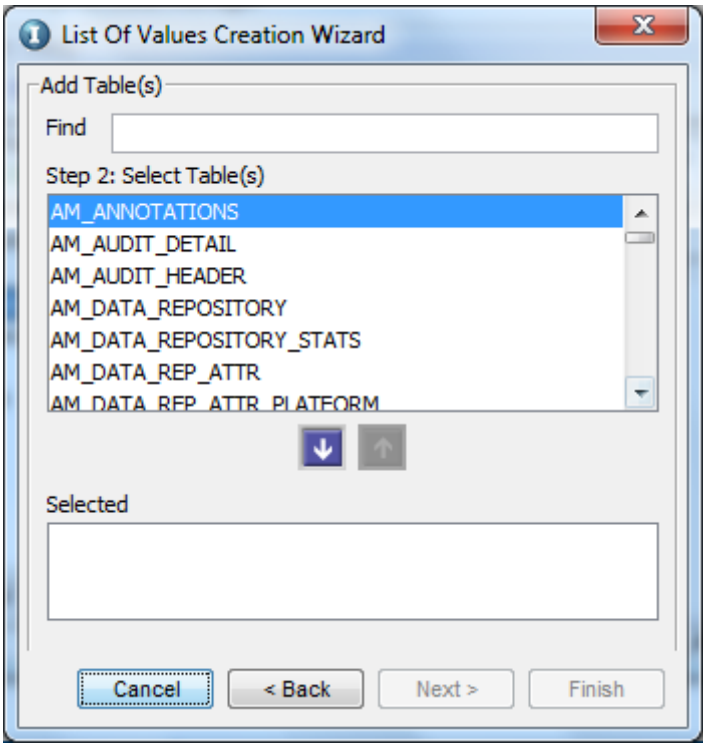
2. Specify a Name and Description for the List of Value and click on **Next**.

   The following page appears:



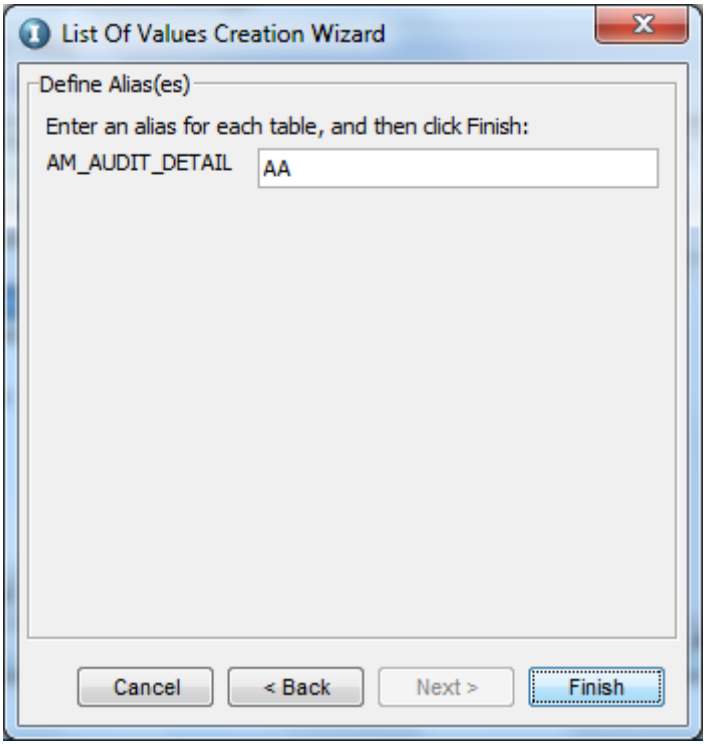
3. Select a Schema from the “Schemas” pane and click Down Arrow to add it to the “Selected” pane. Contrarily, selecting the (pre-selected) Schema and clicking on Up Arrow deselects it (Step 2). When done, click **Next**.

   The following page appears:



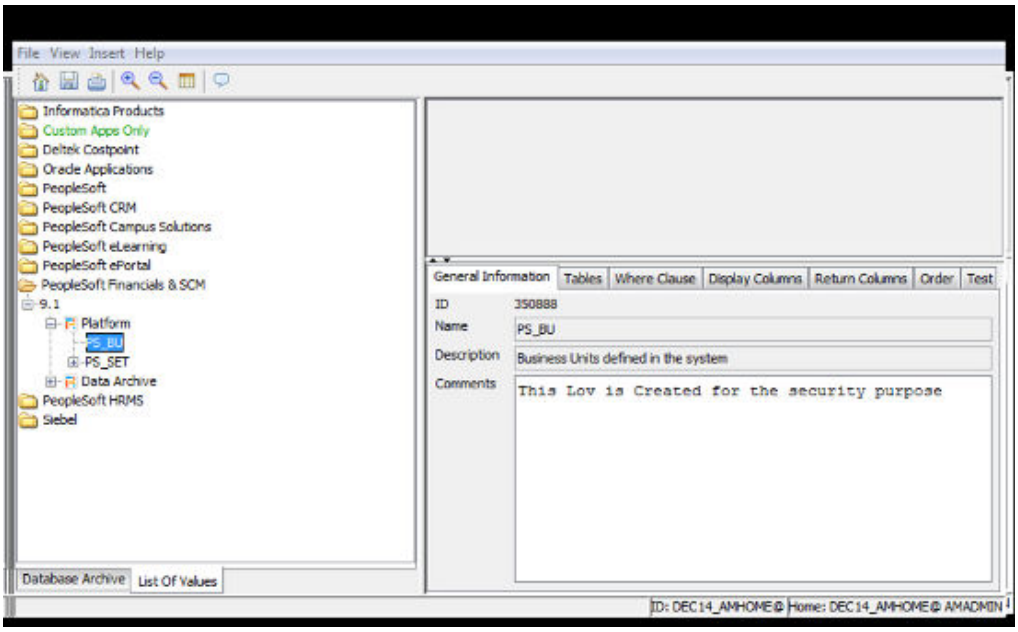
4. Select Business Tables (Step 3) using the Down Arrow (or clicking on Up Arrow to deselect it) and click **Next**.

   The following page appears:



5. Specify aliases for each of the selected Tables (Step 4) and click **Finish**.

   On completion of the wizard, the List of Value is created and appears in the “List of Value” tab of the Explorer Pane. Each List of Value appears with the following tabs in the Details Pane:



List of Value Detail Tab	Function
General Information	Basic information about the List of Value including: - Auto generated ID - Name (specified in List of Values Creation Wizard) - Description (specified in List of Values Creation Wizard) - Comments
Tables	Selected Tables with the following information: - Order of Tables according to selection from List of Values Creation Wizard. Order can be changed using Up Arrow and Down Arrow buttons available to the right. - Table Name - Table Alias (as specified in List of Values Creation Wizard). - Check boxes to determine whether the table is to be included in the FROM clause of the SQL query. Tables can be added or deleted using Add and Delete buttons, on the extreme right.
Where Clause	The WHERE clause for filtering data.

List of Value Detail Tab	Function
Display Columns	Columns to select in the SQL query. Each Column is displayed with the following information: - Order of selection. Order can be changed using Up Arrow and Down Arrow buttons available on the right. - Expression (if any) to be executed on the Column. For example, TOCHAR(). Generally, the Column name is specified here. - Data Type for the Column. - Label / Alias for the column (Column heading) Columns can be added or deleted using Add and Delete buttons, on the extreme right.
Return Columns	Return value for the SQL query (not displayed in the List of Value).
Order	Columns for The ORDER BY clause for sorting data.
Test	The SQL query, built from the information specified above. In case invalid values are supplied and the query fails to comply with SQL syntax, relevant Error messages are displayed in this area.

6. Click the **Save** button to save changes made to the List of Value from its Details Pane.

   The Detail Pane of an List of Value is displayed, with the “Test” tab selected, displaying the SQL query which will fetch values for this List of Value.

   Note that the selected Tables are displayed in the Canvas Pane.

Creating a List of Value Variant

To create a list of value variant, select **Create List of Value Variant** from the context menu of the respective List of Value.

Updating Entity Information

You can update information for an entity from the **Details** pane.

To view details about an entity, click the entity from the **Explorer** pane. The details for the entity appear on the **Details** pane within the **General Information**, **Parameters**, and **Steps** tabs.

The following table describes the information in each tab:

Entity Detail Tab	Function
General Information	The **General Information** tab lists the following pre-specified information: - Auto-generated ID - Entity name - Entity description - Driving table name
Parameters	The **Parameters** tab contains the archive definition parameters. You can set the following parameters: - Order of Parameters. Change the order of parameters by using the up and down arrows on the right. - Parameter Name. Enter a name for the parameter. - Parameter Title. Enter a title to display in the user interface. - Parameter Data Type. Specify the SQL data type for the parameter. - Required. Click the box if the entity specification step for an archive definition requires the parameter. - The List of Values. Specify a name of the previously created list of values for this parameter. - Security Group. Enforce constraints on a particular list of value.
Steps	The **Steps** tab lists steps that the job engine uses to run the archive job. Click **Insert Default Steps** to insert the standard set and the order of steps. You can configure the following columns on the **Steps** tab: - Order. To change the order of execution for SQL statements, use the up arrow and down arrow buttons on the right. - Action. Select an action from the menu. - Interim Table. Select an interim table. - Entity Table. If the action relates to operations on a business table, select a business table. - Procedure. If the action relates to operations on a procedure, enter a procedure. - Business Rule Order. This order matches the order specified for business rules for a particular interim table. - Enabled. Select boxes in this column to include the step in the archive job.

To delete entities, right-click the entity and select **Delete Entity**.

Copying Archived Entities to Customize

To customize an entity that you previously archived to the Data Vault or a database, create a copy of the entity and then add or remove tables.

1. Go to **View** > **Database Archive**.
2. Navigate to the entity.
3. Right-click the entity and select **Copy Entity**.
4. Right-click the same application module folder you copied the entity from and select **Paste Entity**.
5. Rename the copied entity.
6. Add or delete tables to the entity.

Copying Standard Entity Metadata for Data Vault Search

After you retire an entity, you can use the prebuilt application-accelerator metadata to search and examine data in this entity through Data Vault search.

To search for retired entities in Data Vault and to keep the table relationships that existed at the time of retirement, you must copy the application module from a prebuilt application version to the custom application version folder. This triggers the Copy Entity from PFV job to run. When the job completes, assign the appropriate Data Vault access role to the entities to access these entities in Data Vault search.

You can copy entities between versions in an application but not between applications.

1. Go to **View** > **Database Archive**.
2. Navigate to the application version folder.
3. Right-click the application version folder and select **Copy Entities from Application Version**.

   The **Copy Entities from Application Version** window appears.
4. Select the application version that contains the application module you want to copy.
5. Enter a descriptive prefix to identify the application module in Data Vault search.
6. Click **OK**.

   The **Copy Entity from PFV** job begins.
7. From the Data Archive interface, go to **Jobs** > **Monitor Jobs**.
8. After the **Copy Entity from PFV** job completes, expand the row and view the job log.

   The job log shows details such as mismatched tables between the accelerator and source database.
9. Assign Data Vault access roles to the entities.

   The data in these entities can be searched and examined through Data Vault search.

Adding and Deleting Interim Tables

To add interim tables, click on **New Interim** from the respective entity's context menu.

To delete an interim table, click on **Delete Interim** from the context menu.

Enter the interim table name in uppercase letters.

Inserting and Removing Business Tables

To add a business table to an entity, click on **Insert Table** from the entity's context menu.

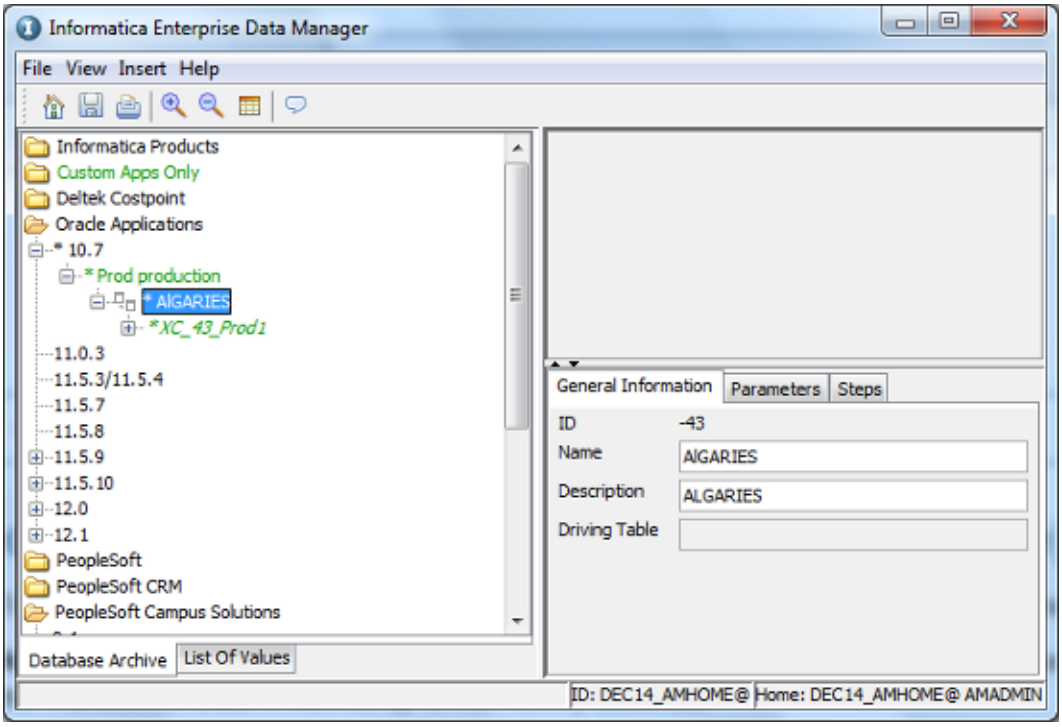
Select a schema and target tables.

To view a list of available tables that were mined for the respective application, select the **Show only tables that have been mined for the application** check box.

To disassociate a table from an entity, select **Remove Table** from the former's context menu.

Business Tables

Each Business Table pertaining to an Entity is accompanied by the following information, as displayed in its Details Pane.



Each tab is discussed in detail in the following table:

Table Detail Tab	Function
General Information	Usually contains the Table name.
Columns	Columns available in the Table, displayed with the following info: - Name - Data type - Check box to identify participation in the Primary key constraint - Check box to identify whether it is Nullable - Check box to identify whether it is a Translation Column
Constraints	Available Integrity Constraints for that particular Table

Specifying Translation Columns (Independent Archive)

Translation Columns, specified here are meant for Data Archive job executions containing Independent Archive.

Note: In case of Independent Archive, Constraints as well as Data is exported, whereas for Database archive, only the Transaction Data is backed up to the History database.

A Column in a Table is a candidate for Translation when data needs to be extracted from a second Table based on a Referential integrity constraint.

Also, there might be circumstances where data is extracted from the first and a third Table based on similar Foreign key constraints, linking the first with second, and second with third Table. In such a case, the Column in the second Table is De-referenced to make way for displaying a Column in the third Table.

Note: Column Translation occurs when two Tables participate in a JOIN. Columns are De-Referenced to include Constraints when more than two Tables are involved in data extraction through a JOIN query.

De-Referencing can be achieved from the “Columns” tab of the Details Pane for a Table.

The steps involved in specifying a Translation Column are summarized below:

1. Select the checkbox under “Translation” for that Column.
   - Application Version: “Oracle Apps 11.5.10”
   - Application Module: “AP Payables”
   - Entity: “AP_INV_APRVL_HIST_ALL”
   - Table: “AP_ENCUMBRANCE_LINES_ALL”

Name	Type	Private	Nullable	Translation
ENCUMBRANCE_TYPE_ID	NUMBER	False	False	True

2. When the *Column* is selected for *Translation*, the following information is displayed in its sub-pane:

Table Column	Constraint	Where Clause	Parent Table Alias	Ref Table Alias	Display Column	De-Reference
ENCUMBRANCE_TYPE_ID	AP_ENCUMBRANCE_LINES_FK3 (GL_ENCUMBRANCE_TYPES)		A	B	ENCUMBRANCE_TYPE	False

3. In case a third *Table* is included in the JOIN, then the *De-Reference* checkbox beside the *Column* (“ENCUMBERANCE_TYPE” in this example) must be checked and used to link it to a third *Table*. The *Column* for a *De-Reference* must also participate in a Referential Integrity constraint.

Specifying Integrity Constraints

The following information is available from the Details Pane when a particular Table is selected from the Explorer Pane:

Constraint Detail Tab	Function
General Information	Displays name and description for the Constraint.
Columns	Displays Columns in the selected Table and with its Name, Data type and checkbox to indicate if it serves as a Primary Key for that table.
Constraints	The constraint, with the following information: - Name: A unique name for the constraint. - Type: This can take one of the values, CHECK, PRIMARY, REFERENTIAL and UNIQUE. - Constraint: A condition for the Constraint can be specified here. For example CHECK (NumParts > 5). - Enabled: There is also an option to disable a constraint, where it will not participate in defining data integrity of the respective Table.

To view existing constraints on tables and specify new constraints, go to **View** > **Constraints**. The **Constraints** tab in the Explorer Pane appears with the schema and its related tables in a hierarchy.

You can also create constraints between tables by using a drag-drop mechanism for target columns from the Canvas pane.

Interim Tables

Adding an Interim table to an Entity is discussed in "Adding and Deleting Interim Tables" on page 30.

The following information can be modified for the Metadata as discussed in the table below:

Interim Detail Tab	Function
General Information	An (auto-generated) Name and (pre-specified) Description of the Interim table is displayed.
Tables	All Tables involved in the Data Archive job executions are listed here. Tables can be added or deleted using Add and Delete. The following information is specified for each Table involved: - Order: INSERT, UPDATE and DELETE statements are executed on the Business Tables in the order specified. This order of execution can be changed using the Up Arrow and Down Arrow buttons. - Table Name - Insert Statement: Generally executed while copying ROWS into Data Destination during a job execution. - Delete Statement: Generally executed while purging ROWS from the Data Source in a job execution. - Update Statements, if any.

Interim Detail Tab	Function
Default Columns	A number of Default Columns can be specified for the Interim tables, each accompanied with the representation of a column from a Business Table. This is specified using the values indicated below: - Order: Order of Columns in the Interim Table. Use using Up Arrow and Down Arrow to promote / demote a Column. - Name: Column name - Type: Data Type for records to be inserted. This generally takes the Data Type of the Business Table column as specified in the SELECT clause. - Length: Maximum number of characters allowed, according to specified Data Type. - PK: Allows one to assign a particular column as a Primary Key of the Interim Table. This is facilitated through check boxes. - Select Clause: Column names from one or more Business Tables, whose values are fetched to populate the respective Default Column for Interim Table. **Note:** When an Interim table is first created the Columns “stats_date”, “org_id” and “purgeable_flag” are created by default. While “stats_date” and “org_id” can be assigned Business Table Columns in the SELECT clause, “purgeable_flag” consists of a flag to determine whether the record is archive-able. From Clause: FROM clause in the SQL query, specifying target Columns to be selected while building the Interim Table. Where Clause: WHERE clause to filter ROWS for Columns selected while building the Interim Table.
Business Rules	Business rules can be specified, which occupy one column each in the interim table (as a flag which indicates its validation status). **Note:** The value inserted in the “purgeable_flag” column is actually a binary summation of the values in the Columns complying to each of the Business Rules Business Rules are defined with the following information: - Order: Order of columns in the Interim table - Name: Column name in interim table which stores validation flag for this business rule. - Description: A short description of the business rule. - Business Rule Order: Order of execution of UPDATE statements on Interim Table intended for applying that Business Rule to the data. - Type: Type of clause used to construct the business rule. A drop-down list containing FULL, INLINE, LOOP, SET and WHERE clauses is available for selection. - Purge Qualification: Statement for the clause type selected above. - Exclude from Evaluation: When this checked, the corresponding business rule is excluded from the binary summation that determines the value in “purgeable_flag” column. - Condition: Actual conditional statement that identifies the business rule.

Interim Detail Tab	Function
Indexes	An INDEX can be specified by clicking on Add and deleted using Delete. Order of execution of Indexes can be altered using Up Arrow and Down Arrow. These buttons are available to the right of the Details Pane. Generally, default Indexes are generated by clicking on Insert Default Index, at the bottom of the Details Pane.
Report Statements	Statements for generating Detail Reports can be specified here. The following information is specified in order to associate a Report Statement with an Interim table: - Order: Order of display for Columns. - Statement Label: A drop down list consisting of pre-created (Reporting) Statements is available to choose from. - Literals (Comma Separated): Labels for Display Columns can be specified here - Group: A Column can be part of a Primary or Secondary group. This basically specifies the placement of Columns in the Detail report of an Archive Definition. The following information is automatically displayed on the right, when a Statement is selected. - Parameters: Allows one to specify Parameters in the same order as they occur in the SQL query. - Statement: Displays the SQL query or "INTERIM_VALUE", whichever was specified during Statement creation.

Entities

An entity is a group of business tables that may or may not be related, and an interim table. An entity with related tables can also contain information on table relationships, a master table, and reference tables.

You create entities to perform the following tasks in Data Archive:

- To archive data to a database.
- To archive data to the Data Vault.
- To retire an application to the Data Vault.
- To use the Data Discovery portal in Data Archive.
- To create and run data visualization reports.
- To partition the source database.
- To archive attachments.

Before You Create an Entity

You must understand and verify the table relationships of the tables you want to archive before you create an entity.

When you create an entity, the entity creation process creates the data model based on the constraints you enable. If the constraints are not correctly enabled, then the data in the entity might not be archived correctly.

For example, you archive an entity with a table that contains multiple paths to the driving table. When you archive the entity, you might encounter the following issues:

- The archive job archives records more than once.
- The Delete from Source step fails.

To avoid these issues, verify that the constraints for each table you plan to include in the entity are correct. If you have a child table related to more than one parent table, you must define the constraint conditions to specify the rows that belong to that relationship. After you verify that the constraints are correct, create the entity.

Verifying Constraints Before You Create an Entity

Before you create an entity, verify that the constraints for each table in the entity are correct.

1. Click **View** > **Constraints**.
2. On the **Explorer** pane, expand the application node, application version, and application module to view the tables.
3. On the **Details** pane, select the **Constraints** tab.
4. Verify if the constraints are correct.
5. If necessary, disable or modify a constraint.

Multiple Entity Generation

You can use wizards to generate multiple entities at a time. Generate multiple entities to reduce the amount of time required to create entities manually. Manual creation of entities can be time-consuming. You can generate entities for live archiving or for retirement.

The method that you select to generate entities depends on the archive scenario. Use one of the following methods to generate multiple entities:

Automatic entity generation

Use the Multiple Entity Creation wizard to generate entities for live archiving and for data discovery. The wizard uses the constraints in the ILM repository to generate the entities. You can use the entities for archive projects and for Data Discovery portal searches.

Retirement entity generation

Use the **Generate Retirement Entity** wizard to generate entities for retirement. The wizard uses the imported schema and table metadata to generate the entities. You can use the entities for retirement projects. If you want to discover data from the Data Discovery portal, you must generate entities based on constraints.

Entity Naming Convention

Data Archive names an entity based on the name of the driving table in the entity and the number of entities in the application version.

When you create an entity, Data Archive assigns a unique name to the entity. The entity name contains two parts. The first part is the name of the driving table for that entity. The second part is a number. The number indicates the entity count at the time the entity was created.

For example, you have application modules A, B, and C in an application version. The application modules have the following entity history:

- Module A has one entity named `Orders_001`.

- Module B has one entity named `Customers_003`. Module B had an entity named `CustomerAddress_002` that you deleted.
- Module C does not have entities.

You create an entity in Module C so you can archive a group of tables. You select the table Supplies to include in the entity. You specify the Supplies table to be the driving table in the entity. Data Archive assigns the name Supplies_004 to the entity. The Supplies part of the entity name indicates the driving table in the entity. The 004 part of the entity name indicates that this is the fourth entity in the application version.

Verifying Constraints Before You Create an Entity

Before you create an entity, verify that the constraints for each table in the entity are correct.

1. Click **View** > **Constraints**.
2. On the **Explorer** pane, expand the application node, application version, and application module to view the tables.
3. On the **Details** pane, select the **Constraints** tab.
4. Verify if the constraints are correct.
5. If necessary, disable or modify a constraint.

Automatic Entity Generation

Use the **Multiple Entity Creation** wizard to generate entities for live archiving and for data discovery. The wizard uses the constraints in the ILM repository to generate the entities. You can use the entities for archive projects and for Data Discovery portal searches. You might want to generate entities for custom applications.

Entity generation for custom applications includes the following high-level steps:

1. Import metadata from the source database.
2. Discover table relationships.
3. Review and import the table relationships that you want to create constraints from.
4. Use the Multiple Entity Creation wizard to create entities based on the constraints that you imported from the table relationship discovery process.

Generating Entities Automatically

Use the Multiple Entity Creation wizard to generate entities. You might want to generate entities after you use one of the table relationship discovery methods to create constraints.

The ILM repository must contain constraints for the imported source metadata.

1. Click **View** > **Constraints**.
2. In the **Explorer** pane, expand the application node to view the application versions.
3. Right-click the application version that you want to create entities in, and select **Auto-create new entity(s)**.

   The **Multiple Entity Creation** wizard appears.
4. Select the schemas, and click the down arrow.
5. Click **Next**.
6. Select the table types, and click the down arrow.
7. Click **Next**.

8. Select the tables and click the down arrow.

   Note: If the source database is Oracle, you can archive tables that contain up to 998 columns.
9. Click **Next**.
10. Select the child tables that you want to include, and click the down arrow.
11. Click **Next**.
12. Select the table types for the child tables that you selected, and click the down arrow.
13. Click **Finish**.

    The **Entity Creation Process** page displays the status of the creation process. A message appears that prompts you to save the data.

Retirement Entity Generation

Use the Generate Retirement Entity wizard to automatically generate entities for retirement projects. When you use the wizard, you specify parameters for the Retirement Auto Entity Creation job. The job simultaneously creates entities for all tables in the application that you want to retire. The job replaces the time-intensive task of creating entities one at a time.

The entities do not include constraints and are only available for data extract. If you want to use the Data Discovery portal to search the retired data, you must generate entities based on constraints. To generate entities based on constraints, use the Multiple Entity Creation wizard.

Retirement Auto Entity Creation Job

The Retirement Auto Entity Creation job creates entities for all tables in the application that you want to retire. You use the entities in a retirement project. The job uses the imported schema and table metadata in the ILM repository and the parameters that you configure in the Generate Retirement Entity wizard to automatically generate entities.

The job uses the following naming convention to create entities:

```
<prefix>_<entity number>_<suffix>
```

The job generates the entity number in sequential order. For example, if you use `Finance` for the prefix, the job generates `Finance_1` for the first entity, `Finance_2` as the second entity, and so on. The job uses the meta schema ID and the meta table ID to determine the order in which it adds tables to the entity.

The number of entities the job creates depends on the parameters that you provide. You specify the number of tables in each entity, the prefix and suffix for the entity name, and whether you want to add tables from more than one schema.

The best practice is to create one entity for every 200 tables in the application. The job can create entities with up to 999 tables in each entity. By default, if the retirement application contains multiple schemas, the job creates an entity with tables from more than one schema. An entity can have tables from multiple schemas if the maximum number of tables in an entity exceeds the amount of tables in the first schema. For example, the application has two schemas with 150 tables in each schema. You configure the job to create 200 tables in each entity. The job uses 150 tables from the first schema and then 50 tables from the second schema to create the first entity.

If you do not want to combine tables from multiple schemas in one entity, you can disable this feature. In the example above, the job creates two entities. The first entity includes 150 tables from the first schema. The second entity includes 150 tables from the second schema.

Retirement Auto Entity Creation Job Parameters

Enter the parameters when you run the Generate Retirement Entity wizard to automatically generate entities for application retirement. The wizard runs the Retirement Auto Entity Creation job.

The Retirement Auto Entity Creation job includes the following parameters:

Number of Tables in each Entity

Required. Maximum number of tables to include in each entity. You can create up to 999 tables in each entity. The recommended number of tables in each entity is 200.

Use up to three digits to specify the number of tables to create. For example, enter 1, 01, or 001 to specify one table.

Prefix for Entity

Required. Text to append before the entity name. The prefix helps identify what application the entity corresponds to. The full entity name, including the prefix and suffix, cannot exceed 45 characters.

Suffix for Entity

Optional. Text to append after the entity name. The full entity name, including the prefix and suffix, cannot exceed 45 characters.

Group Tables beyond the Schemas

Optional. Determines whether the entity can include tables from multiple schemas. If enabled, the wizard creates the maximum number of tables that you specify from across schemas. If disabled, the wizard creates the entity with the maximum number of tables within one schema. Default is enabled.

Retirement Auto Entity Creation for SAP Application Retirement

The Retirement Auto Entity Creation job runs steps that are specific to SAP applications. When you run the Import Metadata job for SAP sources, the job adds a table type to imported metadata tables in the ILM repository. The Retirement Entity Creation job uses the table types to determine the tables the job adds to entities and the configuration of the default entity steps.

By default, the job inserts default entity steps. However, only the Insert into Archive Tables step is enabled. The Insert into Archive Tables step determines from where the retirement job reads data for tables in the entity. By default, the step reads data directly from the database. SAP applications includes special tables that the retirement job cannot read directly from the database. The job cannot read data directly from the database for pool and cluster tables and any tables that have archived data in ADK files. The tables require a additional procedure for the Insert into Archive Tables step.

In addition, the retirement job may need to read data from the same table from multiples sources. SAP tables can store data in both the database and in archived ADK files in an external file system. For a table that has data in the database and in ADK files, the table is included in two separate entities. Separate entities are required because entities can only use one source to read data from. One entity reads data from the database. The other entity reads data from the ADK files in the external file system. Although the retirement job uses two entities to read data for the same table, the table data is consolidated when the Data Vault Loader job loads the data in to the Data Vault. You only see one table in the Data Vault.

The job uses the following naming convention to create the entities for SAP applications:

```
<Prefix>_<Entity Number>_<Suffix>_<SAP-Specific Suffix>
```

The following table describes the parameters in the naming convention:

Parameter	Description
Prefix	User-defined entry in the job parameters.
Entity Number	Job-defined entry.
Suffix	User-defined entry in the job parameters.
SAP-Specific Suffix	Job-defined entry. The job appends the following SAP-specific suffixes: - `_SAP`. For entities that include pool and cluster tables. - `_SAP_ADK`. For entities that include tables with archived data.

Table Types for SAP Application Retirement

The Retirement Entity Creation job adds tables to entities based table types in the ILM repository. The Import Metadata job adds SAP-specific table types when you import metadata from SAP applications.

The Retirement Entity Creation job uses the following table types to determine how to create entities:

Converted transparent table with archived data

Converted transparent tables are logical tables that are not stored in the database. This table type is specifically for transparent HR PCL1-PCL5 cluster tables and the STXL text table that store data in ADK files, in addition to data in the database. The job inserts tables with this table type into a separate entity and inserts the `ArchiveSAPArchivedTable` procedure for the Insert into Archive Table entity step. The procedure overrides the default entity step to connect to the database. The procedure calls an ABAP function module to read data from the external storage that stores the ADK files.

Note that tables with the converted transparent table types are already included in the pre-packaged SAP entities. The Retirement Auto Entity Creation job does not add those table types to entities because the tables are already included in pre-packaged SAP entities. However, the job needs to add tables to new entities if the tables store data in ADK files.

Pool cluster

Pool and cluster tables are logical tables that store data in the database in an SAP proprietary format. The job adds the pool and cluster tables to entities and inserts the `ArchiveSAPLogicalTable` procedure for the Insert into Archive Table entity step. The procedure overrides the default entity step to connect to the database. The procedure calls an ABAP function module to read data from the SAP application layer.

Pool cluster with archived data

Pool and cluster tables with archived data are pool and cluster tables that store data in ADK files, in addition to data in the database. The job inserts tables with this table type into a separate entity and inserts the `ArchiveSAPArchivedTable` procedure for the Insert into Archive Table entity step. The procedure overrides the default entity step to connect to the database. The procedure calls an ABAP function module to read data from the external storage that stores the ADK files.

Transparent table

Transparent tables are physical tables that are stored in the database. The job adds transparent tables to entities and inserts the standard Insert into Table entity step. The step reads data directly from the database.

Transparent table with archived data

Transparent tables with archived data are transparent tables that store data in ADK files, in addition to data in the database. The job inserts tables with this table type into a separate entity and inserts the `ArchiveSAPArchivedTable` procedure for the Insert into Archive Table entity step. The procedure overrides the default entity step to connect to the database. The procedure calls an ABAP function module to read data from the external storage that stores the ADK files.

Generating Retirement Entities

Use the Generate Retirement Entity wizard to automatically generate entities for application retirement.

1. Click **View** > **Constraints** or **View** > **Data Vault**.
2. Navigate to the application version.
3. Right-click on the application version and choose **Generate Retirement Entities**.

   The **Generate Retirement Entity** wizard appears.
4. Enter the retirement entity generation parameters.
5. Click **OK**.

   You receive a message that the wizard started the Retirement Entity Creation job in the background. You can use the job number in the message to monitor the job status in Data Archive.

After the job completes, exit and log back in to view the generated entities.

Troubleshooting Entities

After I create an entity, a message informs me that there are multiple paths for tables in the entity.

First, verify the constraints for each table listed in the message. If the constraints are incorrect, delete the entity. Then, disable or modify the constraints. Then, create the entity.

To verify the constraints, perform the following steps:

1. Click **View** > **Constraints**.
2. On the **Explorer** pane, select a table listed in the message.
3. On the **Details** pane, select the **Constraints** tab.
4. Verify if the constraints are correct.
5. If the constraints are correct, you can ignore the message.
6. If the constraints are incorrect, you must perform the following steps:
   a. Delete the entity.
   b. Disable or modify the constraints for the tables you want in the entity.
   c. Create the entity.
7. Repeat steps 4 to 6 for each table listed in the message.

To delete the entity, perform the following steps:

1. Click **View** > **Database Archive**.
2. Right-click the entity.
3. Select **Delete Entity**.
4. Click **Delete** to confirm the delete action.

To disable a constraint, perform the following steps:

1. Click **View** > **Constraints**.
2. On the **Explorer** pane, select a table listed in the message.
3. On the **Details** pane, select the **Constraints** tab.
4. Click the **Enabled** check box.

   Note: If the constraint you are deleting is a self-referential physical constraint, note that this constraint still exists in the source database. After you archive the entity, you must delete this data from the source database.
5. Click **View** > **Refresh**.

To modify a constraint, perform the following steps:

1. Click **View** > **Constraints**.
2. On the **Explorer** pane, select a table listed in the message.
3. On the **Details** pane, select the **Constraints** tab.
4. Click **Constraint Condition**.
5. In the text box that appears, specify the rows that apply to the constraint.
6. Click **View** > **Refresh**.

To create the entity, perform the following steps:

1. Click **View** > **Refresh**.
2. Click **View** > **Constraints**.
3. On the **Explorer** pane, select the driving table.
4. Right-click the driving table and select **New Entity**.
5. Specify the child tables and table types in the **Multiple Entity Creation Wizard**.

Exporting Accelerators

Enterprise Data Manager allows one to export Metadata from existing ERP applications, commonly referred to as Accelerators. This is achieved through one of the File >Export submenu items.

When you export accelerators, run the export task as a background job if the volume of metadata is high to avoid memory-related issues.

The following Metadata objects can be exported from Archive for the selected Application Version:

Data Model Metadata

Type	Description
Both	Exports Custom and Standard Data Model metadata.
Custom Only	Export Data Model Metadata for used-defined objects.
Standard Only	Imports Data Model Metadata for pre-defined standard objects in the ERP Application.

Data Archive Metadata (Custom/Standard Entities)

Type	Description
Both	Export only Data Archive specific data for user-defined (Custom) and pre-defined (Standard) Objects. Refer options below.
Custom Only	Exports only Data Archive specific Objects that were created by user.
Standard Only	Exports Standard Objects specific only to Data Archive.

Component (Standard Data)

Type	Description
Component Only	Exports Data Archive specific data for selected Component (Entity / Application).

Exporting and Importing a Custom Product Family Version

You can export a custom product family version to an XML file. You might want to export a custom product family version so that you can import the product family version to a different environment.

When you export a product family version, you export both the data model metadata and the application metadata. When you export the data model metadata, you choose to submit the export job as a background job or continue the export through the Enterprise Data Manager.

You also have the option of selecting "Enhanced Export." Select the enhanced export option if you need to import the PFV to an existing environment. If you do not choose enhanced export, and you try to import the exported data model metadata to an environment where any PFV already exists, the data becomes corrupted. You do not need to select enhanced export if you plan to import the PFV to a new environment without any existing PFV's.

After you export a custom product family version, you can import it into a different environment. The process is the same as importing an accelerator. When you import the PFV, you again have the option of choosing enhanced import.

Exporting a Custom Product Family Version

When you export a product family version, export both the data model metadata and the application metadata to a location of your choice.

1. In the Enterprise Data Manager, select the product family version that you want to export and click **File** > **Export** > **Data Model Metadata**.

   The **Export Database Metadata Options** window appears.
2. Select **Both**.
3. Click **OK**.

   The **Export Database Metadata Options** window appears.

4. Choose to run the export job as a background job or through the Enterprise Data Manager.
5. If you plan to import the PFV to an existing environment, select the **Enhanced Export** option.
6. Click **OK**.
7. Select a location to export the XML file to and click **Save**.

   A popup appears to indicate that the export has completed successfully.

Importing Accelerators and Custom Product Family Versions

You can import accelerators and PFV's as a background job or in the foreground. The background job runs on the Data Archive server instead of on the EDM client machine. You may want to run the job on the Data Archive server to improve the job performance and to reduce the amount of resources consumed on the EDM client machine. To import accelerators or PFV's as a background job, you must start EDM from Data Archive. Standalone EDM does not support offline import.

When you import accelerators or PFV's, you specify the job execution mode, the location of the xml metadata files, and the job parameters.

Job Parameters

Enter the job parameters when you import accelerators.

The import accelerators job includes the following parameters:

Commit after Each File

Determines when the database transaction gets committed if you import multiple files.

Enable to commit after each file is imported. The commit occurs if the file is imported successfully and is independent of other files. For example, if you import ten files and there was an error with one of the files, then nine files are imported.

Disable to commit after all of the files are imported. The commit occurs if all of the files are imported successfully. If an error occurs with one of the files, no files are committed. For example, if you import ten files and there was an error with one of the files, then no files are imported.

Continue Processing on Error

Determines whether the system continues to process multiple files if there is an error with one of the files.

Enable to process files subsequent to the file that has the error. For example, you import ten files. An error occurred for the sixth file. The job ignores the sixth file and continues with the rest of the files.

Disable to stop processing the files subsequent to the file that has the error. For example, you import ten files. An error occurred for the sixth file. The job does not process the rest of the files.

Importing Accelerators or Custom Product Family Versions as a Background Job

You can import accelerators or PFV's as a background job. The background job runs on the Data Archive server instead of on the Enterprise Data Manager client machine.

To import accelerators or PFV's as a background job, you must start the Enterprise Data Manager from Data Archive. The standalone version of the Enterprise Data Manager does not support offline import.

You may want to run the job on the Data Archive server to improve the job performance and to reduce the amount of resources consumed on the Enterprise Data Manager client machine. Run on the Data Archive server to avoid memory-related issues if you need to import a large volume of metadata.

1. Click **File** > **Import** > **Accelerators**

   The **Import Metadata Options** screen appears.
2. Choose **Submit Import Metadata as a Background Job**.
3. Select the machine and location of the .xml metadata files.
   - Location on Server. Select if the .xml metadata files exist on the Data Archive server. Enter the full path to the files. For example, `/home/oracle/ILM/export`
   - Location on Client. Select if the .xml metadata files exist on the EDM client machine. Click **Browse** to navigate to the directory.
4. Configure the job parameters. If you used the Enhanced Export option to export the data, select **Enhanced Import**.
5. Click **OK**.

   The system immediately runs the job in the background and provides a message that the job was submitted. The message includes the job ID. You can monitor the job status in Data Archive.

Importing Accelerators and Custom Product Family Versions in the Foreground

You can import accelerators or PFV's in the foreground. When you run the job in the foreground, the job runs on the Enterprise Data Manager client machine.

1. Click **File** > **Import** > **Accelerators**

   The **Import Metadata Options** screen appears.
2. Choose **Continue Import through EDM**.
3. Configure the job parameters. If you used the Enhanced Export option to export the data, select **Enhanced Import**.
4. Click **OK**.

   The Windows Explorer dialog box appears.
5. Navigate to the directory that contains the metadata files that you want to import.
6. Click **OK**.

Virtual Views

A virtual view is a type of table that uses SQL logic to convert the format of retired data. You can create virtual views for any task that you can specify in the SQL.

Create virtual views to store retired data in a different format from data on the production database. For example, the date-of-birth values in a table are in integer format such as 01021991. When you retire this table to the Data Vault, the date-of-birth values remain in the same integer format. You can create a virtual view table to convert these date values to a more readable, alphanumeric format, such as 01-Feb-1991, on the retirement database.

To display the converted data, Data Archive uses the SQL created in the virtual view instead of the dynamically generated SQL used during an archive or a retirement job. You can create virtual views for data that you plan to retire. You can also create virtual views when you cannot create views on a read-only source database.

Create a virtual view to accomplish the following goals:

- Hide the complexity of the underlying data to create a simpler user experience.
- Save space and cost because the table is virtual.
- Simplify the process of capturing metadata from a retirement database.

Virtual Views Taskflow

Add virtual views to the entities you want to retire.

1. Identify data that you want to retire.
2. Create virtual views of source databases in the ILM repository.
3. Add virtual views to entities.
4. Include entities in retirement projects.

Creating a Virtual View

To create a virtual view, use the **Virtual View Wizard** wizard.

1. Go to **View** > **Constraints**.
2. Navigate to the application version.
3. Right-click the application version and select **Create Virtual View**.

   The **Virtual View Wizard** wizard appears.
4. Enter a name and a description for the virtual view that you want to create.
5. Click **Next**.

   A list of the available schemas for the home database appears.
6. Select the schemas for the virtual view, and click the down arrow.
7. Click **Next**.
8. Click the **Add** button to insert a blank line. Enter a name for the column, select the data type, and specify the length.
9. Click **Next**.

   The **Virtual View's SQL Text** page appears.

10. Enter the SQL code that the ILM Engine uses to archive the virtual view. Prefix the SQL code with the schema name so that the row count report generates correctly.

    You must provide the correct number of columns in the SQL code for the job to run successfully.
11. Click **Finish**.

    The virtual view appears under the application module of the application version that you selected.

Enabling Search Data Vault

Use keywords to search for records across entities and applications in Data Vault.

To enable Search Data Vault, you must create a search index for each table you want to include in the search. Each search index contains a list of columns you specify in Enterprise Data Manager. When you search Data Vault with a keyword, the search engine looks for the keyword in indexed columns. If the keyword is in an indexed column, the corresponding record appears in the search results. If the keyword is not in an indexed column, the record does not appear in the results even if the record contains the keyword in another column.

Selecting Columns for the Search Index

Specify columns to include in the search index. Optionally, specify the columns for the record header in the search results.

1. Select an application version and a table.
2. Click **View** > **Constraints**.
3. Click the **Columns** tab.

   A list of all the columns in the table appears.
4. Enable the **Index for Search** field for each column you want to add to the search index.
5. Click the **Constraints** tab.
6. Click the add icon to add a constraint.

   A row appears with the name `NEW CONSTRAINT`.
7. Double-click the **Name** field and enter a name.
8. Optionally, to specify a column name as the record heading on the results page, click the **Type** field and select **Search Header**.

   If you do not specify a column for the record heading, Data Archive uses the column name of the primary key. In the absence of a primary key, Data Archive uses the row ID for the record heading.
9. Enable the **Enabled** field.
10. Click the add icon in between the Child Table Columns and Parent Table Columns sections of the Constraints tab.

    A row appears in the Child Table Columns section.
11. Click the **Type** field and select **Column**.
12. Click the **Name** field and select a column.
13. To add more columns to the record header, repeat steps 10 to 12.
14. Click **File** > **Save**.

Specifying Large Number of Columns at a Time

To add many columns to the search indexes, update the index metadata file in Enterprise Data Manager.

1. Click **View** > **Constraints**.
2. Right-click an application version and select **Export Index Metadata**.

   A csv file appears with a list of columns for all the tables in the application version.
3. Enter **Y** in the **Quick_Search_Index** column for the columns you want to include in the search index.
4. Save the csv file.
5. Right-click the application version and select **Import Index Metadata**.

Troubleshooting Enterprise Data Manager

When I mine data, the mining job fails with JDBC driver errors.

You may receive an error if the source database includes datatypes that are not supported by the JDBC driver.

To resolve the error, perform one of the following tasks:

- Convert the table to a view. Use SQL functions, such as CONVERT, to convert the table to a view. Translate the column datatype to a generic datatype such as CHAR, VARCHAR, or NUMBER. Then, mine the view.
- Use a custom JDBC driver to mine the data. When you mine the database, use the CUSTOM_JDBC option. If you use a custom JDBC driver, you must use the standalone version of the Enterprise Data Manager. You cannot assign custom jar files for the web-based Enterprise Data Manager. Copy the jar files to the local machine from where you run the mining job.

When I mine data, export metadata, or import metadata, the job hangs. I get insufficient memory or heap errors.

The Enterprise Data Manager is a client application that is launched using JNLP protocol. Every client application has memory limitations. The Enterprise Data Manager may run out of memory when you use the client to mine data, export metadata, or import metadata.

You may receive a `java.lang.OutOfMemoryException` error when you mine large numbers of tables. Or, when you import or export a metadata for an application version that includes a large number of tables and constraints. When the Enterprise Data Manager is out of memory, the job does not progress in the status window. Java can use up to 2 gigabytes of RAM, which is usually not enough for large applications.

To resolve this error, run the jobs in the background. When you run the job in the background, the job runs on the Data Archive server instead of on the Enterprise Data Manager client machine.

When I import metadata, I cannot see my table listed in the Import Metadata from Database Wizard.

The reason you cannot see a table in the Import Metadata from Database Wizard might be because the table contains a special character in the table name.

You cannot import metadata if the table name contains one of the following special characters: `~`, `!`, `@`, `#`, `$`, `?`, `%`, `^`, `&`, `*`, `_`, `+`, `-`, and `=`.

CHAPTER 4

ILM Repository Constraints

This chapter includes the following topics:

ILM Repository Constraints Overview

The ILM repository stores constraints for source metadata. Data Archive stores primary, referential, and unique constraint types. Constraints may be added to the ILM repository when you import metadata from the source database. If the constraints are not imported, you must define the constraints manually or use tools to help you identify the table relationships.

Constraints are required to automatically create entities for data discovery. When you use the auto-create entity wizard, the wizard adds all related child tables for the parent table you select and adds all related tables to the entity. Constraints are also required to search the Data Vault from the entity level. The entity that you search on must have the relationships required to related tables to show related information at the transaction level. Constraints are not needed to archive data. When you run an archive or retirement job, the ILM engine creates a WHERE clause that determines what data needs to be archived.

When you import metadata from a source database to the ILM repository, all constraints that are defined on the database level are imported. If the source database does not include constraint information at the database level, such as in the application level, then you must add the constraints to the ILM repository. You can add constraints manually. Or, you can use tools to help you discover table relationships and create constraints based on the suggested relationships.

Constraint Definition

Constraints may be included in the ILM repository when you import metadata for the source application. If the constraints are not included in the ILM repository, you can manually define the constraints. Or, use tools

to help you discover the table relationships and use the suggested relationships to create constraints for the source metadata in the ILM repository.

You can use one of the following options to define constraints:

Manual Definition

You can manually create the constraints if you are familiar with the data model of the source and if the data model is not too large. You may want to manually create constraints if you have a small amount of tables to maintain constraints for.

Definition from Table Relationships

You can use tools to help discover table relationships. The tools help identify primary and foreign key relationships and column uniqueness. Then, create constraints from the suggested table relationships. You may want to define constraints from table relationships if you are not familiar with the data model of the source database. Or, if the data model is too large to manually create the constraints.

After you import or define the constraints for source metadata in the ILM repository, you can maintain the constraints if you need to change them. For example, you can create, delete, or change constraints. The constraints are changed for the source metadata in the ILM repository, not in the source database.

After you import or define the constraints, you can use the constraints to automatically create entities.

Pre-Packaged Applications

For pre-packaged applications, constraint definition is required only if you have custom tables or if you modified standard application tables. Constraints for standard application tables are included in the application accelerators. When you install the accelerators, the metadata and the constraints are imported to the ILM repository.

Custom Applications

For custom applications, constraint definition depends on how the source database stores constraints. If the source database maintains constraints at the database level, the constraints are automatically created in the ILM repository when you import metadata from the source database. If the source database maintains constraints other than at the database level, such as in the application logic, you must define the constraints. You must define constraints if constraint information is not included when you import metadata from the source application.

SAP Applications

For SAP applications, constraint definition is required. When you import metadata from SAP applications, only contraints that are defined at the database level are imported. Constraints that are defined in the SAP ABAP Dictionary are not imported. Constraints are not required to retire the SAP application. Constraints are required to create entities for Data Discovery portal searches. You may want to define the constraints after you retire the application.

Definition from Table Relationships

Use tools to help you analyze and discover table relationships in the source database. Then, review the suggested relationships and create constraints from the suggestions.

Define constraints from table relationships if you are not familiar with the data model of the source database, or if you do not want to maintain constraints manually. You can use table relationships to help you discover primary and foreign key relationships and column uniqueness.

Complete the following tasks to define constraints from table relationships:

Step 1. Discover Table Relationships

You can discover table relationships from ERwin data models, data models in CSV files, profile results that you generated in Informatica Data Quality, or Informatica Data Quality profiles that you generate from Enterprise Data Manager.

Step 2. Import Table Relationships

After you discover table relationships, view and analyze the suggested relationships. Then, use the suggestions to create constraints for the related source metadata in the ILM repository.

You can discover relationships from any tables in the source. However, you can only create constraints for tables that exist in the ILM repository. Before you define constraints from table relationships, you must import metadata for all the tables that you want to discover relationships from.

If you retired the source database, you can discover table relationships from the Data Vault. You initiate the discovery from the metadata that you imported from the source database. You do not have to import metadata from the Data Vault.

Step 1. Discover Table Relationships

Use tools to help you analyze and discover table relationships in source databases. Use the tools to discover unique keys, primary keys, and foreign keys. Use the tools if you are not familiar with the data model of the source database, or if you do not want to maintain constraints manually.

You can use any of the following tools to help you discover table relationships:

Discovery from ERwin data models

You can export files from ERwin data models of source databases. You discover relationships from the exported file. Use ERwin data models to identify unique, primary, and foreign key constraints. All of the information for the table discovery is contained in the exported file. You do not need to connect to the source database nor the ERwin data model.

Discovery from Informatica Data Quality

You can connect to Data Quality to generate profiles for source databases. You discover relationships from the generated profiles. Use Data Quality to identify unique, primary, and foreign key constraints. A connection to the source database and Data Quality is required. If you retired the source database, you can connect to the Data Vault.

Discovery from Informatica Data Quality Profile Results

You can export profile results that are generated from Informatica Developer for source databases. You discover relationships from the exported profile results. Use the profile results to identify unique, primary, and foreign key constraints. A connection to the source database and Data Qulity is required. If you retired the source database, you can connect to the Data Vault.

Discovery from CSV files

You can maintain data models of source databases in CSV files. You discover relationships from the CSV file. Use CSV files to identify primary and foreign key constraints. All of the information for the table discovery is contained in the CSV file. You do not need a connection to the source database.

The table relationship discovery process is driven from the source table metadata in the ILM repository. You initiate the discovery from the application version or from the application module levels of the source in the ILM repository. You choose the tables that you want to discover relationships for. When you discover table relationships at the application version level, you can discover relationships for tables across all schemas in

the application version. When you discover table relationships from the application module level, you can discover relationships for tables within the schema only.

When you discover table relationships, the ILM engine stores the suggested unique keys and table relationships in a profiling suggestion table in the ILM repository. All of the tools store the suggestions in the same table. If you discover table relationships multiple times or if you use multiple discovery tools, the system appends the suggestions to the profiling suggestion table. Because the tools store the results in the same table, only use one tool to discover relationships from a schema. If you use multiple tools for the same schema, you may have difficulty interpreting the results.

After you view the suggestions and import the suggestions to create constraints, the ILM engine creates constraints for the corresponding table metadata in the ILM repository.

Note: Although the discovery process identifies primary keys, when you import the suggested table relationships, the ILM engine creates the discovered primary keys as unique keys to avoid a potential database conflict. A conflict can occur if the table already contains a primary key.

Discovery from ERwin Data Models

You can use ERwin logical data models to discover relationships between tables in source applications. Export the data model to discover unique, primary, and foreign key constraints. Then, use the suggested relationships to create unique and referential constraint types for the related source metadata in the ILM repository.

You export the logical data model to an XML file and then run the discovery on the exported file. When you export the data model, you export the file in PowerCenter format. The ILM engine uses a PowerCenter DTD file to import the metadata from the XML file into the ILM repository. The PowerCenter DTD file is included as part of the Data Archive installation.

The ILM engine uses the imported source metadata in the ILM repository to drive the import of the metadata from the data model. You initiate the discovery from the application version or from the application module levels of the source metadata. You specify the tables that you want to import metadata for. When you discover relationships at the application version level, the discovery occurs across all schemas in the application version. When you discover from the application module level, then the discovery occurs for that schema only.

All of the information for the table discovery is contained in the exported XML file. You do not need to connect to the ERwin data model or to the source database. You can export data models from any ERwin version that PowerCenter supports.

Import Metadata Job

The Import Metadata job initiates the table relationship discovery process when you discover relationships from ERwin data models. When you submit the criteria for the discovery, the ILM engine immediately runs the Import Metadata job in the background.

The Import Metadata job uses the parameters you specify in the wizard to locate the exported XML file. The job uses the PowerCenter DTD file to interpret the results in the exported XML file. The job finds the profiling information for the tables within the application version or application module that you initiated the discovery from. The job imports all the suggested unique, foreign, and primary key relationships to a profiling suggestion table in the ILM repository. The job does not create the constraints in the ILM repository. You must review the suggestions and then choose from which suggestions you want to create constraints.

When you submit the job, a message provides the job ID. Use the job ID to monitor the job status and the job log. The job log summarizes the parameters that you used to initiate the discovery, such as the full path of the exported xml file, the application version or application module, and the related tables that you ran the discovery for.

Rules and Guidelines for Discovering Relationships from ERwin Data Models

Consider the following rules and guidelines when you discover relationships from ERwin data models:

- Create an export file for each schema that you want to discover relationships for. If the constraints are defined across schemas, then create one export file for all the schemas.
- When you export the data model, use the `Informatica PowerCenter` export destination type. Configure the `DBDNAME` export parameter with the schema name that owns the tables. If you do not configure the export parameter, the imported suggestions do not show schema names. If the suggestions do not include schema names, you must select the schema name for the parent and the child tables before you can create the constraints.
- Optionally, after you export the data model to an XML file, you may want to validate that the XML file is complete. The export file may not be complete if the data model is very large. Use the PowerCenter Designer or Repository Manager to import the file into the repository. If you can import the file without any errors, then the file is complete.

Discovering Relationships from ERwin Data Models

To import relationships from ERwin data models, export the data model to an XML file. Then, initiate the discovery from the application version or application module of the source metadata.

Verify that the metadata for all of the tables in the exported XML file is imported in the ILM repository. The ILM engine can use suggested table relationships to create constraints only for tables that exist in the ILM repository.

1. Save the exported XML file to a location that is accessible to the machine that hosts Data Archive.
2. Copy the `powrmart.dtd` file to the same directory where you saved the exported XML file.

   You can find the `powrmart.dtd` in the optional directory of the Data Archive installation.
3. In the Enterprise Data Manager, click **File** > **View Constraints** to open the constraints tab.
4. In the navigation pane, select the application version or application module.
5. Click **File** > **Discover Keys and Relationships**.

   The **Metadata Import Wizard** appears.
6. Select **Import keys and table relationships from Erwin Model** and click **Next**.
7. Enter the full path including the file name of the exported XML file location.
8. Click **Finish**.

The Enterprise Data Manager submits the Import Metadata job and displays a message with the job ID. Use the job ID to monitor the job status and to view the job log. After the job completes, review the suggested unique keys and table relationships.

Discovery from Informatica Data Quality

You can connect to Data Quality to generate profiles for source databases. Generate profiles to identify primary and foreign key constraints. Then, view the results and use the suggested relationships to create unique and referential constraint types for the related source metadata in the ILM repository.

You can generate profiles if you have the Data Discovery option and if you have native connectivity from the Informatica server to the source database server. A connection to the source database and Data Quality is required. If you retired the source database, you can connect to the Data Vault.

When you discover table relationships from Data Quality, you generate a profile for primary key and foreign key profiling. The profile analyzes the percentage of matched values across table columns to determine primary key and foreign key relationship probability and uniqueness.

Data Quality uses the source metadata in the ILM repository to drive the profile analysis on the source database. You initiate the discovery from the application version or from the application module levels of the source metadata in the ILM repository. You choose the tables to profile and configure parameters that determine how Data Quality generates and runs the profile. You can generate profiles for one schema at a time. When you discover relationships at the application version level, the discovery occurs for the default schema. When you discover from the application module level, then the discovery occurs for that schema only.

The profile results are stored in the profiling database. After you generate the profile, you can view the profile results within the Enterprise Data Manager. The job ID has the information required to connect to Data Quality and to obtain the profile results. When you view the results, the Enterprise Data Manager connects to the profiling database and imports the results from the profiling database to the ILM repository. Then, you can import the suggestions as constraints.

If you retired the application, you might need to define constraints for imported source metadata after you retired and decommissioned the source database. You want to use the constraints to create entities for data discovery. You can connect to and generate profiles for the Data Vault instead of the source database.

When you discover table relationships from Data Quality, you initiate the discovery from the source metadata that you originally imported from the source database. You do not have to import metadata from the Data Vault.

Import Metadata Job

The Import Metadata job initiates the table relationship discovery process when you discover relationships from Data Quality. When you submit the discovery criteria, the ILM engine immediately runs the Import Metadata job in the background.

The Import Metadata job gets a list of all imported source metadata in the ILM repository for the application version or application module that you initiated the discovery from and the tables that you selected to profile. The job connects to Data Quality from the metadata import connection properties that are configured on the **Profiling and Discovery** tab of the **System Profile** page. The job uses a list of the source metadata and the parameters that you specify to connect to the source database and to generate the profile. The profiling job profiles for the primary key and determines the best possible key for the table. Then, the job uses the results of the primary key profile and runs the foreign key profile. You configure whether the primary keys can include composite keys on the **Profiling and Discovery** tab of the **System Profile** page.

The job imports the suggested primary key and the foreign key relationships to a profiling suggestion table in the ILM repository. The job does not create the constraints in the ILM repository. You must review the suggestions and then choose suggestions that you want to create constraints from.

When you submit the job, a message provides the job ID. Use the job ID to monitor the job status and the job log. The job log includes the parameters of the discovery, such as the discovery method you used and the parameters that you entered for the discovery tool. The job log shows the application version or application module and the related tables that you ran the discovery for, and the other parameters you used for the discovery.

Connection Parameters

When you discover relationships from Data Quality, you configure connection parameters for the source database. The Import Metadata job passes the connection parameters to Data Quality. Data Quality connects to the source database to generate the profile.

When you configure the connection parameters, you configure a connection that points to the same source database that you imported metadata from. You can choose the same connection that you used to import metadata from the source database. However, the connection must have native connectivity to the source database. Data Quality only connect to source databases through native connectivity.

Most of the connection parameters are populated based on the connection name that you choose. In addition, you must specify a source connection string and possibly change the user name and password.

To use the Data Vault as the source database, configure the same connection properties that the retirement job used for the Data Vault target connection.

The connection includes the following parameters:

Connection Name

Name of the connection to the source database. Choose the connection that you used to import metadata from the source database.

Database Type

Database connection type that determines how you connect to the source database. The connection type depends on the database and the database version that contains the source data. Data Quality supports native connectivity only.

By default, the value is populated based on the connection that you choose.

For Data Vault sources, use the `Data Vault Service` connection to connect to the Data Vault. If your applications were retired into the Default Administrator Schema on Data Vault Service, check the **Use Administrator's default schema(dbo)** option.

Database Host

Name of the machine that hosts the source database. By default, the value is populated based on the connection that you choose.

Database Port

Port number used by the source database. By default, the value is populated based on the connection that you choose.

Database Name/SID

Unique identifier or system identifier for the source database server.

The following table describes the database name for each source database:

Source Database	Description	Connect String Syntax
IBM DB2	Name of the remote database configured from the IBM DB2 Connect client.	`dbname`
Microsoft SQL Server	Name of the host of the Microsoft SQL Server database.	`servername@dbname`

Source Database	Description	Connect String Syntax
Oracle	Full service name or SID for Oracle databases. TNS name that is configured from the ILM application server to the database server. The name is defined in the application server `TNSNAMES.ORA` file.	`dbname.world`
Data Vault	The ODBC data source name created on the machine hosting Informatica Data Quality.	`dsn`

By default, the value is populated based on the connection that you choose.

Source Connection String

Connect string that determines how the database native client connects to the database. Data Quality uses the connect string to create a connection object to the source database.

For Oracle, use the TNS name. For IBM DB2, use the database alias. For Microsoft SQL Server, use the remote hostname and database name in the format `<hostname>@<database name>`. For Data Vault, use the ODBC data source name.

User Name

User name that connects to the source database. The user name is the owner of the schema that you want to profile. The user must own the tables that you want to profile and must have privileges to select table catalogs.

For Oracle and IBM DB2, the user name is the same as the schema name.

Password

Password for the database user name.

Advanced Parameters

Provide the advanced parameters when you discover relationships from Data Quality. The parameters determine how Data Quality generates and runs the profile. The parameters to generate profiles from the Enterprise Data Manager are the same as if you generate the profiles from Informatica Developer. For more information about the parameters, see the *Informatica Data Quality Profile Guide*.

Primary Key

Configure the following advanced parameters for primary key discovery:

Maximum Key Columns

Maximum number of columns that can make up a primary key.

Maximum Rows

Maximum number of rows to profile for primary key discovery.

Conformance Criteria

Minimum percentage or maximum number of rows of key violations that the profile allows for primary key discovery.

Choose one of the following conformance criteria:

- Minimum Percentage. Minimum percentage of key violations that the profile allows for primary key discovery. For example, if you enter 90, the profile returns primary keys that have more than 90% accuracy.
- Maximum Violation Rows. Maximum number of rows of key violations that the profile allows for primary key discovery.

Foreign Key

Configure the following advanced parameters for foreign key discovery:

Datatypes used in Comparison

Determines if the profile uses data dictionary datatypes instead of generic datatypes.

Choose one of the following options:

- Inferred Datatypes. The profile uses generic datatypes.
- Metadata. The profile uses data dictionary datatypes.

Comparison Case Sensitivity

Determines if comparison includes case sensitive differences.

Choose one of the following options:

- Case Sensitive. Determines if the profile runs case sensitive comparisons. If you select this option, the profile job performance may be impacted.
- Not Case Sensitive. Does not display case sensitive differences.

Trim Spaces before Comparison

Determines if Informatica Developer includes leading or trailing spaces in column data.

Maximum Foreign Keys Returned

The maximum number of inferred columns that Informatica Developer returns.

Minimum Confidence Percent

Minimum percentage of key violations that the profile allows for foreign key discovery. For example, if you enter 90, the profile returns foreign keys that have more than 90% accuracy.

Regenerate Signature

When checked, reloads column signatures if the source data changes.

Discovering Relationships from Informatica Data Quality

To generate profiles from Informatica Data Quality, initiate the discovery from the application version or the application module. Choose the tables to profile, configure the source connection properties, and configure profiling parameters that determine the type of profile that Data Quality generates.

- Verify that the metadata import properties are configured on the **Informatica Data Quality** tab of the **System Profile** page. The Import Metadata job uses the properties to connect to and generate profiles from Data Quality.

- Verify that the metadata is imported in the ILM repository for the tables you want to profile. The ILM engine can use suggested table relationships to create constraints only for tables that exist in the ILM repository.
  If you use the Data Vault as the source, you initiate the discovery from the metadata that you imported from the source database. You do not have to import metadata from the Data Vault.

1. Click **File** > **View Constraints** to open the constraints tab.
2. In the navigation pane, select the application version or application module.
3. Click **File** > **Discover Relationships from Informatica Data Quality**.

   The **Metadata Import Wizard** dialog box appears.
4. Select **Discovery and Profiling with Informatica Data Quality**.
5. Select one or both discovery checkboxes and click **Next**.

   The **Select Connection Parameters** page of the wizard appears.
6. Enter the connection parameters.
7. Optionally, click **Advanced Options** and enter inference information. Click **Save**.

   The **Select Connection Parameters** page of the wizard appears.
8. Click **Next**.

   The **Select Tables** page of the wizard appears.
9. Select the tables that you want to discover relationships for. Click the down arrow button to insert the tables in to the **Selected** box.

   Use the control key to select all tables or multiple tables. The tables that you can select depend on where you initiated the relationship discovery from. If you discover relationships from the application version level, then you can select tables across all schemas in the application version. If you discover relationships from the application module level, then you can select tables in the schema in the application module.
10. Click **Finish**.

The Enterprise Data Manager submits the Import Metadata job and displays a message with the job ID. Use the job ID to monitor the job status and to view the job log. After the job completes, review the suggested unique keys and table relationships.

Discovery from Informatica Data Quality Profile Results

You can use Data Quality profile results to discover relationships between tables in source databases. Import the profile results to identify primary and foreign key constraints. Then, use the suggested relationships to create unique and referential constraint types for the table metadata in the ILM repository.

You can import profile results if you have the Data Discovery option.

The Enterprise Data Manager can discover table relationships from profile results that you export from Data Quality. You export profile results from the Informatica Developer to an XML file. The exported XML file includes references to the profile job in Data Quality, such as the profile definition, function definition, and the profile fields. The exported XML file does not contain all of the profiling results. The profiling results remain in the profiling warehouse. A connection to Data Quality is required to obtain the profiling results. Note that the Enterprise Data Manager imports the profile results, not the profile model.

The ILM engine uses the imported source metadata in the ILM repository to drive the import of the profile results. You initiate the discovery from the application version or from the application module levels of the source metadata. When you discover relationships at the application version level, the discovery occurs across all schemas in the application version. When you discover from the application module level, then the discovery occurs for that schema only.

If you retired the application, you may need to define constraints for imported source metadata after you retired and decommissioned the source database. You want to use the constraints to create entities for data discovery. You can generate profiles for the Data Vault instead of the source database.

When you discover relationships from Data Quality profile results, you initiate the discovery from the source metadata that you originally imported from the source database. You do not have to import metadata from the Data Vault.

Import Metadata Job

The Import Metadata job initiates the table relationship discovery process when you discover relationships from profile results that you exported from Informatica Developer. When you submit the discovery criteria, the ILM engine immediately runs the Import Metadata job in the background.

The Import Metadata job gets a list of all imported source metadata in the ILM repository for the application version or application module that you initiated the discovery from. The job connects to Data Quality from the metadata import connection properties that are configured on the **Profiling and Discovery** tab of the **System Profile** page. The job uses the values in the exported file to collect the full results from the profiling warehouse for the tables in the application version or the application module that you specify. The job imports all the suggested unique keys, suggested foreign keys, and primary key relationships to a profiling suggestion table in the ILM repository. The job does not create the constraints in the ILM repository. You must review the suggestions and then choose which suggestions you want to create constraints from.

When you submit the job, a message provides the job ID. Use the job ID to monitor the job status and the job log. The job log includes the parameters of the discovery, such as the discovery method you used and the parameters that you entered for the discovery tool. The job log shows the application version or application module and the related tables that you ran the discovery for, and the other parameters that initiated the discovery.

Rules and Guidelines for Discovery from Informatica Data Quality Profile Results

Review the rules and guidelines before you import profile results from Data Quality.

General Rules and Guidelines

Consider the following rules and guidelines when you discover relationships from Data Quality profile results:

- When you export the profile results from Informatica Developer to an XML file, choose Informatica export object metadata file as the export destination.
- The Enterprise Data Manager discovers unique columns one profile at a time. For primary key profile types, export all profiles as individual XML files. Then, run the discovery process separately for each profile. If you combine multiple profiles into a single file, the Enterprise Data Manager discovers relationships only for the first schema.
- You can export the profile results to any location, however, you must move the exported file to the server that hosts Data Archive.
- The Enterprise Data Manager and Data Quality must be connected to the same Model Repository Service. The Enterprise Data Manager only contains references to the profile job. The Enterprise Data Manager must connect to Data Quality to read the full profiling results. You can only discover relationships from files that were profiled on the same server. You configure a connection from the Enterprise Data Manager to the Model Repository Service on the **Profiling and Discovery** tab of the **System Profile** page.

Importing Profile Results from Informatica Data Quality

To import profile results that were generated in Informatica Developer, export the profile results to an XML file. Then, initiate the discovery from the application version or application module of the source metadata.

- Verify that the metadata import properties are configured on the **Profiling and Discovery** tab of the **System Profile** page. The Enterprise Data Manager uses the properties to connect to Data Quality to obtain the profile results.
- Verify that the metadata for all of the tables in the profile results is imported in the ILM repository. The ILM engine can use suggested table relationships to create constraints only for tables that exist in the ILM repository.
  If you use the Data Vault as the source, you initiate the discovery from the metadata that you imported from the source database. You do not have to import metadata from the Data Vault.

1. Copy the exported profile results XML file to a location that is accessible to the machine that hosts Data Archive.
2. In the Enterprise Data Manager, click **View** > **Constraints** to open the constraints tab.
3. In the navigation pane, select the application version or application module that you want to discover relationships for.
4. Click **File** > **Discover Keys and Relationships**.

   The Metadata Import Wizard appears.
5. Select **Import Profile Results from Informatica Data Quality** and click **Next**.
6. Enter the full path of the location of the exported XML file, including the file name.

   For example, `/home/archive/ilm/optional/pk_export.xml`.
7. Choose the profile type that the profile results include.
8. Click **Finish**.

The Enterprise Data Manager connects to Informatica Data Quality to obtain the profile results and imports the suggestions to the profiling suggestion table in the ILM repository. Review the suggested table relationships to import the suggestions as constraints.

Discovery from CSV Files

You can maintain data models of source databases in CSV files. Use the relationships defined in the CSV file to discover primary and foreign key constraints. Then, use the suggested relationships to create unique and referential constraint types for the related source metadata in the ILM repository.

The CSV file must be in a predefined format. The format is based on the table that stores the suggested table relationships.

When you import the relationships defined in the CSV file, the Enterprise Data Manager uses the values in the file for the tables in the application version or the application module that you specify. The Enterprise Data Manager imports the suggested foreign and primary key relationships to a profiling suggestion table in the ILM repository. The Enterprise Data Manager does not create the constraints. You must review the suggestions and then choose which suggestions you want to create constraints from. No jobs are involved in this process.

CSV File Format

To import table relationships from CSV files, the CSV file must be in the required format. When you import the file, the Enterprise Data Manager imports the contents of the file to a table relationship suggestions table in the ILM repository.

The table relationship suggestion table dictates the format of the CSV file. The CSV file requires the following columns:

Column	Description
PARENT_SCHEMA	Schema of the parent table that contains the primary key.
PARENT_TABLE	Table that contains the primary key.
PARENT_COLUMNS	Column that contains the primary key. If you have more than one column that identifies the primary key, use the following separator between columns: `#<Key column index>#` There is no limit to the amount of columns that can identify a primary key. To specify the end of a column, use the following separator: `#<key column index>`. For example, enter `TICKET_NUMBER#1` if the primary key is based on one column. Enter `TICKET_NUM#1#CREATION_DATE#2` if the primary key is based on two columns.
CHILD_SCHEMA	Schema of the child table that contains the foreign key.
CHILD_TABLE	Table that contains the foreign key.
CHILD_COLUMNS	Column that includes the foreign key.
OVERLAP_PERCENTAGE	Optional. Percentage of overlap between the parent and child columns. User defined value. You can enter any number. For example, you can use a range of 0-100. 0 is no overlap.
PARENT_UNMATCHED	Optional. Percentage of values in the parent column that do not have any matches in the child column.
CHILD_UNMATCHED	Optional. Percentages of values in the child column that do not have any matches in the parent column.

Use the following rules and guidelines when you create a CSV file:

- When you discover relationships from CSV files, you import one file at a time.
- You can include multiple schemas in the same file.
- The header row is required.
- All columns are required in the file. However, values for the `OVERLAP_PERCENTAGE`, `PARENT_UNMATCHED`, and `CHILD_UNMATCHED` columns are optional.
- Enter one table relationship per row.
- Case-sensitivity depends on the source database. Use the same case that the source database uses.

If you import a file that is not in the correct format, the Enterprise Data Manager cannot import the contents to the ILM repository.

Example

The following table displays sample rows of a CSV file:

PARENT_SCHEMA	PARENT_TABLE	PARENT_COLUMNS	CHILD_SCHEMA	CHILD_TABLE	CHILD_COLU
LOYALTY	CUSTOMER	CUST_ID#1	LOYALTY	CUSTOMER_TICKET	CUST_ID#1
LOYALTY	TICKET	TICKET_NUM#1#CREATION_DATE#2	LOYALTY	TICKET_COUPON	TICKET_NUM

Importing Relationships from CSV Files

Use CSV files to import table relationships suggestions to the ILM repository. You can import suggested table relationships for tables with an application version or an application module.

The table metadata for all of the tables in the CSV file must be imported in the ILM repository. The ILM engine can use suggested table relationships to create constraints only for tables that exist in the ILM repository.

1. Copy the CSV file to a location that is accessible to the machine that hosts Data Archive.
2. In the Enterprise Data Manager, click **File** > **View Constraints** to open the constraints tab.
3. In the navigation pane, select the application version or application module that you want to discover relationships for.
4. Click **File** > **Discover Relationships from Informatica Data Explorer**.

   The Metadata Import Wizard appears.
5. Select **Import Table Relationships from CSV** and click **Next**.
6. Enter the full path of the location of the CSV file, including the file name.

   For example, `/home/archive/optional/csv_import_kf.csv`.
7. Click **Finish**.

The Enterprise Data Manager imports the suggestions to the profiling suggestion table in the ILM repository. Review the suggested table relationships to import the suggestions as constraints.

Step 2. Import Table Relationships

Import of suggested table relationships is the second phase of constraint definition from table relationships. When you define constraints from table relationships, you must view the suggested relationships and import the relationships that you want to create constraints from. You can view suggested unique columns and suggested table relationships.

Constraint definition from table relationships consists of discovering table relationships and then importing the discovered relationships as constraints. The discovery process identifies relationships and stores the suggestions in a profiling suggestion table in the ILM repository. The discovery process does not automatically create the constraints for the tables in the ILM repository. After you discover table relationships, you must review the suggestions. Then, you import the suggestions that you want to create constraints from. You import table relationships after you use any of the available tools to discover relationships.

All of the discovery tools store the suggestions in the same table. If you discovery table relationships multiple times or if you use multiple discovery tools, the system appends the suggestions to the profiling suggestion table.

You can review and import the following types of suggestions:

Suggested Table Relationships

View the suggested table relationships to determine which relationships you want to use to create constraints for. Then, import the relationships as constraints for the source metadata in the ILM repository. When you import the suggested relationships, the ILM engine creates an unique constraint on the parent table and a referential constraint on the child table. The child table includes a reference to the unique key on the parent table. Although the discovery process identifies primary keys, when you import the relationships, the ILM engine creates the discovered primary keys as unique keys to avoid a database conflict. A conflict can occur if a primary key already exists on the table.

Suggested Unique Columns

View the suggested unique columns to create unique key constraints for columns that are not included as part of the primary key-foreign key relationship discovery. Then, choose the suggested unique columns that you want to import. When you import the unique columns, the ILM engine creates unique key constraints for the source metadata in the ILM repository.

The import sequence order is independent. You can import suggested unique columns or table relationships in any order. For example, if you import table relationships first, and the foreign key does not find a primary key on the same column in the parent, then the import creates the unique key on the parent and then creates the foreign key.

When you import the suggestions, the ILM engine checks if the constraint exists in the ILM repository. If the constraint exists in the ILM repository, then the import ignores the suggestion. If the constraint does not exist, the ILM engine uses the suggestions that you selected to create the constraints.

The import creates constraints with the following naming conventions:

- Unique keys. `PROF_<TABLE NAME>_UK`
- Foreign keys. `PROF_<TABLE NAME>_FK`

The `PROF_` prefix identifies constraints that you imported from table relationship discovery versus constraints that you created when you imported metadata from the source database or that you manually created.

You can only import constraints for tables that exist in the ILM repository. If you try to import constraints for tables that do not exist in the ILM repository, the import generates an error. You must import all metadata for the source database before you define constraints from table relationships.

After you import the suggestions as constraints, the suggestions remain in the ILM repository. After you create the constraints, you can maintain the constraints if required. For example, you can change or delete constraints that you imported.

Suggested Table Relationships

When you discover table relationships, the ILM engine stores the suggested relationships in a profiling suggestion table in the ILM repository. You can view the suggested table relationships after you discover table relationships. View the results to determine which suggestions you want to use to create constraints from.

The suggested table relationship results are displayed in a table. You access the table from the same application version or application module that you ran the discovery for. All of the discovery tools store the results of the discovery in the same suggestion table. The columns that are in the results depend on the discovery tool that you used to discover the table relationships. Some of the columns in the results apply to

all discovery tools. If you used multiple discovery tools to discover relationships, then you can see columns that are relevant to all discovery tools.

For discovery from ERwin data models and CSV files, the results show all of the results from the exported files. Further interpretation of the results may not be necessary. You import the relationships that are already defined in the export or csv files.

For discovery from Informatica Data Quality and discovery from Informatica Developer profile results, the results are filtered based on how you generated and ran the profiles. Further interpretation of the results is necessary. View the suggested table relationships to determine the appropriate threshold to use for constraint creation. The quality of the suggested relationships depends on the quality of the data that you discovered relationships for. If you are not familiar with the quality of the data that you discovered relationships for, you may want to use a threshold of 90%. If you use a low threshold, then you may create spurious constraints. If you use a high threshold, you may ignore potential relationships.

The suggested table relationship results include the following columns:

Parent Schema

Schema of the parent table that contains primary keys. In some cases, the parent schema may not be populated. If the parent schema is not populated, then you must select a schema name when you import the suggestions.

For table relationship discovery from ERwin data models, the schema name appears if you configured the DBDNAME export parameter when you exported the data model.

Parent Table

Table that contains primary keys.

Parent Column

Column that contains the primary key.

For table relationship discovery from Informatica Data Quality and from Informatica Data Quality profile results, the primary key column that meets the primary foreign key inference criteria that you defined in the profile.

Child Schema

Schema of the child table that contains the foreign key.

Child Table

Table that contains the foreign keys.

Child Column

Column that includes the foreign key.

For table relationship discovery from Informatica Data Quality and from Informatica Data Quality profile results, the foreign key column that meets the primary foreign key inference criteria that you defined in the profile.

Overlap Percentage

Percentage of overlap between the parent and child columns.

Available for table relationship discovery from Informatica Data Quality and discovery from Informatica Data Quality profile results. Available for discovery from CSV files if the value is maintained in the CSV file.

Parent Unmatched

Percentage of values in the parent column that do not have any matches in the child column.

Available for table relationship discovery from Informatica Data Quality and discovery from Informatica Data Quality profile results. Available for discovery from CSV files if the value is maintained in the CSV file.

Child Unmatched

Percentages of values in the child column that do not have any matches in the parent column.

Available for table relationship discovery from Informatica Data Quality and discovery from Informatica Data Quality profile results. Available for discovery from CSV files if the value is maintained in the CSV file.

Viewing and Importing Suggested Table Relationships

After you discover table relationships, view the suggested relationships. Choose the relationships that you want to import as constraints. When you import the suggested relationships as constraints, the ILM engine creates the constraints for the related table metadata in the ILM repository.

1. Click **File** > **View Constraints** to open the Constraints tab.
2. In the navigation pane, right-click the application version or application module that you want to import relationships for. Select the same application version or application module that you used when you discovered relationships.
3. Select **View Suggested Table Relationships**.

   A table appears with the suggestions from the table relationship discovery process.
4. Review the suggestions.
5. If you discovered relationships from ERwin data models or from profile results that were generated from Informatica Developer, you may need to select the parent and child schema names from the drop-down box.

   If the discovery tool cannot identify the parent and child schema names, then the schema name is populated with an underscore.
6. Select the suggestions that you want to import and click **Submit**.

The ILM engine creates constraints for the related tables in the ILM repository.

Suggested Unique Columns

When you discover table relationships, the ILM engine stores the suggested unique columns in a profiling suggestion table in the ILM repository. You can view the suggested unique columns after you discover table relationships.

View the unique columns when you want to import unique keys that are not identified in table relationships. For example, you may want to maintain unique keys in the ILM repository that are not part of a primary key and foreign key relationship. When you import suggested table relationships, the ILM engine automatically creates unique keys from the suggested table relationships.

All of the discovery tools store the results of the discovery in the same suggestion table. The results may not display all of the table columns. For discovery from Data Quality, the results are filtered based on the conformance criteria that you configured when you generated the profile. For example, if you configured 90% as the conformance criteria, then you only see columns that contain 90% and higher uniqueness.

You access the suggested unique columns from the same application version or application module that you ran the discovery from. Review the list of suggested unique columns. Then, select which suggestions you want to import to the source metadata in the ILM repository. When you import the suggestions, the ILM engine adds the unique constraint type.

The suggested unique column results show the following columns:

Schema

Schema of the table that contains unique keys.

Table

Table that contains unique keys.

Column

Column that contains the unique key.

Uniqueness Percentage

Percentage of uniqueness for the column.

Available for discovery from Data Quality and discovery from Data Quality profile results.

Viewing and Importing Suggested Unique Columns

After you discover table relationships, view the suggested unique columns. Choose the unique columns that you want to import as constraints. When you import the suggested unique columns as constraints, the ILM engine creates the constraints for the related table metadata in the ILM repository.

1. Click **File** > **View Constraints**.
2. In the navigation pane, right-click the application version or the application module that you want to import the suggested unique columns for.
3. Select **View Suggested Unique Columns**.

   A table appears with the suggestions that were identified from the table relationship discovery tool.
4. Review the suggestions.
5. If you discovered relationships from ERwin data models or from profile results that were generated from Informatica Developer, you may need to select the parent schema and the child schema from the drop-down box.

   If the table relationship discovery tool cannot identify the schema names, then the table shows an underscore for the schema name.
6. Select the suggestions that you want to import and click **Submit.**

Deleting Profile Results

After you view and analyze constraints, you can delete a profile result.

1. Click **View** > **Constraints** to open the constraints tab.
2. In the navigation pane, right-click the application version or the application module.
3. Select **Show Profiled Results**.

   The **Show Profiled Results** window appears.
4. Select the profiled result and click the **Delete** icon.

CHAPTER 5

Partition Exchange Purging

This chapter includes the following topics:

Partition Exchange Purging Overview

You can use Oracle partition exchange to delete data from sources on Oracle databases, version 10g or later. Configure partition exchange when you want to delete a significant percentage of the source data. For example, you want to delete twenty to thirty percent or more records from the source table.

Partition exchange increases the performance of the archive job when the job deletes data from the source. When you use partition exchange, the archive job calls a procedure that creates a partition for the records to keep in the source. The procedure swaps the original source segment with the new partition. It is more efficient to swap partitions than to individually delete a large amount of records from the source.

You can configure partition exchange for applications on Oracle databases that support partitioning. You can use partition exchange to delete data from partitioned and non-partitioned source tables. You can use partition exchange for archive and restore jobs.

Configure the entity to use partition exchange. When the archive job deletes data from the source, the job uses partition exchange for the tables that you configured in the entity. The job uses individual record deletion for the remainder of the tables in the entity that are not configured for partition exchange.

You can also disable or delete the configuration if you want do not want jobs to use partition exchange.

Working with Partition Exchange

When to configure partition exchange depends on several factors. The main factor is the volume of data that you want to delete from the source. Other factors include the archive or restore use case, the processing time, and the database down time.

Configure partition exchange based on the following factors:

Amount of data to delete from the source

The biggest factor to determine if you should configure partition exchange is the amount of data that you want to delete from the source. Configure partition exchange when you want to delete a significant percentage of data from the source. For example, you want to delete twenty percent or more records from the source table.

Archive use case

You can enable or disable partition exchange based on the archive job requirements, such as initial archiving or frequency of archiving.

You may want to configure partition exchange for the initial archive run if you have a lot of data to archive and purge. Then, disable the partition exchange for next regularly scheduled archive jobs. For example, you have 10 years worth of data in your source. For the first archive run on the source, you want to archive and purge the last 7 years worth of data. Then, you schedule a job that runs on a monthly schedule. The monthly jobs will not archive a large percentage of data. You configure partition exchange for the first archive run because there is a significant amount of data to delete. You disable the partition exchange configuration for the remainder of the jobs as the volume of the data to archive and purge will be significantly less.

Use the frequency of how often you plan to run an archive job to determine if you should configure partition exchange or not. For example, you may want to configure partition exchange for archive jobs that are scheduled on a less frequent basis, such as annual archiving. You may not want to configure partition exchange for jobs that run on a more frequent basis, such as monthly archiving. However, you should still consider the volume of the data even for archive jobs that run on a frequent basis.

Restore use case

You can enable or disable partition exchange to restore and purge data from the history database to the source database. The type of restore job and the amount of data that you need to restore determines if you should configure partition exchange.

For transaction restore jobs, partition exchange is not relevant because the restore job does not involve a large amount of data.

For cycle restore jobs, you can configure partition exchange if the restore cycle restores and purges a large percentage of the data in the history table.

Processing time

The main steps in the partition exchange procedure include creating a keep table, swapping the partitions, and rebuilding the source indexes. The time it takes to swap the partitions is minimal. However, the processing time to create the keep table and to rebuild the source indexes depends on the size of the data that remains in the table and the amount of indexes on the source. The processing time increases with the size of the data and the number of indexes on the source.

Database down time

When the archive job deletes data from the source using partition exchange, the database must be off line to perform the partition exchange. To reduce the impact of the database down time, you can configure the job to pause before the delete from source step. Then, take the database off line.

Partition Exchange Procedure

The `AA_PARTITION_PKG.exchange_partition` procedure runs the Oracle partition exchange logic for the archive job when the job deletes from the source. When you configure the entity steps, you add the procedure for the delete from source step. A single call to the procedure processes the partition exchange for all tables.

When the archive job calls the procedure, the procedure performs the following steps for each table that is configured for partition exchange:

1. The procedure creates a keep table in the staging user. The keep table contains all records from the source table that are not flagged for archiving. The run procedure uses the following naming convention to create the keep table:

   ```
   <table name>_KEEP
   ```

2. The procedure adds records to the keep table. To determine which records to add, the procedure subtracts the records in the staging table from the source table. The remaining records are not included in the archive job and should remain in the source.
3. The procedure invokes the Oracle partition exchange feature to swap the segments from the source table with the segments from the keep table.
4. The procedure rebuilds the source table indexes.
5. The procedure drops the keep table.

The procedure performs the steps for one source table at a time. After the procedure processes the first source table, the procedure performs the cycle again for the next source table. If the source table includes multiple partitions, then the procedure performs the steps for one partition at a time.

Database Requirements

Additional staging space in the source staging tablespace is required to create the keep table and to store data in the keep table. The staging space amount depends on the amount of data that is going to be kept in the source table.

When the archive job runs the partition exchange procedure, the procedure only creates one keep table. When the database completes the partition exchange for the source table, the procedure drops the keep table. This process is repeated for every table that is configured for partition exchange. The size of the keep table depends on each source table. Verify that you have enough staging space for the maximum difference between the source and staging tables.

To estimate the amount of required staging space, configure the archive job to pause after the copy to staging step. Review the log to calculate the sizes of the staging tables and the source tables.

Configuring Partition Exchange

To configure partition exchange, create and configure the interim tables, configure the entity, and verify primary key constraints.

1. Create and configure interim tables.
   Create an interim table and add the source tables. The archive job uses partition exchange to delete records from the tables that you add to the interim table. Then, configure the default columns in the interim table.
2. Configure the entity steps.
   Add steps to process the interim table you created, to disable the individual record deletion, and to enable the partition exchange run procedure.
3. Verify that the primary key constraints are defined for the source tables.

Step 1. Create and Configure the Interim Tables

Create an interim table and add the source tables. The archive job uses partition exchange to delete records from the tables that you add to the interim table. Then, configure the default columns in the interim table.

Add columns for the table owner, table name, and partition name and define the corresponding select clauses. The partition name select clause depends on the partitioned status of the source tables in the interim table. The configuration depends on if all of the source tables in the interim table are partitioned, not partitioned, or both.

Select Clause for Partitioned Source Tables

If all tables in the interim table are partitioned in the source, the archive job uses the ALL_TAB_PARTITIONS Oracle view to populate the interim table values.

The select clause you configure depends on if the tables have the same table owner or multiple table owners. You can add multiple tables for each table owner if the tables have the same partition name. The procedure can process one partition for each table owner. If the table includes multiple partitions, then you must list the table owner, table name, and each partition name.

Single Table Owner Select Clause

Use the following syntax to form a select clause for the partition name when the interim tables have the same table owner:

```
SELECT A.table_owner, A.table_name, A.partition_name
FROM all_tab_partitions A
WHERE table_owner='<table owner>' AND table_name IN ('<table name>') AND
partition_name='<partition name>'
```

For example, you want to add the following values in the interim table:

Table Owner	Table Name	Table Partition
XLA	XLA_AE_HEADERS	CST

Use the following syntax:

```
SELECT A.table_owner, A.table_name, A.partition_name
FROM all_tab_partitions A
WHERE table_owner='XLA' AND table_name IN ('XLA_AE_HEADERS') AND partition_name='CST'
```

Single Table Owner with Multiple Partitions within a Table Select Clause

Use the following syntax to form a select clause for the partition name when the interim tables have the same table owner and include multiple partitions for a table:

```
SELECT A.table_owner, A.table_name, A.partition_name
FROM all_tab_partitions A
WHERE table_owner='<table owner>' AND table_name IN ('<table name>') AND partition_name
IN ('<partition name>','<partition name>')
```

For example, you want to add the following values in the interim table:

Table Owner	Table Name	Table Partition
XLA	XLA_AE_HEADERS	CST
XLA	XLA_AE_HEADERS	AR

Use the following syntax:

```
SELECT A.table_owner, A.table_name, A.partition_name
FROM all_tab_partitions A
WHERE table_owner='XLA' AND table_name IN ('XLA_AE_HEADERS') AND partition_name IN
('CST','AR')
```

Multiple Table Owners Select Clause

Use one select statement for each table owner. Use a UNION statement to join multiple select statements.

Use the following syntax to form multiple select statements when the interim tables have different table owners:

```
SELECT A.table_owner, A.table_name, A.partition_name
FROM all_tab_partitions A
WHERE A.table_owner='<table owner 1>' AND A.table_name IN ('<table name>') AND
A.partition_name='<partition name>'
UNION
SELECT B.table_owner, B.table_name, B.partition_name, 'Y', null, null
FROM all_tab_partitions B
WHERE B.table_owner='<table owner 2>' AND B.table_name IN ('<table name>') AND
B.partition_name='<partition name>'
UNION
SELECT C.table_owner, C.table_name, C.partition_name, 'Y', null, null
FROM all_tab_partitions C
WHERE C.table_owner='<table owner 3>' AND C.table_name IN ('<table name>') and
C.partition_name='<partition name>'
```

For example, you want to add the following values in the interim table:

Table Owner	Table Name	Table Partition
XLA	XLA_AE_LINES	CST
APPLSYS	WF_ITEM_ACTIVITY_STATUSES	WF_ITEM45
PA	PA_SUMM_BALANCES	PA_SUMM_BALANCES_PAR11

Use the following syntax:

```
SELECT A.table_owner, A.table_name, A.partition_name
FROM all_tab_partitions A
WHERE A.table_owner='XLA' AND A.table_name IN ('XLA_AE_LINES') AND
A.partition_name='CST'
UNION
SELECT B.table_owner, B.table_name, B.partition_name, 'Y', null, null
FROM all_tab_partitions B
WHERE B.table_owner='APPLSYS' AND B.table_name IN ('WF_ITEM_ACTIVITY_STATUSES') AND
B.partition_name='WF_ITEM45'
UNION
SELECT C.table_owner, C.table_name, C.partition_name, 'Y', null, null
FROM all_tab_partitions C
WHERE C.table_owner='PA' AND C.table_name IN ('PA_SUMM_BALANCES') AND
C.partition_name='PA_SUMM_BALANCES_PAR11'
```

Select Clause for Non-Partitioned Source Tables

If all tables in the interim table are not partitioned in the source, the archive job uses the ALL_TABLES Oracle view to populate the interim table values. Note that the partition name column will not include any values.

The select clause you configure depends on if the tables have the same table owner or multiple table owners. You can add multiple tables for each table owner.

Single Table Owner Select Clause

Use the following syntax to form the select clause for the table name when the interim tables have the same table owner:

```
SELECT A.owner, A.table_name
FROM all_tables A
WHERE A.owner='<table owner>' AND A.table_name IN ('<table name>')
```

For example, you want to add the following values in the interim table:

Table Owner	Table Name
GL	GL_IMPORT_REFERENCES

Use the following syntax:

```
SELECT A.owner, A.table_name
FROM all_tables A
WHERE A.owner='GL' AND A.table_name IN ('GL_IMPORT_REFERENCES')
```

Multiple Table Owners Select Clause

Use one select statement for each table owner. Use a UNION statement to join multiple select statements.

Use the following syntax to form the select clause for multiple select statements when the interim tables have different table owners:

```
SELECT A.owner, A.table_name, null, 'Y', null, null
FROM all_tables A
WHERE A.owner='<table owner 1>' AND A.table_name IN ('<table 1 name>', '<table 2 name>')
UNION
SELECT B.owner, B.table_name, null, 'Y', null, null
FROM all_tables B
B.owner='<table owner 2>' AND B.table_name IN ('<table 1 name>', '<table 2 name>')
UNION
SELECT C.owner, C.table_name, null, 'Y', null, null
FROM all_tables C
C.owner='<table owner 3>' AND C.table_name IN ('<table 1 name>', '<table 2 name>')
```

For example, you want to add the following values in the interim table:

Table Owner	Table Name
GL	GL_IMPORT_REFERENCES
	GL_JE_LINES
XLA	XLA_AE_HEADERS
	XLA_AE_LINES
APPLSYS	WF_ITEMS
	WF_ITEM_ATTRIBUTE_VALUES

Use the following syntax:

```
SELECT A.owner, A.table_name, null, 'Y', null, null
FROM all_tables A
WHERE A.owner='GL' AND A.table_name IN ('GL_IMPORT_REFERENCES', 'GL_JE_LINES')
UNION
SELECT B.owner, B.table_name, null, 'Y', null, null
FROM all_tables B
B.owner='XLA' AND B.table_name IN ('XLA_AE_HEADERS', 'XLA_AE_LINES')
UNION
```

```
SELECT C.owner, C.table_name, null, 'Y', null, null
FROM all_tables C
C.owner='APPLSYS' AND C.table_name IN ('WF_ITEMS', 'WF_ITEM_ATTRIBUTE_VALUES')
```

Select Clause for Partitioned and Non-Partitioned Source Tables

If the interim tables include a combination of partitioned tables and non-partitioned tables, the archive job uses a union of the ALL_TAB_PARTITIONS and ALL_TABLES Oracle views to populate the interim table values.

Use the following syntax to form the select clause for the partition name:

```
SELECT A.table_owner, A.table_name, A.partition_name
FROM all_tab_partitions A
WHERE table_owner='<table owner>' AND table_name IN ('<table name>') AND
partition_name='<partition name>'
UNION
SELECT B.owner, B.table_name,null,'Y',null,null
FROM all_tables B
WHERE B.owner='<table owner>' AND B.table_name IN ('<table name>')
```

For example, you want to add the following values in the interim table:

Table Owner	Table Name	Partition Name
XLA	XLA_AE_HEADERS	CST
	XLA_AE_LINES	
GL	GL_IMPORT_REFERENCES	(None)
	GL_JE_LINES	

Use the following syntax:

```
SELECT A.table_owner, A.table_name, A.partition_name
FROM all_tab_partitions A
WHERE table_owner='XLA' AND table_name IN ('XLA_AE_HEADERS','XLA_AE_LINES') AND
partition_name='CST'
UNION
SELECT B.owner, B.table_name,null,'Y',null,null
FROM all_tables B
WHERE B.owner='GL' AND B.table_name IN ('GL_IMPORT_REFERENCES','GL_JE_LINES')
```

Creating and Configuring the Interim Tables

Before you begin, identify the tables that you want to delete data from by using partition exchange. In the corresponding entities, create interim tables and add the tables. Then, configure the interim table.

1. Access the entity that includes the tables.
2. Create an interim table.

   You may want to indicate in the description that the interim table is for partition exchange.
3. In the **Tables** tab, add the tables.
4. In the **Default Columns** tab, add the following entries for the table owner, table name, and partition name:

Order	Name	Type	Length	Select Clause
1	TABLE_OWNER	VARCHAR2	50	A.TABLE_OWNER
2	TABLE_NAME	VARCHAR2	50	A.TABLE_NAME
3	PARTITION_NAME	VARCHAR2	50	A.PARTITION_NAME

5. Configure the select clause for the partition name.

The configuration depends on if all of the source tables in the interim table are partitioned, not partitioned, or both. If the source table is partitioned and you archive data from multiple partitions in the table, then list each partition from which you archive data.

6. Save the interim table.

Step 2. Configure the Entities

After you add an interim table to the entity, configure the entity steps. Configure steps to process the interim table you created, to disable the individual record deletion, and to enable the partition exchange run procedure.

1. Access the entity **Steps** tab.
2. Add the Drop Interim Table, Create Interim Table, and Insert Interim Table steps for the partition exchange interim table.

   The order can be at any point in candidate generation.
3. Add a Run Procedure in Generate Candidates step for the partition exchange interim table and add the `am_plsqlutil_pkg.compile_self` procedure.

   The procedure is required for the `AA_PARTITION_PKG.exchange_partition` procedure to run the partition exchange logic.
4. Add a Run Procedure in Generate Candidates step for the partition exchange interim table and add the `aa_util_pkg.compile_self` procedure.

   The procedure is required for the `AA_PARTITION_PKG.exchange_partition` procedure to run the partition exchange logic.
5. Add a Run Procedure in Generate Candidates step for the partition exchange interim table and add the `aa_partition_pkg.compile_self` procedure.

   The procedure allows the `AA_PARTITION_PKG` package to compile objects in the staging user.
6. Delete or disable the corresponding Delete Apps Tables step for each table that you added to the interim table.
7. After the Insert into Archive Tables steps, add a Run Procedure in Delete from Source step for the partition exchange interim table and add the `AA_PARTITION_PKG.exchange_partition` procedure.
8. Save the entity.

Example

You created the XC_384_XLA2 Exchange Partition interim table for the XLA Subledger Accounting entity. You added the XLA_AE_HEADERS and XLA_AE_LINES tables to the interim.

The following table lists the steps that you configure in the entity Steps tab:

Configuration	Action	Interim Table	Entity Table	Procedure	Enabled
Add	Drop Interim Table	XC_384_XLA2 Exchange Partition	(None)	(None)	Yes
Add	Create Interim Table	XC_384_XLA2 Exchange Partition	(None)	(None)	Yes

Configuration	Action	Interim Table	Entity Table	Procedure	Enabled
Add	Insert Interim Table	XC_384_XLA2 Exchange Partition	(None)	(None)	Yes
Add	Run Procedure in Generate Candidates	XC_384_XLA2 Exchange Partition	(None)	am_plsqlutil_pkg.compile_self	Yes
Add	Run Procedure in Generate Candidates	XC_384_XLA2 Exchange Partition	(None)	aa_util_pkg.compile_self	Yes
Add	Run Procedure in Generate Candidates	XC_384_XLA2 Exchange Partition	(None)	aa_partition_pkg.compile_self	Yes
Delete or Disable	Delete Apps Tables	XC_384_XLA1 Subledger Accounting	XLA_AE_HEADERS	(None)	No
Delete or Disable	Delete Apps Tables	XC_384_XLA1 Subledger Accounting	XLA_AE_LINES	(None)	No
Add	Run Procedure in Delete From Source	XC_384_XLA2 Exchange Partition	(None)	AA_PARTITION_PKG.exchange_partition	Yes

After you configure the entities, configure the source connection to use staging. Staging is required because the procedure creates the keep table in the staging database user.

Step 3. Verify the Primary Key Constraints

Verify that the primary key constraints are defined for the source tables that you added to the partition exchange interim tables. By default, pre-packaged accelerators include key constraints. If you modified or added metadata within the pre-packaged accelerators or if you have custom applications, verify that the primary key constraints exist.

The archive job uses a combination of the row ID and the primary key to delete from the source. If the primary key constraint is not defined, then you receive an error in the job log. The job cannot populate the keep table without the primary key.

Managing Partition Exchange

You can enable or disable partition exchange configuration based on the archive job requirements. After you configure partition exchange and run an archive job, you may want to disable or remove the partition exchange configuration. Whether you disable or remove the configuration depends on if you want to enable the configuration again in the future.

You may want to enable the configuration for some jobs and disable it for other jobs. For example, you configured partition exchange for the first time you ran an archive job due to the volume of the data to archive. You want to disable or remove the partition exchange because the subsequent archive job volume is considerably less than the initial archive. However, if the volume changes in the future, you can enable partition exchange again.

You may want to disable or remove partition exchange for restore jobs as partition exchange may not be the optimized method of deletion for some restore jobs. For example, partition exchange is not recommended to restore transactions or to restore a cycle that has a small percentage of data in the history table.

Disabling or Removing Partition Exchange

To remove or disable partition exchange, configure the entity steps.

1. In the entity steps, disable or remove the Run Procedure in Delete from Source step that calls the `AA_PARTITION_PKG.exchange_partition` procedure.
2. Enable or add a Delete Apps Tables step for each table that you included in the partition exchange interim table.

CHAPTER 6

APIs

This chapter includes the following topic:

TPT_EXTRACT_SCRIPT_TEMPLATE

TPT_EXTRACT_SCRIPT_TEMPLATE is the Teradata Parallel Transporter default script template. Archive or retirement jobs can use Teradata Parallel Transporter to extract data from Teradata to the Data Vault. You enable Teradata Parallel Transporter in the Teradata source connection.

The script template includes supported Teradata scripting language and parameters. The job uses the script template that is configured in the Teradata source connection to create one script for each table the job exports data from. During the script creation, the job replaces the parameters with values that are specific to the table. The job uses the script to pass the query request to Teradata Parallel Transporter. Teradata Parallel Transporter rewrites the query request to optimize the query for maximum performance and throughput. Then, Teradata Parallel Transporter creates one BCP file for each table.

No additional configuration is required for the script template. TPT_EXTRACT_SCRIPT_TEMPLATE is the default script template name in the Teradata source connection.

Optionally, you can copy the script template and modify the copied script template. You may want to modify the script template if you want to use additional parameters or to change the scripting language. For example, you may modify the script to optimize performance. If you copy the script template, you must update the script template name in the Teradata source connection.

Script Template Parameters

The Teradata Parallel Transporter script template includes default parameters. The archive or retirement job replaces the parameters at runtime when the job creates scripts for tables.

The default parameters are required. No additional configuration is needed. Optionally, you can add parameters to the script. For example, you may want to use a different login name. You can use any property from the Teradata source connection as a script parameter. To add a parameter, use the technical name from the AM_DATA_REP_ATTR table in the ILM repository.

The script template includes the following required parameters:

&TABLE_NAME

Table name that the job needs to export data from.

&JOB_ID

Job ID of the archive or retirement job. Teradata Parallel Transporter uses the job ID as part of the BCP file name.

&DIR_PATH

BCP file staging directory in the Teradata source connection.

&TDPID

Host name in the Teradata source connection.

&SESSION

Maximum number of parallel sessions in the Teradata source connection.

&USER_NAME

Application login name in the Teradata source connection.

&PASSWORD

Password for the application login name in the Teradata source connection.

&SELECT_STMT

SQL select statement that the job creates to extract the data from Teradata.

&FILE_NAME

Name of the BCP file that the Teradata Parallel Transporter creates. By default, the job uses the following convention to create BCP files:

```
<Job ID>_<Table Name>.bcp
```

CHAPTER 7

Smart Partitioning

This chapter includes the following topics:

Smart Partitioning Overview

Smart partitioning is a process that divides application databases into independent, relationally intact data sets called segments. Segments are database partitions with a business definition.

Smart partitioning uses native database partitioning methods to create segments that increase application performance and help you manage application data growth. You can query and manage segments independently, which increases application response time and simplifies processes such as database compression. You can also restrict access to segments based on application, database, or OS users.

Before you create segments, you group the application data into segmentation groups that define database and application data relationships. You also create dimensions, which are criteria like time or region, that add a business definition to the segments you create. You configure segmentation groups and dimensions in the Enterprise Data Manager.

After you configure segmentation groups and dimensions, you add the dimensions to a data classification based on business practices and the application data model. Then you create a segmentation policy to assign the segmentation group to a data classification and create the segments. When you run the segmentation policy, the ILM Engine divides the segmentation group consistently to create segments based on the dimensions that you associate with a data classification.

Smart Partitioning Example

To increase application response time, you want to divide the transactions in a high-volume general ledger application module so that the application queries specific segments. You decide to create segments based on calendar year so that users can easily query the module by year. In the Enterprise Data Manager you create a time dimension that you configure as a date range. Then you create a general ledger segmentation group that contains related general ledger tables. In the Data Archive user interface you create a data classification and configure the time dimension to create segments for all general ledger transactions by year, from 2010 through the current year.

When you run the segmentation policy, the ILM Engine divides each table across the general ledger segmentation group based on the dimension you defined. The smart partitioning process creates a segment that contains a tablespace for each year of general ledger transactions. The smart partitioning process also creates a default segment for new transactions and transactions that do not meet business rule requirements for the other segments. As users enter new transactions in the general ledger, the application inserts the transactions in the default segment where they remain until you move them to another segment or create a new default segment.

Dimensions

A dimension determines the method by which the segmentation process creates segments, such as by time or business unit. Dimensions add a business definition to data, so that you can manage it easily and access it quickly. You create dimensions during smart partitioning implementation.

The Data Archive metadata includes a dimension called time. You can use the time dimension to classify the data in a segment by date or time. You can also create custom dimensions based on business needs, such as business unit, product line, or region.

When you implement smart partitioning, you choose to use single-dimensional or multidimensional data classifications. A single-dimensional data classification uses one dimension to create segments. A multidimensional data classification uses more than one dimension to create segments.

When you create a dimension, you must configure the type and datatype. The type can be list or range, and the datatype can be date, number, or string. If you want to reuse a dimension, you can associate it with any segmentation group.

Before you begin the segmentation process, you create dimension slices that specify how the data is organized on the dimension. The first time you create segments for a segmentation group, the ILM Engine leverages native database partitioning methods to create segments for each dimension slice in the data classification.

Dimension Example

A group of application users need access to only the current fiscal year of transactions in an accounts receivable application module. To save space on your production database you decide to create segments for every fiscal year of transactions in the AR module and then compress the segments that contain data from previous years. You choose the time dimension with a type of range and a datatype of date. In the Data Archive user interface you add dimension slices for the time dimension you created. When you schedule the segmentation process, the ILM Engine creates a segment for each dimension slice.

The following table shows dimension slices for the time dimension that you configured:

Dimension Slice Name	Dimension Slice Value
FY2012	Jan 1, 2012 – Dec 31, 2012
FY2011	Jan 1, 2011 – Dec 31, 2011
FY2010	Jan 1, 2010 – Dec 31, 2010

Single-dimensional Data Classification

A single-dimensional data classification uses one dimension to divide the application data into segments.

Use a single-dimensional data classification when you want to create segments for application data in one way, such as by year or quarter. Single-dimensional data classifications often use the time dimension.

For example, you manage three years of application data that you want to divide into segments by quarter. You create a time dimension and select range as the dimension type and date as the datatype. When you run the segmentation policy the ILM Engine creates 13 segments, one for each quarter and one default segment. The data in the default segment includes all of the data that does not meet the requirements for a segment, such as transactions that are still active. The default segment also includes new transactional data.

Multidimensional Data Classification

A multidimensional data classification uses more than one dimension to divide the application data into segments.

You can include more than one dimension in a data classification, so that Data Archive uses all the dimensions to create the segments. When you apply multidimensional data classifications, the ILM Engine creates segments for each combination of values.

For example, you want to create segments for a segmentation group by both a date range and sales office region. The application has three years of data that you want to create segments for. You choose the time dimension to create segments by year. You also create a custom dimension called region. You configure the region dimension to create segments based on whether each sales office is in the Eastern or Western sales territory.

When you run the segmentation policy, the ILM Engine creates seven segments, one for each year of data from the Western region, one for each year of data from the Eastern region, and a default segment. Each non-default segment contains the combination of one year of data from either the East or West sales offices. All remaining data is placed in the default segment. The remaining data includes data that does not meet the requirements for a segment, such as transactions that are still active, plus all new transactions.

Dimension Properties

You can view and configure dimension properties, such as dimension name, type, datatype, and parameters.

The following table describes the dimension properties:

Field Name	Description
Name	Name of the dimension. Populated with the name you enter when you create the dimension. Choose dimension names that are short and descriptive, such as "year" or "division."
Description	Short description of the dimension. Populated with the description you enter when you create the segmentation group. This field is optional.
Dimension Type	Type of the dimension, either list or range. Use the range dimension type with the time dimension to create segments by date range. Use the list dimension type with dimensions such as business unit, region, or product line.

Field Name	Description
Data Type	Data type of the dimension. Select date, number, or string. - Use the date data type if you want to create segments by date. - Use the number data type when the values associated with the dimension are numeric. - Use the string data type when the values associated with the dimension are alphanumeric, such as region.
Interim Table Column Name (Time dimension)	Name of the interim table column. If you choose to use interim tables to pre-process business rules for a segmentation group, enter a name for the interim table column. The column will be added to the interim table when it is created. The column will also be referenced in the dimension rule.
Parameter Name	Name of the parameters. Populated based on the dimension type that you chose. If you create a range dimension called "YEAR," the parameter name defaults to "YEAR_From" for a parameter type of FROM.
Parameter Type	Predetermined parameters based on the dimension type that you choose. A range dimension type has a FROM and TO parameter. A list dimension type has a TO parameter. This field is read-only.

Creating a Dimension

Create a dimension to add to a data classification. The dimensions and dimension slices in a data classification determine how the segmentation process creates segments.

1. From the **View** menu, select **Dimensions**.

   The **Dimensions** tab appears at the bottom of the Explorer pane.
2. Right-click **Dimensions** in the Explorer pane.
3. Select **New Dimension**.

   The **Add New Dimension** wizard appears.
4. Enter a dimension name. Optionally, enter a dimension description.
5. Click **Finish.**

   The dimension appears in the Explorer pane.
6. On the **General Information** tab in the Details pane, select a dimension type of LIST or RANGE.
7. Select a datatype of DATE, NUMBER, or STRING.

After you create a dimension, you can associate it with a data classification in the Data Archive user interface.

Segmentation Groups

A segmentation group is a group of tables that defines database and application relationships. A segmentation group represents a business function, such as order management, accounts payable, or human resources. Segmentation groups are similar to entities in archiving.

The smart partitioning process creates segments by dividing each table consistently across the segmentation group. This process organizes transactions in the segmentation group by the dimensions you associated with the group. This method of partitioning maintains the referential integrity of each transaction.

With smart partitioning, a table can belong to one segmentation group at a time. Segmentation groups might not include every table in an application module because not every table is an ideal candidate for segmentation. Tables with a large and growing number of transactions are the best candidates for segmentation.

If you purchased an application accelerator such as the Oracle E-Business Suite or PeopleSoft accelerator, some segmentation groups are predefined for you. If you need to create a segmentation group manually, you must first import metadata from the application database and create an application module. Then you can select the tables you want to include in the segmentation group.

After you select the segmentation group tables you must mark the driving tables, enter constraints, associate dimensions with the segmentation group, and define business rules.

Segmentation Group Properties

You can view and configure segmentation group properties, such as name, description, and driving table.

The following table describes general segmentation group properties:

Field Name	Description
ID	ID number for the segmentation group. ID is an internal identifier that is generated by the database. This field is read-only.
Name	Name of the segmentation group. Populated with the name you enter when you create the segmentation group. Name is restricted to 13 alphanumeric characters and cannot contain special characters. When you begin the smart partitioning process in the Data Archive user interface, the ILM Engine generates metadata for the segmentation group and you cannot change the segmentation group name.
Description	Short description of the segmentation group. Populated with the description you enter when you create the segmentation group. This field is optional.
Driving Table	Selected driving table for the segmentation group. This field is read-only.

Tables

You must select the tables that you want to include in a segmentation group. Any table included in the application entity is eligible for inclusion in the segmentation group.

Consider the following guidelines when you add tables to a segmentation group:

- Large tables and tables with a negative impact on application performance are more likely to benefit from segmentation.
- If you want to include a child table, you must include the parent.

- If you want to archive the segmentation group in the future, especially to the Data Vault, you might want the application module to be complete.

If you do not include an application entity table in the segmentation group, the table remains visible and available without functional issues in the application. You will not achieve performance benefits for excluded tables.

Table indexes can help optimize smart partitioning performance and significantly reduce smart partitioning run time. Typically smart partitioning uses the index predefined by the application. If multiple potential indexes exist in the application schema for the selected table, you must register an index after you add the table to the segmentation group.

If the table does not have an index sufficient for partitioning, you can choose to create a temporary optimization index. When you configure a temporary optimization index, you give the index a name and specify the columns you want to use for indexing. The smart partitioning process creates the index dynamically when you run a segmentation policy and drops the index when segmentation is complete.

When you register a predefined index or create a temporary optimization index, you must decide when the partitioning process uses the index. You can configure the index to be used in the default segment creation process, non-default segment creation process, or both.

Table Properties

You can view and configure properties for the tables included in a segmentation group. Required table properties include the table name and desired alias name.

The following table describes segmentation group table properties:

Field Name	Description
Table Name	Name of a table included in the segmentation group. The drop-down menu lists available tables.
Table Alias Name	Alias name for an included table. Because you can use the same table multiple times in an segmentation group, you may need to create alias names for some tables. If the table has one or more aliases, select the desired alias.
Use row count for space allocation	Displays the row count of each segment before you schedule the segmentation process. This field is optional.
History Segment Select Hint	Text used for selection SQL during segmentation and creation of split segments. This field is optional.
Insert_Update_Hint_Text	Text used to improve performance of SQL INSERT or UPDATE. This includes segmentation and all incremental data movement. This field is optional.
Optimization Index	Indicates whether the table has an associated optimization index.
Default Segment Select Hint	Inserts hints in the SELECT query that creates the default segment. This field is optional.
Default Segment Sub Query Hint	Inserts hints in the SUB query that creates a default segment. This field is optional.
History Segment Sub Query Hint	Inserts hints in the SUB query that creates a history segment. This field is optional.

Field Name	Description
ROWID Select Hint	Inserts hints in the ROWID SELECT query. This field is optional.
Row Count Hint	Inserts hints for the row count query during the Create Audit Snapshot job. This field is optional.
GatherStats PD	Inserts a table-specific parallel degree for gathering statistics. This field is optional.

Table Tokens

You can use tokens to help create some of the performance hints available under the **Tables** tab. During runtime, the segmentation process replaces the tokens with the actual table or index names, and parameter values.

The following table describes table tokens:

Token Name	Hint Column	Description
SEL_TABLE	Insert Update Hint Text	Use as a substitute for the selection table name.
$PARTITIONED_TAB_INDEX$	Default Segment Select Hint	Use as a substitute for the index name of the segmented table.
$PARTITIONED_TABLE$	Default Segment Select Hint	Use as a substitute for the name of the segmented table.
$INTERIM_TABLE$	History Segment Sub Query Hint	Use as a substitute for the name of the interim table.
$INTERIM_INDEX$	History Segment Sub Query Hint	Use as a substitute for the name of the interim table index.
$DML_PARALLEL_DEGREE$	Insert Update Hint Text	Use as a substitute for the value of the "Parallel_Degree_Insert_Only" parameter in the **Manage Segmentation Setup** page.

Optimization Index Properties

You can view and configure properties for the indexes associated with tables in a segmentation group. Required optimization index properties include the name and whether or not the index is temporary.

Field Name	Description
Index Name	Name of the optimization index for the selected table. If the index is predefined in the application schema, enter the name of the index you want to use. If you want to create a temporary index, enter a unique name.
Index Usage	Type of segment creation process that you want the index to be used for. Select default, non-default, or all.
Temporary	Allows you to designate whether the index is temporary or predefined.

Field Name	Description
Index Columns	Columns to use for indexing. If the index is predefined, this field is read-only.
Unique	Designates whether the index columns are unique. If the index is predefined, this field is read-only.

Business Rules

Use business rules to identify transactions that are still active. Active transactions remain in the default segment until the business rule criteria is met.

When you apply a business rule to the driving table of a segmentation group, the transactions that do not meet the business rule criteria will remain in the default segment. For example, you can create a business rule requiring that the invoices are fully paid and the checks are cleared for an invoice driving table.

If you want to use interim table processing to preprocess the business rules you create, transactions that meet the business rule criteria you specify will remain in the default segment. If you plan to use interim table processing for business rules, create business rules with criteria for keeping the transactions in the default segment.

Business Rule Properties

Enter business rule properties to create business rules that identify active transactions.

The following table describes the business rule properties:

Field Name	Description
Driving Table Alias	Name of the driving table for the segmentation group. Select from a drop-down list of available tables.
Business Rule Name	Name to describe the business rule.
Business Rule	A WHERE clause fragment containing logic that identifies active transactions. For example: TDI_DEMO_ORDER_HEADERS.CLOSED_DATE between DECODE(:DATE_FROM, 'All', CLOSED_DATE, :DATE_FROM) and DECODE(:DATE_TO,'All', CLOSED_DATE, :DATE_TO)
Interim	Indicates whether you have chosen to pre-process the business rules using interim tables. This field is read-only.

Entity Constraints

Constraints identify the relationships in the data model that are used to create segments for the segmentation group.

Every table in a segmentation group that is not a driving table must have a constraint that identifies its relationship with the parent table. Add only the constraints that define the necessary relationships for segmentation.

For example, you want to create an Orders segmentation group with three tables named ORDERS, ORDERS_LINES, and ORDER_SHIPMENTS. You mark the ORDERS table as the driving table, ORDER_LINES as a child table of ORDERS, and ORDER_SHIPMENTS as a child table of ORDER_LINES. The application defined

foreign keys in ORDER_SHIPMENTS to both of its parent tables, ORDERS and ORDER_LINES. When you create the constraints for smart partitioning, you use only one of the constraints to define the relationship.

Segmentation groups may have tables with multiple aliases. You must identify the constraints that exist between parent and child tables, as well as their aliases if they exist. For example, if the DOCUMENTS table is a child of both INVOICES_PARENT and CHECKS_PARENT, you must add the DOCUMENTS table to the model twice. The DOCUMENTS_1 alias has a constraint that associates the DOCUMENTS table to INVOICES_PARENT. The DOCUMENTS_2 alias has a constraint that associates the DOCUMENTS table with CHECKS_PARENT.

You can also add an expression to each constraint. Use an expression to restrict the rows in a child table to those that apply to the parent. For example, a general documents table could be referenced by many parents. The expression for a relationship to INVOICE_PARENT could be:

```
and DOCUMENTS.DOCUMENT_TYPE='INVOICES'
```

Entity Constraint Properties

Enter constraint properties to define the relationships between parent and child tables in the segmentation group.

The following table describes the segmentation group constraint properties:

Field Name	Description
Child Table	Name of child table included in the segmentation group.
Child Alias Name	Alias name of the selected child table. Identifies the specific table instance associated in this relationship.
Constraint	Reference to AM_META_CONSTRAINTS that defines the relationship between the parent and child tables.
Parent Table	Name of the parent table for the selected child table.
Parent Alias Name	Alias name of the selected parent table. Identifies the specific table instance associated in this relationship.
Expression	SQL fragment that restricts the number of rows in the child table to those that belong to the parent. This field is optional.

Entity Dimensions

Use dimension rules to associate one or more dimensions with a segmentation group.

If you want to create segments for a segmentation group using a new dimension, you need to add a dimension rule to the driving table of the segmentation group. A dimension rule is a WHERE clause fragment that associates the segmentation group driving table with the dimension parameters.

The dimension rule defines the logic used to segment the driving table by that dimension. You can associate one or more dimensions with a segmentation group using dimension rules. The dimensions you associate with a segmentation group define the segmentation criteria.

Entity Dimension Properties

Enter dimension rule properties to associate a dimension with a segmentation group.

The following table describes dimension rule properties:

Field Name	Description
Order	Order in which the dimension appears in the Data Archive user interface.
Driving Table Name	Name of the segmentation group driving table. The drop-down menu lists available driving tables.
Dimension Rule Name	Name of the dimension rule. Enter a descriptive name for the dimension rule.
Dimension Name	Name of the dimension associated with the driving table. The drop-down menu lists available dimensions.
Datatype	Datatype of the selected dimension. Datatype is date, number, or string. This field is read-only.
Dimension Parameters	Parameters used to associate the driving table to the dimension. The dimension rule must contain references to these parameters. For example, if the dimension is time, the parameters are :TIME_FROM and :TIME_TO Dimension parameters are determined by the type of dimension, list or range. This field is read-only.
Dimension Rule	The WHERE clause fragment that associates the driving table with the dimension parameters. For example: CHECK_PARENT.CLEARED_DATE between :TIME_FROM and :TIME_TO For interim table preprocessing, the TIME dimension rule is a SELECT statement. For example: SELECT r.header_id, MAX(r.closed_date) max_date FROM DEMO_SRC.HEADERS5 r GROUP BY r_header.id
Interim	Indicates whether you have chosen to pre-process the business rules using interim tables. This field is read-only.

Driving Tables

A segmentation group must have at least one driving table to which all tables in the segmentation group are mapped with constraints.

For example, an Accounts Payable module could have segmentation groups named invoices and checks. You select the tables INVOICES_PARENTS and CHECKS_PARENTS as the driving tables for the segmentation groups. Then you associate each table in the module with one of the segmentation groups.

When you designate the driving tables for a segmentation group you must also add business rules and dimension rules to each driving table to determine how the table is segmented. During segmentation, the ILM Engine applies the business rules and dimension rules to each child table of the driving table.

You cannot delete a driving table from a segmentation group if the driving table has any child tables or constraints.

You can find driving table properties in a metadata table named XP_ENTITY_DRIVING_TABLES_EXT.

Business Rules Pre-processing

When you select a driving table for a segmentation group, you can also choose to pre-process the segmentation group business rules if the segmentation group tables are defined for interim table processing.

Business rule pre-processing optimizes the import and export process for smart partitioning, improves partitioning performance, and reduces the downtime associated with initial segmentation. When you select business rule pre-processing, Data Archive adds a step called Preprocess Business Rules to the segmentation policy job when you run it.

When you run the segmentation policy, the partitioning process creates three interim tables: a main interim table, a date interim table if you have applied the time dimension, and a business rule interim table. The job applies the dimension and business rules you provided and populates the interim tables accordingly in preparation to create segments.

If you define a segmentation group for interim table processing, the logic used for business rules and dimension rules changes. Transactions that meet the business rule criteria you specify will remain in the default segment. If you plan to use interim table processing for business rules, create business rules with criteria for keeping the transactions in the default segment.

Driving Table Properties

You can view and configure properties for the segmentation group driving tables. You can also choose to use interim tables for business rule pre-processing. If you choose to enable business rule pre-processing, the names and attributes of the interim tables are displayed below the driving table properties.

Field Name	Description
Driving Table	Driving table for the segmentation group. Choose from available tables.
Driving Table Alias	Alias of the driving table. Choose from available tables.
Use Interim Processing	Option to use interim tables to pre-process business rules. Select the checkbox to use interim table processing for the segmentation group.
Primary Key	Primary or unique key of the driving table.

Creating and Configuring a Segmentation Group

Before you schedule or run the segmentation process, create segmentation groups for each application module that you want to create segments for.

Complete the following tasks to create a segmentation group:

1. Import metadata from the application database.
2. Create an application module.
3. Create the segmentation group and enter the segmentation group details.

Step 1. Import Metadata from the Database

You must import metadata from the application database before you create a segmentation group.

1. Select an application version and click **File** > **Import Metadata from Database**.

   The **Connect to Import Metadata from Database** window appears.

2. Enter the database connection details and click **OK**.

   The **Import Metadata from Database** wizard appears.

3. Select the schemas to import metadata from.

   Click the down arrow button to insert the tables in to the Selected box. Use the control keyboard options to select multiple tables or all tables.

4. Click **Next.**

5. Choose one of the following options to run the job:

   - **Submit Import Metadata as a Background Job**. Runs the job in the background. If you run the job in the background, you can continue to perform other tasks. Run the job in the background if you have a large volume of metadata to import and to avoid memory errors.
   - **Continue Import Metadata through EDM**. Runs the job in the foreground. If you run the job in the foreground, you must wait until the job completes to perform an other task. Additionally, if you run the job in the foreground and you have a large volume of metadata to import, you may receive memory errors. You may want to run the job in the foreground if you have a low volume of metadata to import.

6. Click **Next**.

7. Enter the mining parameters.

8. Click **Finish.**

9. Click **OK**.

Step 2. Create an Application Module

Create an application module to which the segmentation group belongs.

1. From the **View** menu, select **Segmentation**.

   The **Segmentation** tab appears at the bottom of the Explorer pane.

2. Right-click the application version and select **New Application Module**. You can also select the application version, click on the **Insert** menu, and select **New Application Module**.

   The **Application Module** window appears.

3. Give the application module a name. Optionally, enter a description for the module.

4. Click **Finish**.

   The application module appears under the application version in the Explorer pane.

Step 3. Create and Configure the Segmentation Group

Create and configure the segmentation group. To configure the segmentation group you must select the included tables, mark the driving tables, add constraints, apply business rules, and add dimension rules.

1. In the Explorer pane, right-click on the application module you created in step 2.

2. Click on **New Entity**.

   The **Entity Wizard** appears.

3. Enter a name and optionally a description for the segmentation group.
4. Click **Finish**.

   The new segmentation group appears under the application module.
5. Click on the segmentation group you created.
6. Click on the **Tables** tab.
7. In the top pane, click on the green **Add** button and select a table to include in the segmentation group from the drop-down list of available tables.
8. If the table has multiple aliases, add the table multiple times and select each alias for inclusion.
9. If you want to register a table index or create a temporary index, select the table in the **Tables** pane.
10. In the **Optimization Indexes** pane, enter an index name.
11. Click in the **Index Usage** column and select whether the index will be used for default segment creation, non-default segment creation, or both.
12. If the index is temporary, select the **Temporary** checkbox.

    The **Index Columns** window appears.
13. Enter the columns you want to use for indexing.
14. For temporary indexes, select the **Unique** checkbox if the index is unique.
15. Click on the **Driving Tables** tab.
16. Click on the green **Add** button and select a driving table from the drop-down list of available tables. Repeat for multiple driving tables.
17. If you plan to use interim table processing to pre-process business rules for the segmentation group, select the **Use Interim Processing** checkbox and choose the primary key for the table.
18.
19. Click on the **Constraints** tab.
20. In the **Constraint** field, select the type of constraint that exists for each listed child table and child alias table.
21. Click on the **Dimension Rules** tab.
22. Double-click in the **Driving Table** field. From the drop-down menu of available driving tables, select the driving table on which to associate the dimension.
23. Double-click in the **Dimension Name** field. From the drop-down menu of available dimensions, select a dimension to associate with the segmentation group.

    The EDM populates the **Data Type** and **Dimension Parameters** fields based on the dimension you chose.
24. In the **Dimension Rule** field, enter the WHERE or SELECT clause that implements the dimension rule.

    For example: ORDER_DATE BETWEEN :DATE_FROM AND :DATE_TO
25. In the **Dimension Rule Name** field, give the dimension rule a name.
26. Click on the **Business Rules** tab.
27. In the **Driving Table Name** field, select the driving table on which to associate the business rule.
28. In the **Business Rule** field, enter the WHERE clause that implements the business rule.

    For example: ORDER_STATUS = 'CLOSED'
29. In the **Business Rule Name** field, give the business rule a name.

APPENDIX A

SAP Application Retirement Entities

This appendix includes the following topics:

SAP Application Retirement Entities Overview

Data Archive includes pre-packaged entities for SAP applications. No installation is required. Entities are available for transparent HR tables, transparent text tables, and attachments.

The Enterprise Data Manager includes an application for SAP. The application includes application modules for CRM, ERP, SCM, and SRM. Each application module has attachment entities. In addition, the ERP application module includes entities for the transparent HR PCL1-PCL5 cluster tables and the STXL text table.

The pre-packaged entities are for retirement only. The entities do not include constraints. If you need to use the Data Discovery portal to search the retired data, you must create entities that include constraints.

The pre-packaged entities are standard and are read-only. No installation is required. To modify the entities, use the Copy Application job to copy the SAP application to a customer-defined application. Copy the pre-packaged entities to the customer defined application, and then modify the copied entities. You may want to modify entities to change the run procedure statement. For example, you may want to change a procedure parameter to keep external attachments that are stored in an external file system in the original file system.

Entities for Attachments

Attachment entities are included for each SAP application module. The entities include tables that store attachments for each application module. There is an entity that includes the attachment link table, an entity

for tables that store attachments in the database, and an entity for tables that store attachments in ADK files in an external file system.

Each entity includes default entity steps. However, only the Insert into Archive Tables step is enabled. The rest of the steps are not relevant for retirement. The Insert into Archive Tables step determines from where the retirement job reads data for tables in the entity.

Each application module includes the following attachment entities:

<Application Module> Attachments Links

Includes the attachment link table, ZINFA_ATTCH_LINK. The entity includes the default Insert Into Archive Table step that reads directly from the database.

The ZINFA_ATTCH_LINK table includes information on the downloaded attachments so you can view the attachments after you retire them. If you archive attachments to a file system or external storage, then you can use the link table to build the path and file name for Data Discovery style sheets. If you archive attachments to the Data Vault, the table includes information to link attachments with the AM_ATTACHMENTS table.

<Application Module> Attachments Part 1

Extracts the attachments that are stored in the SAP database. The entity includes the default Insert Into Archive Table step that reads directly from the database.

<Application Module> Attachments Part 2

Extracts the attachments that are located in a file system or external storage.

The entity includes a `ArchiveSAPAttachmentTable` procedure for the Insert into Archive Table entity step. The procedure overrides the default entity step to connect to the database. The procedure calls an ABAP function module to read data from the file system or external storage.

The full path for the procedure is:

```
java://
com.application.archive.dao.impl.ArchiveSAPAttachmentTable(TABNAME=<attachment table
name>,ABAP_PRG_NAME=ZINFA_DOWNLOAD_ATTACHMENTS,EXT_STORAGE=N,DTFORMAT=yyyy-MM-dd-
HH.mm.ss)
```

The parameters are included in parenthesis. For attachments that are stored in file systems or external storage, you can configure if the retirement job keeps the attachments in the original location or if the job moves the attachments to the attachment staging directory that is configured in the source connection. By default, the retirement job moves all attachments to the attachment staging directory.

To keep the attachments in the original location, configure the following parameter:

```
EXT_STORAGE=Y
```

The metadata for the attachment is included in the ILM repository, however, the attachment remains in the original location.

Entities for Transparent HR Cluster Tables and Text Tables

The ERP application module includes entities for the transparent HR PCL1-PCL5 cluster tables and the STXL text table.

Each entity includes default entity steps. However, only the Insert into Archive Tables step is enabled. The rest of the steps are disabled as the steps are not relevant for retirement. The Insert into Archive Tables step determines from where the retirement job reads data for tables in the entity.

The default entities are only for tables that store data in the database. The entities include the `ArchiveSAPLogicalTable` procedure for the Insert into Archive Table entity step. The procedure overrides the default entity step to connect to the database. The procedure calls an ABAP function module to read data from the SAP application layer.

If you archived data from the HR cluster tables or the STXL text table, the Retirement Auto Entity Creation job creates separate entities for the tables that store data in ADK files.

The ERP application module includes the following entities:

HR Cluster Tables

Includes tables for the PCL1-PCL5 cluster tables.

Text Tables

Includes the STXL text table.

APPENDIX B

Import Formats for Constraints

This appendix includes the following topics:

Import Formats for Constraints Overview

This section enlists sample formats for bulk import of Constraints specified in an Excel Sheet.

Importing Referential Integrity Constraints

The following table explains each column of a sample Referential Integrity Constraints XLS file:

Column Name	Value
Table Name	Name of the Primary Table.
Constraint Name	Name of the Constraint.
Constraint Column	Column names where Constraint is imposed in the Primary and Secondary Tables.
Primary Key Name	Column name in the Primary Table which functions as its Primary key.
Ref Table	Secondary Table name.
Owner	Owner of the Primary Table (i.e. its Parent Application Module).
Ref Owner	Owner of the Secondary Table (i.e. its Parent Application Module).
Constraint Type	Type of Constraint (“R” for Referential Integrity).
Status	Specifies whether the Constraint is to be imposed currently.

Importing Primary Integrity Constraints

The following table explains each column of a sample Primary Integrity Constraints' XLS file.

Column Name	Value
Table Name	Name of the Table.
Constraint Name	Name of the Constraint.
Constraint Column	Column(s) which will act as Primary key.
Owner	Owner of the Table (i.e. its Parent Application Module).
Constraint Type	Type of Constraint ("P" for Primary Integrity).

Importing Column Translation Data

Column Translation data can also be imported in bulk by specifying a format for XLS. A brief explanation of fields is included in the following table:

Column Name	Value
Table Name	Name of the "first" Table.
Field Name	Column in the "first" Table that needs to be translated.
Related Table Name	Name of the "second" Table.
Related Field Name	Column in the "second" Table that will be either displayed or De-referenced for connecting to a "third" Table.
Constraint	Constraint that defines a relation between the "first" and "second" Tables.
Parent Table Alias	Alias for the "first" Table.
Ref Table alias	Alias for the "second" Table.
Where Clause	WHERE clause for the JOIN between the two Tables.

APPENDIX C

Glossary

application

A list of brand names that identify a range of ERP applications, such as Oracle and PeopleSoft.

application module

A list of supported Application Modules for a particular Application Version.

application version

A list of versions for a particular application.

archive

Informatica refers to a backed up database as an Archive. From the process perspective, it also means archiving data from an ERP/CRM instance to an online database using business rules.

archive definition

An Archive Definition defines what data is archived, where it is archived from, and where it is archived to.

archive engine

A set of software components that work together to archive data.

business rule

A business rule is criteria that determines if a transaction is eligible to be archived or moved from the default segment.

business table

Any Table that (with other Tables and Interim Tables) contributes to define an Entity.

custom object

An Object (Application / Entity) defined by an Enterprise Data Manager Developer.

Data Archive

A product of the Information Lifecycle Management Suite, which provides flexible archive / purge functionality that quickly, reduces the overall size of production databases. Archiving for performance, compliance, and retirement are the three main use cases for Data Archive.

Data Archive metadata

Metadata that is specific to a particular ILM product.

data destination

The database containing the archived data.

Data Model Metadata

Metadata that is common to Data Archive.

data source

The database containing the data to be archived.

de-referencing

The process of specifying inter-Column Constraints for JOIN queries when more than two Tables are involved. Enterprise Data Manager dictates that two Tables are specified first and then a Column in the Second Table is De-Referenced to specify a Third Table (and so on) for building the final JOIN query.

entity

An entity is a hierarchy of ERP/CRM tables whose data collectively comprise a set of business transactions. Each table in the hierarchy is connected to one or more other tables via primary key/foreign key relationships. Entity table relationships are defined in a set of metadata tables.

ERP application

ERP applications are software suites used to create business transaction documents such as purchase orders and sales orders.

home database

The database containing metadata and other tables used to persist application data. This is contained in the Schema, usually known as “AMHOME”.

interim

Temporary tables generated for an Entity for data archive.

metadata

Metadata is data about data. It not only contains details about the structure of database tables and objects, but also information on how data is extracted, transformed, and loaded from source to target. It can also contain information about the origin of the data.

Metadata contains Business Rules that determines whether a given transaction is archivable.

restore (Cycle)

A restore operation based on a pre-created Archive Project.

security group

A Security Group is used to limit a User (during an Archive Project) to archive only data that conforms to certain scope restrictions for an Entity.

staging schema

A staging schema is created at the Data Source for validations during data backup.

standard object

An Object (Application / Entity) which is pre-defined by the ERP Application and is a candidate for Import into Enterprise Data Manager.

transaction data

Transaction data contain the information within the business documents created using the master data, such as purchase orders, sales orders etc. Transactional Data can change very often and is not constant. Transaction data is created using ERP applications.

Transaction data is located in relational database table hierarchies. These hierarchies enforce top-down data dependencies, in which a parent table has one or more child tables whose data is linked together using primary key/foreign key relationships.

translation column

A column which is a candidate for a JOIN query on two or more Tables. This is applicable to Independent Archive, as Referential Data is backed up exclusively for XML based Archives and not Database archive, which contain only Transaction Data.

Note that such a scenario occurs when a Data Archive job execution extracts data from two Tables that are indirectly related to each other through Constrains on an intermediary Table.

user

Data Archive is accessible through an authentication mechanism, only to individuals who have an account created by the Administrator. The level of access is governed by the assigned system-defined roles.

virtual view

A virtual tables that includes SQL logic to convert the format of retired data.