Oracle® Fusion Middleware

User's Guide for Oracle Business Intelligence Data Warehouse Administration Console

11*g* Release 1 (11.1.1)

E14849-04

April 2012

Explains how to use the Data Warehouse Administration Console to create, configure, execute, and monitor modular data warehouse applications. Includes instructions for using the console to load, administer, and monitor the Oracle Business Analytics Warehouse.

ORACLE

Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Data Warehouse Administration Console, 11*g* Release 1 (11.1.1)

E14849-04

Primary Author: Jill Arehart

Contributors: Oracle Business Intelligence development, product management, and quality assurance teams.

Contents

3 Setup Tasks and Concepts for DAC Administrators

4 DAC Quick Start

5 Building and Running Execution Plans

6 Customizing ETL Processes

7 Defining and Managing Parameters

8 Designing Subject Areas

9 Managing Data Warehouse Schemas

10 Performance Tuning With DAC

11 Working With DAC Metadata Patches

12 Common Tasks Performed in the DAC

13 Integrating DAC With Other ETL Tools

14 Upgrading, Comparing and Merging DAC Repositories

15 DAC Functional Reference

Index

Preface

The *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Data Warehouse Administration Console* contains information about using the Data Warehouse Administration Console (DAC), a centralized console for management, configuration, administration, loading, and monitoring of the Oracle Business Analytics Warehouse.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Audience

This document is intended for data warehouse administrators and ETL developers and operators.

Related Documents

For more information, see the following Oracle Business Intelligence Applications 11*g* Release 1 (11.1.1) documents:

- The Oracle Business Intelligence Applications chapter in the *Oracle Fusion Middleware Release Notes* for your platform: http://docs.oracle.com/cd/E23943_01/relnotes.htm
- *Oracle Fusion Middleware Installation and Configuration Guide for Oracle Business Intelligence Applications*
- *Oracle Fusion Middleware Reference Guide for Oracle Business Intelligence Applications*

Also see the following documents in the Oracle Business Intelligence Enterprise Edition 11*g* Release 1 (11.1.1) documentation set:

- The Oracle Business Intelligence chapter in *Oracle Fusion Middleware Release Notes* for your platform

- *Oracle Fusion Middleware System Administrator's Guide for Oracle Business Intelligence Enterprise Edition*
- *Oracle Fusion Middleware Enterprise Deployment Guide for Oracle Business Intelligence*
- *Oracle Fusion Middleware Security Guide for Oracle Business Intelligence Enterprise Edition*
- *Oracle Fusion Middleware Developer's Guide for Oracle Business Intelligence Enterprise Edition*
- *Oracle Fusion Middleware Scheduling Jobs Guide for Oracle Business Intelligence Enterprise Edition*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

What's New in This Release

This chapter lists the new features in the current release of Oracle Business Intelligence Data Warehouse Administration Console (DAC). It also provides information about hardware and software requirements, platforms, and databases.

The chapter contains the following topics:

What's New in DAC

New DAC features include the following:

Functional Setup for Task Level Parameters

The functional setup of task level parameters is an operation that you now perform using the Oracle Business Intelligence Applications Configuration Manager. For more information, see Oracle Fusion Middleware Installation and Configuration Guide for Oracle Business Intelligence Applications.

DAC Security

See Chapter 1, "About DAC Security," for information about the new DAC security model. For information about the differences between standalone mode and Fusion Middleware mode, see "DAC Authentication in Oracle Fusion Middleware (WebLogic Server) Mode" and "DAC Authentication in Standalone Mode".

Enhancement for Index Creation

DAC now supports the creation of query indexes in parallel immediately following the creation of ETL-related indexes. DAC notifies the successor tasks as soon as the ETL indexes have been created, allowing query indexes and additional unique indexes to be created sooner. See "Managing Indexes" for more information.

Creation of Indexes in Parallel for a Specific Table

DAC now enables you to specify how many indexes can be created in parallel for a specific table or how many indexes can be created in parallel for all tables associated with a specific physical data source connection. See "Specifying How Many Indexes Can Be Created in Parallel" for more information.

Testing an Individual Workflow

You can test an individual workflow before running a full execution plan. See "Unit Testing Execution Plan Tasks" for more information.

Enhancement for Prune Time Property

See "Execution Plans Tab: Connectivity Parameters Subtab" for more information.

Running Execution Plans Concurrently

You can configure DAC to run multiple execution plans concurrently if the execution plans are independent of one another, that is, as long as the execution plans do not load data into the same table on the same physical data source. See "Running Execution Plans Concurrently" for more information.

Looping of Workflows

You can configure the full and incremental load commands for tasks to repeat (or loop) multiple times during the execution of an ETL process. See "Looping of Workflows" for more information.

Event Delays

You can set up event delays to configure the extracts for different data sources to occur independently. See "Setting Up Extract Delays, Event Delays and Data Source Notifications" for more information.

Using Heuristics to Manage Tasks, Tables and Indexes

The DAC heuristics feature enables you to gather intelligence about the amount of incremental data that will be processed during an ETL execution. To optimize performance, you can use this intelligence to decide how to manage the behavior of tasks, tables, and indexes. See "Using Heuristics to Manage Tasks, Tables and Indexes" for more information.

Integrating DAC With Other ETL Tools

The external executor framework enables you to integrate DAC with ETL engines other than Informatica. See "Integrating DAC With Other ETL Tools" for more information.

Working With DAC Metadata Patch Sets

The DAC metadata patch feature enables you to import and export subsets of DAC metadata at a fine grain. See "Working With DAC Metadata Patches" for more information.

Creating and Upgrading the Data Warehouse Schema

DAC provides new functionality to create or upgrade data warehouse schemas. See Chapter 9, "Managing Data Warehouse Schemas," for more information.

EM Beans Implementation

Fusion Middleware Control MBean Browser is an Oracle Web application (based on JMX MBean containers) that you can use to view information about running, failed and queued execution plans and the status of the DAC Server. See "Viewing DAC Metrics Using Fusion Middleware Control MBean Browser" for more information.

Moving a DAC Environment

Oracle Fusion Middleware provides movement scripts that enable you to move components and configurations from a development or test environment to a production environment. When DAC is installed in an Oracle Fusion Middleware environment and you run the movement scripts, DAC components and configurations are moved along with the other Oracle Fusion Middleware components. The section "Moving a DAC Environment Using Oracle Fusion Middleware Movement Scripts" provides information about this process.

Updates to Revision 3 of This Guide

Revision 3 of *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Data Warehouse Administration Console (11.1.1)* contains the following updates:

- Support for handling failed tasks while an execution plan is still running. See "Using DAC's Hotfix Capability to Handle Failed Tasks While an Execution Plan Is Still Running".
- Support for the parameterization of default values for table columns. See "Managing Data Warehouse Schemas" and "Parameterizing Default Values for Table Columns".
- Enhancements to the DAC patching feature that enable you to have subject areas automatically assembled and execution plans automatically rebuilt when these objects are applied to a repository as patch contents. See "Working With DAC Metadata Patches".
- Enhancements to the concurrent ETL feature that enable you to define an execution plan prefix in order to run concurrent execution plans that share common tasks. See "Execution Plans That Are Eligible to Run Concurrently".
- A new Task by Depth right-click command in the Execution Plans tab of the Execute view, which shows how many tasks in an execution plan are at each depth level. See "Performance Tuning the ETL Process Using Tasks by Depth Command".
- Enhancements to performance tuning by creating logical partitions on load tasks. See "Creating Logical Partitions on Load Tasks to Increase Performance".
- General quality enhancements throughout the guide.

Updates to Revision 4 of This Guide

Revision 4 of *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Data Warehouse Administration Console (11.1.1)* contains the following updates:

- Support for creating multiple instances of an execution plan that can run concurrently. See "Running Multiple Instances of an Execution Plan".
- A new extended property in the Extended Properties subtab of the Physical Data Sources tab in the Setup view called Time Difference Override. This property is used to specify the time difference between the DAC Server machine and the physical data source database machine. See "Physical Data Sources Tab: Extended Properties Subtab".
- General quality enhancements throughout the guide.

System Requirements and Certification

Refer to the system requirements and certification documentation for information about hardware and software requirements, platforms, databases, and other information. Both of these documents are available on Oracle Technology Network (OTN).

The system requirements document covers information such as hardware and software requirements, minimum disk space and memory requirements, and required system libraries, packages, or patches:

http://www.oracle.com/technology/software/products/ias/files/fusion_requirements.htm

The certification document covers supported installation types, platforms, operating systems, databases, JDKs, and third-party products:

http://www.oracle.com/technology/software/products/ias/files/fusion_certification.html

1

About DAC Security

This chapter provides information about the DAC security model, including an explanation about authentication and the recommended DAC security setup.

This chapter contains the following topics:

DAC Security Overview

Oracle Business Intelligence Data Warehouse Administration Console (DAC) comprises the following components:

- **DAC Repository.** Resides on a database and stores the metadata (semantics of the Oracle Business Analytics Warehouse) that represents the data warehouse processes.
- **DAC Client**. A thick client (Swing GUI).
- **DAC Server**. Can be deployed as an enterprise application on the Web Logic Server (referred to as Web mode) or as a standalone Java application (referred to as standalone mode).
- **Non-interactive automation tools**
- **Non-interactive command line tools**

When DAC runs in **Fusion Middleware mode**, users are defined in the WebLogic Server identity store (LDAP) and authenticated against a BI domain. The Fusion Middleware tier authenticates the users for access to the DAC repository. The LDAP credentials indicate: 1) whether the user is valid, and 2) the user's role. The DAC Client also accesses database credentials stored in an encrypted cwallet.sso file in the file system to access the DAC repository database. The database credentials are used to manipulate objects in the repository through direct JDBC access.

When DAC runs in **DAC standalone authentication mode**, the DAC Client authenticates users and gets user permissions against user credentials stored in the DAC repository.

DAC Client Installation Requirements

For production environments, in both Fusion Middleware and DAC standalone authentication deployment modes, the DAC Client has access to highly sensitive password information that allows connectivity to the DAC repository, to all of the data sources accessed by the BI Server (including the transactional data source), and to the data warehouse.

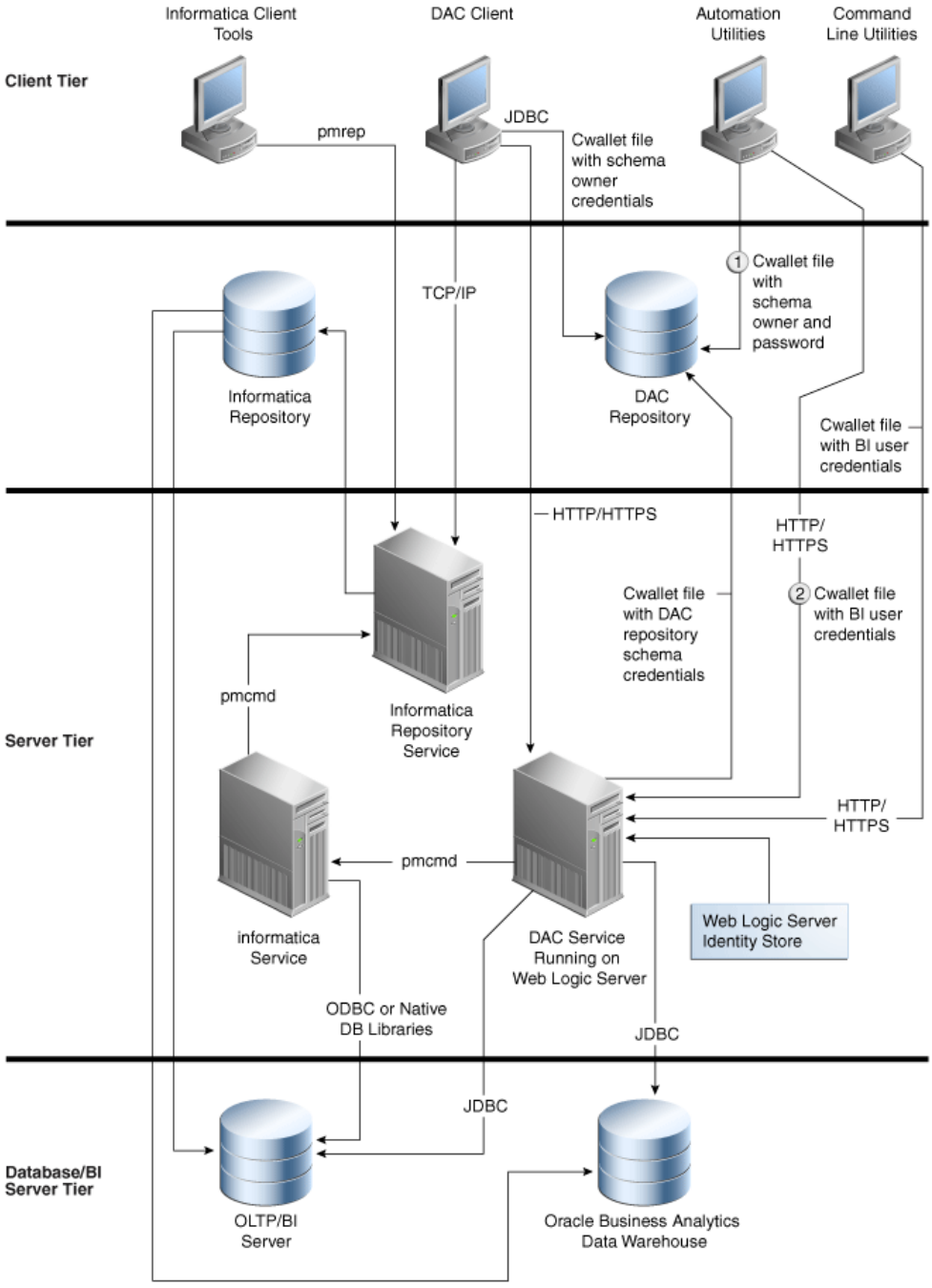
Therefore, for production environments, in both Fusion Middleware and DAC standalone authentication deployment modes, you must install the DAC Client according to the following requirements:

- The DAC Client must be physically located in the server tier with the other middle-tier components.
- The DAC Client should be accessed only by trusted users.
- The DAC Client should be accessible only through remote log in tools if it is accessed outside of the server tier.
- The DAC Client should not be installed on the administrator's desktop.

DAC Authentication in Oracle Fusion Middleware (WebLogic Server) Mode

Figure 1–1 illustrates the process of securing DAC when the DAC Server is running as a service on WebLogic Server.

Figure 1–1 DAC Server Running as Service on WebLogic Server



This process is as follows:

1. DAC Client logs in using FMW authentication:
   a. Gets user name and password from user (can be optionally saved on the file system).
   b. Reads the database connection information from the encrypted cwallet.sso file stored on the file system.

c. Logs into the DAC repository.

d. Reads the DAC Server URL from the DAC repository.

e. Authenticates and gets permissions through the DAC Server in the BI domain using the BI domain URL.

2. DAC Server reads the database connection information from the file system and connects to the DAC repository upon startup.

3. Automation utilities read the database connection information from the file system and connect to the DAC repository.

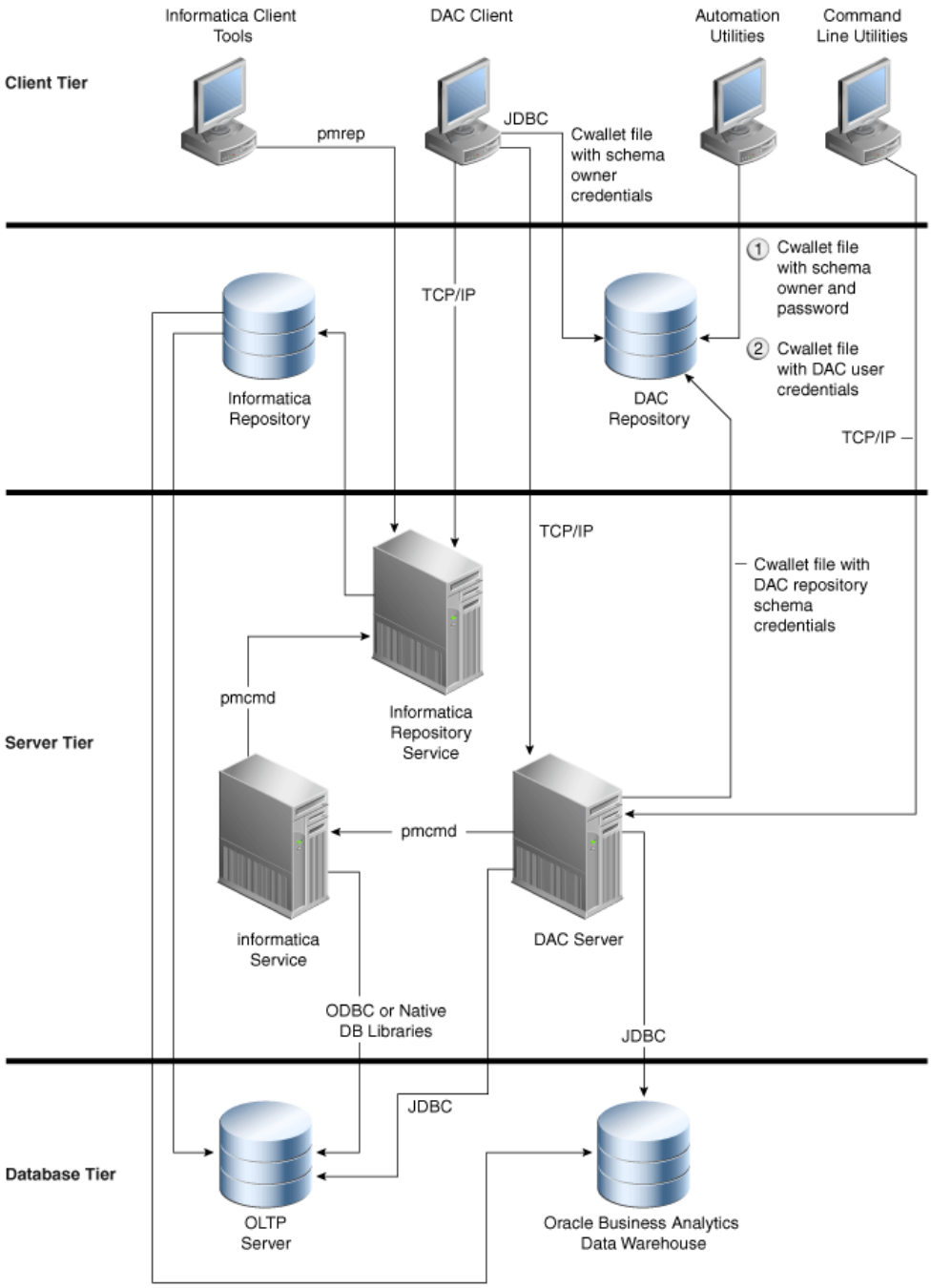
   Note: The automation utilities are not interactive

4. DAC Server command line utilities read the DAC Server information from the file system and send it as a Web service request, which is authenticated with proper user credentials.

DAC Authentication in Standalone Mode

Figure 1–2 illustrates the process of securing DAC when the DAC Server is running as a standalone JVM process.

Figure 1–2 DAC Server Running in Standalone Mode



This process is as follows:

1. DAC Client logs in using DAC authentication:
   a. Gets user name and password from user (can be optionally saved on the file system).
   b. Reads the database connection information from the encrypted cwallet.sso file stored on the file system.

c. Logs into the DAC repository.

d. Authenticates and gets permissions against user credentials stored in the DAC repository.

2. DAC Server reads the database connection information from the file system and connects to the DAC repository upon startup.

3. Automation utilities read the database connection information from the file system and connect to the DAC repository. **Note:** The automation utilities are not interactive.

4. DAC Server command line utilities read the DAC Server information from the file system and send it as a Web service request, which is authenticated with proper user credentials.

Recommended DAC Security Setup

The recommended DAC security setup includes the following points:

- DAC is used for orchestrating ETL processes, and, therefore, should be accessed by a limited number of administrators with the appropriate privileges. The schema level operations that require administrator privileges include but are not limited to the following:
  - Truncating tables
  - Managing indexes
  - Collecting statistics on tables after the data is populated
  - Querying system catalog tables
  - Creating the data warehouse schema

  Because of the sensitive nature of schema level operations, DAC should also be secured by the operating system level security.
- The DAC repository should be stored in a different database from the data warehouse and transactional applications databases. This allows for restriction of DAC users, if necessary.

2

Before You Begin Using DAC

This chapter provides an overview of DAC and explains fundamental concepts that you need to know before you begin using DAC.

This chapter contains the following topics:

DAC Architecture Overview

DAC has a distributed client-server architecture in which the DAC Client issues service requests of the DAC Server. However, because of security considerations, the DAC Client must be physically located in the server tier with the other server-tier components. For more information about DAC Client security, see "DAC Client Installation Requirements".

DAC runs either as an enterprise application on WebLogic Server (Web mode), or as a standalone Java application (standalone mode). For topology diagrams of these two modes, see Figure 1–1 in "DAC Authentication in Oracle Fusion Middleware (WebLogic Server) Mode" and Figure 1–2 in "DAC Authentication in Standalone Mode".

As shown in Figure 1–1 and Figure 1–2:

- The **Client tier** contains the Informatica PowerCenter client tools and the DAC Client.
- The **Server tier** contains the following:
  - **DAC Server.** The DAC Server executes the instructions from the DAC Client. It manages data warehouse processes, including scheduling, loading of the ETL, and configuring the subject areas to be loaded. It dynamically adjusts its actions based on information in the DAC repository. Depending on your business needs, you might incrementally refresh the Oracle Business Analytics

Warehouse once a day, once a week, once a month, or on another similar schedule.

  - **DAC Repository.** The DAC repository stores the metadata (semantics of the Oracle Business Analytics Warehouse) that represents the data warehouse processes.
  - **Informatica PowerCenter Integration Service.** The Integration Service reads workflow information from the repository. The Integration Service connects to the repository through the Repository Service to retrieve metadata from the repository.
  - **Informatica PowerCenter Repository Service.** The Repository Service manages connections to the Informatica repository from client applications. The Repository Service is a separate, multi-threaded process that retrieves, inserts, and updates metadata in the repository database tables.
  - **Informatica Repository.** Stores the metadata related to Informatica workflows.
- The **Database tier** contains the transactional and data warehouse databases.

Introduction to DAC

DAC provides a framework for the entire life cycle of data warehouse implementations, including the setup, configuration, administration, and loading of data warehouses. DAC enables you to create, configure, execute, and monitor modular data warehouse applications in a parallel, high-performing environment.

DAC is a metadata-driven ETL orchestration tool that complements ETL platforms, such as Informatica. It provides *application-specific* capabilities that are not prebuilt into ETL platforms. For example, ETL platforms are not aware of the semantics of the subject areas being populated in the data warehouse nor the method in which they are populated.

DAC provides application capabilities at a layer of abstraction above the ETL execution platform that enable you to do the following:

- **Minimize installation, setup, and configuration time**
  - Create a physical data model in the data warehouse
  - Design subject areas and build execution plans
- **Manage metadata driven dependencies and relationships**
  - Generate custom ETL execution plans
  - Capture deleted records
  - Manage indexes
  - Perform test runs of execution plans
- **Provide reporting and monitoring to isolate bottlenecks**
  - Perform error monitoring and email alerting
  - Perform structured ETL analysis and reporting
- **Utilize performance execution techniques**
  - Automate full and incremental mode optimization rules
  - Set the level of ETL session concurrency

- Load balance across multiple ETL servers
- Restart from point of failure
- Create indexes in parallel
- Run appropriate tasks before query indexes have been created
- Queue execution tasks for performance (see Figure 2–1)

  DAC manages the task execution queue based on metadata driven priorities and scores computed at runtime. This combination allows for flexible and optimized execution. Tasks are dynamically assigned a priority based on their number of dependents, number of sources, and average duration.

Task Execution and Queue Management

Figure 2–1 illustrates how tasks are assigned a position in the execution queue. When a task becomes eligible to run, DAC assesses first its phase priority and then the task execution priority. Next, DAC assigns an order of execution based on the number of task dependents, the number of sources, and the estimated duration of the task.

Figure 2–1 Task Execution Queue

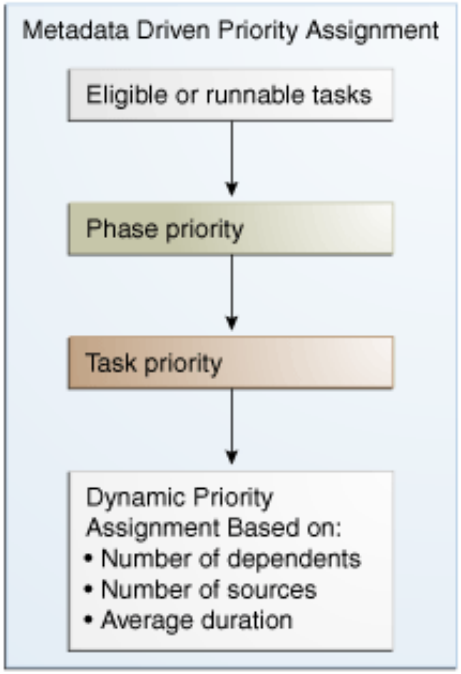


Figure 2–2 and Figure 2–3 show a comparison between poor utilization of resources and optimum utilization of resources. Figure 2–2 illustrates a queue with no task management, in which wait periods can be extensive because resources are not optimized. Figure 2–3 illustrates the strategy used by DAC, in which the queue is managed dynamically based on information gathered about each task, as illustrated in Figure 2–2.

Figure 2–2 Poor Resource Utilization

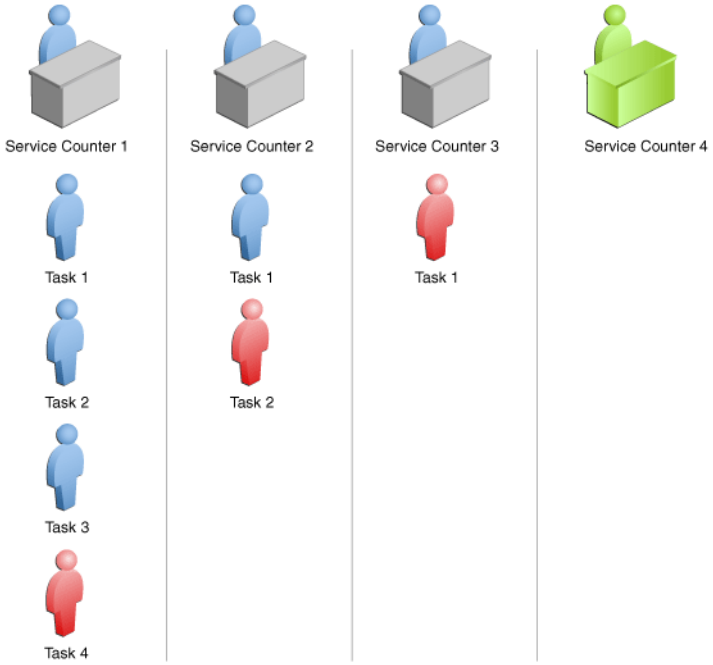
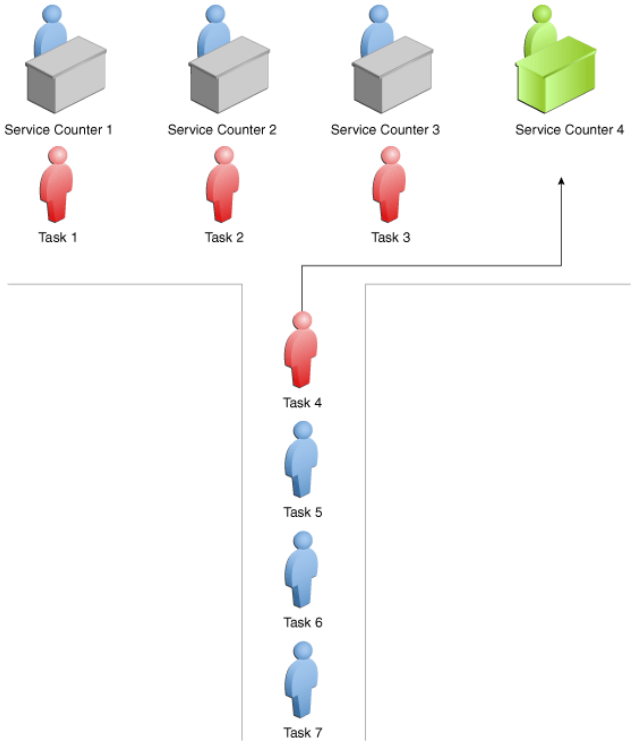


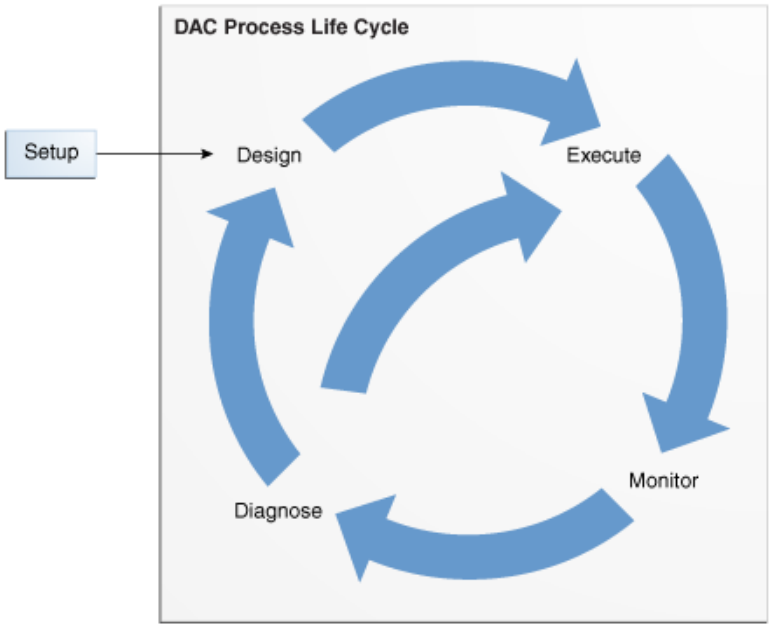
Figure 2–3 DAC's Optimum Resource Utilization



About the DAC Process Life Cycle

DAC is used by different user groups to design, execute, monitor, and diagnose execution plans. These phases together make up the DAC process life cycle, as shown in Figure 2–4.

Figure 2–4 DAC Process Life Cycle



The phases of the process and the actions associated with them are as follows:

- **Setup**
  - Set up database connections
  - Set up ETL processes
  - Set up email recipients
- **Design**
  - Define application objects
  - Design execution plans
- **Execute**
  - Define scheduling parameters to run execution plans
  - Access runtime controls to restart or stop currently running schedules
- **Monitor**
  - Monitor runtime execution of data warehouse applications
  - Monitor users, DAC repository, and application maintenance jobs

About Source System Containers

Source system containers hold repository objects that correspond to a specific source system. You cannot modify objects in the predefined source system containers. You

can make a copy of a predefined container and then modify the metadata to create your own custom source system container. This enables the DAC Client to track customizations you make in the copy of the source system container, such as newly created objects and modified objects. DAC is also able to compare the modified objects with the predefined object definitions in the predefined container. This feature enables DAC to rollback changes to the objects if necessary. For more information, see "About Object Ownership in DAC".

Caution: You cannot modify objects in the predefined source system containers either through the DAC Client or directly through SQL statements to the DAC repository. You must make a copy of a predefined container in order to make any changes to it.

For instructions on creating a new source system container or copying an existing container, see "Creating or Copying a Source System Container".

About DAC Repository Objects

All DAC repository objects are associated with a source system container. The DAC repository stores objects in a hierarchical framework that defines a data warehouse application. DAC enables you to view the repository application objects based on specific source system containers. The source system container holds the metadata that corresponds to the source system with which you are working.

A data warehouse application includes but is not limited to the following repository objects:

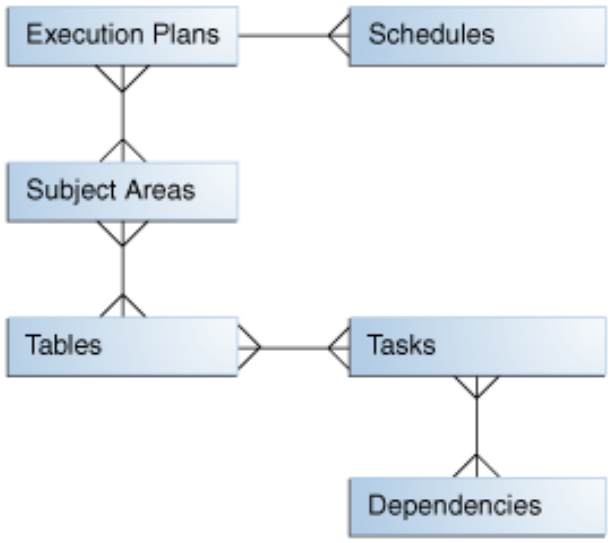
- **Execution plan.** A data transformation plan that is defined on subject areas and is transformed at certain frequencies of time. Execution plans are defined based on business requirements for when the data warehouse needs to be loaded. Execution plans can be scheduled for full or incremental loads.
- **Subject area.** A logical grouping of tables related to a particular subject or application context. A subject area includes the tasks that are associated with the tables, as well as the tasks required to load the tables. Subject areas are assigned to execution plans.
- **Tables.** Physical database tables defined in the database schema. The tables can be transactional database tables or data warehouse tables. Table types include dimension, hierarchy, aggregate, and so on. Flat files can also be used as sources or targets.
- **Tasks.** Units of work for loading tables. Tasks comprise the following: source and target tables, phase, execution type, truncate properties, and commands for full or incremental loads. Tasks can do the following: execute Informatica workflows, execute batch files, call database stored procedures, and execute SQL, XML, and operating system commands. When you assemble a subject area, DAC automatically assigns tasks to it. Tasks that are automatically assigned to the subject area by DAC are indicated by the Autogenerated flag in the Tasks subtab of the Subject Areas tab.

  Task properties are critical in ETL design and execution. DAC automatically assembles tasks into subject areas based on task properties, such as source and target tables. Tasks in the ETL queue are prioritized by the DAC Server, based on task properties, such as phase, source and target connections, and truncate properties.

- **Task group.** A group of tasks that you define because you want to impose a specific order of execution. A task group is considered to be a "special task."
- **Indexes.** Physical database indexes to be defined in the database schema to improve the performance of the ETL processes or the queries for reporting purposes.
- **Schedule.** A schedule specifies when and how often an execution plan runs. An execution plan can be scheduled for different frequencies or for recurrences by defining multiple schedules.

Figure 2–5 illustrates the hierarchy among the DAC repository objects.

Figure 2–5 DAC Repository Object Hierarchy



For best practice tips about the DAC repository objects, see"Considerations When Defining Repository Objects".

About Object Ownership in DAC

The source system container in which an object originates is known as the *owner* container. The tabs in the DAC Design view display the owner of the various repository objects.

You can reuse an object among different source system containers by *referencing* the object. A reference works like a symbolic link or shortcut. You can use the referenced object just as you would an original object; the object's ownership remains unchanged.

For example, W_INVOICE_F is a fact table whose owner is the data warehouse source system container. You can reuse W_INVOICE_F in any other container by referencing it, but the owner is always the data warehouse.

You can reference an object from its owner container, and you can also reference an object that has already been referenced by another source system container.

If you modify a referenced object, the modified object becomes a *clone* and the ownership changes to the source system container in which you performed the modification.

When you make changes to an original object that has been referenced by other containers, any updates to the original object are immediately reflected in the referenced object. If you delete the original object, all referenced objects are also deleted.

Changes to an original object's child objects are not automatically reflected in the referenced object's child objects. You need to *push* the changes to the referenced object's child objects by using the Push to References right-click command. And, conversely, you can import into a referenced object the changes made to an original object. This function is referred to as a *re-reference*.

For a description of the ownership functionality available in the Design view right-click menu, see "Ownership Right-Click Commands".

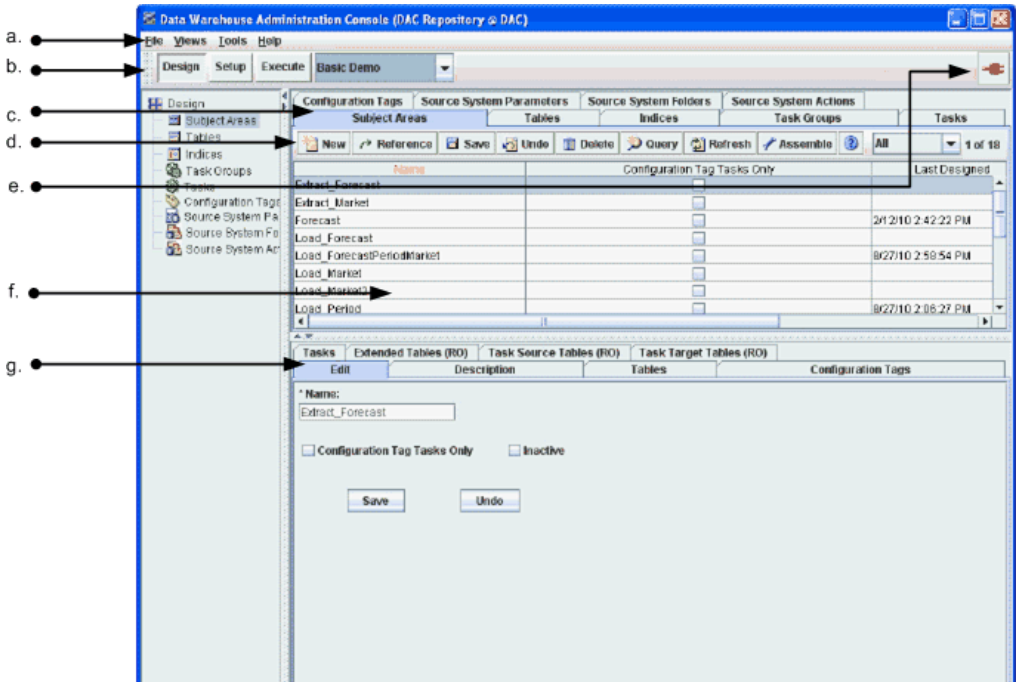
DAC User Interface Overview

This section describes the main elements of the DAC user interface. It contains the following topics:

Main DAC Window

Figure 2–6 shows the main elements of the DAC window.

Figure 2–6 DAC Main Window



Key to figure:

a. File menus

b. Views button and container drop-down list

c. Top pane tabs

d. Top pane toolbar

e. DAC Server monitor icon

f. Editable lists

g. Bottom pane tabs

Menu Bar Commands

The menu bar provides access to the File, Views, Tools, and Help menu commands. For a detailed description of these commands, see the following sections in Chapter 15, "DAC Functional Reference":

- File Menu Commands
- Views Menu Commands
- Tools Menu Commands
- Help Menu Commands

Views Buttons

The View buttons are located directly under the menu bar and provide access to the Design, Setup, and Execute views. For detailed information about the functionality provided in the DAC views, see the following sections in Chapter 15, "DAC Functional Reference":

- Design View Tabs
- Setup View Tabs
- Execute View Tabs

Top Pane Toolbar

For a detailed description of the commands available in the top pane toolbar, see the section titled "Top Pane Toolbar Commands" in Chapter 15, "DAC Functional Reference."

Right-Click Menus

For a detailed description of the commands available in right-click menus, see the section titled "Right-Click Menu Commands" in Chapter 15, "DAC Functional Reference."

DAC Server Monitor Icons

The Server Monitor is located in the upper-right corner of the DAC Client. Its color and shape change based on the DAC Server status. When the DAC Client cannot establish a connection to the DAC Server, the Server Monitor icon resembles a red electrical plug, as shown in Figure 2–7. When Accessibility Mode is enabled, the text "Not connected to DAC Server" is displayed.

When the client is connected to the server and the server is idle, the icon resembles an orange electrical plug in a socket, as shown in Figure 2–8. When Accessibility Mode is enabled, the text "Connected to idle DAC Server" is displayed.

If the client is connected to a server that is running an ETL process, the icon resembles a green electrical plug with a lightning sign superimposed on it, as shown in Figure 2–9. When Accessibility Mode is enabled, the text "Connected to active DAC Server" is displayed. In addition, in standalone mode, clicking on the icon when there is a connection to the server opens a text box that displays data related to the ETL process.

Figure 2–7 DAC Server Down Icon

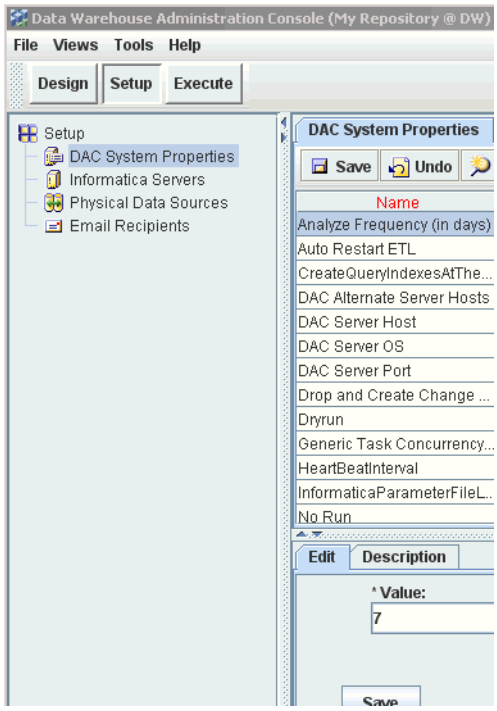
Figure 2–8 DAC Server Idle Icon

Figure 2–9 DAC Server Running Icon

Navigation Tree

The navigation tree appears on the left side of the DAC window, as shown in Figure 2–10. It displays the top-level tabs of the selected view.

Figure 2–10 DAC Navigation Tree



Editable Lists

The top and bottom panes of the DAC window display records in a list format. Some of the columns in the list are editable, and others are read-only. The toolbar at the top of each pane enables you to perform various tasks associated with a selected record in the list. For a description of the toolbar commands, see "Top Pane Toolbar".

A right-click menu is also accessible from the lists in both the top and bottom panes. For a description of these commands, see "Right-Click Menus".

The list format enables you to do the following:

- Edit the data in place and save the record by either clicking another record in the list or clicking the Save button.
- Reorder the columns.
- Sort the data in the list by clicking on the column name.
- Select predefined values from picklists.
- Use the query functionality in pop-up dialogs for fields that refer to values from other entities.
- Use Ctrl+C to copy an editable string to the clipboard (not available for read-only strings).

- Use Ctrl+V to paste a string from the clipboard into a selected cell that supports a string data type.

Font Variations of Objects Displayed in the DAC

The different categories of objects are represented in the DAC with differing fonts. For a description of the object types, see "About Object Ownership in DAC".

Table 2–1 Font Variations Displayed in the DAC

Object Type	Font
Original object or new object in custom container	(System dependent) Black by default, regular style.
Referenced object	Green color, italic style.
Clone	Blue color, regular style.

Using the DAC Query Functionality

Querying is a way to locate one or more records that meet your specified criteria. Query functionality is available in every DAC screen. When you enter query mode, the Edit and Description subtabs in the bottom pane are not available.

This section contains the following topics:

- DAC Query Commands and Operators
- Common DAC Query Procedures
- Common DAC Query Procedures
- Using Flat Views Querying

DAC Query Commands and Operators

Table 2–2 describes the query commands and operators you can use to define your query criteria.

Table 2–2 DAC Query Commands and Operators

Operator	Description
=	Placed before a value, returns records containing a value equal to the query value.
<	Placed before a value, returns records containing a value less than the query value.
>	Placed before a value, returns records containing a value greater than the query value.
<>	Placed before a value, returns records containing a value that is not equal to the query value.
<=	Placed before a value, returns records containing a value less than or equal to the query value.
>=	Placed before a value, returns records containing a value greater than or equal to the query value.
*	Wildcard that can be placed in the middle, or at the beginning or end of a text string.
!	Used for negation.
""	Surrounds a string that, unless modified by a wildcard, must be matched exactly.

Table 2–2 (Cont.) DAC Query Commands and Operators

Operator	Description
\	Escape symbol is used when double quotes should not be processed as a special symbol. For example, `!("*null text" or(\"*\"))` is a value expression for a text field. The query returns values that do not end with a string `null text` and that are not surrounded by double quotes.
()	Surrounds the values and operators that will be processed first.
NULL	Returns records for which the query field is blank.
AND	Placed between values, returns only records for which all the given conditions are true. (Not case sensitive.)
OR	Placed between values, returns records for which at least one condition is true. (Not case sensitive.)

DAC Query Examples

The following examples show different ways you can query on the Name column of the Tasks tab.

- `Extract*` lists all tasks whose name starts with Extract.
- `*Extract*` lists all tasks whose name contains the word `Extract`.
- `!Extract*` lists all tasks whose name does not start with the word `Extract`.
- `!null` lists all tasks whose name is not null.
- `Extract*` or `Aggregate*` lists all tasks whose name starts with `Extract` or `Aggregate`.
- `Load*` and `*Aggregate*` lists all tasks whose name starts with `Load` and also contains the word `Aggregate`.
- `"Extract for Wave Dimension"` or `"Load into Wave Dimension"` lists tasks whose name is either *Extract for Wave Dimension* or *Load into Wave Dimension*.

> **Note:** When using spaces within strings, you need to enclose the string with quotes ("").

Common DAC Query Procedures

This section includes instructions for common query procedures.

To create and execute a query in the DAC

1. In the top or bottom pane of the DAC Client, click **Query** on the toolbar or in the right-click menu.

   A blank row in a list is displayed.
2. Enter the query criteria in the appropriate fields.
3. Click **Run Query** on the toolbar.

   The query is executed and the records appear.

To enter a query value in a date field

1. In the date field, click the **Calendar** icon on the right side of the cell.

   The Date dialog is displayed.

2. Enter the date and time for which you want to search, and select the appropriate query condition.

Using Flat Views Querying

You can use the Flat Views query feature to query for various objects, modify data, and do mass updates. This feature is available in the right-click menu in the Tables, Indices, and Tasks tabs of the Design view. The Flat Views right-click command is context-sensitive and enables you to query only on certain columns.

You can modify individual records in the query results window, or you can use the Update Records right-click command to update multiple records.

To update multiple records using the Flat Views query feature

1. In the Design view, right-click in the Tables, Tasks or Indices tab.
2. Select **Flat Views**, and then select a context-sensitive column on which you want to query.
3. In the query dialog, enter search criteria, and click **Go**.
4. In the query results dialog, right-click and select **Update Records**.
5. In the Update Record Set dialog, select the column you want to update, and then click **Set Value**.
6. Enter a value for the column.
7. To update records that are referenced objects, select **Update Referenced Records**.

   If you select this check box, referenced objects as well as original and cloned objects will be updated. The referenced objects will become clones, and the ownership column for these records will be updated to reflect the new ownership.

   If you do not select this check box, only the columns in records that are original or cloned objects (objects owned by the source system container) will be modified.
8. Click **OK**.
9. Click **Yes** when asked if you want to proceed.

   A message dialog tells you which records were updated.
10. Click **OK** to close the window.

About Query Mode

Some DAC tabs and dialogs open in Query mode, which displays an empty record in which you can enter query criteria. DAC tabs that open in Query mode do not display the list of records contained in the tab. You need to exit Query mode in order to see the list of records. To exit Query mode, click Go in the toolbar.

DAC Accessibility Keyboard Options

Table 2–3 describes the DAC accessibility keyboard options.

Table 2–3 DAC Accessibility Keyboard Options

To Do This	Press
Expand the Views menu to the first level.	ALT+V.

Table 2–3 (Cont.) DAC Accessibility Keyboard Options

To Do This	Press
Move between menu options.	Arrow keys.
In a tabular form, move one cell to the right.	Tab or right arrow key.
Select a menu option.	Enter.
In a tabular form, move one cell to the left.	Shift+Tab or left arrow key.
In a tabular form, move one row up.	Up arrow key.
In a tabular form, move one row down.	Down arrow key.
Activate a cell.	Spacebar.
Activate a cell with a drop-down list.	F2.
Move between options in an open drop-down list.	Arrow keys.
Select an option from a drop-down list.	Enter.
To escape from a cell in edit mode.	Esc.
Add a new value to a cell with a drop-down list.	F2 to enter edit mode, then enter text.
Select a check box.	F2 to activate the check box, spacebar to toggle between selecting and clearing the check box.
To activate a multi-value group (MVG) cell.	Spacebar to activate the cell, F2 to enter edit mode, Ctrl+E to display the dialog (such as a date dialog).
Move to the next focusable object.	Tab
Move to the previous focusable object.	Shift+Tab.
Move between tabs.	Arrow keys.
Move between panels of a dialog.	F6.
Select a button on a dialog.	Alt+the letter underlined on the button.
To display the context menu for a dialog.	Shift+F10.
To activate the menu bar on a dialog.	F10.
To display the default Windows menu in a dialog.	Alt+spacebar.

DAC Installation Scenarios

You can install the DAC Client and Server in various ways, as described below.

- **Oracle Fusion Applications Provisioning**. During the Oracle Fusion Applications installation and provisioning process, the DAC Client and DAC Server software

files are installed in the Oracle Home directory for Oracle Business Intelligence. After you complete the Fusion Applications provisioning process, you must perform additional setup and configuration tasks related to DAC. For more information about the required tasks you must perform, see *Oracle Fusion Middleware Installation and Configuration Guide for Oracle Business Intelligence Applications*.

- **Oracle Business Intelligence Applications Client Installer**. Installs the DAC Client on Windows machines. For instructions on running this installer, see *Oracle Fusion Middleware Installation and Configuration Guide for Oracle Business Intelligence Applications*. For important security considerations when installing the DAC Client, see "DAC Client Installation Requirements".

DAC Installation Directory Paths

Table 2–4 lists the DAC installation directory paths.

Table 2–4 DAC Installation Directory Paths

ORACLE_HOME (read only)	$DOMAIN_HOME/dac	DAC_CONFIG_ LOCATION (shared network location)	Client Config
DACSystem.jar	/*.bat, *.sh	/conf-shared/*.*	/conf-client/login.xml
/lib/*.*	/utilities/*.*	/conf-shared/connection_ template.xml	/conf-client/deleteTrigger s.list
/documentation/*.*	/conf/numeric.precision.o verride.conf	/conf-shared/extralibs.pro perties	/conf-client/ws/
unix_script_bkp/*.*	/conf/sqlgen/*.*	/conf-shared/infa_ command.xml	/conf-client/connections/* .*
version.txt	/export/*.*	/conf-shared/security/ma il	/conf-client/security
	/Informatica/*.*	/conf-shared/security/rep ository	
	/log/*.*	/conf-shared/server.prope rties	
		/conf-shared/task_ restartability.xml	
		/conf-shared/upgrade/*.*	
		/CustomSQLs/*.*	

About the /conf-client/ws Directory

Note the following points about the /conf-client/ws directory:

- The oracle-webservice-client.xml file resides in the dac/conf-client/ws directory. This file declares the Oracle Web Services Manager policies for the DAC Client. When you configure the DAC Client to run over SSL, this policy needs to be replaced by `oracle/wss_username_token_over_ssl_client_policy`.
- `oracle/wsmtom_policy` supports log file transfers.

3

Setup Tasks and Concepts for DAC Administrators

This chapter provides information about the administrative tasks that must be performed in order for a DAC environment to be operable. It also includes various concepts related to DAC operations that a DAC administrator should be aware of.

Note: If you installed and configured DAC using the Oracle BI Applications installation process and by following the post-installation instructions in *Oracle Fusion Middleware Installation and Configuration Guide for Oracle Business Intelligence Applications*, you do not need to repeat the same tasks that appear in this chapter.

This chapter contains the following topics:

About DAC Authentication Modes

When the DAC Client, automation utilities, or command line utilities connect to the DAC Server or DAC repository, they use either Fusion Middleware mode (the option

appears in the DAC Client as "FMW") or DAC standalone authentication mode (the option appears in the DAC Client as "DAC"). See "DAC Authentication in Oracle Fusion Middleware (WebLogic Server) Mode" and "DAC Authentication in Standalone Mode" for topology diagrams that illustrate these two authentication modes.

Fusion Middleware Mode

The Fusion Middleware mode uses the Fusion Middleware security model. This model requires the DAC repository to be configured for Web mode and a valid DAC Server URL to be present in the DAC repository. Authentication occurs through Web service calls to the DAC enterprise application deployed in WebLogic Server. The DAC enterprise application that the URL points to must be running for the authentication to occur.

DAC Standalone Authentication Mode

DAC standalone authentication mode uses the DAC user account security model. DAC user account information is stored in the DAC repository. A DAC user with the Administrator role creates user accounts with the appropriate user names, passwords, and roles. The DAC administrator needs to distribute the user account information (name and password) to the users to enable them to log into the DAC Client and repository. In standalone mode, the DAC Server does not participate in user authentication.

Using the DAC standalone authentication mode, a user can connect to a DAC repository that is configured for either Fusion Middleware or standalone mode. However, when the DAC repository is in Fusion Middleware (Web) mode and the user is authenticated through standalone mode, the user cannot run ETLs, because the user cannot connect to the DAC Server. When standalone mode is used for both the user and DAC repository, connecting to the DAC Server is possible, and, therefore, ETLs can be run.

About the DAC Authentication File

The DAC authentication file is a cwallet.sso file and is used by the DAC Client and automation utilities to connect to the DAC repository. It contains the repository database schema owner and password and a DAC encryption key. An authentication file and a DAC repository have a one-to-one relationship.

The authentication file is automatically generated when the Oracle BI Applications Installer is run or when the BI domain is extended by running the install_dwtools.py installation script. When automatically generated, the authentication file is saved by default in <DAC_Config_Location>\conf-shared\security\repository.

The authentication file can also be created by a DAC user at the time of the first connection to the DAC repository. The first time users connect to a DAC repository, they must either specify an authentication file that was previously created or create a new authentication file. When creating a new authentication file, the user must specify the database schema owner and password.

A DAC administrator needs to distribute the authentication file to users who need access to the repository.

Logging Into DAC for the First Time as an Administrator

When you log into DAC for the first time, you must configure a connection to the DAC repository. DAC stores this connection information for subsequent logins.

The initial login process for an administrator varies depending on the authentication mode being used and whether an authentication file was already created.

If you are using Fusion Middleware authentication, the authentication file was created during the Oracle BI Applications set up and configuration process. You will need to know the location of this file when you log into DAC for the first time.

If you are using DAC standalone authentication, you will create an authentication file while logging into DAC for the first time. To do so, you will need to know the database schema owner and password.

Regardless of the authentication mode, an administrator must distribute the authentication file to all user accounts (those authenticated through DAC as well as through WebLogic Server) that need to access the specified DAC repository. For more information about the authentication file, see "About the DAC Authentication File".

To log in to DAC for the first time

1. Start the DAC Client by navigating to the <Domain_Home>\dac directory and double-clicking the **startclient.bat** file.

   The Login... dialog is displayed.

2. Click **Configure**.
3. In the Configuring... dialog, select **Create Connection**, and then click **Next**.
4. Enter the appropriate connection information:

Field	Required Value
Name	Enter a unique name for the connection to the DAC repository.
Connection type	Select the type of database in which the DAC repository will be stored.

Field	Required Value
Connection String, or **Database name**, or **TNS Name**, or **Service Name**	Select the database name or database account name of the DAC repository. If you are using: ■ Oracle (OCI8), use the tnsnames entry. ■ Oracle (Thin), use the service name. ■ SQL Server, use the database name. ■ DB2-UDB, use the connect string as defined in the DB2 configuration.
Database Host	Enter the name of the machine where the DAC repository will reside.
Database Port	Enter the port number on which the database listens. For example, for an Oracle database the default port is 1521, or for a SQL Server database the default port is 1433.
DB URL (Optional)	Can be used to override the standard URL for this connection.
DB Driver (Optional)	Can be used to override the standard driver for this connection.
Authentication File	Click in this field to do one of the following: ■ Select an existing authentication file. Proceed to step 5 for detailed instructions. ■ Create a new authentication file. Proceed to step 6 for detailed instructions.
SSL Trust Store File	(Optional) For deployments in Web mode, location of the SSL Trust Store file.
SSL JKS Password File	(Optional) For deployments in Web mode, location of the SSL JKS password file.
Log Level	Specifies a client log level for the client session. The logs are saved in <Domain_Home>\dac\log\client <client's logical connection name>.

5. To select an existing authentication file, do the following:
   a. Click in the **Authentication File** field.
   b. In the Authentication File dialog, select **Choose existing authentication file**, and click **OK**.
   c. Navigate to the appropriate directory, and select the appropriate cwallet.sso file. Click **OK**.
   d. In the Configuring... dialog, click **Test Connection** to confirm the connection works.
   e. Click **Apply**, and then click **Finish**.

   Note: A DAC administrator must distribute this authentication file to all user accounts that need to access this DAC repository.

6. To create a new authentication file, do the following:
   a. Click in the **Authentication File** field of the Configuring... dialog.

b. In the Authentication File dialog, select **Create authentication file,** and click **OK**.

c. Navigate to the directory where you want to save the new authentication file, and click **OK**.

d. In the Create Authentication File dialog, enter the **Table Owner Name** and **Password** for the database where the repository will reside.

e. Click **Generate Random Encryption Key** to generate an encryption key for the authentication file. The key is displayed in the Key field.

Alternatively, you can enter a key in the **Key** field. The key must be at least 24 characters long.

f. Click **OK** to close the Create Authentication File dialog.

g. In the Configuring... dialog, click **Test Connection** to confirm the connection works.

Note: You must distribute this authentication file to all user accounts that need to access this DAC repository.

---

7. Click **Apply**, and then click **Finish**

8. To log in using Web mode, do the following:

a. In the Login... dialog, select the appropriate **Connection** from the drop-down list.

b. Enter your **User Name** and **Password**.

This must match the user name and password stored in the WebLogic Server identity store.

c. Select **FMW** as the Authentication Type.

d. If you want DAC to remember the password for this connection, select **Remember Password**.

e. Click **Login**.

The DAC Client is displayed.

9. To log in using DAC standalone mode, do the following:

a. In the Login... dialog, select the appropriate **Connection** from the drop-down list.

b. For a first-time login, you can enter any values in the **User Name** and **Password** fields, because the DAC repository that you just created does not contain any user names or passwords.

A user account will be created with the user name and password you enter in this step. This user account is automatically assigned the Administrator role.

c. Select **DAC** as the Authentication Type.

d. If you want DAC to remember the password for this connection, select **Remember Password**.

e. Click **Login**.

10. (Optional) To specify the DAC view and tab that you want to appear each time you log in, click **Options**.

11. If a DAC repository has not already been created, you will be prompted to create one. Click **Yes**.

    This process creates DAC repository tables.

    Depending on your database type, you may have the option to specify a tablespace.

    The Unicode check box is available for a repository on SQL Server or DB2 databases. Check the Unicode check box if your deployment requires a Unicode schema to be created.

Managing the DAC Server

This section contains the following topics:

Accessing the DAC Server

You use the DAC Server to manage data warehouse ETL processes. The DAC Server can run in standalone mode or Web mode.

- In **standalone DAC mode**, the DAC Server connects to other components in the DAC environment using TCP/IP. When the DAC Server runs in standalone mode, the DAC repository must also be configured as standalone (by selecting Tools, DAC Server Management, Repository Configuration). For a description and topology of standalone mode, see "DAC Authentication in Standalone Mode".
- In **Web mode**, the DAC Server runs as an enterprise application on WebLogic Server. The DAC repository must also be configured as Web mode. For a description and topology of Web mode, see "DAC Authentication in Oracle Fusion Middleware (WebLogic Server) Mode".

You can access the DAC Server using the following methods:

- **DAC Client**. The DAC Client is a command and control interface that enables you to perform schema management tasks, and configure, administer and monitor execution plans. The DAC Client accesses the DAC Server through TCP/IP. You can access the DAC Client through the Start menu or by double-clicking the startclient.bat file in the <Domain_Home>\dac directory.
- **Oracle Enterprise Manager Fusion Middleware Control MBean Browser**. You can view information about running, failed and queued execution plans and the status of the DAC Server. See "Viewing DAC Metrics Using Fusion Middleware Control MBean Browser" for information about accessing DAC through Fusion Middleware Control MBean Browser.
- **Oracle WebLogic Server**. The DAC Server runs as an enterprise application on the WebLogic Server. You can monitor the status of the DAC Server application through the WebLogic Server Administration Console. Only one DAC Server

application can be run against the DAC repository at one time. Therefore, the DAC Server application should be deployed to only one managed server. If more than one DAC Server application is run against the DAC repository, the repository will be corrupted.

See "Monitoring the DAC Server Using WebLogic Server" for information about accessing the DAC Server through the WebLogic Server Administration Console.

- **Command Line.** You can access the DAC Server using the command line in order to start and stop execution plans and to get status information for servers, databases, and execution plans. See "Accessing the DAC Server Using the Command Line" for more information.
- **WSDL File**. You can use a WSDL (Web service definition language) file in order to write service calls to the DAC Server. To access the WSDL file, invoke the following URL:

  ```
  http//<host of the managed WebLogic Server>:<port of the managed WebLogic Server>/DACServer/DACServerManagerService?wsdl
  ```

  where the name of the DAC Server enterprise application is DACServer.

About DAC Server High-Availability

The DAC Server can be configured for high-availability. This process involves scaling out the Oracle BI Applications installation. For more information about scaling out Oracle BI Applications, see *Oracle Fusion Applications Enterprise Deployment Guide*.

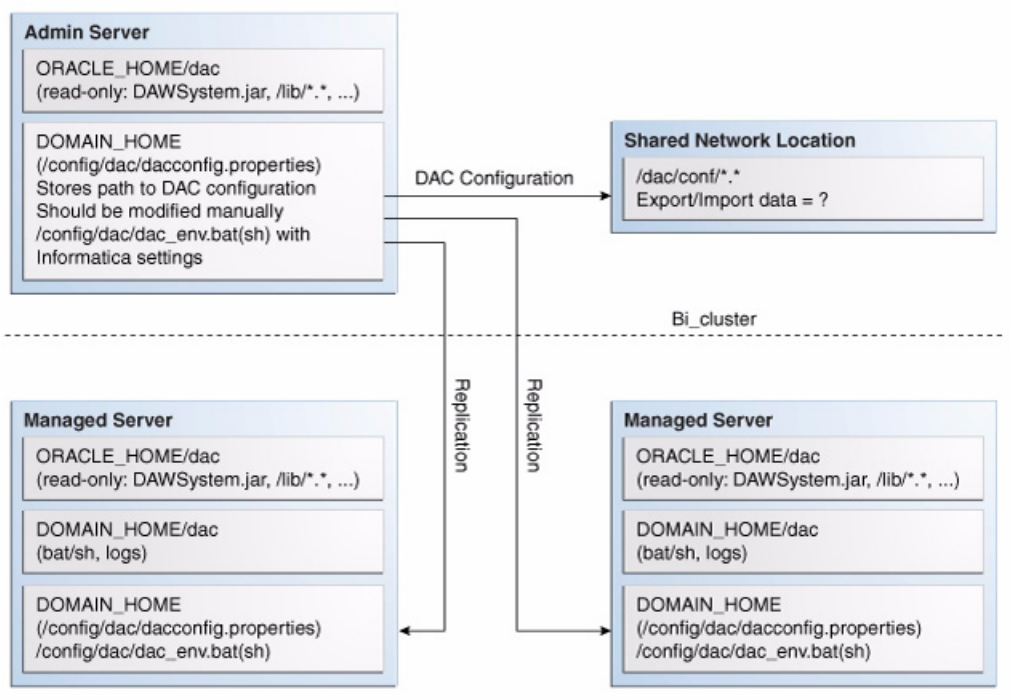
The DAC Server uses an active-passive high-availability configuration. There can be only one active DAC Server and one or more passive servers. Once you have scaled out Oracle BI Applications, the DAC Server failover process occurs automatically. The scale out process uses the server migration approach, which is explained in "DAC Server Migration Approach to Failover". For more information about configuring DAC for high-availability, see the section titled "Configuring Data Warehouse Administration Console for High-Availability," in *Oracle Fusion Applications Enterprise Deployment Guide*.

Note: If DAC was installed using the Simple Install option in the Oracle BI Applications Installer, you will not be able to scale out DAC.

DAC Server Migration Approach to Failover

Figure 3–1 shows the architecture of how DAC can be deployed to achieve high availability and failover capability.

Figure 3–1 DAC Server Migration Approach to Failover



In the BI cluster, when one of the managed servers fails unexpectedly, the second managed server will automatically continue an ETL execution from the point at which it stopped. To enable this operation, the DAC system property "Auto Restart ETL" must be set to True (in the DAC System Properties tab in the Setup view). When the second managed server is automatically started, the execution graph status is reconstructed from the run history tables.

The following scenarios are possible for individual workflows that were started but not completed by the first managed server:

- If the workflow is still running on the Informatica Integration Service grid, DAC will wait for it to complete.
- If the workflow completed, DAC will not restart this workflow.
- If the workflow failed, DAC will restart the workflow.

The second managed server is also capable of authenticating any DAC user defined in a common credential store.

How DAC High Availability Relates to Informatica

The Informatica binaries and domains file need to be on a shared network location that is highly available. DAC will be configured to use them at the time the first managed server is created by the installer.

DAC uses the Informatica pmrep and pmcmd command line programs to communicate with the Informatica Integration Service and the Repository Service.

- **pmcmd.exe** - Used to start and stop workflows and to get workflow status.
- **pmrep.exe** - Used to get the list of sessions used in a workflow.

In addition, Oracle recommends that the DAC Server and Informatica Integration Service are on a shared network drive, because DAC produces parameters files that are consumed by Informatica.

For information about setting up Informatica for high availability, see the Informatica documentation.

Configuring the DAC Repository to Allow DAC Server Connections

In this procedure you specify which DAC Server will be allowed to connect to the DAC repository. Only one DAC Server at a time should connect to a DAC repository. However, you can specify alternative DAC Server hosts that will be allowed to connect to the DAC repository if the main DAC Server host fails. In standalone mode, you must manually start an alternative DAC Server. In Web mode, if high availability is configured, the alternative DAC Server will start automatically.

To configure the DAC repository to allow DAC Server connections

1. In the DAC Client, on the **Tools** menu, select **DAC Server Management**, and then select **Repository Configuration**.
2. In the Repository Configuration dialog, enter the following information:

Field	Description
Mode	Select the DAC Server mode. Possible values are **Web** and **Standalone**.
Web Mode Configuration: URL	If you are using Web mode, enter the URL for the DAC enterprise application that is deployed on WebLogic Server.
Standalone Mode Configuration:	If you are using standalone mode, enter the following information: ■ **Host.** DAC Server host machine ■ **Alternative Hosts.** Additional machines hosting DAC Servers that can be used to connect to the DAC repository if the main DAC Server host machine fails. Enter commas between multiple host names. ■ **Port.** DAC Server port.

3. Click **Save**.

Configuring the Connection Between the DAC Server and DAC Repository

On Windows, you can use the DAC Client to configure the connection between the DAC Server and the DAC repository if the DAC Client and DAC Server are running in the same \dac folder.

Optionally, or to configure a DAC Server installed in another folder or on another Windows machine, use the serverSetupPrompt.bat file to configure the repository connection from the additional machine or folder

On UNIX or Linux, use the serverSetupPrompt.sh script to configure the connection between the DAC Server and the DAC repository.

Note the following points:

- In Web mode, you must first complete the procedure "Configuring the DAC Repository to Allow DAC Server Connections" before you perform the procedure in this section.
- In standalone mode, you must configure the connection between the DAC Server and DAC repository from the location where the DAC Server will be running.
- Only one DAC Server at a time should connect to a DAC repository. It is possible to configure connections between multiple DAC Servers and a single DAC repository, but only one DAC Server at a time should be allowed to connect to the DAC repository.

To use the DAC Client to configure the connection between the DAC Server and the DAC repository

1. Make sure an authentication file (cwallet.sso) has been created and placed in the <DAC_Config_Location>\conf-shared\security\repository directory.
2. In the DAC Client, on the **Tools** menu, select **DAC Server Management**, and then select **DAC Server Setup**.
3. Click **Yes** to display the Server Configuration dialog.
4. In the Repository Connection Information tab, click **Populate from preconfigured client connection**. This will populate the fields with connection details from the DAC Client.
5. Click **Test Connection** to make sure the DAC repository connection works.
6. Click **Save**.

To use the serverSetupPrompt files to configure the connection between the DAC Server and the DAC repository

1. Run the serverSetupPrompt script, as follows:
   - On Windows, double-click the serverSetupPrompt.bat located in the <Domain_Home>\dac directory.
   - On UNIX or Linux, run serverSetupPrompt.sh located in the <Domain_Home>/dac directory.
2. Enter **1** in the 'Please make your selection' prompt to enter repository connection information.
3. Enter the number for the type of database storing the DAC repository from the list of connection type choices.
4. Enter the appropriate connection information for the database.
5. Enter **2** to test the DAC repository connection.
6. Enter **5** to save changes.
7. Enter **6** to exit.

Starting and Stopping the DAC Server (Standalone Mode)

This section explains how to start and stop the DAC Server when the DAC Server is running in standalone mode.

To start or stop the DAC Server in standalone mode

- On Windows, do one of the following:

  - Select **Start**, **Programs**, **Oracle Business Intelligence**, **Oracle DAC**, and then **Start Server** or **Stop Server**.
  - Go to the <Domain_Home>\dac folder and double-click the startserver.bat or stopserver.bat file, depending on which action you want to perform.

- On UNIX or Linux, issue one of the following commands:

  ```
  ./startserver.sh
  ```

  ```
  ./stopserver.sh
  ```

- On AIX, issue one of the following commands:

  ```
  ./startserver_aix.sh
  ```

  ```
  ./stopserver_aix.sh
  ```

  Note: When you execute ./stopserver.csh or ./stopserver.sh, the server will issue a warning about the shutdown request. When the server receives the request, it will shut down even if there is an ETL in progress. The statuses for the ETL run will not be set correctly. The next time the DAC Server starts, it will set the status to Failed for any uncompleted run.

  Tip: When you start the DAC Server, look at the DAC Server status icon in the DAC console to make sure that the DAC Server has started. The DAC Server status icon should either be orange (idle) or green (active). If Accessibility mode is enabled, the icon for an idle DAC Server is replaced with the text "Connected to idle DAC Server," and the icon for an active DAC Server is replaced with the text "Connected to active DAC Server."

  Note: In Web mode, the process for the DAC Server to move from idle status to active can take a number of minutes.

To run the DAC Server in the background

- In bash-related shells, issue the following command:

  ```
  nohup startserver.sh 2>&1 &
  ```

  The nohup command allows the DAC Server to run in the background on UNIX.

  Note: To stop the DAC Server running as a background process, use stopserver.sh or stopserver.csh.

Note: When you execute ./stopserver.csh or ./stopserver.sh, the server will issue a warning about the shutdown request. When the server receives the request, it will shut down even if there is an ETL in progress. The statuses for the ETL run will not be set correctly. The next time the DAC Server starts, it will set the status to Failed for any uncompleted run.

Tip: When you start the DAC Server, look at the DAC Server status icon in the DAC console to make sure that the DAC Server has started. The DAC Server status icon should either be orange (idle) or green (active). If Accessibility mode is enabled, the icon for an idle DAC Server is replaced with the text "Connected to idle DAC Server," and the icon for an active DAC Server is replaced with the text "Connected to active DAC Server."

Note: In Web mode, the process for the DAC Server to move from idle status to active can take a number of minutes.

Starting and Stopping the DAC Server (Web Mode)

When the DAC Server is running in Web mode, you can start and stop it using the following methods:

- In the DAC Client, on the Tools menu, select one of the following:
  - Start DAC Server
  - Restart DAC Server
  - Stop DAC Server
- Using the command line. For instructions, see "Using the Command Line to Access the DAC Server".

Setting Up Email Notifications in the DAC Client and Server

This section provides instructions for setting up users to receive email notifications about ETL statuses, task statuses, and data source usage. To set up email notifications, you need to complete the steps in both of the following procedures:

- Configuring Email Recipients in the DAC Client
- Configuring Email in the DAC Server

For data source usage notifications, you need to complete additional configuration steps, as described in "Setting Up Extract Delays, Event Delays and Data Source Notifications".

Configuring Email Recipients in the DAC Client

Follow this procedure to configure email recipients in the DAC Client.

To configure email recipients in the DAC Client

1. In the Setup view, click the **Email Recipients** tab.
2. Click **New** in the toolbar.
3. In the Edit subtab, enter the following information:

Field	Description
Name	Logical name of the user to be notified.
Email Address	Email address where the notification will be sent.

Field	Description
Notification Level	The notification levels are as follows: ■ 1 -- Notifies recipient of ETL statuses Failed and Aborted. ■ 5 -- Notifies recipient of ETL statuses Failed, Aborted, and Completed; and data source usage. ■ 10 -- Notifies recipient of the same notifications as level 5 and the tasks status Failed.
Inactive	Indicates whether the email notification for this recipient is active or inactive.

Configuring Email in the DAC Server

Follow this procedure to configure the email administrator account in the DAC Server to enable recipients to receive email notifications.

The DAC Server has a built-in login-authentication based email (SMTP) client, which connects to any SMTP login-authenticating server.

Note the following points:

- In Web mode, you must first complete the procedure "Configuring the DAC Repository to Allow DAC Server Connections" before you perform the procedure in this section.
- In standalone mode, you must configure email administrator account from the location where the DAC Server will be running.
- For the email functionality to work, you must be using an SMTP server in which the SMTP authentication mode LOGIN is enabled. For example, if you are using Microsoft Exchange Server, you must enable the SMTP module and enable authentication in the Basic Mode. The SMTP server outbound email authentication must be turned on for the DAC Server to be able to send email notifications.

To configure the email administrator account in the DAC Server

1. In the DAC Client, on the **Tools** menu, select **DAC Server Management**, and then select **DAC Server Setup**.
2. Click **Yes**.

   The Server Configuration dialog appears.
3. Click the **Email Configuration** tab, and enter the email details for the email address to which you want to send DAC Server information notifications.

Field	Description
User Name	User name for the email account.
Password	User password on the email server. (Only required if you select Needs Authentication.)
Email Server	Host name of the email server.
Email Server Port	Port where the email server listens.
Email Address	Email address of the user.
Needs Authentication	Specifies whether the corporate email server requires authentication.
Needs SSL	Specifies whether an SSL connection is required.

4. Click **Send Test Email** to test the connection.
5. Click **Save**.

Setting Up Physical Data Sources

This section provides instructions for specifying the connection pool information for the transactional and data warehouse databases.

To specify transactional and data warehouse data source connections

1. In the Setup view, select the **Physical Data Sources** tab.

   The Physical Data Sources tab displays a precreated record for the data warehouse with the name DataWarehouse, and one or more records for the transactional sources.

2. For each record, enter the following information in the Edit subtab:

Field	Description
Name	Logical name for the data warehouse or transactional database connection pool. You should not change the names of the precreated records. **Note**: When you create an execution plan to perform an ETL, you need to use the data warehouse and transactional data source names that you specify here as the Parameter values for the execution plan parameters DBConnection_OLTP and DBConnection_OLAP.
Type	■ **Source**. Select this option for a transactional (OLTP) database. ■ **Warehouse**. Select this option for a data warehouse (OLAP) database. ■ **Informatica Repository**. Select this option for the Informatica repository database. ■ **DAC Repository**. Select this option for the DAC repository database. ■ **Other**
Connection Type	Database type.
Service Name, **TNS Name**, **Connection String**, or **Database Name** (Note: Field name changes based on the Connection Type selection.)	Enter the value appropriate for your database. For an Oracle TNS Name, enter the TNS name that is specified in the tnsnames.ora file in \network\admin\.
Table Owner	Valid database user.
Password	Password for the database user.

Field	Description
Num Connections per EP	Number of connections the DAC Server will open to this physical data source during the ETL process. **Note:** Typically, DAC needs more connections for data warehouse databases on which it truncates and analyzes tables, and drops and creates indexes. DAC also requires more connections for source systems, such as Siebel, for which it performs change capture operations. In such cases, the value of the **Num Connections per EP** property determines how many change capture processes can run concurrently. If you have a powerful transactional database server and are going to run ETL processes during off-peak times, you can increase the **Num Connections per EP** value to 15 or 20 (10 is the default). If you have a less powerful transactional database server, you should not overload the system with ETL processes. Therefore, you should set the value below 10. Source systems for which DAC does not perform transactions usually do not require as many connections. For data warehouse databases, when the DAC Server is analyzing and truncating tables and dropping and creating indexes, the **Num Connections per EP** value can be higher than the **Num Parallel Workflows per EP** parameter value (set in the Informatica Servers tab) because when DAC creates indexes for a table in parallel, it needs additional relational connections. For example, if you are running 10 workflows, and you want to create two indexes per table in parallel, you will need 20 connections. This is true even though not all 10 tasks will be doing database-related activities (such as dropping and creating indexes, analyzing tables, and so on) at the same time.
Host	Machine name or instance where the database resides.
Port	Port number where the database listens (for example, 1521 is the default for an Oracle database).
Source Priority	Priority of the physical data source connection. Used for multi-source execution plans to resolve conflicts when data is extracted into common data warehouse tables. The DAC dependency algorithm ensures that no more than one task writes to a target table at a time (unless overridden in task groups), and that the reads and writes are staggered. For example, if you have multi-source scenario in which you want to extract data from Oracle EBS and Siebel transactional sources, and you want to extract data for the common data warehouse entities from Oracle EBS first and then from Siebel, the source priority for Oracle EBS should be 1 and Siebel should be 2.
Data Source Number	Unique number assigned to the data source category so that the data can be identified in the data warehouse. If you are editing a data source template for a data source type, Oracle recommends that you do not change the default value. If you are specifying a data source without using a pre-defined template, you must use the correct value for that data source category. This value is passed as a parameter to the Informatica workflows. If you are using multiple sources, each data source has a unique number. Typically, all source dependent extracts will use this parameter to populate the DATASOURCE_NUM_ID column, and the source independent workflows will carry these values to the final dimension and fact tables.

Field	Description
Default Index Space	(Oracle specific) Specifies the table space in which DAC creates indexes on the database. To use the default table space, leave this field blank.
Num Parallel Indexes per Table	Specifies how many indexes are to be created in parallel for each table associated with the physical data source connection. For example, if you give this property a value of 2, then two indexes will be created in parallel per table during the ETL process. **Note:** The number of indexes that can be created in parallel is limited by the value you set in the Num Connections per EP property and the Max Num Workflows per EP property on the Informatica Servers tab. Each task that is creating an index on a table requires a connection. Therefore, if you have two tables and each table has three indexes, and you have set the Num Parallel Indexes per Table property to 3, the Num Connections per EP property should have a value of at least 6 in order for all the indexes to be created in parallel. **Note:** Use caution when setting this property. Setting this number too high as a default for every table that participates in the ETL process can put stress on the database. You may want to consider setting the Number of Parallel Indexes property (in the Parallel Indexes subtab) for individual tables after you identify tasks that are creating bottlenecks or blockages.
JDBC Driver (Optional)	A JDBC driver for the data source connection. The value in this field must conform to the database specifications. Use this option to specify a unique driver for a particular physical data source.
URL (Optional)	A JDBC URL for the data source connection. The value in this field must conform to the database specifications. Use this option to specify a unique URL for a particular data source. For example, this option can be useful if this physical data source uses Oracle RAC and other data sources use a different database type.

3. Click **Test Connection** to make sure the connection works.
4. Click **Save**.

Database Connectivity

You must install and configure the appropriate database connectivity software on the machines that host the Informatica PowerCenter Services, DAC Server, and DAC Client.

- Informatica Integration Service connects to the BI Server using ODBC or native drivers.
- Informatica Repository Service connects to the Informatica repository using native connectivity.
- The DAC Client and Server and the BI Server require JDBC drivers for database connectivity. Only the Oracle JDBC driver is shipped with Oracle BI Applications.
- The DAC Client requires an ODBC connection to the Oracle Business Analytics Warehouse when the database type is DB2, DB2-390, or SQL Server, in order to create or upgrade the data warehouse schema using the DDL Import Utility.

Setting Up Communication Between DAC and Informatica

This section includes information about the requirements needed for DAC and Informatica PowerCenter to communicate.

Note: If you installed and configured DAC using the Oracle BI Applications installation process and by following the post-installation instructions in *Oracle Fusion Middleware Installation and Configuration Guide for Oracle Business Intelligence Applications*, you do not need to repeat the same tasks that appear in this section.

This section contains the following topics:

Introduction to DAC and Informatica Interaction

DAC orchestrates ETL routines written in Informatica and uses the Informatica pmcmd and pmrep command line interface tools to run workflows. At runtime, DAC consolidates all parameters—including runtime, database specific, SQL, and application level—into a parameter file while invoking the workflows. DAC also sets the session log file name, and, at the end of the routine execution, collects valid statistics, such as workflow status, number of row processed, and read/write throughputs.

DAC Session Log File Naming Conventions

DAC provides a log file name for each Informatica session it executes. If a workflow does not run concurrently, and, therefore, does not have multiple instances, DAC uses the following naming convention for the log file:

```
[session_name].[source_database_name].log
```

If a workflow runs concurrently because it is configured to loop (see "Looping of Workflows"), DAC uses the following naming convention for the log file

```
[dac_task_name].S[session_number].[source_database_name].log
```

If a workflow runs concurrently because a DAC_EP_PREFIX execution plan level parameter has been defined (see "Execution Plans That Are Eligible to Run Concurrently"), DAC uses the following naming convention for the log file:

```
[prefix].[session_name].[source_database_name].log
```

The source database name is appended to allow for multi-source ETL processes in which the same task may be run as many times as there are sources, and the log files must be distinguishable.

DAC will shorten the name of the session log file if the naming convention produces files names greater than 100 characters.

If the length of the absolute file name (fully qualified with directory names) is more than 250 characters, Informatica may fail and return a "File name too long" error when it tries to create the workflow/session log files. To help avoid this error, Oracle recommends that the variables $PMSessLogDir and $PMWorkflowDir have values that are short, such as C:\infalogs, rather than the typical value C:\Informatica\PowerCenter<version number>\server\infa_shared\SessLogs.

However, even if you use short values for the variables $PMSessLogDir and $PMWorkflowDir, DAC may still shorten the length of the log file names. To enable DAC to use longer file names (over 100 characters, which is the default), edit the variable informatica.max.sesslogfile.length in the infa-config.properties file, which is located in the <DAC_Config_Location>\conf-shared directory.

Informatica Log File Naming Conventions

The naming conventions for Informatica log files is different depending on whether the DAC_EP_PREFIX execution plan level parameter has been defined. Table 3–1 lists the naming conventions for the Informatica log files.

Table 3–1 Informatica Log File Naming Conventions

Command That Generates Log	Log File Name Without DAC_EP_PREFIX Defined	Log File Name With DAC_EP_PREFIX Defined
Pmcmd: startworkflow getworkflowstatus	WF.log	<PREFIX>_WF.log
Pmcmd: getSessionDetails	WF_DETAIL.log	<PREFIX>_WF_DETAIL.log
Pmrep: getSessionSet	WF_SESSIONS.log	<PREFIX>_WF_SESSIONS.log

Connectivity Requirements

You should be familiar with the following connectivity requirements for DAC and Informatica.

Informatica and DAC Server Connectivity

The DAC Server requires access to the following Informatica command line programs:

- **pmrep**. For communicating with the Informatica Repository Service to query the sessions associated with a workflow.
- **pmcmd**. For communicating with the Informatica Integration Service in order to start and stop the Informatica workflows as well as collect statistics for the workflow.

To ensure DAC has access to pmrep and pmcmd, do one of the following:

- Co-locate the DAC Server and Informatica on the same machine.
- Make the Informatica binary files accessible on a shared network drive.
- Copy the Informatica\PowerCenter<version number>\CMD_Utilities directory to the machine where the DAC Server is installed

Informatica and DAC Client Connectivity

The DAC Client requires access to pmrep in order to synchronize tasks with Informatica workflows and to keep the DAC task source and target tables information up to date. You can either co-locate the Informatica client tools and the DAC Client on the same machine, or you can install the Informatica client tools as a shared network resource.

Procedure DAC Follows When Interacting With Informatica

The procedure DAC follows when interacting with Informatica is as follows.

1. At the beginning of the ETL process, DAC executes the pmrep command "connect" (CONNECT_REP_SERVER section) and the pmcmd command "pingserver" (PING_SERVER section) to the Repository Service entry and also to all the Informatica Integration Services.

   Any non-zero return codes indicate an error occurred when connecting to the Informatica Integration Services, resulting in failure of the ETL process. The outputs are captured in pmrepCommand.log and ping.log in the appropriate log directory for the run.

2. The following steps are followed when a workflow is executed:

   a. The list of sessions associated with the workflow are identified. This is done by invoking the pmrep "listobjectdependencies" command (GET_SESSIONS section in the infa_command.xml). The output of the command is parsed to get the list of sessions. This list is then used for composing a parameter file where the parameter name-value pairs are repeated for each session under the workflow. The output is captured in <WF_NAME>_SESSIONS.log.

   b. The workflow execution is triggered in a polling mode for ENU environments, and with -WAIT mode for the non-ENU environments, using the pmcmd "startworkflow" command (START_WORKFLOW section).

      For ENU environments, the exit code for the pmcmd command to start the workflow indicates whether the request to start the workflow succeeded. The output of this command is captured in <WFNAME>.log. Once the workflow starts to execute, DAC tries to poll the status of the workflow every 30 seconds using pmcmd "getworkflowdetails" command (GET_WORKFLOW_DETAILS section). As long as the status is "Running," DAC waits. If the workflow return code is 0 and the status is "Succeeded," then the workflow is considered to have completed without errors. DAC looks for both workflow return code and also the status to make a decision. The output of the command is also written to WF_NAME.log file.

      For non-ENU environments, the exit code of the pmcmd command to start the workflow indicates whether the workflow completed successfully. Also, some of the strings that DAC looks for in the output files are localized. In such cases, you need to find the equivalent strings from the respective outputs and populate them in the infa_commands.xml file under "Informatica Internationalization" section. This step is essential for DAC to interpret the output provided by the pmcmd and pmrep utilities. For example, DAC interprets a workflow to be successful if the workflow status is listed by the pmcmd command "getworkflowdetails," with the tag "Workflow Run Status," and a value of "Succeeded." These strings in a non-ENU environment could appear in localized strings. For DAC to be able to understand and interpret this, you need to fill in the equivalent localized strings in the body.

c. For each of the sessions under the workflow, DAC also tries to get the session statistics by executing the pmcmd "getSessionStatistics" command (GET_SESSION_STATISTICS section). The output of the command is written to the file <SESSION_NAME>_DETAIL.log.

Defining the Informatica Domains File Path for the DAC Client and DAC Server

In order for the DAC Client and Server to be able to use the pmrep and pmcmd programs, the path of the Informatica Domains file 'domains.infa' must be defined in the dac_env.cmd or dac_env.sh file on both the DAC Client machine and the DAC Server machine.

Configuring Relational Connections in Informatica Workflow Manager

The Informatica Integration Service requires database connection information for the source and target databases. You use Informatica Workflow Manager to configure the database connections. The connections you configure in Informatica map to physical data source definitions in DAC, as specified in the Physical Data Sources tab of the Setup view. DAC also specifies the name of the Informatica database connection in the parameter file that is generated for the workflow when an ETL process is executed.

Note: The name for the connection specified in Informatica must match the name of the physical data source in DAC to which it maps.

To configure relational connections in Informatica Workflow Manager

1. In Informatica PowerCenter Workflow Manager, connect to the repository.
2. Select **Connections**, and then select **Relational**.

   The Relational Connection Browser is displayed.
3. Create a connection for each transactional (OLTP) database, and a connection for the Oracle Business Analytics Warehouse (OLAP) database. For each database connection, do the following:
   a. Click **New** to display the Select Subtype dialog.
   b. Select the appropriate database type, and click **OK**.

      The Connection Object Definition dialog displays options appropriate to the specified database type.
   c. Define the relational connection.

      Name: Enter a name for the database connection exactly as it appears in the Physical Data Sources tab in the DAC Setup View.

      User Name: Database user name with the appropriate read and write database permissions to access the database.

      Password: Password for the user name.

      Connect string: Connect string used to communicate with the database (refer to the Informatica online help for information about specifying this value).

      Code Page: Code page compatible with the code page of the database client. If NLS_LANG (for Oracle database) or DB2CODPAGE (for DB2 database) has been set, then the Code Page value should be a code page compatible with the language set by these variables.
   d. Click **OK** to save the details.

Registering Informatica Services in DAC

This section explains how to register the Informatica Integration Service and the Informatica Repository Service in DAC.

When you register Informatica Services in DAC, note the following:

- You can register multiple Integration Service services, but you must register at least one service.
- You must register one Repository Service. You should only register one Repository Service.

> **Tip:** To locate Informatica services properties that you may need to complete this procedure, log into the Informatica Administrator and select the appropriate service.

To register Informatica Services in DAC

1. In the DAC Setup view, click the **Informatica Servers** tab.
2. For the Informatica Integration Service that you want to register, do the following:
   a. Modify the record with Name = INFORMATICA_DW_SERVER by entering the following information in the Edit subtab:

Field	Description
Name	Enter a logical name for the Integration Service.
Type	Select Informatica.
Service	Enter the name of the Integration Service as specified in the Informatica domain (as seen in Informatica Administrator).
Domain	Enter the Informatica domain name (as seen in Informatica Administrator).
Login	Informatica repository user name who has Administrator privileges for the Informatica repository. **Note:** DAC must log in to Informatica as an Informatica repository Administrator user that is configured in the native security domain.
Password	Password for the user specified in the Login field.
Num Parallel Workflows per EP	Maximum number of workflows that can be executed in parallel on the Informatica Integration Service. If the number of sessions is zero or is not specified, the DAC Server assigns the default value of 10. For more information about setting this property, see "Determining the Num Parallel Workflows per EP Parameter Setting".
Repository Name	Name of the Informatica Repository Service that manages the Informatica repository for Oracle BI Applications.
Inactive	Indicates whether the Integration Service will participate in the ETL process.

   b. Click **Test Connection** to make sure that the connection works.

      Note: The Integration Service must be running.

   c. Click **Save** to save the details.

3. For the Informatica Repository Service you want to register, do the following:

   a. Modify the record with Name = INFORMATICA_REP_SERVER by entering the following information in the Edit subtab:

Field	Description
Name	Enter a logical name for the Repository Service.
Type	Select Repository.
Hostname	Enter the name of the Informatica Gateway Host machine.
Server Port	Enter the Gateway port (as specified during the installation of Informatica PowerCenter Services).
Login	Informatica repository user name who has Administrator privileges for the Informatica repository.
Password	Password for the user specified in the Login field.
Num Parallel Workflows per EP	Maximum number of workflows that can be executed in parallel on the Informatica Integration Service. If the number of sessions is zero or is not specified, the DAC Server assigns the default value of 10. For more information about setting this property, see "Determining the Num Parallel Workflows per EP Parameter Setting".
Repository Name	Name of the Informatica Repository Service that manages the Informatica repository for Oracle BI Applications.
Inactive	Indicates whether the Repository Service will participate in the ETL process.

   b. Click **Test Connection** to make sure that the connection works.

      Note: The Repository Service must be running.

   c. Click **Save** to save the details.

Determining the Num Parallel Workflows per EP Parameter Setting

You set the **Num Parallel Workflows per EP** parameter value when you register the Informatica Integration Service in the Informatica Servers tab of the DAC Client. This parameter specifies the maximum number of workflows that can be executed in parallel on the Integration Service. If the number of sessions is zero or is not specified, the DAC Server assigns the default value of 10.

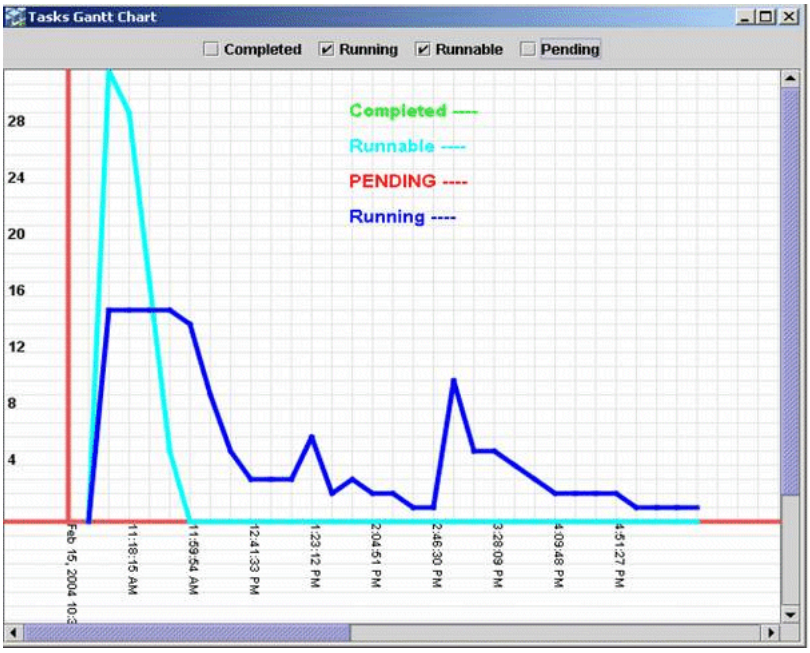
You should consider the following factors when determining the Num Parallel Workflows per EP parameter value:

- How powerful the machine is that hosts the Integration Service.
- How many instances of Integration Services are available.
- The number of Runnable tasks in the queue. A Runnable task is a task for which the Depends On tasks have completed and is ready to be run but is waiting for an Informatica slot to be available. For information about the different task run statuses, see "Current Runs Tab".

  For an optimal run, the runnable queue should be at zero or should reach zero in a short time. For example, Figure 3–2 shows an ideal performance run when 15 sessions were run in parallel. There were many runnable tasks before the process began, but the queue soon reached zero.

You can generate a run analysis such as Figure 3–2 from the right-click menu (select Get Run Information, then select Get Graph) on the DAC Current Runs and Run History tabs. If you find that the runnable curve does not reach zero in a short time, you should increase the Num Parallel Workflows per EP parameter value to make more Informatica slots available.

Figure 3–2 Sample Performance Run



Setting Informatica Integration Service Relaxed Code Page Validation

The Informatica Integration Service must be configured for relaxed code page validation. For more information, refer to the Informatica documentation.

To configure the Informatica Integration Service for relaxed code page validation

1. Log into Informatica Administrator.
2. Select the appropriate Integration Service.
3. Select the **Properties** tab.
4. In the Configuration Properties area, click **Edit**.
5. Deselect the **ValidateDataCodePages** check box.

Setting Informatica Integration Service Custom Properties

Follow this procedure to set custom properties for the Informatica Integration Service.

To set custom properties for the Informatica Integration Service

1. In Informatica Administrator, select the appropriate Integration Service.
2. Click the **Properties** tab.
3. In the Custom Properties area, click **Edit**.
4. Use the **New** option to display the New Custom Property dialog, and add the following custom properties:

Table 3–2 Custom Properties for Informatica Integration Service

Custom Properties Name	Custom Properties Value	Notes
ServerPort	<Server Port Number>	For <Server port number>, enter the number of an available port. For example, 4006. This custom property configures the Integration Service to listen on <Server port number>. DAC communicates with the Integration Service using this port.
overrideMpltVarWithMapVar	Yes	Enables Informatica to evaluate parameters within mapplets.
DisableDB2BulkMode	Yes	Add this custom property and set value to Yes if your Oracle Business Analytics Warehouse is on a DB2/390 or a DB2 UDB database.

5. Click **OK** to save the details.
6. Make sure that the Integration Service and Repository Service that you created during the Informatica PowerCenter installation are running.

Creating the Repository Administrator User in the Native Security Domain

For DAC to be able to access Informatica and perform tasks in pmcmd and pmrep command line programs, DAC must log in to Informatica as an Informatica Repository Administrator user. This user must be configured in the native security domain.

You need to create such a Repository Administrator user, or, if your security policies allow, you can use the default Administrator user (whose privileges include Repository administration) for DAC connectivity to Informatica.

For more information on creating users and Informatica security domains, see the Informatica documentation.

To use the default Administrator user in the native security domain

1. Log in to Informatica Administrator as Administrator.
2. Click the **Configure Security** icon in the top, right corner of the Informatica Administrator work area to display the Security tab.
3. In the Users area, expand the Native directory and select **Administrator**.

   Note: Do not select an existing user from an LDAP domain.
4. Click the **Privileges** tab, and make sure that the correct Domain and Repository Service are selected.
5. Click **OK**.

To create a new Repository Administrator defined in the native security domain

1. Log in to Informatica Administrator as Administrator.
2. In the Users area, click **Actions**, then **Create User** to display the Create User dialog.

   Note: Do not create this user in an LDAP domain.
3. Use the Create User dialog to specify the user details, then click **OK**.

4. In the Users area, select the new user, click the **Privileges** tab, then click **Edit** to display the Edit Roles and Privileges dialog.
5. Use the Roles tab and Privileges tab to select the appropriate Domain and Repository Service.
6. Click **OK**.

Setting the Data Movement Mode

You must set the data movement mode appropriately in the Informatica Integration Service.

To configure the data movement mode

1. Log in to Informatica Administrator.
2. Select the appropriate Integration Service.
3. Select the **Properties** tab.
4. In the General Properties area, click **Edit**.
5. Use the **DataMovementMode** drop down list to specify either ASCII or Unicode, then click **OK** to save the change.

Encrypting Passwords When DAC Interacts With Informatica

DAC sends the Informatica Integration Service and Repository Service passwords un-encrypted when communicating with Informatica through `pmcmd` and `pmrep` commands. This section includes instructions for forcing password encryption.

To force password encryption

In the examples included in the following procedure, the Informatica Integration Service and Informatica Repository Service use the password Administrator.

1. Open a command window and type the following command to create an encrypted Informatica password for `pmcmd` and `pmrep`

   ```
   pmpasswd Administrator -e CRYPT_SYSTEM
   ```

   This step will produce something similar to the following text:

   ```
   Informatica PowerMart Encryption Utility, 32-bit
   Copyright (c) Informatica Corporation 1996-2008
   All Rights Reserved


   Encrypted string
   ->dMGpMvpsuQwXD5UvRmq0OZxhppTWK0Y7fzBtxHL04Gg=<-
   Will decrypt to ->Administrator<-
   ```

2. Create a new environment variable with the following properties.

   Name– `INFA_PASSWORD` (Or any name that you choose.)

   Value– `dMGpMvpsuQwXD5UvRmq0OZxhppTWK0Y7fzBtxHL04Gg=`

   Note: The value should be exactly as shown in the encrypted message in the Command window (the value between --> and <--).

3. Modify the file <DAC_Config_Location>\conf-shared\infa_command.xml by replacing all occurrences of <-p> with <-pv> and <-x> with <-X>.

4. Stop the DAC Server.
5. In DAC, go to the Setup view, and select the **Informatica Servers** tab.
6. Select the appropriate Informatica Service record and enter the name of the environment variable that you created in step 2 of this procedure as the value in the Password field. Save the record.
7. Repeat step 6 for the Informatica Repository Service.
8. Close and re-open the DAC Client to test the connections.
9. If the DAC Server is located on the same machine, start the DAC Server and run an execution plan.
10. Verify that DAC issues the following `pmcmd` command.

```
pmcmd startworkflow -sv BI_DW_Server -d <Domain> -u Administrator -pv **** -f
<folder> -lpf <filename><workflow>
INFORMATICS TASK:<workflow> has finished execution with Completed status.
```

11. If the DAC Server is on a different Windows machine, do the following:
    a. Set the environmental variable on the server machine and modify the <DAC_Config_Location>\conf-shared\infa_command.xml.
    b. Shut down the DAC Server when testing Informatica connectivity or unit testing a DAC task using the DAC Client.
12. If the DAC Server is on a non-Windows machine, do the following:
    a. Set the environmental variable on the server machine and modify the <DAC_Config_Location>\conf-shared\infa_command.xml.
    b. Shut down the DAC Server when testing Informatica connectivity or unit testing a DAC task using the DAC Client.

Integrating DAC and Oracle BI Server

If you are using an Oracle Fusion Applications source system, you need to configure the Oracle BI Server as a physical data source. DAC communicates with the BI Server for the following reasons:

- To adjust refresh dates for data sources

  DAC adjusts the ETL start time for each data source. This information is retrieved using one of the BI Server connection pools that access the transactional database. For information about how DAC handles refresh dates, see "About Refresh Dates and DAC's Incremental Load Strategy".

- To allow you to use the DAC Actions feature to execute any SQL statement

  The Actions feature allows you to do the following:

  – Write any populate or aggregate persistence command
  – Write any executable command that could be executed using nqcmd
  – Allow for replacing the Informatica mapping with a populate command on the BI Server or design a population strategy for populating any table using the BI Server syntax

  For more information about the DAC Actions feature, see "Using Actions to Optimize Indexes and Collect Statistics on Tables".

To configure the BI Server as a physical data source

1. In the **Setup** view, select the **Physical Data Sources** tab.
2. Click **New** in the toolbar.
3. In the Edit subtab, enter the appropriate information for the BI Server. For a description of the fields, see "Setting Up Physical Data Sources".
4. Click **Test Connection** to make sure the connection works.
5. Click **Save**.
6. Specify the BI Server Connection Pool name:
   a. Click the **Extended Properties** subtab.
   b. Click **New** in the bottom pane toolbar.
   c. Select **BIPool** from the Name drop-down list.
   d. Double-click in the **Value** field to open the **Property Value** dialog.
   e. Enter the connection pool name.
   f. Enter the database type of the transactional data source.
   g. Click **Save** in the bottom pane toolbar.

   Note: For Oracle Fusion Applications source systems, after performing this step, you might receive an error in the log files for the run stating the connection pool does not have privileges to run a script for obtaining the timestamp of the data source database. You can ignore this error. However, if physical data source and DAC Server are in different time zones, make sure you define the Time Difference Override property as described in the following step.

7. If the DAC Server machine and the transactional system's database are in different time zones, set the Time Difference Override extended property to specify the time difference between the DAC Server machine and the transactional system's database:
   a. In the Extended Properties subtab, click **New** in the bottom pane toolbar.
   b. Select **TimeDifferenceOverride** from the Name drop-down list.
   c. Click in the Value field to open the Property Value dialog.
   d. Enter the time difference, in minutes, between the DAC Server machine and the transactional system's database.

      This value can be a positive or negative number. It needs to specify the minutes that must be added to or subtracted from the DAC Server time to match the physical data source database time. For example, if the DAC Server is in Pacific Standard Time and the physical data source database is in Eastern Standard Time, the value should be 180. And, conversely, if the DAC Server is in Eastern Standard Time and the physical data source database is in Pacific Standard Time, the value should be -180.

      This value is static and is not automatically updated for Daylight Savings Time differences throughout the year. You will need to update this value for any such time changes.

For more information about the Time Difference Override extended property, see "Physical Data Sources Tab: Extended Properties Subtab".

Setting Up DAC System Properties

You set DAC system properties in the DAC System Properties tab in the Setup view. The Description subtab provides a description of each property. Also see "DAC System Properties Tab" for more information about system properties.

Creating or Upgrading the Data Warehouse Schema

DAC provides functionality for creating and upgrading data warehouse schemas. For information about this functionality, see Chapter 9, "Managing Data Warehouse Schemas."

Distributing DAC Metadata

Typically, you may have multiple environments, such as development, test, QA, production, and so on. The DAC Import/Export feature enables you to make changes in one environment, then export the whole environment as XML files, and then import the XML files into another environment.

For instructions on exporting DAC metadata, see "Exporting DAC Metadata".

For instructions on importing DAC metadata, see "Importing DAC Metadata".

Exporting DAC Metadata

The DAC's Import/Export feature enables you to export source system-specific DAC metadata out of the DAC repository. You can use this feature to migrate DAC metadata from one environment to another, such as from a development environment to test or production environments.

Caution: When you export metadata, all files in the target folder are deleted.

DAC behavior relating to the target folder is as follows:

- If the target folder is empty, DAC exports the metadata without a warning.
- If the target folder contains DAC metadata, DAC issues a warning and you must click OK to proceed with the export. The export process replaces all content in the target folder.
- If the target folder has non-DAC metadata as well as DAC metadata, DAC issues a warning, and you must click OK to proceed with the export. The export process replaces all content in the folder. All non-DAC metadata is deleted.
- If the target folder has only non-DAC metadata, DAC cannot export into the specified target folder.

To export DAC metadata

1. In the DAC Client, on the **Tools** menu, select **DAC Repository Management**, and then select **Export**.

2. Select the directory to which you want to export DAC metadata, or accept the default directory.
3. Select the source system containers that hold the metadata you want to export.
4. Select the appropriate categories of metadata you want to export:
   - **Logical.** Exports all information contained in the Design view and metadata defined in the Seed Data menu.
   - **System.** Exports all information contained in the Setup view, except passwords for servers and database connections.
   - **Run Time.** Exports information about ETL runs and schedules (contained in the Execute view).
   - **User Data.** (Applicable to DAC standalone authentication only) Exports the users, roles, and passwords. **Note:** When importing roles and passwords, if the encryption key is different in the repository to which you are importing from that in which the metadata was exported, the roles and passwords will be unreadable.
5. Click **OK**.
6. Verify the export process by reviewing the log file <Domain_ Home>\dac\log\export.log.

Importing DAC Metadata

The DAC's Import/Export feature enables you to import source system-specific DAC metadata into the DAC repository. You can also use this feature to migrate DAC metadata from one environment to another, such as from a development environment to test or production environments.

To import DAC metadata

1. In the DAC Client, on the **Tools** menu, select **DAC Repository Management**, and then select **Import**.
2. Select the directory from which you want to import DAC metadata, or accept the default directory.
3. Select the appropriate applications for which you want to import metadata.
4. Select the appropriate categories of metadata you want to import:
   - **Logical.** Exports all information contained in the Design view and metadata defined in the Seed Data menu.
   - **System.** Exports all information contained in the Setup view, except passwords for servers and database connections.
   - **Run Time.** Exports information about ETL runs and schedules (contained in the Execute view).
   - **User Data.** (Applicable to DAC standalone authentication only) Exports the users, roles, and passwords. **Note:** When importing roles and passwords, if the encryption key is different in the repository to which you are importing from that in which the metadata was exported, the roles and passwords will be unreadable.
5. If you are importing metadata into a blank repository or to completely replace selected categories of the current metadata in the repository, select **Truncate**

Repository Tables. This option overwrites the content in the current repository. It also greatly increases the speed of the import process.

6. If you want DAC to import new records and update existing records, select **Update Existing Records**. If you do not select this check box, DAC will only insert new records. This option is only available if you do not select the Truncate Repository Tables option.

   Note: When importing DAC metadata into a container, if you select the Update Existing Records option (and not the Truncate Repository Tables option), you also need to import the metadata into any child containers. Otherwise, the child containers will be inconsistent with the parent container.

7. (Optional) Select **Enable bulk mode** to insert the imported metadata into the repository as an array insert. You should elect this option only if you also selected the Truncate Repository Tables option.

   This action increases the speed of the import process.

8. Click **OK**.

9. Verify the import process by reviewing the log file <Domain_Home>\dac\log\import.log.

Applying Patches

The DAC metadata patch feature enables you to do the following:

- Update the predefined ("out-of-the-box") Oracle BI Applications metadata provided by Oracle. In this process, DAC works in conjunction with OPatch, a utility supplied by Oracle.
- Move subsets of DAC metadata from one environment to another.

For more information about the DAC metadata patch feature, see Chapter 11, "Working With DAC Metadata Patches."

Integrating DAC With ETL Tools Other Than Informatica

The DAC external executor framework provides interfaces that enable you to integrate it with ETL tools other than Informatica. The interfaces are described in Javadocs, which are located in the <Oracle_Home>\dac\documentation\External_Executors\Javadocs directory.

For more information about the DAC external executor framework, see Chapter 13, "Integrating DAC With Other ETL Tools."

Managing DAC User Accounts

This section contains the following topics:

How DAC Permissions Map to Oracle Business Intelligence Applications Roles

Oracle Business Intelligence Applications includes preconfigured security roles that map to the preconfigured DAC permissions. Table 3–3 shows the mapping between the Oracle BI Applications role and the DAC permissions.

For more information about Oracle BI Applications roles, see the chapter titled "Oracle BI Applications Security" in Oracle Fusion Middleware Reference Guide for Oracle Business Intelligence Applications.

Table 3–3 Oracle BI Applications Roles Mapped to DAC Permissions

Oracle BI Applications Role	DAC Permission
BI Administrator	Administrator
BI Author	Developer, Operator
BI Consumer	Guest
BIA_ADMINISTRATOR_DUTY	Administrator
BIA_FUNCTIONAL_DEVELOPER_DUTY	Developer
BIA_IMPLRMENTATION_MANAGER_DUTY	Operator

As shown in Table 3–4, each DAC permission has a set of privileges that determines what DAC functionality the permission can access.

Table 3–4 User Account Permissions and Privileges

Permission	Privileges
Administrator	Read and write privileges on all DAC tabs and dialogs, including all administrative tasks.
Developer	Read and write privileges on the following: ■ All Design view tabs ■ All Execute view tabs ■ Export dialog ■ New Source System Container dialog ■ Rename Source System Container dialog ■ Delete Source System Container dialog ■ Purge Run Details ■ All functionality in the Seed Data menu
Operator	Read and write privileges on all Execute view tabs
Guest	Read privileges in all views.

Creating, Deleting, Inactivating User Accounts

In DAC standalone authentication mode, the User Management dialog enables a user with the Administrator permission to manage user accounts. A user account includes a unique identifier, password, and one or more permissions. The Administrator account is created during the initial login to a new DAC installation. See "Logging Into DAC for the First Time as an Administrator" for more information about creating the Administrator account.

After a user account is created, the Administrator must distribute the DAC repository authentication file to users in order for the users to be able to log into the DAC Client (and access the DAC repository) using their own user name and password.

To create a user account

1. In the DAC Client, on the **File** menu, select **DAC User Management**.
2. In the User Management dialog, click **New**.
3. In the new record field, do the following:
   a. Enter a unique name and password.
   b. Click in the **Roles** field, and then select the roles you want to associate with this user account.
4. Click **Save**.
5. Click **Close** to exit the DAC User Management dialog.
6. Distribute the authentication file for the database where the DAC repository resides to the user.

To delete a user account

1. On the **File** menu, select **DAC User Management**.
2. In the **DAC User Management** dialog, select the user account you want to delete.
3. Click **Delete**.
4. Click **Close** to exit the DAC User Management dialog.

To inactivate a user account

1. On the elect **File**, and then **DAC User Management**.
2. In the **DAC User Management** dialog, select the user account you want to inactivate.
3. Click the **Inactive** check box.
4. Click **Save**.
5. Click **Close** to exit the DAC User Management dialog.

Importing DAC User Account Information From a Text File

You can create multiple DAC user accounts by importing the user account information from a text file.

To import DAC user account information from a text file

1. Create and save a text file containing user account information in one of the following formats:
   - <user name>

     The default role is Developer, and the default password is the same as the user name.
   - <user name>, <password>

     The default role is Developer.
   - <user name>, <password>, <role 1>[,<role 2>][...]

   You can use different formats on different lines in the text file.

2. In the **File** menu, select **DAC User Management**.
3. Right-click in the DAC User Management dialog, and select **Read Users From File**.
4. Navigate to the text file, select it, and then click **OK**.

   The new user account information will appear in the DAC User Management dialog.

Changing the DAC Repository Encryption Key

When you change the DAC repository encryption key, the following changes occur:

- All encrypted information is re-encrypted in the DAC repository.
- The DAC repository is restamped with the new encryption key.
- The authentication file (cwallet.sso) file is updated with the new key.

To change the DAC encryption key

1. In the DAC Client, on the **Tools** menu, select **DAC Repository Management**, and then select **Change Encryption Key**.
2. Click **Generate Random Encryption Key** to generate an encryption key. The key is displayed in the Key field. Alternatively, you can enter a key in the Key field. The key must be at least 24 characters long.
3. Click **OK**.
4. Distribute the updated authentication file (cwallet.sso) to all users that connect to this repository.

Modifying an Existing Authentication File

A DAC administrator can change the DAC repository schema owner and password in an existing authentication file.

To modify an existing authentication file

1. Start the DAC Client by navigating to the <Domain_Home>\dac directory and double-clicking the **startclient.bat** file.
2. In the Login... dialog, click **Configure**.
3. In the Configuring... dialog, select **Modify Connection**, and then click **Next**.
4. Click in the Authentication File field to open the Authentication File dialog.
5. From the drop-down list, select **Modify existing authentication file**, and then click **OK**.
6. Select the authentication file, and then click **OK**.
7. Enter a new table owner name and password, and then click **OK.**
8. Click **Apply**, and then click **Finish**.
9. Distribute the updated authentication file (cwallet.sso) to all user accounts that connect to this repository.

Moving a DAC Environment Using Oracle Fusion Middleware Movement Scripts

Oracle Fusion Middleware provides movement scripts that enable you to move components and configurations from a development or test environment to a production environment.

When DAC is installed in an Oracle Fusion Middleware environment and you run the movement scripts, DAC components and configurations are moved along with the other Oracle Fusion Middleware components.

Detailed information about moving an Oracle Fusion Middleware environment is provided in the *Oracle Fusion Middleware Administrator's Guide*. You should thoroughly review and follow the instructions in the *Oracle Fusion Middleware Administrator's Guide* when moving an Oracle Fusion Middleware environment (including DAC).

This section describes only the movement of the DAC components and configurations when you use the movement scripts provided by Oracle Fusion Middleware.

This section contains the following topics:

Preconditions for Moving a DAC Environment

The following preconditions must be met before a DAC environment is moved:

- The source system is installed and properly configured.
- The source and target systems have equivalent hardware. There will be a one-to-one mapping from each host in the source system to an equivalent host in the target system.
- The operating system for the source and target systems are the same.
- Informatica is installed on both the source and target systems.

  For instructions on installing Informatica, use the Informatica PowerCenter installation documentation in conjunction with information about Oracle BI Applications requirements that appears in the section titled "Setting Up Oracle BI Applications," in the *Oracle Fusion Middleware Installation and Configuration Guide for Oracle Business Intelligence Applications*.
- The Informatica Repository has been restored on the target system.

  For instructions on restoring an Informatica Repository, see the section titled, "Restoring the Prebuilt Informatica Repository, in the *Oracle Fusion Middleware Installation and Configuration Guide for Oracle Business Intelligence Applications*.
- The Informatica Repository has been configured on the target system.

  For information about the tasks required to configure the Informatica Repository, see "Setting Up Communication Between DAC and Informatica".

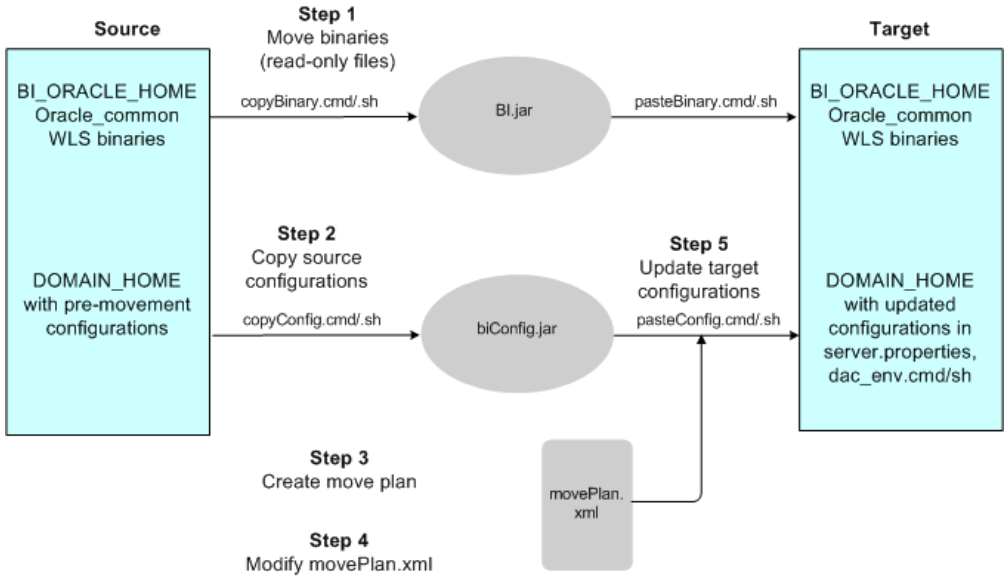
Process for Moving DAC Components

This section describes the process for how DAC components are moved when the Oracle Fusion Middleware movement scripts are used to move an Oracle Fusion Middleware environment.

Detailed instructions for moving an Oracle Fusion Middleware environment are provided in the *Oracle Fusion Middleware Administrator's Guide*.

Figure 3–3 illustrates the sequence of steps of the DAC movement process. These steps are further described below. The movement scripts described in this section are located in the ORACLE_COMMON_HOME/bin directory.

Figure 3–3 Flow for Moving DAC Components



1. **Move the binary files.** In this step, you create an archive of the source Middleware home and use the archive to create a copy of the Middleware home. You first execute copyBinary.cmd/.sh file on the source system, which creates the BI.jar archive. Then, you execute pasteBinary.cmd/sh and specify the target destination.
2. **Copy the source configurations.** In this step, you execute copyConfig.cmd/.sh, which creates an archive of the source configurations. This archive is named biConfig.jar.
3. **Create a move plan.** In this step, you execute extractMovePlan.cmd/.sh, which extracts from biConfig.jar the configuration settings of the source environment. Table 3–5 lists the DAC-related configuration information that is contained in the movePlan.xml file.
4. **Modify the move plan.** In this step, you manually edit the move plan to specify properties for the target environment.
5. **Update the target environment with the settings specified in the move plan.** In this step, you execute pasteConfig.cmd/.sh. This step also drops and creates the data warehouse schema and updates the DAC server.properties file and dac_env.cmd/.sh.

Editing DAC Properties in moveplan.xml

Table 3–5 describes the move plan properties related to DAC that appear in the moveplan.xml file.

Note: You should only modify properties in the moveplan.xml file that have a READ_WRITE scope. Do not modify properties that have a READ_ONLY scope.

Table 3–5 DAC-Related Properties in moveplan.xml

Property	Description	Example
DAC Server Configuration	**The following properties appear under the DAC-Server-Configuration config group.**	
jdbc.url	URL to connect to the DAC repository.	jdbc:oracle:thin:@example.us.oracle.com:1521/orcl.us.oracle.com
jdbc.driver	Name of the JDBC driver.	oracle.jdbc.driver.OracleDriver
Username	User name used to connect to the DAC repository.	
Password File	Absolute path of a secure file containing the password for the user to connect to the DAC repository. You must provide a password file, even if you are not changing the configuration.	/scratch/biplatform/cloning/password_.txt
Email Configuration	**The following properties are in the Email-Configuration config group.**	
email_host	Host name of the email server.	examplehost
email_protocol	Protocol for the email server.	smtp
email_address	Email address of the user.	example@example.com
needs_authentication	The flag indicating whether the corporate email server requires authentication. Valid values are `true` or `false`.	
needs_ssl	The flag indicating whether an SSL connection is required. Valid values are `true` or `false`.	
email_host_port	Port where the email server listens.	
email_user	User name for the email account.	
email_password	Absolute path of a secure file containing the password for the user of the email server. (Only required if the needs_ authentication property is set to `true`.	/scratch/biplatform/cloning/password_.txt
Data Warehouse Configuration	**The following properties are in the Datawarehouse-Configuration config group**.	

Table 3–5 (Cont.) DAC-Related Properties in moveplan.xml

Property	Description	Example
jdbc.url	URL to connect to the data warehouse.	jdbc:oracle:thin:@example.com:1521/example.com
jdbc.driver	Name of the JDBC driver.	oracle.jdbc.driver.OracleDriver
Username	User name used to connect to the data warehouse.	
Password File	Absolute path of a secure file containing the password for the user to connect to the data warehouse. You must provide a password file, even if you are not changing the configuration.	/scratch/biplatform/cloning/password_DW.txt
Informatica Configuration	**The following properties are in the Informatica-Configuration config group**.	
Informatica server home	Informatica server home.	scratch/infahome/
Domains infa file location	Location of the domains.infa file.	/scratch/infahome/domains.infa
InformaticaParameter FileLocation	Directory where the Informatica parameter files are stored (or `DEFAULT`). The default location is dac\Informatica\parameters.	DEFAULT
Data Source Configuration	**The following properties are in the Datasources-Connection-Details config group.**	
Type	The physical data source type. Possible values are: `Source`, `Warehouse`, `Informatica Repository`, `DAC Repository`, and `Other`.	
Connection Type	The type of database connection. Possible values are `BI Server`, `Oracle (OCI8)`, `Oracle (Thin)`, `DB2`, `DB2-390`, `MSSQL`, `Teradata`, `Flat File`.	N/A
Connection String	The data source connection string. If you are using: ■ Oracle (OCI8): Use the tnsnames entry. ■ Oracle (Thin): Use the instance name. ■ SQL Server: Use the database name. ■ DB2-UDB/DB2-390: Use the connect string as defined in the DB2 configuration. ■ Teradata: Use the database name.	orcl.example.com
Table Owner	Name of the table owner.	DB_User

Table 3–5 (Cont.) DAC-Related Properties in moveplan.xml

Property	Description	Example
Host	Host name of the server where the database resides.	example.com
Port	Port where the database receives requests.	1521
JDBC Driver (Optional)	JDBC driver for the data source connection. The value in this field must conform to the database specifications.	oracle.jdbc.driver.OracleDriver
URL (Optional)	The JDBC URL for the data source connection. The value in this field must conform to the database specifications.	jdbc:oracle:thin:@example.com:1521/orcl.example.com
Password File		/scratch/biplatform/cloning/password_ds.txt
Connection Pool Name (BIPool)	Connection pool name.	FSCM_OLTP."Connection Pool"
Database Type (BIPool)	Database type of the transactional data source.	Oracle
External Executors Configuration	**The following properties are in the External_Executors config group.**	N/A
Execution type	Execution type for the tasks that will be executed by the external executor.	N/A
Name	Name of the property that must be configured to integrate DAC with other ETL tools. There are multiple properties for the external executors. `Name` is the name of the property. `Value` is the value that defines the property.	`<name>ODIUser</name>` `<value>TestUser</value>`

Post Movement Steps

The steps below should be followed after moving a DAC environment:

- Review the log file specified by the argument -ldl in pasteConfig.cmd/.sh. The log file should indicate that the operation completed successfully, such as "CLONE-21007 Cloning operation completed successfully." If the log file does not indicate the operation completed successfully, you should execute the pasteConfig.cmd/.sh script again and enter the correct parameters.
- Start the DAC Client on the target system, and create a new DAC administrator account. For instructions, see "Logging Into DAC for the First Time as an Administrator".
- Check the DAC Server configuration and test the connection between the DAC Server and DAC repository. For instructions, see "Configuring the Connection Between the DAC Server and DAC Repository"
- Test the connection to the physical data sources. For instructions, see "Setting Up Physical Data Sources".
- Test the connection to the Informatica Integration Service. For instructions, see "Registering Informatica Services in DAC".

Note: If you log into the DAC Client and receive an error message stating, "Can't authenticate user" and if the log file states, "INFO: Authentication failed. Cause: Couldn't connect to DAC application," this problem may be caused by an incorrect migration of the DAC Server URL. To correct this issue, you can execute the following SQL statement against the DAC repository:

```
UPDATE W_ETL_REPOS SET VALUE =
'http://example.us.oracle.com:10217/DACServer'
WHERE NAME = 'DACServerURL';
COMMIT;
```

Alternatively, if the schema owner and password are unknown but you have the authentication file for the DAC repository, you can manually update the DAC Server URL by logging into the DAC Client using DAC standalone authentication mode. You will then need to restart the DAC Server.

4

DAC Quick Start

This chapter provides the essential information you need to get started using DAC to run ETL processes.

Note: Before you can perform the operations in this chapter, you need to have completed certain administrative setup tasks. If you installed and set up Oracle BI Applications by following the steps in *Oracle Fusion Middleware Installation and Configuration Guide for Oracle Business Intelligence Applications*, you have already completed the necessary tasks, and you are ready to begin running ETL processes, as described in this chapter.

In this guide, Chapter 3, "Setup Tasks and Concepts for DAC Administrators," also provides the administrative tasks that must be performed in order for a DAC environment to be operable.

The key configurations that you must perform before running an ETL process are the following:

- Configure a connection to the DAC repository.
- Configure data sources with database connectivity information.
- Configure Informatica connectivity information.
- Create custom containers in which you perform customizations.
- Register email recipients
- Create execution plan schedules
- Create data warehouse tables

Before running ETL processes, you should also be familiar with the concepts discussed in "About Source System Containers".

This chapter contains the following topics:

- About Refresh Dates and DAC's Incremental Load Strategy

Logging into DAC for the First Time as a DAC User

To perform this procedure you must have a DAC user account and you must know the location of the authentication file that you will use to connect to the DAC repository.

If you are a DAC administrator and are logging into DAC for the first time, see "Logging Into DAC for the First Time as an Administrator".

When you log into DAC for the first time as a DAC user, you configure a connection to connect to the DAC repository. DAC stores this connection information for subsequent logins.

To log into DAC for the first time as a DAC user

1. Start the DAC Client by navigating to the <Domain_Home>\dac directory and double-clicking the **startclient.bat** file.

   The Login... dialog is displayed.

2. Click **Configure**.
3. In the Configuring... dialog, select **Create Connection**, and then click **Next**.
4. Enter the appropriate connection information:

Field	Required Value
Name	Enter a unique name for the connection to the DAC repository.
Connection type	Select the type of database in which the DAC repository will be stored.

Field	Required Value
Connection String, or **Database name**, or **TNS Name**, or **Service Name**	Select the database name or database account name of the DAC repository. If you are using: ■ Oracle (OCI8), use the tnsnames entry. ■ Oracle (Thin), use the service name. ■ SQL Server, use the database name. ■ DB2-UDB, use the connect string as defined in the DB2 configuration.
Database Host	Enter the name of the machine where the DAC repository resides.
Database Port	Enter the port number on which the database listens. The default port for an Oracle database is 1521. The default port for a SQL Server database is 1433.
DB URL (Optional)	Use this option to specify a unique URL for this connection.
DB Driver (Optional)	Use this option to specify a unique driver for this particular connection.
Authentication File	Click in this field to select an existing authentication file. **Proceed to step 5 for detailed instructions.**
SSL Trust Store File	(Optional) For deployments in Web mode, location of the SSL Trust Store file.
SSL JKS Password File	(Optional) For deployments in Web mode, location of the SSL JKS password file.
Log Level	Specifies a client log level for the client session. The logs are saved in <Domain_Home>\dac\log\client <client's logical connection name>.

5. To select an existing authentication file, do the following:
   a. Click in the **Authentication File** field.
   b. In the Authentication File dialog, select **Choose existing authentication file**, and click **OK**.
   c. Navigate to the appropriate directory, and select the appropriate cwallet.sso file. Click **OK**.
   d. In the Configuring... dialog, click **Test Connection** to confirm the connection works.
   e. Click **Apply**, and then click **Finish**.

   Note: A DAC administrator must distribute this authentication file to all user accounts that need to access this DAC repository.

6. Click **Apply**, and then click **Finish**
7. To log in using Web mode, do the following:
   a. In the Login... dialog, select the appropriate **Connection** from the drop-down list.
   b. Enter your **User Name** and **Password**.

This must match the user name and password stored in the WebLogic Server identity store.

c. Select **FMW** as the Authentication Type.

d. If you want DAC to remember the password for this connection, select **Remember Password**.

e. Click **Login**.

The DAC Client is displayed.

8. To log in using DAC standalone mode, do the following:

a. In the Login... dialog, select the appropriate **Connection** from the drop-down list.

b. Enter your DAC user account **User Name** and **Password**.

c. Select **DAC** as the Authentication Type.

d. If you want DAC to remember the password for this connection, select **Remember Password**.

e. Click **Login**.

9. In the Starting tab dialog, specify the initial view and tab that you want to appear each time you log in, or leave the default. Click **OK**.

Running an Execution Plan

An execution plan is an ETL process that comprises one or more subject areas, connectivity parameters for the physical source and target data sources, and pre- and post-ETL tasks. Before an execution plan runs, DAC checks the connections to the data sources and establishes connections to the ETL tool.

Predefined execution plans are available with an Oracle BI Applications installation. You can run predefined execution plans from the predefined container if you do not make any changes to the predefined source system container.

Predefined execution plans should be run mainly for proof-of-concept purposes. After running a predefined execution plan, you can then review the predefined dashboards and reports in order to perform a gap analysis that will help you understand what customizations are required for your organization. With this knowledge, you can then create a copy of the source system container and modify the predefined execution plan or create a new one by selecting the appropriate subject areas for your ongoing ETL requirements.

Note: You cannot change any of the metadata in a predefined container. You must first make a copy of the container, and then modify objects within the custom container. See "Creating or Copying a Source System Container" for instructions.

For more detailed information about execution plans, see "Building and Running Execution Plans".

To run a predefined execution plan

1. Make sure the DAC Server is running. For instructions on starting the DAC Server, see "Starting and Stopping the DAC Server (Standalone Mode)".

2. In the Execute view, go to the **Execution Plans** tab.

3. Select the execution plan you want to run.

4. Start the execution plan:

a. Click **Run Now**.

b. In the **Starting ETL** dialog, click **Yes**.

Note: The **Honor Time Delays** property determines whether the Delay property in the Connectivity Parameters subtab will be active. The Delay property indicates how many minutes an extract of a data source will be delayed after the first extract of a multiple source extract process has started. For information about the Delay property, see "Setting Up Extract Delays".

Once the execution plan starts running you can monitor its progress in the Current Runs tab. See "Monitoring Execution Plan Processes" for instructions.

To run a customized execution plan

1. Create a custom container based on the existing predefined container. See "Creating or Copying a Source System Container" for instructions.
2. In the Execute view, go to the **Execution Plans** tab, and click **New**.
3. Enter a name for the execution plan, and click **Save**.
4. Associate one or more subject areas with the execution plan.
   a. Click the **Subject Areas** subtab.
   b. Click **Add/Remove** in the bottom pane toolbar.
   c. In the Choose Subject Areas dialog, select the custom container from the drop-down list that you created in step 1.
   d. Query for the subject area you want to associate with the execution plan.
   e. Select the subject area and click **Add**.

      You can repeat this process to associate multiple subject areas from any available source system container with an execution plan.

   f. Click **OK** to close the window.
5. Generate the runtime connectivity parameters.
   a. Click the **Connectivity Parameters** subtab, and then click **Generate** in the bottom pane toolbar.

      The Generating Parameters... dialog lists the containers that are involved in this execution plan.

   b. In the Generating Parameters dialog, enter the number of copies for each source system container. Unless you are running a multi-source execution plan, enter the value **1**, and then click **OK**.

      DAC automatically generates the parameters required for the source system container.

   c. In the **Value** column, for each folder or database connection, select the appropriate physical folder name or database connection.

      Note: For the data source type of FlatFileConnection, make sure you have copied all files into the directory specified in the DAC system property InformaticaParameterFileLocation.

   d. Leave the default values in the **Delay** and **Prune Time** fields.

For a description of all the fields in this subtab, see "Execution Plans Tab: Connectivity Parameters Subtab".

6. (Optional) Add one or more tasks that will run before the ordered tasks generated by DAC.
   a. Click the **Preceding Tasks** subtab, and then click **Add/Remove**.
   b. Select the appropriate container from the drop-down list.
   c. Query for and select the task you want to add, and then click **Add**. Note: Only tasks that have the task phase Pre ETL Process will appear in this list.
7. (Optional) Add one or more tasks that will run after the ordered tasks generated by DAC.
   a. Click the **Following Tasks** subtab, and then click **Add/Remove**.
   b. Select the appropriate container from the drop-down list.
   c. Query for and select the task you want to add, and then click **Add**. Note: Only tasks that have the task phase Post ETL Process will appear in this list
8. In the top pane of the Execution Plans tab, make sure the new execution plan is highlighted, and click **Build**.
9. In the Building... dialog, select the option **Selected Record Only**, to build only the selected execution plan.
10. Review the ordered tasks for the execution plan:
    a. Click the **Ordered Tasks** subtab.
    b. Click **Refresh** in the bottom pane toolbar to populate the list of ordered tasks.
    c. To view details about a specific task, select the task, and then click **Details** in the bottom pane toolbar.

       For instructions on unit testing a task, see "Unit Testing Execution Plan Tasks".
11. Make sure the DAC Server is running. For instructions on starting the DAC Server, see "Starting and Stopping the DAC Server (Standalone Mode)".
12. Start the execution plan:
    a. Click **Run Now**.
    b. In the **Starting ETL** dialog, click **Yes**.

    Once the execution plan starts running you can monitor its progress in the Current Runs tab. For instructions, "Monitoring Execution Plan Processes".

    For information about how refresh dates are tracked, see "About Refresh Dates and DAC's Incremental Load Strategy".

    To schedule an execution plan, see "Scheduling an Execution Plan".

Creating or Copying a Source System Container

In DAC, the metadata for a source system is held in a container. Included with Oracle BI Applications are predefined containers that hold metadata specific for the supported source systems. An important feature of DAC is that the metadata for predefined containers *cannot be changed*. To customize the metadata in a predefined container, you must first make a copy of the container. DAC keeps track of all customizations in the copied container, so that at any time you can find the newly created objects and modified objects, as well as the original objects. DAC is also able to

compare the modified objects with the predefined object definitions in the predefined container. This feature enables you to selectively rollback changes to the objects if necessary.

For information about managing object ownership, see "Ownership Right-Click Commands".

You can also create a new, empty container if you want to build your own container with customized metadata.

To create a new container or copy an existing container

1. In DAC menu bar, select **File**, and then **New Source System Container**.
2. Enter an **ID** and a **Name** for the container.

   The ID and Name fields are alphanumeric. The Name can contain spaces but the ID cannot. The Name field must be at least five characters long. It is strongly recommended that you use a descriptive Name and ID. For example:

   - Name: Customized Fusion V1
   - ID: CUST_FUSN_V1
3. Select one of the following:
   - **Create Empty New Source System Container**
   - **Create as a Copy of Source System Container**
4. If you are creating an empty, new container, click **OK**.
5. If you are making a copy of an existing container, select the existing container from the drop-down list, and then click **OK**.

   Note: All records in the base container will be referenced to the newly created custom container, and parent-child relationships between the containers will be established.

Monitoring Execution Plan Processes

The Current Runs tab in the Execute view lists all execution plans that are currently running or that have not yet completed. It provides a number of predefined reports that enable you to monitor execution plan processes and diagnose errors and performance issues.

The Tasks and Task Details subtabs of the Current Runs tab provide detailed information about the status of tasks associated with the execution plan.

This section contains the following topics:

Generating Run Reports

Follow this procedure to generate reports for execution plans that are running or have not yet completed.

To generate run reports

1. Go to the **Current Runs** tab in the Execute view.
2. Right-click and select **Get Run Information**.
3. The following options are available:
   - **Get Log File**. Saves a log file in the log\statistics directory.
   - **Analyze Run**. Saves a description of the run in HTML format in the <Domain_Home>\dac\log\statistics directory. For the period of the run, the file summarizes the number of queued, running, runnable, completed, failed, and stopped tasks. You can review this report to determine whether any tasks are creating performance issues that could be resolved to reduce the overall ETL process time.
   - **Get Task Gantt Chart**. Displays a Gantt chart in a separate window showing all tasks based on the start and end times. This chart can be useful to identify any long running ETL phases or to summarize the ETL run.
   - **Get Phase Gantt Chart**. Displays a Gantt chart in a separate window showing all the execution plan phases based on the start and end times. This chart can be useful to identify any long running execution plan phases or to summarize the ETL run.
   - **Get Run Graph.** Displays a graph showing how the number of tasks belonging to different statuses varies across the ETL run.
   - **Statistics**. Displays the number of times the execution plan attempted to run, the time from the start of the first attempt until the end of the last attempt, and the cumulative running time of all attempts.

Viewing the Life Span of a Task

Once an execution plan is started, the life span of a task consists of the following statuses. You can view the status and other detailed information about the task in the Task and Task Details subtabs of the Current Runs tab. The list of tasks in these subtabs is color coded based on the status.

1. **Queued.** Task is waiting for one or more predecessor tasks to complete. Appears as light yellow.
2. **Runnable.** Task is waiting on a resource token to be available. All predecessor tasks have completed. Appears as yellow.
3. **Waiting.** Task is eligible to run but is waiting because a time delay for the source was defined in the connectivity parameters or because an event delay was defined at the data source level (see "Setting Up Extract Delays, Event Delays and Data Source Notifications"). Appears as white.
4. **Running**. Task obtained a resource token and has started running. Appears as blue.
5. **Paused.** Task is a member of a task group and has paused until the child tasks of the task group are completed. Appears as blue.
6. **Completed.** All task details of the task have executed successfully. Appears as green.

7. **Failed.** One or more of the task details of the task have failed. Appears as red.
8. **Stopped.** Task has stopped because one or more predecessor tasks failed. Appears as ochre.
9. **Not Executed.** Task has a heuristics definition that prevents it from running (see "Using Heuristics to Manage Tasks, Tables and Indexes"). Appears as white.

In general, when an execution plan starts, the first tasks to run are those with no dependencies (or predecessors). A task becomes "Runnable" if all of its predecessor tasks are completed. When all of the tasks of an execution plan are completed, the status of the execution plan becomes "Completed."

Identifying Why a Task Failed

When a task fails, DAC provides detailed information about the cause of failure.

To identify why a task failed

1. Go to the **Current Runs** tab in the Execute view.
2. Click the **Tasks** subtab.
3. In the drop-down list on the right side of the subtab toolbar, select **Failed**.

   The failed tasks are displayed.
4. Select a task, and click **Details**.

   Information about the task details is displayed. (Task details are the building blocks of the task.) The Status Description field provides extensive information about the cause of failure of a task detail. After you have understood and fixed the problem, you can restart the execution plan. DAC will attempt to re-execute the failed task.

   If you fixed the problem and executed the task outside of DAC (for example, by running the task in Informatica), within DAC you can set the task details status to Completed. Then, when you re-run the execution plan, this task will be ignored.

   See the following topics for more information:

   - Using DAC's Hotfix Capability to Handle Failed Tasks While an Execution Plan Is Still Running
   - Restarting an Execution Plan That Failed

Restarting an Execution Plan That Failed

When an execution plan runs, if a task fails, the status of the tasks that are dependent on the failed task is changed to Stopped. While tasks are still running, the status of the execution plan is Running. Even though there may be failed tasks, the tasks unrelated to the failure will run to completion. After all the tasks have run, if one or more tasks failed, the execution plan's status is changed to Failed.

When restarting an execution plan, you can restart from the point of failure, or you can manually run a task detail using an ETL tool, and then in DAC mark the task detail as Completed, so that DAC skips the task when the execution plan restarts.

To restart an execution plan from the point of failure

1. In the Current Runs tab, select the execution plan you want to restart.
2. In the toolbar, click **Restart**.

DAC automatically starts the execution plan from the point of failure. No further action is necessary.

To restart an execution plan after manually running a task detail

1. Identify the task or tasks that failed.

   For instructions, see "Identifying Why a Task Failed".

2. Fix the problem that caused the task detail to fail, and then re-execute the task detail using an ETL tool (such as Informatica).

3. In DAC, mark the task detail as Completed:

   a. In the Current Runs tab, select the appropriate execution plan.

   b. Click the **Tasks** subtab.

   c. In the drop-down list on the right side of the subtab toolbar, select **Failed**.

      The failed tasks are displayed.

   d. Select the task that you re-executed, and click **Details**.

   e. In the Details dialog, click the **Task Details** tab.

   f. In the Status column, select **Completed** from the drop-down list.

4. Restart the execution plan that failed by selecting it in the top pane of the Current Run tab, and then clicking **Restart**.

   DAC will rerun all the tasks with a status other than Completed.

> **Caution:** The DAC Server does not validate tasks that have been run manually.

Using DAC's Hotfix Capability to Handle Failed Tasks While an Execution Plan Is Still Running

When an execution plan is running and a task fails, you can fix the issue that caused the task to fail and then re-run the task, without having to wait for the entire execution plan to stop. You re-run the task by requeuing it in DAC.

When one or more tasks in an execution plan fails, DAC continues to run all the tasks that are possible to run. Therefore, the failure of one or more tasks does not cause the execution plan to stop. The status of the execution plan is changed to Failed when the status of all tasks has changed to Stopped, Failed, or Completed.

When a task fails, the status of the dependent tasks changes as follows:

- For tasks that do not belong to a task group, the status of all the dependent tasks is changed to Stopped.
- For tasks that do belong to a task group and have the Restart All property set to False, the status of all the child tasks is changed to Stopped. When all the tasks in the task group have completed, the status of the task group is changed from Paused to Failed.
- For tasks that do belong to a task group and have the Restart All property set to True, the status of *all* the child tasks is changed to Stopped. Even if tasks belonging to the task group have completed, their status is still changed to Stopped. Also, the status of the current task that failed is changed to Stopped.

When all the tasks in the task group fail, the task group status is changed from Paused to Failed.

Note: To requeue a task that belongs to a task group, you must wait until the task group has completed and then requeue the task group, not the individual task.

Requeuing a Failed Task

A task comprises a main task detail and additional secondary task details. For example, a typical load task has a main task detail that loads data into the target table and secondary task details that drop and create indexes and analyze the table.

When a task detail fails during the running of an execution plan, the status of the task is changed to Failed, and the status of the dependent tasks is changed to Stopped. After you fix the issue that caused the task detail to fail, you can requeue the task in DAC by manually changing the task detail status to Queued. When DAC consumes the requeued task for execution, it changes the task status to Runnable. At this point, DAC also changes the status of the dependent tasks to Queued for all dependent tasks that have no failed predecessors.

You can also choose to re-execute the task detail outside of DAC, using an ETL tool (such as Informatica). You then need to change the task detail status in DAC to Completed. This manual update will trigger DAC to consume the task back into execution, which, in turn, will update the status of the dependent tasks to Queued for all dependent tasks that have no failed predecessors. During execution of the task, DAC will skip the task details that have a Completed status.

To requeue a failed task in DAC

1. In the Current Runs tab of the Execute view, select the appropriate execution plan.
2. Click the **Tasks** subtab.
3. In the drop-down list on the right side of the subtab toolbar, select **Failed**.

   The failed tasks are displayed.
4. Select the appropriate task, and click **Details**.
5. Requeue the failed task:
   a. In the Details dialog, click the **Task Details** tab.
   b. In the Status column, select **Queued** from the drop-down list.
6. If you re-executed a failed task detail to completion outside of DAC, change the task detail in DAC to Completed:
   a. In the Details dialog, click the **Task Details** tab.
   b. In the Status column, select **Completed** from the drop-down list.

Additional Points to Consider

You should be familiar with the following points before you use the Hotfix capability:

- DAC will requeue the successors of a failed task only when all of the predecessors have completed.
- DAC polls for any failed tasks that were fixed as long as the execution plan is running. The polling stops when the status of the execution plan becomes Failed
- If the task detail that failed is the main task detail (that is, the same as the command for full or incremental load), any "Truncate Table" or "Upon Failure Restart" actions defined for this task will be re-executed.

As an example, an error in an ETL mapping would cause a main task detail to fail. After fixing the error, you would requeue the main task detail by manually setting the status to Queued. When DAC is ready to run the main task detail, it would change the status to Runnable. At this time, DAC would also change the status of all dependent task details, such as a truncate table statement, to Queued.

- If the task detail that failed is not the main task detail, then DAC will continue to run the execution plan from the point of failure.

  For example, suppose the main task detail loaded a target table and completed successfully, but it loaded duplicate records, which caused a dependent task detail for creating a unique index to fail. In this case, you would clean up the duplicate records, and then manually change the status of the task detail for creating the unique index to Queued. This will trigger DAC to re-consume the task and run the dependent task details that have a status other than Completed.

- You can only change the status of a task detail from Failed to Completed or Failed to Queued. If you want to force a task detail with a Completed status to run again, query for the task detail, and then use the Update Record right-click command to set the status to Queued.

Scheduling an Execution Plan

Follow this procedure to schedule an execution plan.

To schedule an execution plan

1. Go to the **Scheduler** tab in the Execute view.

   The current list of schedules is displayed in the top pane.

2. Click **New** in the top pane toolbar.

   The Edit tab in the bottom pane becomes active.

3. Enter a name for the schedule.
4. Select an execution plan from the drop-down list.
5. Do one of the following:
   - If you want the schedule to run once, select the **Run Only Once** check box, and then select a start date.
   - To create a periodic schedule, select a recurrence pattern, and enter the appropriate date and time parameters.
6. Click **Save**.

Unit Testing Execution Plan Tasks

You can test how DAC will execute an individual task (and its details) without running a fully functional execution plan. You can test any task regardless of its position in the dependency graph. For example, you can test a task at depth 10, without running the execution plan. You can also test tasks that belong to single-source or multi-source execution plans. From the Unit Test dialog, you can execute the task after reviewing the task details. The unit test occurs within the DAC Client; the command is not sent to the DAC Server.

Note: When you test a task, DAC generates the required parameter files and issues the proper pmcmd command to execute the appropriate workflow. You cannot re-run the

corresponding workflow in Informatica because DAC generates parameter files and log files using variable names.

To unit test an execution plan task

1. Go to the **Execution Plans** tab in the Execute view, and select the appropriate execution plan.
2. Click the **Ordered Tasks** subtab.
3. Query for and select the task that you want to test.
4. Click **Unit Test** in the toolbar.

   The Unit Test dialog displays the steps that DAC will carry out if the task is executed.
5. To execute the task, click **Execute** in the Unit Test dialog.

   Note: The DAC system property Dryrun must be set to False for the task to run.

About Refresh Dates and DAC's Incremental Load Strategy

In DAC, refresh dates refer to the timestamp on the source and target tables that are associated with an execution plan. DAC uses refresh dates (indicated in the Refresh Dates subtab of the Physical Data Sources tab) to determine whether to issue full or incremental load commands. DAC automatically populates and updates refresh dates for all tables involved in the ETL process. DAC stores refresh dates for the appropriate tables by data source.

DAC tracks refresh dates for primary source and auxiliary tables and primary target tables on tasks in a successful, completed run of an execution plan. Note that DAC updates refresh dates only when the ETL process completed successfully.

DAC runs the full load command for tasks if the refresh date against the table is null. When there are multiple primary source or auxiliary tables, the earliest of the refresh dates will trigger a full load or an incremental load. If any one of the primary source or auxiliary tables has no refresh date, then DAC will run the full load command. A source refresh date is the minimum of both the source and target tables. A target refresh date is the minimum of the refresh dates for only the target tables.

At the successful end of the ETL process, DAC updates the refresh dates for all the source (primary and auxiliary) and target tables with the actual ETL start time as the refresh date. Even if the ETL process fails multiple times, the timestamp of the first attempt of the ETL process is used as the last refresh timestamp.

For example, suppose an ETL process started on January 1, 2011 at 10:00 PM Central time and ran for one hour before some tasks failed, thus causing the ETL process to fail. The ETL process was then restarted at 8:00 AM on January 2, 2011, and completed successfully. At the end of the ETL process, DAC updates the refresh dates for all primary/auxiliary source and target tables that participated in the ETL process as January 1, 2011, 10:00 PM, adjusted to the time zone of the transaction system. This means that if the transactional system from which the data is sourced is in the Pacific time zone (-2 hours), the tables for the source database will have a refresh timestamp as January 1, 2011, 9:00 PM. And, if the data warehouse is in the Eastern time zone (+1), the data warehouse tables will have a refresh timestamp as January 1, 2011, 11:00 PM.

Note: If an extract task fails and is restarted any number of times, the last_extract_ timestamp value provided by DAC will always be the same as the last ETL process start time. The last_extract_timestamp is not the last time the task was attempted. This

behavior is an intentional design feature in DAC. If a task fails N number of times, the value of this variable should not fluctuate, because there could be a potential for data loss.

Refresh Date Scenarios

Table 4–1 shows the possible scenarios regarding refresh dates and load and truncate commands.

Table 4–1 Refresh Date Scenarios

Scenario	Source Refresh Date (Primary and Auxiliary only)	Target Refresh Date	Command DAC Will Use	Truncate Target Table?
1	Null	Null	Full Load	Yes
2	Null	Not Null	Full Load	No
3	Not Null	Null	Full Load	Yes
4	Not Null	Not Null	Incremental Load	No

Note: Refresh dates for micro ETL execution plans are kept separate from the refresh dates for regular execution plans. DAC saves refresh dates for micro ETL execution plans in the Micro ETL Refresh Dates subtab of the Execution Plans tab (Execute view) and refresh dates for regular execution plans in the Refresh Dates subtab of the Physical Data Sources tab (Setup view).

Adding or Resetting Refresh Dates

You can manually add refresh dates to tables in the DAC repository or reset refresh dates before running an execution plan. The Add Refresh Dates command in the Execution Plans tab right-click menu prepopulates records with the appropriate table name in the Refresh Dates subtab of the Physical Data Sources tab. After you execute the Add Refresh Dates right-click command, you can manually add refresh dates to the prepopulated records.

How DAC Computes Timestamps for Refresh Dates

The baseline timestamp for a refresh date is the ETL process start time of the DAC Server host machine. DAC computes and uses timestamps in the following ways:

- For timestamps that appear in the Run History tab of the Execute view, DAC uses the ETL start time of the DAC Server host.
- For refresh dates, which are stored in the Refresh Dates subtab of the Physical Data Sources tab in the Setup view, DAC stores the timestamp of the data source database.

  For example, an ETL process is started on 1/1/2011 at 10 PM Central standard time. It runs for an hour and some tasks fail, and, therefore, the ETL process fails. The ETL process is restarted on 1/2/2011 at 8:00 AM. This time the ETL process successfully completes. At the end of the ETL process, DAC updates the refresh dates for all the primary and auxiliary source tables and the target tables that participated in the ETL process as 1/1/2011 10:00 PM Central standard time but adjusts the times based on the time zone of the transactional system and the data warehouse. In this example, the data is sourced in Pacific standard time (-2 hours), so the timestamp of the source tables is 1/1/2011 8:00 PM. The data warehouse is

in the Eastern standard time (+1 hour), so the timestamp of the target tables is 1/1/2011 11:00 PM.

- For flat file sources, DAC uses the timestamp of the DAC Server host.
- For incremental ETL processes, the ETL logic uses two types of timestamps:
  - **LAST_REFRESH_DATE**. The last time data was extracted from the table.
  - **PRUNED_LAST_REFRESH_DATE**. The last refresh timestamp minus the prune date period, to ensure that no records are missed from previous extracts.

5

Building and Running Execution Plans

This chapter provides information about building and running execution plans.

This chapter contains the following topics:

Introduction to Execution Plans and Load Processes

An execution plan is the unit of work you use to organize, schedule, execute, and monitor ETL processes. To generate an execution plan, you specify one or more subject areas, connectivity parameters, and, optionally, pre- and post-ETL tasks. DAC then generates an ordered set of tasks, with dependencies, that will carry out the ETL process.

When you run an execution plan, data is extracted from one or more tables in the source system database, loaded into staging tables, and then transformed and loaded into tables in the data warehouse. The ETL process is carried out in either full or incremental mode, depending on the refresh dates for the tables associated with the execution plan.

If the source or target table refresh dates for a task are null, then DAC invokes a full load workflow command. If both the source and target tables have refresh dates, then DAC invokes the incremental workflow command. For a detailed description of refresh dates, see "About Refresh Dates and DAC's Incremental Load Strategy".

DAC supports the following extract and load combinations:

- **Full extract and full load**

  This extract and load combination is used for the very first extract and load. All data is extracted from one or more source systems and loaded into the data

warehouse in the following manner: Target tables are truncated; the bulk insert mode is used; and indexes are dropped and created.

- **Full extract and incremental load**

  This extract and load combination supports modular execution plan deployments, enabling you to deploy a new source system or new subject area that will write to data warehouse tables that already contain data. When you deploy a new source system or subject area, you should use the same staging tables for the conforming data that are in use for the original source. These staging tables will already have refresh dates, because they will have been populated with data from the original source. However, when a new source is added, the source tables will not have refresh dates. This will cause DAC to perform a full extract and incremental load. On the original source, DAC will perform an incremental extract and an incremental load. The Mode attribute in the Tasks subtab of the Current Runs tab indicates the source and load type.

  The task level properties Truncate Always and Truncate for Full Load determine whether tables are truncated during the ETL process. For any given task, if the Truncate Always property is set to True, then both full and incremental loads will truncate the target tables. If the Truncate for Full load property is set to True, then only full loads will truncate the target tables.

  When data is loaded into a staging table that has the Truncate Always property set to True, the bulk loader option can be used for both full and incremental commands. When data is written directly to a dimension or fact table that has the Truncate for Full Load property set to True, DAC will use the command for full load to extract all data from the source system but will not truncate the target tables, and, therefore, will not drop indexes. Therefore, the command should be able to write to a table that already has data and indexes. This also means that such a full command does not use bulk loaders, and may need an update strategy. Typically, such logic for using bulk loaders and update strategies are implemented in Informatica (or another external ETL tool).

  The incremental load process requires additional logic to determine whether a record should be inserted or updated. Therefore, if you add a new source connection to populate an existing data warehouse, you should expect the incremental load to be slower than when running a full load.

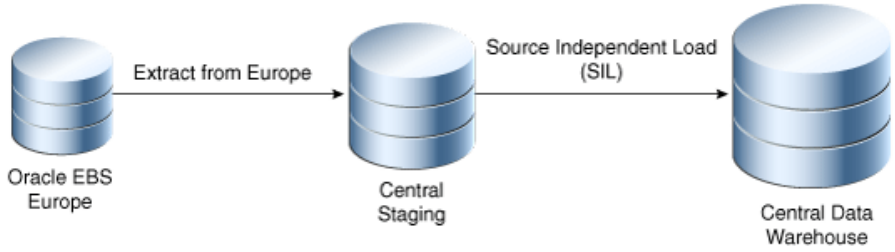
- **Incremental extract and incremental load**

  This extract and load combination is used for regular nightly or weekly ETL processes. New or changed records are extracted from one or more source systems and loaded into the data warehouse.

About Single-Source Execution Plans

A single-source execution plan extracts data from a single instance of a single source system container. As illustrated in Figure 5–1, in this scenario data is extracted from a single source, loaded into staging tables, and then loaded into the data warehouse.

Figure 5–1 Single Extract and Single Load Option



Truncate Table Behavior in a Single Extract Scenario

When DAC truncates a table, it also drops and recreates indexes (if the table has indexes) and analyzes the table after loading the data. In a single-extract scenario, the extract tasks truncation process for a table is as follows:

1. Truncate table.
2. Drop indexes.
3. Run the Informatica mapping.
4. Create indexes.
5. Analyze table.

For instructions on building a single-source execution plan, see "Building and Running Execution Plans".

About Multi-Source Execution Plans

When you have a multi-source environment, you can use a single execution plan to extract data for a given subject area for all the sources. When you build the execution plan, DAC creates as many instances of the extract tasks as there are sources, without having the need to duplicate the workflows or DAC metadata.

There are two types of multi-source execution plans:

- **Homogeneous**. This type of execution plan extracts data from multiple instances of the same source system. For example, a business might have an instance of Oracle Fusion Applications in one location and time zone and another instance of Oracle Fusion Applications in another location and time zone. In such cases, the timing of data extraction from the different instances can be staggered to meet your business requirements.
- **Heterogeneous**. This type of execution plan extracts data from multiple instances of *dissimilar* source systems. For example, a business might have an instance of Siebel 8.0 in one location and instance of Oracle EBS 11i in another location. You can also stagger the timing of data extraction when you use this type of execution plan, by setting the Delay property in the Connectivity Parameters subtab of the Execution Plans tab.

For instructions on building a multi-source execution plan, see "Building and Running Execution Plans".

Considerations for Multi-Source Execution Plans

This section describes properties and behaviors you should consider when setting up multi-source execution plans.

Source Priority

The order in which DAC loads data from the different sources that are participating in the ETL process is determined by the priority of the physical data source connection, as set in the Source Priority field in the Physical Data Sources tab of the Setup view. This property ensures that tasks attempting to write to the same target table will not be in conflict. If two sources are given the same priority, DAC will randomly stagger the tasks.

Truncate Table Behavior

If a target table is shared across different sources, it will be truncated only once. The priority of the data source (specified by the Source Priority property) determines which of the extracts truncates the tables. For more information about how tables are truncated, see "Execution Plan Build Process Rules".

Note that truncate properties should be the same across all source system containers associated with a multi-source execution plan.

In a multiple extract scenario, the truncation process for a table is as follows:

1. Truncate table (first extract task).
2. Drop indexes (first extract task).
3. Run the Informatica mapping (first extract task).
4. Run the Informatica mapping (second extract task).
5. Create indexes (second extract task).
6. Analyze table.

Enabling Failure Restarts When Extracting From Multiple Sources

Typically, extract tasks truncate the target tables, and, therefore, will not have update strategies. To enable failure restarts when extracting from multiple sources, you can create an action that will predelete all the records that are loaded from the source. See "Using a Task Action to Enable Failure Restarts When Extracting From Multiple Sources" for instructions.

Task Groups

Tasks should belong to the same task groups across all source system containers associated with the multi-source execution plan. For example, if a task belongs to a task group in one container but is not associated with a task group in a different container, then an exception will be thrown. Also, if a task with the same name belongs to two different task groups, an exception will be thrown.

Full and Incremental Workflow Commands

Full and incremental workflow commands need to match for tasks across all source system containers.

Configuring Extracts and Notifications

You can delay extracts for a particular source system by using the Delay property in the Connectivity Parameters subtab of the Execution Plans tab. You can also use the Event Delay feature to configure the extracts for different source systems to occur independently.

The Notifications feature enables you to initiate email notifications and the execution of custom SQL based on the first or last read or write operations on a particular data

source. See "Setting Up Extract Delays, Event Delays and Data Source Notifications" for more information.

Task Physical Folder Instance Priority

DAC deduplicates tasks from multiple source containers or multiple instances of the same container based on the task name, the primary physical source, and the primary physical target. DAC executes only a single instance of the task. If multiple containers contain the same task, with the same source and target databases but different task physical folders, then DAC will pick one folder. If folders have different priorities, the folder with the lowest priority will be picked. If folders have the same priority, DAC randomly selects a folder.

In Oracle BI Applications, the Instance Priority for task physical folders is predefined. In certain situations, such as if you are using a vertical application, you may want to execute a task from a different folder. You can specify the preference of task physical folders by modifying the Instance Priority. To do so, on the Tools menu, select Seed Data, and then select Task Physical Folders.

DATASOURCE_NUM_ID

- All tables should contain the column DATASOURCE_NUM_ID, which describes the source from which the data is coming.
- Unique indexes should always include the column DATASOURCE_NUM_ID.
- All the extract mappings should populate the DATASOURCE_NUM_ID column from the parameter file produced by DAC.
- All the load mappings should extract the value of the DATASOURCE_NUM_ID column from the staging tables.

Multi-Source Execution Plan Extract and Load Scenarios

Multi-source execution plans support two load scenarios:

- **Multiple extract and single load**. In this scenario, as illustrated in Figure 5–2, data is extracted from multiple sources and loaded into central staging tables. After all of the extract tasks have completed, the data is loaded into the data warehouse in a single load process.
- **Multiple extract and multiple loads**. In this scenario, as illustrated in Figure 5–3, data is extracted from multiple sources and loaded into non-centralized staging tables. After all of the extract tasks have completed, the data is loaded into multiple data warehouses using multiple load processes.

Figure 5–2 Multiple Extract and Single Load Option

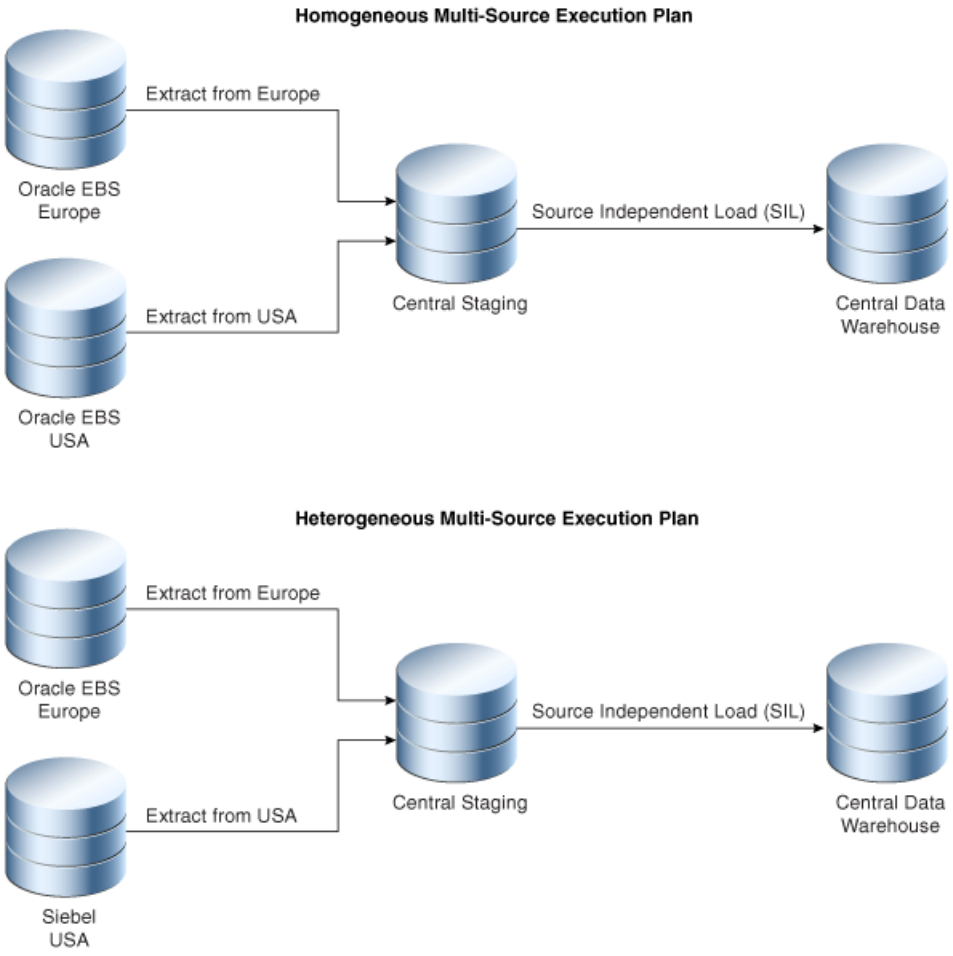
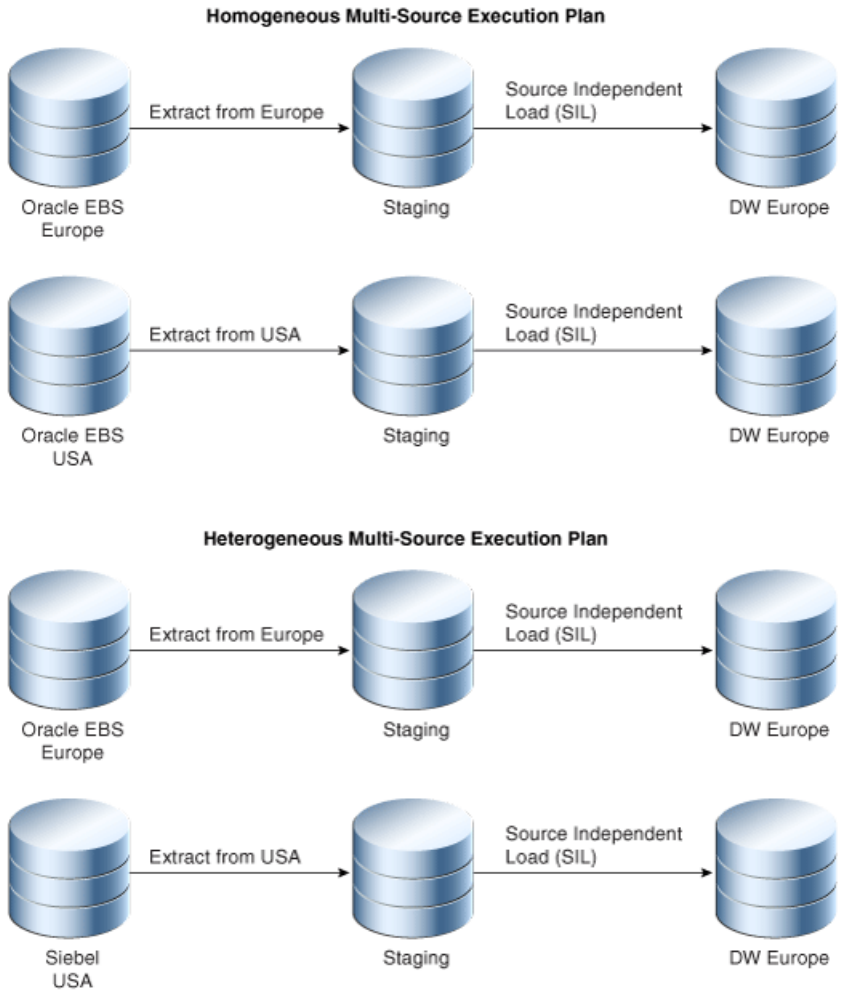


Figure 5–3 Multiple Extract and Multiple Load Scenario



About Micro ETL Execution Plans

Micro ETL execution plans are ETL processes that you schedule at very frequent intervals, such as hourly or half-hourly. They usually handle small subject areas or subsets of larger subject areas.

For instructions on building a micro ETL execution plan, see "Building and Running Execution Plans".

This section contains the following topics:

- Why Use a Micro ETL Execution Plan?
- About Refresh Dates and Micro ETL Execution Plans
- Important Considerations When Using Micro ETL Execution Plans
- Designing a Micro ETL Execution Plan

Why Use a Micro ETL Execution Plan?

A regular execution plan includes one or more fact tables and all content associated with the fact tables, such as dimension tables and related tables, and so on. The more associated content that is added to an execution plan, the longer the ETL process runs. If a subject area is small, for example, with one fact table and two dimension tables, there is no need to create a micro ETL execution plan. Instead, you can schedule an execution plan with a single, small subject area to run at frequent intervals.

Micro ETL execution plans, on the other hand, accommodate a business need for specific users to access updated subsets of data at frequent intervals. For example, some users may find their reports are stale if data is only refreshed once nightly. In such cases, a micro ETL execution plan can refresh the results of a small subject area or a subset of a star schema hourly or half-hourly. If you have a fact table with 50 dimension tables as well as other multiple related stars, an execution plan containing this fact table would run for a long period of time and could not be scheduled to run at frequent intervals. You could create a micro ETL execution plan by removing the unnecessary tables and tasks, leaving only the fact and dimension tables required for a specific report. See "Designing a Micro ETL Execution Plan" for instructions.

You should be aware that inconsistencies can occur when running micro ETL execution plans, such as dimension keys pointing to "Unspecified" records and discrepancies between facts and aggregates. However, the subset of data extracted for a micro ETL execution plan will be updated and patched during a regularly scheduled nightly ETL process. See "Important Considerations When Using Micro ETL Execution Plans" for other points to consider.

Note: All tasks that need to participate in the micro ETL execution plan should have update strategies that can update the data even though it has already been loaded into the data warehouse.

About Refresh Dates and Micro ETL Execution Plans

DAC tracks refresh dates for tables in micro ETL execution plans separately from other execution plans and uses these refresh dates in the change capture process. After a micro ETL execution plan runs, DAC populates refresh date values in the Micro ETL Refresh Dates subtab of the Execution Plans tab. If a subject area is used in a regular execution plan (an execution plan with the Micro ETL option in the Execution Plans tab *not* selected) as well as a micro ETL execution plan, DAC maintains refresh dates for the tables in the regular execution plan in the Refresh Dates subtab of the Physical Data Sources tab (Setup view).

In cases where a subject area is used in both a regular and micro ETL execution plan and the micro ETL execution plan is suspended for a few days but the regular execution plan runs nightly, DAC automatically detects the last refresh date for the tables common to both execution plans and intelligently extracts only the most recent records for the micro ETL execution plan. Micro ETL refresh dates do not affect the refresh dates of regular execution plans.

As an example, suppose a micro ETL execution plan runs every hour and the complete (regular) execution plan runs once a day at 1:00 PM. When the micro ETL execution plan runs at 10:00 PM, it will get the refresh timestamp as 9:00 PM. When the complete execution plan runs at 1:00 PM, it will get the refresh timestamp as the previous day at 1:00 PM, even though the micro ETL execution plan ran at 12:00 PM. This will cause all the records to be re-extracted that were already loaded into the data warehouse using the micro ETL execution plan. As long as all tasks have proper update strategies, DAC will re-update the records that were loaded during the day.

Note: The image sync for Siebel sources does not occur during micro ETL execution plans. This means that the records from the I image tables do not get moved into the R image tables. Therefore, the records are re-extracted during the complete (regular) execution plan.

Important Considerations When Using Micro ETL Execution Plans

Micro ETL processes can cause issues with data inconsistencies, data availability, and additional load on the transactional database. Therefore, you should consider the following factors before implementing a micro ETL process:

- For related star schemas, if one schema is omitted from a micro ETL execution plan, the cross-star reports may be inaccurate. For example, if the Person fact table is refreshed more frequently than the Revenue fact table, a report that spans the Person and Revenue dimensional schemas may produce inconsistent results.
- If you omit dimension tables from a micro ETL execution plan, the foreign keys for the fact tables will point to Unspecified rows for the new dimension records. The foreign key references will be resolved when the Complete ETL execution plan is run, but users of the reports should be aware of such inconsistencies.
- If you do not include aggregate tables in micro ETL execution plans, the reports that use data from these tables will be inconsistent with the reports that use data from the detailed fact tables. However, if aggregate tables are included in the micro ETL execution plan, the aggregate calculations are performed for each ETL process, which will take a constant amount of time and may be inefficient to perform at such frequent intervals.
- Ensure any tasks or tables for soft-deletes are included in the micro ETL to identify deleted records from the source system.
- Ensure all downstream post load processes are included as well as aggregates. This can be actual business dependencies, such as the GL Reconciliation process in Financial Analytics. For example, suppose a customer wants to refresh just the AP (Payables) fact three times a day. This will require the whole GL Reconciliation process and tasks to run to match the GL with AP. Otherwise, there will be a mismatch between the balance and transaction reports.
- Hierarchy tables are rebuilt during every ETL execution plan by querying the base dimension tables. This operation takes a constant amount of time. If the base tables are big, this operation may take a long time and may be inefficient if the micro ETL execution plan runs several times a day. However, if you avoid populating the hierarchy tables during micro ETL processes, data inconsistencies will occur.
- With micro ETL execution plans, BI Server caching will occur more frequently, which may have performance implications.
- Micro ETL execution plans will put more load on the transactional database because of the frequent extracts.
- Ensure the business is aware of the frequency of micro ETL execution plans and how dashboard results and critical reports are impacted throughout the day.

Designing a Micro ETL Execution Plan

Follow this procedure to design a micro ETL execution plan.

To design a micro ETL execution plan

1. In the Design view, select the appropriate source system container from the drop-down list in the toolbar.
2. Go to the Subject Areas tab, and click **New**.
3. Enter a name for the subject area, and click **Save**.
4. In the Tables subtab, click **Add/Remove**.

   The Choose Tables dialog is displayed. The left-hand window lists all the tables held in the selected container.
5. Query for one or more fact tables.
6. Select the fact table (use Shift+click to select more than one table), and click **Add**.
7. Click **OK** to close the Choose Tables dialog.
8. Click **Assemble** in the Subject Areas tab toolbar.
9. In the Assembling... dialog, select **Selected record only**.

   The tree view on the left side of the Subject Area Assembly dialog displays the fact tables that belong to the subject area. You can expand the fact table node to view its related tables.
10. Deselect the unnecessary fact and dimension tables.
11. Click **Calculate Task List** to assemble the tasks needed to load the tables displayed in the tree view.

    A list of tasks is displayed in the Task List tab on the right side of the window. Also, the Table List tab displays the tables included in the subject area.
12. Click **Accept** to complete the subject area assembly process.
13. Inactivate any unnecessary tasks:
    a. With the appropriate subject area selected in the Subject Areas tab, click the Tasks subtab.
    b. Select the **Inactive** check box for any unnecessary task.
    c. Reassemble the subject area by clicking **Assemble** in the top pane toolbar.

Note: You can also create a micro ETL execution plan by creating a new subject area and then using the Tasks subtab to associate specific tasks with it.

Execution Plan Build Process Rules

DAC builds an execution plan (generates a task dependency graph) based on the following rules for the metadata in the Design and Setup views.

- **Task source and target tables**. The dependency algorithm first looks at a task's source and target tables. For example, suppose table A is populated by task T1 by reading from table B, and table B is populated by task T2 by reading from table C. The algorithm would determine task T2 should be executed before T1.

  A task with a target table that is not a source table in any other task will be a leaf node.
- **Task phase priority.** An ETL process typically goes through a number of phases. The task phase priority determines the order in which the phase will occur. For

Oracle BI Applications, the task phase priorities are predefined. You can view or modify the priorities by selecting Tools, Seed Data, Task Phases.

An example of a typical order in which phases are executed is as follows:

1. Extract Dimension
2. Extract Fact
3. Load Dimension
4. Load Fact and Load Hierarchy (executed in parallel)
5. Load Aggregate tables
6. Update Dimensions

- **Truncate properties**. DAC truncates a target table only once during the life span of an ETL execution. If multiple tasks write to the same table, the following rules apply:
  - The first task truncates the target table. This task has the highest priority. Even if there are multiple tasks that could potentially truncate a table—either from the source container or across multiple source containers—only the first task will truncate the table.
  - The task reading from the data source with the highest priority truncates the tables and drops the indexes. The last task writing to the table from the data source with the highest priority creates the indexes. The last task writing to the table will create query indexes.
  - DAC does not truncate the target tables during the execution of the subsequent tasks even if the Truncate Always or Truncate for Full Load properties are selected.
  - If the task belongs to a task group and the task group's Truncate Always or Truncate for Full Load properties are different from the individual task truncate properties, the task's truncate properties are ignored. Note: You can use a task action if you want to override this behavior. See "Using Actions to Optimize Indexes and Collect Statistics on Tables" for information about the actions feature.
- **Priority of the source connection**. When there is more than one source system, you need to specify the priority of each source connection in the Source Priority field in the Physical Data Sources tab of Setup view.
- **Task groups**. The first task in the group determines the position of the group in the dependency graph.

  DAC randomly organizes tasks that have the same property values, because it does not allow parallel reads and writes on any table. If you want to force tasks to be executed in a particular order, or in parallel, you can create a task group that allows you to specify an execution order. For instructions on creating a task group, see "Creating a Task Group"
- **Task phase dependency**. The task phase dependency is used to force dependencies between tasks that normally would not have a dependency, for example, between extract facts and extract dimensions. For instructions on setting a task phase dependency, see "Setting a Task Phase Dependency".

Building and Running Execution Plans

Follow this procedure to run a single-source, multi-source, or micro ETL execution plan.

Before you attempt to run an execution plan, make sure you have completed the following:

- In the Physical Data Sources tab, set database connections to the transactional and data warehouse databases.
- In the Informatica Servers tab, registered the Informatica Repository Service and Integration Service.
- Created a custom container from which you will select the subject areas for the execution plan.

Before you attempt to run a multi-source execution plan, you must first define the priority for each source. The priority specifies the order in which DAC will load the data from the different sources.

To define a source priority for multi-source execution plans

1. In the Setup view, select the **Physical Data Sources** tab.
2. For each of the physical data sources that will participate in the multi-source execution plan, enter a numerical value in the **Source Priority** field.

   The lower the numerical value, the higher the priority. For example, if you enter a value of 1, data from this source will be loaded first.

To build and run an execution plan

1. In the Execute view, select the **Execution Plans** tab.
2. Create a new execution plan.
   a. In the top pane toolbar, click **New**.
   b. In the top pane window or in the Edit subtab, enter the following information.

Field	Description
Name	Enter a name for the execution plan.
Full Load Always	(Optional) Select to indicate the execution plan will always execute a full load. **Note:** DAC does not store refresh dates when this option is selected.
Micro ETL	(Optional) Select to indicate the execution plan is a micro ETL execution plan. See "About Micro ETL Execution Plans" for more information.
Analyze	(Optional) Select to indicate the tables associated with this execution plan will be analyzed.
Analyze Truncated Tables Only	(Optional) Select to indicate only truncated tables will be analyzed.
Drop/Create Indices	(Optional) Select to indicate indexes on the tables associated with this execution plan will be dropped and created.
Inactive	(Optional) Select to inactivate the execution plan.

   c. Click **Save**.

3. For a micro ETL execution plan, create a subject area in the Subject Areas tab of the Design view by doing one of the following:
   - Create a new subject area by adding the appropriate tasks to it.
   - Create a new subject area by adding the appropriate tables and then deleting the unnecessary tasks.
   - Modify an existing subject area by adding or removing tasks.

   Assemble the subject area by clicking **Assemble** in the top pane toolbar. See "Creating a Subject Area" for more information.
4. Associate one or more subject areas with the execution plan.
   a. Click the **Subject Areas** subtab.
   b. Click **Add/Remove** in the bottom pane toolbar.
   c. In the Choose Subject Areas dialog, select the appropriate source system container from the drop-down list.
   d. Query for the subject area you want to associate with the execution plan. For micro ETL execution plans, query for the subject area you created in step 3.
   e. Select the subject area and click **Add**.

      You can repeat this process to associate multiple subject areas from any available source system container with an execution plan.
   f. Click **OK** to close the window.
5. Generate the runtime connectivity parameters.
   a. Click the **Connectivity Parameters** subtab, and then click **Generate** in the bottom pane toolbar.

      The Generating Parameters... dialog lists the containers that are involved in this execution plan.
   b. Enter the number of copies of each container that are needed, and then click **OK**.

      DAC automatically generates the parameters required for each copy of the source system container. Note that not all copies require all of the possible parameters.
   c. In the Value column, for each folder or database connection, select the appropriate physical folder name or database connection.

      Note: For the data source type of FlatFileConnection, make sure you have copied all files into the directory specified in the DAC system property InformaticaParameterFileLocation.

      ---

   d. (Optional) If you are extracting data from more than one source system and want to stagger the data extracts, in the Delay field for the appropriate data source, enter a value for the number of minutes you want to delay this extract after the first extract process has started.
   e. (Optional) Set the Prune Time property. This setting subtracts the Prune Time value from the LAST_REFRESH_DATE and supplies this value as the value for the $$LAST_EXTRACT_DATE parameter. See "Prune Time Column" for more information.

6. (Optional) Add one or more tasks that will run before the ordered tasks generated by DAC.
   a. Click the **Preceding Tasks** subtab, and then click **Add/Remove**.
   b. Select the appropriate container from the drop-down list.
   c. Query for and select the task you want to add, and then click **Add**.
7. (Optional) Add one or more tasks that will run after the ordered tasks generated by DAC.
   a. Click the **Following Tasks** subtab, and then click **Add/Remove**.
   b. Select the appropriate container from the drop-down list.
   c. Query for and select the task you want to add, and then click **Add**.
8. In the top pane of the Execution Plans tab, make sure the new execution plan is highlighted, and click **Build**.
9. In the Building... dialog, select the option **Selected Record Only**, to build only the selected execution plan.
10. Click the **Ordered Tasks subtab**, and verify the following:
    a. Click **Details** in the toolbar, and review each task's predecessor and successor tasks to confirm tasks common to multiple sources are ordered in a manner consistent with the priority of the source connection.
    b. Confirm that load tasks appear only once even if there are multiple extracts for tables common to multiple sources.
    c. For tasks common to multiple sources, click **Unit Test** in the toolbar, and confirm that the first common task truncates the common target table and the following tasks do not. For instructions on unit testing a task, see "Unit Testing Execution Plan Tasks".
11. Make sure the DAC Server is running. For instructions on starting the DAC Server, see "Managing the DAC Server".
12. Start the execution plan:
    a. Click **Run Now**.
    b. In the **Starting ETL** dialog, click **Yes**.

    Note: The **Honor Time Delays** property determines whether the Delay property in the Connectivity Parameters subtab will be active.

    Once the execution plan starts running you can monitor its progress in the Current Runs tab. For instructions, "Monitoring Execution Plan Processes".

    For information about how refresh dates are tracked, see "About Refresh Dates and DAC's Incremental Load Strategy".

    To schedule an execution plan, see "Scheduling an Execution Plan".

Running Execution Plans Concurrently

This section contains the following topics:

Introduction to Running Execution Plans Concurrently

You can configure DAC to run multiple execution plans concurrently if the execution plans are not dependent on one another. To avoid data inconsistency, execution plans that have dependencies with other execution plans cannot run concurrently (unless you override the default behavior).

Dependent Execution Plans

Execution plans are considered dependent if 1) they load data into tables that have the same name on the same physical data source; and 2) one execution plan writes to a table and the other execution plans reads from the same table on the same physical data source. Execution plans that have dependencies with other execution plans are not be eligible to run concurrently if any of the following scenarios are true:

- The execution plans share one or more target tables on the same physical data source.
- An execution plan has a target table that is a source for another execution plan. For example, execution plan A writes to table 1 on the physical data source DW and execution plan B reads from table 1 on DW.
- An execution plan has a source table that is a target table for another execution plan. For example, execution plan C reads from table 2 on the physical data source DW and execution plan D writes to table 2 on DW.

Execution Plans That Are Eligible to Run Concurrently

DAC allows execution plans to run concurrently as long as they write to unique tables. Tables are considered to be unique in relation to other tables if the table names are unique or if tables have the same name but reside in different physical data sources.

DAC does not distinguish whether the same Informatica workflows are used by multiple execution plans. Therefore, as long as execution plans write to unique tables, if they share the same Informatica workflows, they can still run concurrently.

For example, consider a scenario in which you have development and QA environments that each have a different OLTP and data warehouse but share the same DAC and Informatica repositories, and, therefore, use the same Informatica workflows. Execution plans for these two environments that share Informatica workflows can run concurrently.

DAC provides two options for enabling Informatica workflows to be shared by multiple execution plans:

- **Defining an execution plan prefix**

  You can define a prefix for each execution plan using the DAC_EP_PREFIX execution plan level parameter. When you use this option you are able to use single instances of the DAC Server, Informatica Integration Service, and DAC and Informatica repositories to run execution plans from multiple sources. That is, you can run multiple execution plans at the same time using the same metadata.

This solution is useful if you have multiple sources with execution plans that have the same subject area. It can also be useful if the sources are in the same time zone and the execution plans need to run simultaneously or the sources are in different time zones but have overlapping execution plan schedules.

The workflows that are common to different execution plans must be enabled in Informatica Workflow Manager to run concurrently. DAC will invoke the workflows with an instance name that is in the format <prefix>_workflowname. DAC captures the Informatica pmcmd and pmrep outputs and stores them as log files for each ETL process run. For more information about log file naming conventions, see "DAC Session Log File Naming Conventions" and "Informatica Log File Naming Conventions".

For instructions on defining an execution plan prefix, see "Defining an Execution Plan Prefix".

- **Copying the Informatica workflow folders and creating new folder mappings**

  You can make copies of the Informatica workflow folders so that there is one extract and load folder combination for each execution plan, and then create new mappings in DAC for the physical to logical folders. For instructions, see "Copying Informatica Workflow Folders and Creating New Folder Mappings".

About Resource Usage

When you run execution plans concurrently, the following resource configurations apply to each ETL process. You should consider these properties when deciding how many execution plans you will configure to run concurrently.

- **Num Parallel Workflows per EP**

  This property is located on the Informatica Servers tab in the Setup view. It specifies the maximum number of workflows that can be executed in parallel on the Informatica Server for each ETL run. For example, if this value is set to 10 and three execution plans are running concurrently, 30 Informatica workflows may be running at any given time.

- **Num Connections per EP**

  This property is located on the Physical Data Sources tab in the Setup view. It specifies the number of database connections that each ETL will open to the data source.

- **Generic Task Concurrency Limit per EP**

  This system property is located on the DAC System Properties tab in the Setup view. It specifies for each execution plan the number of tasks with an Execution Type other than Informatica that can run concurrently.

Note: When execution plans run concurrently, they do not share resources. Keep in mind that the values you set for these properties apply to each ETL process.

For example, if you configure DAC to run two execution plans concurrently, your environment must have the resources for twice the value you set for the Num Parallel Workflows per EP, Num Connections per EP, and Generic Task Concurrently Limit properties.

Make sure you have reviewed the system requirements and certification documentation for information about hardware and software requirements, platforms, and databases available on Oracle Technology Network as well as the Oracle BI Applications configuration information in Oracle Fusion Middleware Installation and Configuration Guide for Oracle Business Intelligence Applications.

Viewing Execution Plan Concurrent Dependencies

In considering whether to run execution plans concurrently, you should have an understanding of the concurrent dependencies (common target tables) between the execution plans.

To view execution plan concurrent dependencies

1. In the Execute view, select the **Execution Plans** tab.
2. Right-click one of the execution plans you want to compare, and then select **Concurrent Dependency**.
3. From the drop-down list in the Concurrent Dependency dialog, select the execution plan you want to compare with the execution plan you selected in step 2.

   The common target tables between the two execution plans are displayed as well as the tables that one execution plan writes to and the tables that the other execution plan reads from.

Configuring DAC to Run Execution Plans Concurrently

To configure DAC to run execution plans concurrently, you need to do the following:

- Set the Concurrency Level system property, which controls the number of independent execution plans that DAC will allow to run in parallel. See the instructions below.

  When the value of the Concurrency Level system property is more than one, each execution plan starts in a separate Java process. An identifier for each process is stored in the OS Process Name column on the Current Runs tab in the Execute view.

- Add the directory path to the JAVA_HOME directory to the PATH environment variable.

To set the Concurrency Level system property

1. In the DAC Setup view, select the **DAC System Properties** tab.
2. In the list of system properties, find **Concurrency Level**.

3. Click in the **Value** column, and enter a value that specifies the number of execution plans you want to run in parallel.
4. Click **Save**.
5. Restart the DAC Server by selecting **Tools**, **DAC Server Management**, **Restart DAC Server**.

Enabling Informatica Workflows to Be Shared by Multiple Execution Plans

DAC provides two options for enabling Informatica workflows to be shared by multiple execution plans:

- Defining an Execution Plan Prefix
- Copying Informatica Workflow Folders and Creating New Folder Mappings

For information about the requirements for running execution plans concurrently, see "Execution Plans That Are Eligible to Run Concurrently".

Defining an Execution Plan Prefix

Follow the procedure below to define an execution plan prefix, which enables multiple execution plans to use the same Informatica workflows.

To define an execution plan prefix

1. For each workflow associated with an execution plan that will run concurrently, enable the Informatica workflow property **Configure Concurrent Execution**:
   a. In Informatica Workflow Manager, open the appropriate workflow in the Workflow Designer.
   b. On the menu bar, select **Workflows**, and then select **Edit**.
   c. In the Edit Workflow dialog, select the **Enabled** check box for the **Configure Concurrent Execution** property.
   d. Click the **Configure Concurrent Execution** button.
   e. In the Configure Concurrent Execution dialog, select **Allow concurrent run with same instance name**.
   f. Click **OK**.
2. In the DAC Execute view, go to the **Execution Plans** tab.
3. Query for and select the execution plan for which you want to define a prefix.
4. Click the **Execution Parameters** subtab.
5. Click **New** in the subtab toolbar.
6. In the Name field, enter **DAC_EP_PREFIX**.

   Note: This value is case sensitive. You must enter the value exactly as it appears, using upper-case characters.
7. Click **Save** in the subtab toolbar.
8. Leave the default values as Text for the Data Type and Both as the Load Type.
9. Click in the **Value** field to open the Enter Parameter Value dialog.
10. Make sure **Static** is selected.
11. In the text box, enter a text string of one to four characters that defines the value of the DAC_EP_PREFIX parameter.

This value will be appended as a prefix to the execution plan name. For example, if you define the parameter value as `AMER`, and the execution plan name is `Extract_Forecast`. When the execution plan runs, its name will appear in the Current Runs tab and the log files as `AMER_Extract_Forecast`.

Copying Informatica Workflow Folders and Creating New Folder Mappings

Follow this procedure to copy Informatica workflow folders and create new folder mappings in order to configure Informatica workflows so that they can be shared by multiple execution plans.

To make copies of the Informatica workflow folders and create new folder mappings

1. In Informatica Workflow Manager, make copies of the SDE and SIL folders. You will need one copy of each folder for each execution plan.
2. Define the new physical folders in DAC:
   a. In the DAC menu bar, select **Tools**, **Seed Data**, **Task Physical Folders**.
   b. In the Task Physical Folders dialog, click **New**.
   c. Enter a name and instance priority for the physical folder.

      For information about the instance priority property, see "Task Physical Folder Instance Priority".

      Repeat this step for each copy of the physical folder you made in Informatica Workflow Manager.
   d. Click **OK** to close the Task Physical Folders dialog.
3. Define the new logical folders in DAC:
   a. In the DAC menu bar, select **Tools**, **Seed Data**, **Task Logical Folders**.
   b. In the Task Logical Folders dialog, click **New**.
   c. Enter a name for the logical folder, and click **OK**.

      Repeat this step for each copy of the logical folder you made in Informatica Workflow Manager.
4. Map the logical and physical folders:
   a. In the DAC Design view, go to the **Source System Folders tab**.
   b. In the toolbar, click **New**.
   c. Click in the **Logical Folder** column and select the appropriate folder from the drop-down list.
   d. Click in the **Physical Folder** column and select the appropriate folder from the drop-down list.
   e. Click **Save**.

      Repeat this step for each new combination of physical and logical folders.

Explicitly Defining Execution Plans as Independent or Dependent

You can override the DAC execution engine to define dependent execution plans as independent and independent execution plans as dependent.

Defining Execution Plans as Independent

You should exercise caution when overriding the DAC execution engine to define two dependent execution plans as independent of each other. You should perform this procedure only if: 1) you have an advanced understanding of the DAC ETL process; and 2) you know the load mappings handle mutually exclusive sets of data across execution plans, and the execution plans do not truncate tables; or 3) you have two execution plans that use time-based dimensions and are only loaded during full loads and not incremental loads.

Caution: If your incremental ETL process involves dropping and recreating indexes, you should not declare dependent execution plans as independent.

Defining Execution Plans as Dependent

You may want to define independent execution plans as dependent if you have resource issues. For information about potential resource issues when running execution plans concurrently, see "About Resource Usage".

To define execution plans as independent or dependent

1. In the Execute view, select the **Execution Plans** tab.
2. Select the execution plan for which you want to define the dependency attribute of another execution plan.

   For example, if you want to define the execution plan Oracle 11.5.10 Enterprise Sales as independent from the execution plan Procurement and Spend: Oracle 11.5.10, select Procurement and Spend: Oracle 11.5.10 in this step.
3. Select the **Concurrent Dependency** subtab.
4. Locate the execution plan whose dependency attribute you want to set:
   a. In the bottom pane toolbar, click **Add/Remove**.
   b. Select the execution plan, and click **Add**.
   c. Click **OK** to return to the Concurrent Dependency subtab.
5. In the Concurrent Dependency list, select the appropriate execution plan.
6. Click in the **Type** column, and select one of the following:
   - **Independent**

     Defines the execution plan selected in the Concurrent Dependency subtab as independent of the execution plan selected in the top window.
   - **Dependent**

     Defines the execution plan selected in the Concurrent Dependency subtab as dependent on the execution plan selected in the top window
7. Click **Save** in the bottom pane toolbar.

Running Multiple Instances of an Execution Plan

You can create multiple instances (or copies) of an execution plan and then run the parent execution plan and the additional child instances concurrently.

The following examples illustrate when you might find it useful to run multiple instances of an execution plan:

- You have an execution plan that is designed to extract data from a particular source and load a corresponding data warehouse. You also have three additional countries which require similar execution plans. You can create three child instances of the execution plan and reconfigure the connectivity details. The only difference between the parent execution plan and the child instances will be the connectivity details.
- You have development and QA environments that each have a different OLTP and data warehouse but share the same DAC and Informatica repositories, and, therefore, use the same Informatica workflows. You can create child instances of an execution plan to run concurrently.

  Note: You must enable the Informatica workflow property Configure Concurrent Execution in Informatica Workflow Manager for each workflow. For instructions, see step one in the procedure "Defining an Execution Plan Prefix".

When you create a new instance of an execution plan, you have the option to override the following properties:

- Physical data source.
- Physical folder.
- Context connectivity parameter for external executors. For more information about external executors, see Chapter 13, "Integrating DAC With Other ETL Tools."

Note: The behavior of execution plans and child instances is the same, including behavior related to running execution plans concurrently. The information about running execution plans concurrently in the section "Running Execution Plans Concurrently" also applies to child instances.

Important Points About Execution Plan Child Instances

- After you have created one or more child instances of a parent execution plan, if you make changes to the parent execution plan, these changes will be propagated automatically to all child instances when you rebuild the execution plan.
- All of the properties of a parent execution plan, except for execution plan level parameters, are propagated to the child instances, including connectivity parameters, dependencies, and so on. Therefore, you do not have to rebuild the child instances.
- Even though execution plan level parameters are not copied from the parent execution plan to the child instance, they are used by the instance during runtime. However, you can override one or more of the execution plan level parameters defined in the parent execution plan by creating new execution plan level parameters in the instance and using the same name for the parameter as that of the parent and giving it a different value.

  For example, suppose you have an execution plan that has two execution plan level parameters defined: one for currency, with the value USD; and one for region, with the value California. Then, you create a child instance of the execution plan, in which you want the currency parameter to remain as USD, but you want the value of region to become Nevada. The value of the currency parameter will be propagated to the child instance, so you do not need to modify this parameter. For the region parameter, you would need to define an execution plan level parameter for the child instance. To do so, you would use the same name for the parameter

as that of the parent execution plan (in this case, “region”), and you would give the region parameter the value Nevada.

You can also create execution plan level parameters specifically for a child instance.

To create, configure and run multiple instances of an execution plan

1. In the Execute view, go to the **Execution Plans** tab, right-click the execution plan you want to copy, and select **Create Execution Instance**.

   The Execution Instance Configuration dialog opens.

2. Enter a unique name for the instance.
3. Enter a unique instance code as an identifier. The value must be alphanumeric and from one to five characters.
4. Click **Next**.
5. The physical data source for the parent execution plan is displayed. You can optionally override this property with a different physical data source by doing the following:
   a. Click the drop-down list in the Override With field.
   b. Select the appropriate physical data source for this execution plan instance.
   c. Click **Next** to proceed.
6. The physical folder for the parent execution plan is displayed. You can optionally override this property with a different physical folder by following the substeps in step 5.
7. If a context connectivity parameter for external executors is defined for the parent execution plan, it is displayed. You can optionally override this property with a different context by following the substeps in step 5.
8. Click **Finish**.

   A message dialog indicates whether the process was successful.

9. (Optional) Go to the **Execution Instances** tab, select the appropriate instance, and configure additional properties:
   a. Configure the properties that appear in the top pane window or in the Edit subtab. For a description of these properties, see "Building and Running Execution Plans".
   b. In the Connectivity Parameters subtab, modify the parameters as needed.
   c. In the Execution Parameters subtab, do the following as needed:

      - Override one or more of the execution plan level parameters defined in the parent execution plan by creating new execution plan level parameters in the instance and using the same name for the parameter as that of the parent and giving it a different value.

      - Create new execution plan level parameters.

10. (Optional) To run the execution plan instance, do one of the following:
    - Click **Run Now** in the top pane toolbar.
    - Schedule the execution plan instance to run by following the instructions in the procedure "Scheduling an Execution Plan".

You can monitor execution plan instances the same way you monitor execution plans, using the Current Run and Run History tabs of the Execute view. See "Monitoring Execution Plan Processes" for more information.

11. To reconfigure the override settings of an instance, go to the **Execution Instances** tab, and click **Configure**.

Setting Up Extract Delays, Event Delays and Data Source Notifications

This section contains the following topics. These topics apply to multi-source execution plans.

- Setting Up Extract Delays.

  Use the Extract Delay feature to stagger the timing of data extraction from different source systems.

- Setting Up Event Delays.

  Use the Event Delay feature to configure the extracts for different source systems to occur independently.

- Setting Up Data Source Usage Notifications

  Use the Data Source Notification feature to initiate email notifications about data source usage and to define custom SQL to be executed based on the usage.

Setting Up Extract Delays

You can delay the extract of a source system in a multi-source environment by setting the Delay property in the Connectivity Parameters subtab of the Execution Plans tab. See "Building and Running Execution Plans" for instructions on setting the Delay property.

Setting Up Event Delays

The Event Delay feature applies to multi-source environments. It enables you to configure the extracts for the different data sources to occur independently.

The following are examples of how you might use the Event Delay feature:

- **In a multi-source environment, one data source is not available at the beginning of the ETL.** For example, suppose an ETL process extracts CRM data from one source system and financial data from another source system. Nightly maintenance of the CRM system is scheduled for 11:00 PM, and the system is expected to be back up at 11:30 PM. The ETL process is scheduled to begin at 12:00 AM. If the CRM system is still down at the ETL start time, the ETL process will fail immediately because the connection pool to the CRM source will not be created.

  In such situations, you can define an event delay that would allow the tasks that read and write to the financial data source to be executed while the tasks for the CRM source wait until a connection pool can be created. You define the event delay using the **Lazy Initialization** property, which prevents a connection pool from being created until the connection itself is available. When you enable Lazy Initialization, you also set the following properties:

  – **Polling Frequency** - Specifies in minutes how often DAC will try to create the connection.

  – **Timeout** - A period of time (specified in minutes) indicating how long DAC will continue to poll to see if the connection is available.

  - **Upon Timeout** - The possible action DAC will take if the Timeout period elapses without a connection being available. The options are:

    Fail - DAC will fail all of the tasks that read and write to the unavailable source. The failed tasks are marked as Stopped.

    Skip - DAC will skip the tasks that read and write to the unavailable source. The refresh dates for primary and auxiliary source and target tables will not be persisted. This allows for the next ETL process to pick up the data since the prior ETL event was not successful. The skipped tasks are marked as Completed, and the task details reflect a Not Executed status.

  In this example, the Lazy Initialization property is enabled on the CRM source. The Polling Frequency is set at one minute, and the Timeout period is set to 120 minutes. Therefore, DAC will poll the CRM source every minute for 120 minutes.

  If the connection becomes available within 120 minutes, DAC will create the connection pool and the ETL will proceed. If the Timeout period elapses, and Fail is selected for the Upon Timeout property, the tasks that read and write to the unavailable source will fail. If the Timeout period elapses, and Skip is selected, the tasks will be skipped, and the extract from the financial source will proceed.

- **Specifying a Condition Using SQL**. For example, suppose an automated process inserts and updates contact information into a CRM source, but this data load process does not complete before the scheduled ETL process. If the ETL process starts to extract data, only some of the data will be available for extraction, which will result in incomplete data loading into the data warehouse, and, consequently, inaccurate reports.

  In such cases, you can use write custom SQL to define a condition that must occur before the ETL process starts; for example, you can define a condition in which the ETL process will not start until all of the records have been inserted or updated. The process that loads the contact information can be enhanced to insert a completed status in a relational table, indicating the completion of the process. This information would then signal the ETL to start from the CRM system.

  - **Condition SQL** - A SQL that can be executed against the data source to verify if the data source is ready.
  - **Cleanup SQL** - A SQL that can be executed upon the successful occurrence of the event for the data source and upon successfully executing the Condition SQL. This property applies only if a Condition SQL is defined.

You can set up an event delay for situations similar to those described above as well as a combination of the examples described above.

To set up an event delay

1. Go to the **Physical Data Sources** tab in the Setup view.
2. In the top pane, select the physical data source for which you want to define an event delay.
3. Click the **Extended Properties** subtab.
4. Click **New** in the bottom pane toolbar.
5. Click in the **Name** field to expose the drop-down list, and select **EventBasedDelay**.
6. Click **Save** in the toolbar.
7. Double-click in the **Value** field to display the Property Value dialog.

8. Complete the fields in the Property Value dialog using the following information:

Field	Description
Lazy Initialization	When this check box is selected, DAC prevents a connection pool from being created until the connection itself is available. When you select Lazy Initialization, you then need to set the Timeout, Upon Timeout, and Polling Frequency properties.
Condition SQL	Click in the field to open a text box where you can enter a SQL statement that defines a condition that DAC will execute against the data source to verify if the data source is ready. This SQL should return a value in the form of a number. A result of 0 indicates the data source is not ready. A non-zero result means the event has happened.
Cleanup SQL	A SQL that can be executed upon the successful occurrence of the event defined in the Condition SQL. This property applies only if a Condition SQL is defined.
Timeout (min)	The period of time (specified in minutes) indicating how long DAC will continue to poll to see if the connection or source is available.
Upon Timeout	The possible action DAC will take if the Timeout period elapses without a connection being available or the source being ready. The options are: ■ **Fail**. DAC will fail all of the tasks that read and write to the unavailable source. The failed tasks are marked as Stopped. For example, using the scenario described in the introduction to this section, the financial data extract will occur and the financial-specific load tasks will run, but the CRM data extract will not occur and load tasks that are specific to both financial and CRM data will not run. ■ **Skip**. DAC will skip the tasks that read and write to the unavailable source. The refresh dates for primary and auxiliary source and target tables will not be persisted. This allows for the next ETL process to pick up the data since the prior ETL event was not successful. The skipped tasks are marked as Completed, and the task details reflect a Not Executed status. For example, using the scenario described in the introduction to this section, the tasks related to CRM and to both CRM and financial data will be skipped. Extracts and loads related to the financial source will occur.
Polling Frequency (min)	Specifies in minutes how often DAC will try to create the connection.

9. Click OK to close the Property Value dialog.

10. Click **Save** in the toolbar.

Setting Up Data Source Usage Notifications

The Data Source Usage Notification feature enables you to initiate email notifications about data source usage and to define custom SQL to be executed based on the usage.

The data source usage categories about which DAC will send notifications are the following:

- Notification Before First Read
- Notification Before First Write
- Notification Before First Read or Write
- Notification After Last Read
- Notification After Last Write
- Notification After Last Read and Write

The email notification contains the execution plan name, the ETL run name and timestamp, the data source name, and the data source usage category. The recipients of the email are the email accounts set up in the procedure "Setting Up Email Notifications in the DAC Client and Server".

When you define the data source usage notification properties, you can also define custom SQL that will be executed upon the occurrence of the specified data source usage category. In the custom SQL, you can use the DAC variable @DAC_NOTIFICATION_RECORD in a parameter.

To set up a notification

1. Go to the **Physical Data Sources** tab in the Setup view.
2. In the top pane, select the physical data source for which you want to define an event delay.
3. Click the **Extended Properties** subtab.
4. Click **New** in the bottom pane toolbar.
5. Click in the **Name** field to expose the drop-down list, and select the appropriate data source usage category.
6. Click **Save** in the toolbar.
7. Double-click in the **Value** field to display the Property Value dialog.
8. Complete the fields in the Property Value dialog using the following information:

Field	Description
Send Email	Select this check box to indicate an email should be sent when the selected operation occurs. Email recipients with a Notification Level of 5 or higher will receive the email The DAC email administrator account and email recipients must be set up before DAC can send data source usage notifications. For instructions, see "Setting Up Email Notifications in the DAC Client and Server".
Execute SQL	Click in the field to open a text box where you can enter a SQL statement that DAC will execute against the data source when the selected operation occurs.
Is Stored Procedure	Select this check box if the custom SQL is a stored procedure.

9. Click **OK** to close the Property Value dialog.
10. Click **Save** in the toolbar.

How the DAC Server Handles Requests to Start and Stop Execution Plans

Starting an Execution Plan

When the DAC Server receives a request to start an execution plan, it performs a series of checks to verify that the execution plan can be started. It first checks that an execution plan with the requested name exists and that the execution plan is active.

Next, it checks the status of the execution plan that last ran. If an execution plan is still running and the DAC Server receives a request to start the same execution plan, the request will be rejected. If an execution plan failed, a request to run the same execution plan again will be executed to restart the failed execution plan.

However, a request to run a different execution plan will be put on hold if the execution plan has dependencies on the failed execution plan (dependencies such as having the same source and target tables). If the execution plans are mutually exclusive, the new request will result in a new execution plan run.

When the DAC Server receives a request to start an execution plan, it will issue a warning if any of the following conditions are true. (A warning is for informational purposes and does not mean the execution plan will not start.)

- The Generic Task Concurrency Limit value in the DAC System Properties tab is not a positive number.
- There are no active Informatica Integration Service or Repository Service registered in the Informatica Servers tab.
- One or more Informatica Integration Services do not have the passwords defined in the Informatica Servers tab.
- One or more Informatica Integration Services do not have a Num Parallel Workflows per EP value properly defined in the Informatica Servers tab.
- One or more data sources do not have the Table Owner or Table Owner Password values properly defined in the Physical Data Sources tab.
- One or more data sources do not have a maximum number of connections (Num Connections per EP) value properly defined in the Physical Data Sources tab.
- One or more data sources do not have a Data Source Number defined in the Physical Data Sources tab.

Stopping an Execution Plan

When the DAC Server receives a request to stop the operation of a running execution plan, the request will fail in the following cases:

- The name of the execution plan that is running is different from the name in the request.
- There is no execution plan currently running.

6

Customizing ETL Processes

This chapter provides information about customizing DAC objects for inclusion in ETL processes.

This chapter contains the following topics:

Considerations When Defining Repository Objects

This chapter contains the following topics:

Note: You should be familiar with the information in "About Source System Containers" before considering best practices.

Container Behavior and Best Practices

The following behavior and best practices apply to containers:

- When changes are made to objects in the container that owns them, the change is instantaneous.
- Changes made to parent objects in the owner container are automatically pushed to the parent referenced objects.
- When you add child objects to a parent object, you must use the Push to References right-click command (Design view) to push the changes to the child referenced objects. For example, if you add a column to a table that is registered in DAC, the new column is not automatically added to the references in the other containers referencing the parent object. You must use the Push to References command to effect the column changes in the other referenced tables.
- When you delete a referenced object, only the referenced object is deleted. The original object is not deleted.
- If you delete an object from the owner container, the object is deleted as well as referenced objects in other containers. This is referred to as a **deep delete**. For example, if you delete a table from the owner container, the table and columns are deleted from the owner container and all the containers that reference this object.
- If you delete a column from the owner table, the column is deleted in all the referenced objects.
- If you delete child objects from the owner object, the referenced child objects are automatically deleted.

Task Behavior and Best Practices

The following behavior and best practices apply to tasks:

- Start your work with tasks in Informatica. After you create a workflow, do the following in the DAC Task tab:
  - Create a new task and assign it a logical (readable) name.
  - Enter the command for a full load or incremental load.

    The commands can be the same. If the Command for Incremental Load field is left blank, no action occurs for this task while in incremental mode. If the Command for Full Load field is left blank, no action occurs for this task while in full mode.
  - Ensure all the source and target tables are defined for the task.

    You can use the task synchronize functionality to import data from Informatica. You can also manually assign the source or target tables.
- Select at least one primary table because the incremental and full mode properties are determined based on the refresh dates of the primary table.
- Design tasks so that they load only one table at a time.
- Define granular tasks rather than tasks that include bigger blocks of processes. Granular tasks are more efficient and have better restartability.

- Do not truncate a table on the source system tables (for example, Oracle, Siebel or PeopleSoft sources).
- Ensure the truncate property for the target tables is set properly.
- For tables that need to get truncated regardless of the mode of the run (Full or Incremental), set the Truncate Always property to True.
- For tables that need to get incrementally loaded, set the Truncate for Full Load property to True.
- Select the Analyze Table option if the task should analyze the table. The default value for this option is True if either of the Truncate options are selected.
- Do not truncate a table more than once within the single life span of an ETL.
- If a task that writes to a target table is contingent upon another table being loaded, use conditional tables. This ensures that the task qualifies only if the conditional table is part of the subject area design.
- Assign an appropriate phase to the task. An understanding of task phases is essential to understanding ETL processes.
- If you want to force a relationship where none exists, consider using phase dependencies. For example, if you have an ETL process in which the extract facts and extract dimensions do not share any common source or target tables, but the design requires that the extract facts should run before extracting dimensions, then, for the task that extracts facts, add extract dimension as the phase that waits. For more information about phase dependencies, see "Tasks Tab: Phase Dependency Subtab".
- Ensure you do not introduce conflicting phase dependencies. This can cause the DAC Server to hang.
- If the source qualifier needs to use a data parameter, always use the DAC date parameter that can be formatted to the database-specific syntax.

Task Group Behavior and Best Practices

The following best practices apply to task groups:

- Do not create task groups unnecessarily. Doing so can adversely impact the DAC's auto-dependency functionality, which automatically orders tasks for an execution plan. Create task groups only to satisfy the points listed below.
- Avoid circular relationships among tasks if the tasks are of the same phase. For example, avoid situations in which Task 1 reads from Table A and writes to Table B, and Task 2 reads from Table B and writes to Table A. You can use task groups to avoid these situations. Note: If tasks belong to different phases, this situation is acceptable.
- If you have many tasks belonging to the same phase that write to the same table, you can use task groups to run the tasks in parallel. If the target tables need to be truncated before the tasks are run, select the properties Truncate Always and Truncate Full Load in the Task Group tab.
- Do not mix extracts and loads under a single table group.
- Do not make Task Groups for obvious ordering needs. DAC handles ordering in such cases.
- If a source system container uses a task group, make other containers that reference the task also include the task group.

Table Behavior and Best Practices

The following best practices apply to tables:

- Always use all upper case characters for table names.
- Ensure you set the Table Type property correctly in the Tables tab of the Design view.
- For Teradata databases, pay attention to the Set/Multiset property. If you anticipate that the data will contain duplicate rows, choose Multiset as the value for this property.
- DAC automatically associates foreign key tables with the referenced table. You can also define which other tables need to be loaded, such as aggregate tables, by associating these tables with the referenced table using the Related Tables subtab of the Tables tab.

Index Behavior and Best Practices

The following best practices apply to indexes:

- Index names must be unique. Oracle, DB2 and Teradata databases enforce this rule by default. However, SQL Server databases do not enforce this rule. If you are using a SQL Server database, you must ensure all index names are unique.
- Always use all upper case characters for column names.
- If you have a foreign key column, associate the foreign key table and the join column. DAC uses this information to identify all the related tables to be loaded when a certain table needs to be loaded in the data warehouse.
- Do not register any columns for source system container tables.
- Ensure you add all the appropriate system columns. For example, all tables should have the following:
  - ROW_WID in number format.
  - INTEGRATION_ID in varchar format.
  - DATASOURCE_NUM_ID in number format.
- For Teradata databases:
  - Pay attention to the Teradata Primary Index property.
  - Pay attention to which columns need to gather statistics. Note that column statistics are somewhat equivalent to indexes.
  - If you would have had indexes that span multiple columns for other databases, consider defining multi-column statistics for Teradata.

Column Behavior and Best Practices

The following best practices apply to columns:

- Always use all upper case characters for table names.
- Ensure you set the Table Type property correctly in the Tables tab of the Design view.
- Always use all upper case characters for column names.

- If you have a foreign key column, associate the foreign key table with the join column. DAC uses this information to identify all the related tables to be loaded when a certain table needs to be loaded in the data warehouse.
- For Teradata databases:
  - Pay attention to which columns need to gather statistics. Note that column statistics are somewhat equivalent to indexes.
  - If you would have had indexes that span multiple columns for other databases, consider defining multi-column statistics for Teradata.
  - Pay attention to the Teradata Primary Index property.
- Do not register any columns for source system container tables.
- Ensure you add all the appropriate system columns. For example, all tables should have the following:
  - ROW_WID in number format.
  - INTEGRATION_ID in varchar format.
  - DATASOURCE_NUM_ID in number format.
  - ETL_PROC_WID in number format.

Configuration Tag Behavior and Best Practices

The following best practices apply to configuration tags:

- Use configuration tags to tag tasks that you do not want to be part of all the defined subject areas.
- A tagged task can be re-associated with a subject area by assigning the configuration tag to the subject area.

Source System Parameter Behavior and Best Practices

The following best practices apply to source system parameters:

- Use source system parameters when tasks in the source system container need a particular value.

Subject Area Behavior and Best Practices

The following best practices apply to subject areas:

- To define a subject area, associate only fact tables with it. DAC automatically computes which additional aggregate tables and dimension tables to associate with the subject area based on the related tables you define and foreign key relationships.
- If you delete a task from a subject area using the Delete button on the Task tab, the next time you assemble the subject area the task may be included. However, if you inactivate the task by selecting Inactive in the Task tab, the task will remain inactive when you re-assemble the subject area.
- Avoid adding tasks or inactivating tasks manually.

Execution Plan Behavior and Best Practices

The following best practices apply to execution plans:

- If many tasks with the same name across source system containers read and write to the same data source, DAC will consider them to be the same task.
- If the logical source to physical mapping yields multiple records, DAC will produce as many runtime instances of the task.
- DAC orders tasks in the most efficient manner possible based on the following:
  - Phase of task
  - Source and target tables
  - Truncate table properties
  - Data source priority

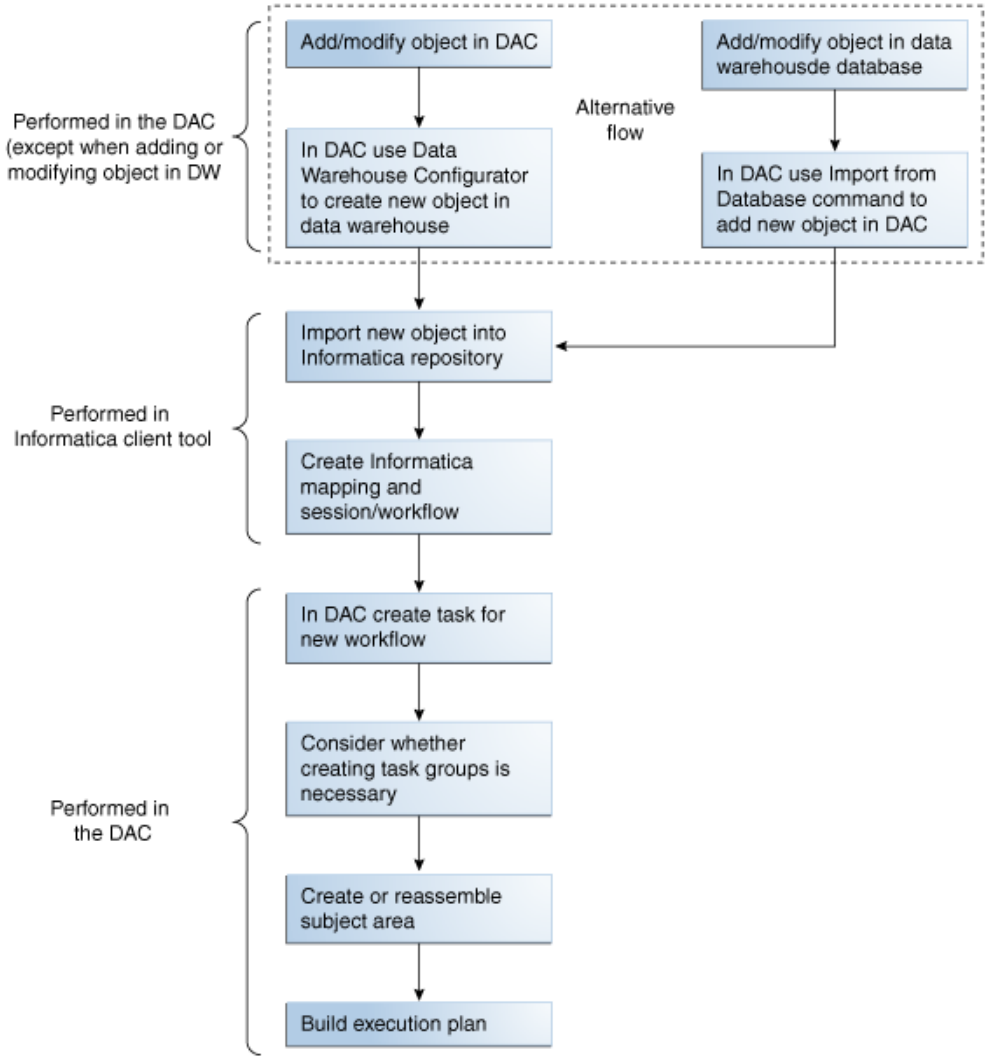
About Customizing the Data Warehouse

Note: You cannot change the metadata for predefined containers. In order to customize the metadata in a predefined container, you must first make a copy of the container. For instructions, see "Creating or Copying a Source System Container".

You can add tables, columns, and indexes to the data warehouse, and you can modify the existing objects. Customizing the data warehouse in this way requires using DAC and Informatica client tools.

Figure 6–1 shows the major steps required for adding a new object to the data warehouse or modifying existing objects. As shown in Figure 6–1, you can begin the customization process by adding or modifying the new data warehouse object in DAC and then using the DAC's schema creation and upgrade functionality to propagate the changes to the data warehouse. Alternatively, you can add or modify the object directly in the data warehouse database and then use the DAC's Import from Database command to add the new object in DAC.

Figure 6–1 Process Flow to Add New Object to Data Warehouse



Adding a New Table and Columns to the Data Warehouse

As shown in Figure 6–1, there are two alternative process flows for adding a new object to the data warehouse. You can enter the table and column definitions in DAC and then use the DAC's schema creation and upgrade functionality to propagate the changes to the data warehouse database. For this method, follow the procedure, "To add new tables and columns to DAC and propagate changes to the data warehouse".

Alternatively, you can add the new table and column definitions directly in the data warehouse database and then use the DAC's Import from Database command to add the new table and columns in DAC. For this method, follow the procedure, "To add a new table and columns using the DAC's Import command".

To add new tables and columns to DAC and propagate changes to the data warehouse

1. In the Design view, select the appropriate source system container from the drop-down list.

2. In the Tables tab, create the new table:
   a. Click **New**.
   b. In the Edit subtab, enter the appropriate information about the table, and click **Save**.

      For a description of the fields in this tab, see "Tables Tab".
3. Add the columns for the new table:
   a. In the Columns subtab, click **New**.
   b. Enter the appropriate column information for each column you want to add to the table, and click **Save**.
   c. Enter the appropriate foreign key table and column information.

      Note: For performance purposes, it is recommended that you do not enter more than 254 columns to a dimension or fact table.
4. Create the new tables and columns in the data warehouse database.

   DAC provides several ways to create and upgrade data warehouse schemas. Carefully review Chapter 9, "Managing Data Warehouse Schemas," and select the method that best suits your requirements.

To add a new table and columns using the DAC's Import command

1. Add the new table and column definitions into the data warehouse database.
2. In the DAC Design view, select the appropriate source system container from the drop-down list.
3. Import the new table definition.
   a. In the Tables tab, right-click and select **Import from Database**, **Import Database Tables**.
   b. In the Import Tables dialog, select **DataWarehouse**.
   c. Optionally, enter filter criteria to identity the table name you entered in Step 1.

      See "DAC Query Commands and Operators" for available filter commands and operators.
   d. Click **Read Tables**.
   e. In the list of tables displayed, select the **Import** check box for the tables you want to import.
   f. Click **Import Tables**.

      A message dialog indicates whether the process was successful.
4. Import the new column definitions.
   a. In the Tables tab, query for the table you imported in Step 3.
   b. With the table highlighted, right-click and select **Import from Database**, **Import Database Columns**.
   c. In the Importing Columns... dialog, select **Selected Record Only**, and then click **OK**.
   d. In the Import Table Columns dialog, click **Read Columns**.

The Changes column displays a description of column changes, which are explained below:

Change	Explanation
The object was added to the database.	The column is in the database but not the DAC repository. Importing it will add the column to the DAC repository.
The object was added to the repository.	The column is in the DAC repository but not in the database. Importing it will delete it from the DAC repository.
The object was modified.	The column definition in the database does not match the definition in the DAC repository.

e. In the list of columns displayed, select the **Import** check box for the columns you want to import.

f. Click **Import Columns**.

A message dialog indicates whether the process was successful.

Adding an Index to the Data Warehouse

Follow this procedure to add a new index to the data warehouse.

To add a new index to the data warehouse

1. Add the new index definition into the data warehouse database.
2. In the DAC Design view, select the appropriate source system container from the drop-down list.
3. In the Tables tab, query for the table for which you want to import index definitions.
4. Right-click and select **Import from Database**, **Import Indices**.
5. Choose to import indexes for a selected table or for all the records retrieved in the query, and click **OK**.
6. In the Import Indices dialog, select the appropriate data warehouse from the Data Sources drop-down list.
7. Click **Read Indices**.
   a. In the list of indexes displayed, select the **Import** check box for the indexes you want to import.
   b. Click **Import Indices**.

   A message dialog indicates whether the process was successful.

Importing New Data Warehouse Objects into the Informatica Repository

This step requires using Informatica client tools to import new data warehouse objects into the Informatica repository. For instructions on this step of customizing the data warehouse, see the Informatica documentation.

Creating Informatica Mappings and Workflows

The process of creating Informatica mappings and workflows for the data warehouse objects that you imported into the Informatica repository requires using Informatica client tools. For instructions on this process, see the Informatica documentation.

Creating Tasks in DAC for New or Modified Informatica Workflows

You need to perform this step for all new workflows you create in Informatica and for all workflows that you modify.

To create a task in the DAC for new or modified Informatica workflows

1. In the DAC Design view, select the appropriate source system container from the drop-down list.
2. Create a custom logical folder for the custom folder you created in the Informatica repository.
   a. On the **Tools menu**, select **Seed Data**, and then select **Task Logical Folders**.
   b. To create a custom logical folder, click **New**.
   c. Enter a name for the custom logical folder, and click **Save**.
   d. In the Type field, select **Logical**.
3. Create a custom physical folder for the custom folder you created in the Informatica repository.
   a. On the **Tools** menu, select **Seed Data**, and then select **Task Physical Folders**.
   b. To create a custom physical folder, click **New**.
   c. Enter a name for the custom physical folder.
   d. Enter a value for the **Instance Priority**.

      See "Task Physical Folder Instance Priority" for information about this property.
4. Register the custom logical and physical folders you created in steps 2 and 3 in the Source System Folders tab.
   a. In the Design view, select the **Source System Folders** tab.
   b. Click **New**.
   c. In the Edit subtab, enter the name of the custom logical folder in the Logical Folder field.
   d. Enter the name of the custom physical folder in the Physical Folder field, and click Save.
5. Create new tasks for the workflows.
   a. In the Design view, select **Tasks**, and click **New** in the top pane toolbar.
   b. In the Edit subtab, enter the workflow name as it appears in Informatica Workflow Manager.
   c. Right-click, and select **Synchronize Tasks**.
   d. Select **Selected Record Only**, and click **OK**.

      This command imports the source and target table definitions for a task from Informatica into the DAC repository.

e. In the Tasks tab, enter the remaining information required for the task.

   For a description of the fields in this tab, see "Tasks Tab".

The new table is now ready to be associated with a subject area. For information about creating a subject area, see "Creating a Subject Area".

Setting a Task Phase Dependency

A task phase dependency enables you to dictate additional dependency hints for tasks that are completely unrelated to each other through source/target table relationships.

To set a task phase dependency

1. In the Design view, select the appropriate source system container from the drop-down list.
2. In the Tasks tab, query for the task for which you want to add a phase dependency, and make sure it is highlighted.
3. Click the **Phase Dependency** subtab.
4. Click **Add/Remove** in the subtab toolbar.
5. In the Choose Phases dialog, query for a task phase, select the phase, and click **Add**.
6. Click **OK** in the message box that states the phase was added.
7. In the window on the right side of the dialog, select the appropriate values for the following properties:

Property	Description
Action	Possible values: ▪ **Wait.** Indicates the task selected in the top pane will wait to be executed until all tasks of the phase specified in the Phase column have been executed. ▪ **Block.** Indicates the task selected in the top pane will block all tasks of the phase specified in the Phase column from being executed.
Grain	Possible values: ▪ **All**. Indicates the action will affect all tasks. ▪ **Related**. Indicates the action will affect related tasks.

Property	Description
Scope	Specifies how the Block action of the phase dependency behaves in relation to multi-source execution plans. Possible values: ▪ **Both.** Indicates the blocking action is active for tasks that have the same source and target physical data source connections ▪ **Source**. Indicates the blocking action is active for tasks that have the same source physical data source connection. ▪ **Target**. Indicates the blocking action is active for tasks that have the same target physical data source connection. ▪ **None. Indicates the blocking action is active for all tasks regardless of the source and target physical data source connections.**

The task phase dependency is displayed in the Phase Dependency subtab.

8. Reassemble the appropriate subject area.
   a. In the Subject Areas tab, query for the appropriate subject area.
   b. Click **Assemble**.
9. Rebuild the execution plan.
   a. In the Execute view, query for the appropriate execution plan.
   b. In the Connectivity Parameters subtab, click **Generate**.

      The Generating Parameters... dialog lists the containers that are involved in this execution plan.
   c. Enter the number of copies of each container that are needed, and then click **OK**.

      DAC automatically generates the parameters required for each copy of the source system container. Note that not all copies require all of the possible parameters.
   d. In the Value column, for each folder or database connection, select the appropriate physical folder name or database connection.

      Note: For the data source type of FlatFileConnection, make sure you have copied all files into the directory specified in the DAC system property InformaticaParameterFileLocation.

   e. In the top pane toolbar, click **Build**.

Creating a Task Group

DAC automatically organizes tasks into a dependency structure based on dependency rules. For information about the DAC's dependency rules, see "Execution Plan Build Process Rules".

DAC does not allow parallel reads and writes on the same table; therefore, DAC randomly assigns priorities to tasks that have the same properties. You can group tasks by using the Task Group feature to enforce a particular behavior.

The task group feature can be useful in the following situations:

- You want to enforce a particular order for multiple tasks writing to the same table.
- You want tasks to run in parallel (because you know the tasks have no dependencies among them).
- You want to create a task action to truncate a table when you have multiple tasks writing to the same table.
- You have circular tables, and need to enforce an order. For example, task 1 reads from table A and writes to table B, and task 2 reads from table B and writes to table A.

To create a task group

1. In the Design view, select the appropriate source system container from the drop-down list.
2. In the Task Groups tab, click **New** in the top pane toolbar.
3. In the Edit subtab, enter a name and select the appropriate properties.
4. Click the Child Tasks subtab, and click **Add/Remove** in the toolbar.
5. In the left-hand window of the Choose Child Tasks dialog, query for the tasks you want to add to the task group.
6. Select the tasks, and click **Add**.
7. In the window on the right side, enter a value in the Dependency Order field to specify an execution order.
8. Specify whether the task is a **Heuristic Driver**.

   See "DAC Heuristics and Task Groups" for more information.
9. Click **Save**, and then click **OK** to close the window.

Working with Configuration Tags

A configuration tag is an object that controls the inclusion of tasks in subject areas. When a task is tagged, it is not eligible to be included in the collection of tasks for a subject area. You can override this behavior by using the "Include Task" property in the subject area definition.

A configuration tag can function in one of the following ways:

- **Remove tasks from all subject areas**

  If you assign a task to a configuration tag, the task will not be eligible to participate in *any* subject area. For instructions, see "To remove tasks from all subject areas".

- **Reassign autogenerated tasks to a specific subject area**

  An autogenerated task is a task that the DAC automatically assigns to a subject area when the subject area is assembled.

  For autogenerated tasks that were removed from participating in a subject area, you can set up the configuration tag to reassign a task to participate in specific subject areas. You do this by associating the configuration tag with the desired

subject area. This method only applies to tasks that are autogenerated tasks of a subject area. For instructions, see "To reassign autogenerated tasks to a subject area".

- **Add non-autogenerated tasks to a subject area**

  You can set up a configuration tag to add non-autogenerated tasks to a subject area. The non-autogenerated tasks will participate in the subject area along with the subject area's autogenerated tasks. For instructions, see "To add non-autogenerated tasks to a subject area".

- **Assign only configuration tag tasks to a subject area (excludes the subject area's autogenerated tasks)**

  You can also set up a configuration tag so that *only* tasks that were assigned to the configuration tag participate in a specific subject area. In this case, the subject area's autogenerated tasks do not participate. For instructions, see "To assign only configuration tag tasks to a subject area (excludes the subject area's autogenerated tasks)"

To remove tasks from all subject areas

1. In the Design view, select the appropriate source system container from the drop-down list.
2. Create a new configuration tag.
   a. In the Configuration Tags tab, click **New** in the top pane toolbar.
   b. In the Edit subtab, enter a name for the configuration tag.
   c. Make sure the **Include Tasks** check box is not selected.
   d. Click **Save**.
3. Add tasks to the configuration tag.
   a. With the new configuration tag highlighted in the top pane, click the **Tasks** subtab.
   b. In the bottom pane toolbar, click **Add/Remove**.
   c. In the Tasks dialog, query for the tasks you want to add to the configuration tag.
   d. Highlight the tasks, and then click **Add**.

      The tasks appear in the right-hand window.
   e. Click **Save**, and then click **OK** to close the window.

      These tasks will not be eligible to participate in any subject area.

To reassign autogenerated tasks to a subject area

1. In Design view, select the appropriate source system container from the drop-down list.
2. In the Configuration Tags tab, query for the configuration tag that contains the tasks you want to reassign to a subject area.
3. Verify the configuration tag contains the appropriate tasks by clicking the Tasks subtab and reviewing the list of tasks associated with this configuration tag.

Note: Only a subject area's autogenerated tasks will be reassigned. If non-autogenerated tasks appear in the list, the DAC will ignore them.

---

4. Associate the configuration tag with the subject areas to which you want to reassign the tasks.
   a. With the configuration tag highlighted in the top pane, click the **Subject Areas** subtab.
   b. Click **Add/Remove** in the bottom pane toolbar.
   c. In the Subject Areas dialog, query for one or more subject areas to which you want to reassign the task or tasks.
   d. Highlight the appropriate subject areas, and click **Add**.
   e. Click **Save**, and then click **OK** to close the window.
5. Reassemble the subject area.
   a. In the Subject Area tab, query for all the subjects areas you added to the configuration tag.
   b. Highlight the subject areas, and click **Assemble**.

To add non-autogenerated tasks to a subject area

1. In the Design view, select the appropriate source system container from the drop-down list.
2. Create a new configuration tag.
   a. In the Configuration Tags tab, click **New** in the top pane toolbar.
   b. In the Edit subtab, enter a name for the configuration tag.
   c. Select the I**nclude Tasks** check box.
   d. Click **Save**.
3. Add the non-autogenerated tasks to the configuration tag.
   a. With the new configuration tag highlighted in the top pane, click the **Tasks** subtab.
   b. In the bottom pane toolbar, click **Add/Remove**.
   c. In the Tasks dialog, query for the extraneous tasks you want to add to the configuration tag.
   d. Highlight the tasks, and then click **Add**.
   e. Click **Save**, and then click **OK** to close the window.
4. Associate the configuration tag with the subject areas to which you want to add the non-autogenerated tasks.
   a. With the configuration tag highlighted in the top pane, click the **Subject Areas** subtab.
   b. Click **Add/Remove** in the bottom pane toolbar.
   c. In the Subject Areas dialog, query for one or more subject areas to which you want to add the non-autogenerated tasks.
   d. Highlight the appropriate subject areas, and click **Add**.
   e. Click **Save**, and then click **OK** to close the window.
5. Reassemble the subject area.
   a. In the Subject Area tab, query for all the subjects areas you added to the configuration tag.

b. Highlight the subject areas, and click **Assemble**.

To assign only configuration tag tasks to a subject area (excludes the subject area's autogenerated tasks)

1. In the Design view, select the appropriate source system container from the drop-down list.
2. In the Subject Areas tab, query for the subject area to which you want to add configuration tag tasks.

   Note: The autogenerated tasks for this subject area will be excluded.

3. Select the **Configuration Tag Tasks Only** check box, and click **Save**.
4. Create a configuration tag.
   a. In the Configuration Tags tab, click **New** in the top pane toolbar.
   b. In the Edit subtab, enter a name for the configuration tag.
   c. Select the **Include Tasks** check box.
   d. Click **Save.**
5. Add the tasks to the configuration tag.
   a. With the new configuration tag highlighted in the top pane, click the **Tasks** subtab.
   b. In the bottom pane toolbar, click Edit.
   c. In the Tasks dialog, query for the tasks you want to add to the configuration tag.
   d. Highlight the tasks, and then click Add.
   e. Click Save, and then click OK to close the window.
6. Associate the configuration tag with the subject area.
   a. With the configuration tag highlighted in the top pane, click the **Subject Areas** subtab.
   b. Click **Add/Remove** in the bottom pane toolbar.
   c. In the Subject Areas dialog, query for the appropriate subject area.
   d. Highlight the subject area, and click **Add**.
   e. Click **Save**, and then click **OK** to close the window.
7. Reassemble the subject area.
   a. In the Subject Area tab, query for all the subjects areas you added to the configuration tag.
   b. Highlight the subject areas, and click **Assemble**.

Using Actions to Manage Indexes, Tables and Tasks

The Actions feature enables you to define and run SQL scripts to carry out various actions in relation to indexes, tables, and tasks.

- **Index actions** can override the default behavior for dropping and creating indexes by mode type (full load, incremental load, or both). The Index action types are Create Index and Drop Index.

  For example, you can define an index action to create indexes for tasks with the commands defined for full loads, or incremental loads, or both types, or you can define an action to drop indexes at a certain time on specific tables. Index actions override all other index properties.

- **Table actions** can override the default behavior for truncating and analyzing tables by mode type. The Table action types are Truncate Table and Analyze Table.

  For example, you can define a table action to truncate tables with the commands defined for full loads, or incremental loads, or both. Table actions override all other table properties.

- **Task actions** can add new functionality based on various task behaviors. The following task action types are available:

  - **Preceding Action**. Executes a SQL script before a task runs.
  - **Success Action**. Executes a SQL script after a task runs successfully.
  - **Failure Action**. Executes a SQL script if a task fails during its execution.
  - **Upon Failure Restart**. Executes a SQL script when a task that previously failed is restarted.

You can also use the Actions Template feature to:

- Combine SQL templates to do synchronized actions, such as create and analyze indexes.
- Combine object level properties with user-defined parameters in SQL statements and stored procedures.

This chapter contains the following topics:

Defining a SQL Script for an Action

The first step in the process of creating an action is to define the SQL script for the action. After completing this procedure, proceed to "Assigning an Action to a Repository Object".

To define a SQL statement for an action

1. In the DAC Client, on the **Tools** menu, select **Seed Data**, and then select one of the following:

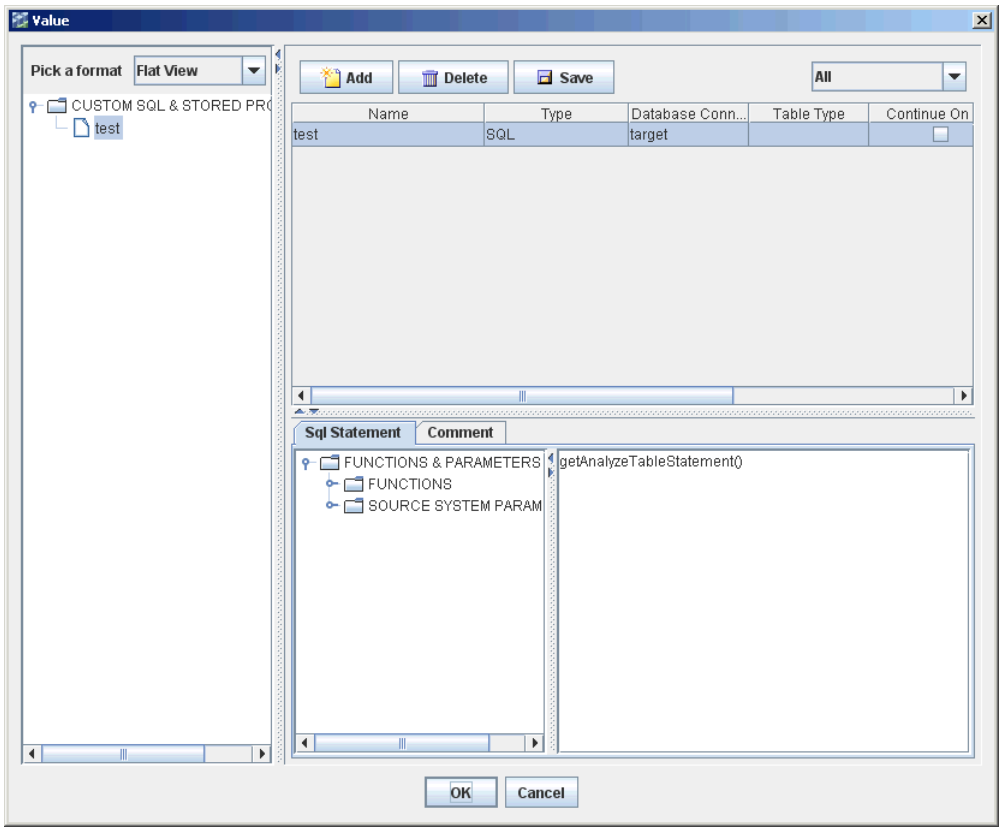
   - **Index Actions**
   - **Table Actions**
   - **Task Actions**

   The Actions dialog is displayed.

2. In the toolbar, click **New**.

3. In the new record field, enter a descriptive name for the action, and then click **Save**.
4. Double-click in the **Value** field.

   The Value dialog is displayed.



5. Select a format for the tree view.
   - **Flat View** displays the SQL entries in a list format in their order of execution.
   - **Category View** displays the entries by the categories SQL and Stored Procedure.

     You can reorder the entries in the tree by dragging and dropping them.
6. Click **Add**.
7. In the new record field, enter or select the appropriate information.

Field	Description
Name	Logical name for the SQL block.
Type	SQL or Stored procedure

Field	Description
Database Connection Type	Should be used only for SQL types (not stored procedures). Defines which database the SQL statement will run against. Possible values are: **Source** - SQL runs against the source connection defined for the task. **Target** - SQL runs against the source connection defined for the task. **Both** - SQL runs against both the source and target connection. **Table Connection** - SQL runs against the table-specific connection if a separate table connection is available.
Table Type	Specifies the table type against which the SQL will run. Possible values are: **All Source** - SQL runs against all source tables defined for the task. **All Target** - SQL runs against all target tables defined for the task. **Source Lookup** - SQL runs against all the source lookup tables defined for the task. **Source Primary** - SQL runs against all the source primary tables defined for the task. **Source Auxiliary** - SQL runs against all the source auxiliary tables defined for the task.
Continue on Fail	Specifies whether an execution should proceed if a given SQL block fails.
Retries	Specifies how many retries are allowed. If the number is not positive, a default number of one (1) will be used.
Valid Database Platforms	Specifies the valid database platforms against which the SQL will run. If this field is left empty, the SQL can be run against any database.

8. In the lower-right side text box, enter a SQL statement.

   The SQL Statement tab to the left of the text box lists all the supported SQL functions and DAC source system parameters that you can use in constructing custom SQLs. Double-click a function or source system parameter to move it into the text box.

   For a description of the available functions, see "Functions for Use with Actions".

   The source systems parameters list contains the names of all source system parameters defined in the DAC repository, with the prefix @DAC_. During runtime, the DAC Server resolves the source system parameter and replaces its name with the runtime value.

   For an example of how to use a source system parameter in a SQL statement, see "Using a DAC Source System Parameter in an Action".

9. (Optional) Enter a comment about the SQL in the Comment tab.

10. Click **OK**.

> **Note:** You can add multiple SQL statements and stored procedures to a single action.

11. To assign this action to a repository object, proceed to "Assigning an Action to a Repository Object".

Assigning an Action to a Repository Object

Follow this procedure to assign an action to a DAC repository object. Before you do this procedure, you must have defined a SQL script for an action. For instructions, see "Defining a SQL Script for an Action".

To assign an action to a repository object

1. In the Design view, navigate to one of the following tabs, depending on the object type for which you want to assign an action:
   - Indices tab
   - Tables tab
   - Tasks tab
2. Select or query for the object for which you want to assign an action.
3. With the appropriate object selected in the top pane, select the **Actions** subtab.
4. Click **New** in the subtab toolbar.
5. In the new record field, do the following:
   a. Select an **Action Type**.

      For a description of the available Action Types, see the following: "Indices Tab: Actions Subtab", "Tables Tab: Actions Subtab", and "Tasks Tab: Actions Subtab".

   b. Select the **Load Type**:

      Full - To assign the action to a full load command.

      Incremental - To assign the action to an incremental load command.

      Both - To assign the action to both full and incremental load commands.

   c. Double-click in the **Action** field to open the Choose Action dialog, and select an action.

   d. Click **OK** to close the Choose Action dialog.

   e. Click **Save** in the subtab toolbar.

Functions for Use with Actions

This section includes a list of functions that are available for Index, Table and Task actions.

- Table 6–1, " Functions for Index Actions"
- Table 6–2, " Functions for Table Actions"
- Table 6–3, " Functions for Task Actions"

Table 6–1 Functions for Index Actions

Function	Description
getAdditionalColumns()	Returns a comma separated list of included index columns. This function is related to the #Unique Columns index property. For DB2 and DB2-390 databases, the list may include a combination of unique and included columns.
getAnalyzeStatement()	Returns the default DAC index analyze statement (for this particular index).
getAnalyzeTableStatement()	Returns the default DAC table analyze statement (for the parent table).
getBitMapString	Resolves to the string `BITMAP` if it is a bitmap index. Otherwise, the string is empty.
getClusteredString	Resolves to the string `CLUTSTERED` if it is a bitmap index. Otherwise, the string is empty.
getCreateIndexStatement()	Returns the default DAC index creation statement.
getDBType()	Returns the physical data source connection type (Oracle OCI8, Oracle Thin, DB2, DB2-390, MSSQL, Teradata, or BI Server).
getDropIndexStatement()	Returns the default DAC index drop statement.
getHashString()	Resolves to the string `HASH` if it is a hash index. Otherwise, the string is empty.
getImageSuffix()	Resolves to the table image suffix if one is specified. Otherwise, the string is empty.
getIndexColumns()	Returns a comma separated list of index columns.
getIndexName()	Returns the index name.
getIndexTableSpace()	Resolves to the index space name if one exists. Otherwise, the string is empty.
getNamedSource()	Returns the DAC physical connection name.
getRvrsScanString()	Resolves to the string `ALLOW REVERSE SCANS` if the index supports reverse scans. Otherwise, the string is empty.
getTableName()	Returns the table name.
getTableOwner()	Returns the table owner name.
getTableSpace()	Returns the table space name if one exists. Otherwise, the string is empty.
getTruncateTableStatement()	Returns the default DAC table truncate statement.
getUniqueColumns()	Returns a comma separated list of unique index columns. This function is a counterpart of the getAdditionalColumns() function.
getUniqueString()	Resolves to the string `UNIQUE` if the index is unique. Otherwise, the string is empty.

Table 6–2 Functions for Table Actions

Function	Description
getAnalyzeTableStatement()	Returns the default DAC table Analyze statement.
getDBType()	Returns the physical data source connection type (Oracle OCI8, Oracle Thin, DB2, DB2-390, MSSQL, Teradata or BI Server).

Table 6–2 (Cont.) Functions for Table Actions

Function	Description
getImageSuffix()	Returns the table image suffix if one exists. Otherwise, the string is empty.
getCreateIndexStatement()	Returns the default DAC index creation statement.
getNamedSource()	Returns the DAC physical connection name.
getDropIndexStatement()	Returns the default DAC index drop statement.
getTableName()	Returns the table name.
getTableOwnerName()	Returns the table owner.
getTableSpace()	Returns the table space name if one exists. Otherwise, the string is empty.
getTruncateTableStatement()	Returns the default DAC table truncate statement.

Table 6–3 Functions for Task Actions

Function	Description
getAnalyzeTableStatement()	Returns the default DAC analyze table statement.
getDBType()	Returns the physical data source connection type (Oracle OCI8, Oracle Thin, DB2, DB2-390, MSSQL, Teradata, or BI Server).
getImageSuffix()	Returns the table image suffix if one exists. Otherwise, the string is empty.
getNamedSource()	Returns the physical connection name in DAC.
getTableName()	Returns the table name.
getTableOwner()	Returns the table owner.
getTableSpace()	Returns the table space name if one exists. Otherwise, the string is empty.
getTruncateTableStatement()	Returns the default DAC truncate table statement.

Note: Table-related task action functions should be used only for SQL blocks with a specified table type. For these blocks, DAC will loop through the tables of the type you specify and execute custom SQL for each table. Functions will be substituted with table-specific values for each iteration.

For example, if you wanted to gather statistics in a particular way after creating specific indexes, you would need to create index actions with two SQL blocks:

```
getCreateIndexStatement()

begin
DBMS_STATS.GATHER_INDEX_STATS(ownname => '@TABLEOWNER',
indname => 'getIndexName()', estimate_percent => 50);
end;
```

Using a DAC Source System Parameter in an Action

The following example illustrates how to use a source system parameter in defining an action.

Assume there is a source system parameter called *COUNTRY* (note that source system parameter names are case sensitive). And, you use this parameter in an action using the following SQL statement:

```
DELETE FROM TABLE1 WHERE COUNRY_NAME='@DAC_COUNTRY'
```

Assume that during the ETL process, `COUNTRY` gets resolved to Canada. The resulting SQL that is executed by the DAC Server would be the following:

```
DELETE FROM TABLE1 WHERE COUNTRY_NAME='Canada'
```

Using a Task Action to Enable Failure Restarts When Extracting From Multiple Sources

Typically, the extract tasks truncate the table, and, therefore, will not have update strategies. To enable failure restarts when extracting from multiple sources, create an action that will predelete all the records that are loaded from the source. For example, you can create a task action for predeleting records called "Predelete Staging Data" using the following SQL in the action definition. This example assumes the data from that source is always loaded with a certain data source number and is defined as a parameter in DAC called $$DATASOURCE_NUM_ID.

```
delete from getTargetTableName()
where datasource_num_id = @DAC_$$DATASOURCE_NUM_ID
```

Associate this action for all the extract tasks from all the sources using the action types "Preceding Action" and "Upon Failure Restart Action." This will ensure that the task will, if restarted from failure, always predelete the records pertinent to that data source before running the command. You can also mass associate such actions to all extract tasks using the right-click command Add Actions.

Mapping Multiple Database-Specific Informatica Workflows to the Same DAC Task

You can map multiple, database-specific Informatica workflows to the same DAC task by parameterizing the Informatica workflow command. At runtime, DAC determines which workflow to run based on the parameterization.

This procedure uses SIL_PersonDimension_Full as an example of a full command and SIL_PersonDimension as an example of an incremental command on an Oracle database and SIL_PersonDimension_Full_TD and SIL_PersonDimension_TD as full and incremental commands, respectively, on a Teradata database.

To map multiple database-specific Informatica workflows to the same DAC task

1. In the DAC Design view, go to the Tasks tab.
2. Query for the task to which you want add multiple workflows.
3. Select the task, and then click the **Parameters** subtab.
4. Create a new parameter for a full load command:
   a. Click **New** in the subtab toolbar.
   b. In the Name field, enter **$$workflow_CMD_PARAMETER**.
   c. In the Data Type field, select **DB Specific Text**.

d. In the Load Type field, select **Full**.

e. Click in the Value field to open the Enter Parameter Value dialog.

f. In the Connection Type field, select **@DAC_TARGET_DBTYPE**.

g. In the appropriate database fields, enter the full command name for both database types.

   For example, enter SIL_PersonDimension_Full in the Oracle field and SIL_PersonDimension_Full_TD in the Teradata field.

5. Create a new parameter for an incremental load command:

   a. Click **New** in the subtab toolbar.

   b. In the Name field, enter **$$workflow_CMD_PARAMETER**.

   c. In the Data Type field, select **DB Specific Text**.

   d. In the Load Type field, select **Incremental**.

   e. Click in the Value field to open the Enter Parameter Value dialog.

   f. In the Connection Type field, select **@DAC_TARGET_DBTYPE**.

   g. In the appropriate database fields, enter the incremental command name for both database types.

      For example, enter SIL_PersonDimension in the Oracle field and SIL_PersonDimension_TD in the Teradata field.

6. With the same task selected, click the **Edit** subtab.

7. In the Command for Incremental Load field, enter **@DAC_$$workflow_CMD_PARAMETER**.

8. In the Command for Full Load field, enter **@DAC_$$workflow_CMD_PARAMETER**.

9. Click **Save**.

7

Defining and Managing Parameters

This chapter explains how parameters are used in DAC as well as instructions for defining and managing parameters.

This chapter contains the following topics:

Overview of Parameters

This section contains the following topics:

What Are Parameters and How Are They Used in DAC?

In an ETL process, a parameter represents an attribute that is not hard coded or sourced from the transactional system. Parameters provide flexibility so that you can adapt ETL processes to fit your business requirements.

The following examples illustrate a few of the ways you can use parameters in ETL processes:

- A parameter is set by default in Oracle BI Applications to track changes on dimension tables. If you do not want to track changes on a particular table, you can change the parameter value "Y" (yes) to "N" (no). This is an example of a parameter that uses the text data type.
- Fact tables tend to be large. For testing purposes, in a development or QA environment, you might want to retrieve data for only one year, whereas in a production environment, you might want to retrieve data for ten years. You can define a parameter that specifies the date from which data needs to be extracted

from the transactional system and change this value when the ETL process is run in different environments. This is an example of a parameter that uses the timestamp data type.

- The ETL start time is an example of a runtime parameter that you can define at the source system, task, or execution plan level. Runtime parameters use predefined DAC variables to access attribute values that vary for different ETL runs. See "About DAC Variables" for more information.
- Performance can be enhanced by defining parameters using database-specific syntax to write query hints. DAC passes database-specific parameters to the Informatica Source Qualifier as a SQL override.

You can define static or runtime parameters in the DAC repository (using the DAC Client) to apply to source systems, tasks, or execution plans. You can also define parameters whose values come from outside of DAC, such as from external relational sources that DAC accesses through SQL interfaces, or from external technology stacks that DAC accesses through a Java API interface.

In addition, some parameters used by Oracle BI Applications are defined by the user in the Oracle BI Applications Configuration Manager. See *Oracle Fusion Middleware Installation and Configuration Guide for Oracle Business Intelligence Applications* for information about the parameters that are defined in the Configuration Manager. Also, depending on your environment, some predefined parameters used by Oracle BI Applications may be held in flat files named parameterfileDW.txt and parameterfileOLTP.txt, which are stored in the directory <Domain_Home>\dac\Informatica\parameters\input. Oracle recommends that you do not modify these parameter flat files. Instead, if you need to customize the value of the parameters, you can define new parameters in the DAC repository.

How DAC Handles Parameters at Runtime

During an ETL execution, DAC reads and evaluates *all* parameters associated with that ETL run, including static and runtime parameters defined in DAC, parameters held in flat files, and parameters defined externally to DAC. DAC consolidates all the parameters for the ETL run, deduplicates any redundant parameters, and then creates an individual parameter file for each Informatica session. This file contains the evaluated name-value pairs for all parameters, both static and runtime, for each workflow that DAC executes. The parameter file contains a section for each session under a workflow. DAC determines the sessions under a workflow during runtime by using the Informatica pmrep function ListObjectDependencies.

The naming convention for the parameter file is

```
<Informatica foldername>.<workflow instance name>.<primary
source name>.<primary target name>.txt
```

DAC writes this file to a location specified in the DAC system property InformaticaParameterFileLocation. The location specified by the property InformaticaParameterFileLocation must be the same as the location specified by the Informatica parameter property $PMSourcefileDir.

Note: Informatica Services must be configured to read parameter files from the location specified in the DAC system property InformaticaParameterFileLocation. You must set the Informatica parameter $PMSourceFileDir (accessible from Informatica Administrator) to match the location specified in InformaticaParameterFileLocation.

Types of Parameters Supported by DAC

DAC supports the following types of parameters:

- **Global External Parameters**

  Global external parameters are defined in an external technology stack outside of DAC, such as a relational database that has a thin client implementation. This type of parameter enables non-DAC users to define parameters for use in ETL processes. DAC consumes global external parameters through either a SQL interface or an external Java implementation.

  You can define global external parameters that apply to 1) all tasks in all source system containers; 2) all tasks reading from a data source; or 3) specific tasks reading from a data source.

  Global external parameters return name-value pairs of Text or Timestamp data types from source or target databases.

  You can view existing global external parameters and define new ones in the Global External Parameters dialog, which is accessible on the Tools menu, by selecting Seed Data and then External Parameters. For instructions on defining global external parameters, see "Defining a Global External Parameter".

  Java interfaces related to global external parameters are defined in dac-external-parameters.jar, which is stored in the Oracle_Home\dac\lib directory. Javadocs provide information about the interfaces and methods, and are located in the <Oracle_Home>\dac\documentation\External_Parameters\Javadocs directory.

- **Source System Parameters**

  Source system parameters apply to all tasks in a source system container. You can view existing source system parameters and define new ones in the Source System Parameters tab in the Design view.

- **Task Parameters**

  Task parameters apply to one particular task in a source system container. You can view existing task parameters and define new ones in the Parameters subtab of the Task tab in the Design view.

- **Execution Plan Parameters**

  Execution plan parameters apply to all tasks associated with a particular execution plan. You can view existing execution plan parameters and define new ones in the Execution Parameters subtab of the Execution Plans tab in the Execute view.

Rules of Precedence

DAC evaluates the hierarchy of precedence for parameters as follows:

1. Execution plan parameters take precedence over all other parameters.
2. Task parameters take precedence over source system parameters and global external parameters.
3. Source system parameters take precedence over global external parameters.

For example, if the same parameter is registered as a source system parameter and as a task parameter, DAC will evaluate the value of the task parameter. Similarly, if the same parameter is registered as a source system parameter and as a global external parameter, DAC will evaluate the value of the source system parameter.

Parameter Data Types

Parameters can have one of the following data types:

Text

Applies to source system and task parameters. The value for the parameter is defined as text. You can use the Text data type for both static and runtime parameters.

For instructions on defining a parameter using the Text data type, see "Defining a Text Type Parameter".

DB Specific Text

Applies to source system and task parameters. The value for the parameter is defined as database-specific text. This parameter should be used only if you have a heterogeneous database environment and the parameter value needs to be different for the different database types. DAC evaluates the text string based on the source or target database type. If you do not specify database-specific text, DAC returns the default value.

For instructions on defining a parameter using the DB Specific Text data type, see "Defining a Database-Specific Text Type Parameter".

Timestamp

Applies to source system and task parameters. The value for the parameter is defined as a timestamp. You can use the Timestamp data type for both static and runtime parameters. A static timestamp can be any time that is constant. A runtime timestamp parameter is a variable for which the value is supplied by DAC at runtime. You can define the timestamp in one of multiple formats or define a custom format. You can also use SQL to fetch any value that can be fired against the specified logical database connection. DAC executes the SQL against a data source that maps to the specified logical connection and then formats the resulting value in the specified format. A SQL statement specified for a given timestamp parameter can include nested DAC parameters. For information about nested parameters, see "Nesting Parameters within Other Parameters".

For instructions on defining a parameter using the Timestamp data type, see "Defining a Timestamp Type Parameter".

SQL

Applies to source system and task parameters. DAC retrieves the value for the parameter from a database using user-defined SQL.

For instructions on defining a parameter using the SQL data type, see "Defining a SQL Type Parameter".

JavaScript

Applies to source system and task parameters. DAC fetches the value for the parameter using user-defined JavaScript.

For instructions on defining a parameter using the JavaScript data type, see "Defining a JavaScript Type Parameter".

Multi-Parameter Text

Applies to source system and task parameters. The value for the parameter is defined as name-value pair strings using user-defined SQL.

For instructions on defining a parameter using the Multi-Parameter Text data type, see "Defining a Multi-Parameter Type Parameter".

Multi-Parameter Timestamp

Applies to source system and task parameters. The value for the parameter is defined as timestamps in name-value pairs using user-defined SQL.

For instructions on defining a parameter using the Multi-Parameter Timestamp data type, see "Defining a Multi-Parameter Type Parameter".

External-Parameter Text

Applies to source system parameters only. The value for the parameter is defined using a parameter producer Java API. The External-Parameter Text data type enables you to create a library and classes to implement the DACTextParameterProducer interface.

For instructions on defining a parameter using the External-Parameter Text data type, see "Defining an External Type Parameter".

External-Parameter Timestamp

Applies to source system parameters only. The value for the parameter is defined using a parameter producer Java interface. The External-Parameter Timestamp data type enables you to create a library and classes to implement the DACTimestampParameterProducer interface.

For instructions on defining a parameter using the External-Parameter Timestamp data type, see "Defining an External Type Parameter".

Global Multi-Parameter Text or Timestamp

Applies to global parameters only. The value for the parameter is defined as name-value pairs using user-defined SQL.

For instructions on defining a global parameter, see "Defining a Global External Parameter".

Static Versus Runtime Parameter Values

Source system and task parameters that have either a Text or Timestamp data type can be specified as either **static** or **runtime**. The values of static parameters are user defined and are not necessarily constant for all ETL processes. For example, static parameter values can be the result of a SQL query or the evaluation of a JavaScript. The values for runtime parameters are provided by DAC and pertain to runtime data. Such values are used in the ETL logic but cannot be predicted, such as process ID, last refresh timestamp, ETL start timestamp, and so on.

Predefined Parameters in Oracle BI Applications

Oracle BI Applications uses predefined parameters for Informatica tasks. The parameters are specific to Informatica sessions. Some of the predefined parameters are held in text files named parameterfileDW.txt and parameterfileOLTP.txt, which are stored in the directory <Domain_Home>\dac\Informatica\parameters\input. Other predefined parameters are held in DAC. The parameters held in DAC are specific to the different source system containers.

Oracle recommends that you do not modify the parameter text files. Instead, if you need to customize the value of the parameters, you can define parameters in the DAC repository at either the source system level or the task level.

Nesting Parameters within Other Parameters

You can nest any parameter definition within another parameter definition. For example, you could nest a runtime text parameter that returns the current run ID within a where clause of a SQL parameter, or you could use database specific text inside another parameter of the text or SQL data types. Parameters that are nested within other parameters must use the prefix @DAC_.

Below is an example of a text parameter that returns the current run ID placed in a where clause of a SQL parameter.

```
SELECT VALUE FROM PARAM_TEST
WHERE ROW_ID=`@DAC_p1'
```

Note: Avoid circular nesting, such as parameter A nested within parameter B, which is nested within parameter A. In such situations, DAC randomly picks one of the parameters in the circle and evaluates it as an empty string.

Parameter Load Type Property

The Load Type property enables you to define parameter values that are specific to the type of load the ETL process is carrying out, that is, a full load or an incremental load. This property can be useful for performance tuning. Note that this property does not apply to global external parameters.

About DAC Variables

DAC contains predefined variables that act as a mechanism to allow parameters to access various ETL-specific and task-specific information. These variables are available when you define a runtime parameter of the Text and Timestamp data types. When you define a runtime parameter and select a DAC variable, the variable gets wrapped inside the parameter and returns the information specified by the variable.

DAC Variables for the Text Data Type

The following variables are available when you define a runtime parameter of the Text data type.

- **@DAC_CURRENT_PROCESS_ID.** Returns the current process ID.
- **@DAC_DATASOURCE_NUM_ID**. Returns the data source number ID.
- **@DAC_DATATARGET_NUM_ID**. Returns the target database number ID.
- **@DAC_EXECUTION_PLAN_NAME**. Returns the name of the current execution plan.
- **@DAC_EXECUTION_PLAN_RUN_NAME**. Returns the run name of the current execution plan.
- **@DAC_READ_MODE**. Returns the "read mode" while running the task. The possible values are FULL and INCREMENTAL.
- **@DAC_SOURCE_DBTYPE**. Returns the task's primary source database type.

- **@DAC_SOURCE_TABLE_OWNER**. Returns the table owner of the source database.
- **@DAC_SOURCE_PRUNE_DAYS**. Returns the number of prune days for the primary source as defined in the execution plan connectivity parameters.
- **@DAC_SOURCE_PRUNE_MINUTES**. Returns the number of prune minutes for the primary source as defined in the execution plan connectivity parameters
- **@DAC_TARGET_DBTYPE**. Returns the task's primary target database type.
- **@DAC_TARGET_PRUNE_DAYS**. Returns the number of prune days for the primary target as defined in the execution plan connectivity parameters.
- **@DAC_TARGET_PRUNE_MINUTES**. Returns the number of prune minutes for the primary target as defined in the execution plan connectivity parameter.
- **@DAC_TARGET_TABLE_OWNER**. Returns the table owner of the target database.
- **@DAC_TASK_NAME**. Returns the task name of the task that is currently running.
- **@DAC_TASK_NUMBER_OF_LOOPS**. Returns the task's total number of loops as defined in the task's extended property.
- **@DAC_TASK_RUN_INSTANCE_NUMBER**. Returns the instance number of the task currently running.
- **@DAC_TASK_RUN_INSTANCE_NUMBER_DESC**. Returns the instance number of the task currently running in descending order.
- **@DAC_TASK_FULL_COMMAND**. Returns the name of the Informatica workflow for a task's full load command.
- **@DAC_TASK_INCREMENTAL_COMMAND**. Returns the name of the Informatica workflow for a task's incremental load command.
- **@DAC_WRITE_MODE**. Returns the "write mode" while running the task. The possible values are FULL and INCREMENTAL

DAC Variables for the Timestamp Data Type

The following variables are available when you define a runtime parameter of the Timestamp data type.

- **@DAC_ETL_START_TIME.** Returns the timestamp for the start time of the ETL process.
- **@DAC_ETL_START_TIME_FOR_SOURCE.** Returns the timestamp for the source database.
- **@DAC_ETL_START_TIME_FOR_TARGET**. This variable returns the timestamp for the target database.
- **@DAC_ETL_PRUNED_START_TIME.** Returns the current execution plan's actual start time minus the prune minutes.
- **@DAC_ETL_PRUNED_START_TIME_FOR_SOURCE**. Returns the current execution plan's actual start time adjusted to the source database time zone, minus the prune minutes.
- **@DAC_ETL_PRUNED_START_TIME_FOR_TARGET**. Returns the current execution plan's actual start time adjusted to the target database time zone, minus the prune minutes.

- **@DAC_CURRENT_TIMESTAMP**. Returns the current timestamp of the DAC Server.
- **@DAC_SOURCE_REFRESH_TIMESTAMP**. Returns the minimum of the task's primary or auxiliary source tables last refresh timestamp.
- **@DAC_TARGET_REFRESH_TIMESTAMP.** Returns the minimum of the task's primary or auxiliary target tables' last refresh timestamp.
- **@DAC_SOURCE_PRUNED_REFRESH_TIMESTAMP**. Returns the minimum of the task's primary or auxiliary source tables last refresh timestamp, minus the prune minutes.
- **@DAC_TARGET_PRUNED_REFRESH_TIMESTAMP**. Returns the minimum of the task's primary or auxiliary target tables last refresh timestamp, minus the prune minutes.

Defining Parameters

This section provides instructions for defining parameters based on the data type.

Defining a Text Type Parameter

Follow this procedure to define a parameter using the Text data type. This procedure applies to parameters defined at both the source system and task levels.

To define a Text data type parameter

1. Do one of the following:
   - To define a source system parameter, in the Design view, click the **Source System Parameters** tab, and then click **New** in the toolbar.
   - To define a task parameter, in the Design view, click the **Tasks** tab, then click the **Parameters** subtab, and then click **New** in the bottom pane toolbar.
2. In the new record field, enter a parameter name in the Name field.
3. Select **Text** as the Data Type.
4. Click **Save** to save the record.
5. Click in the **Value** field to open the Enter Parameter Value dialog.
6. Select one of the following options:
   - **Static.** Specifies a value that remains constant for all ETL runs.
   - **Runtime.** Specifies the value will be provided by DAC during the execution of the task.

7. If you selected the Static option, enter a text value in the text window, and click OK.
8. If you selected the Runtime option, select a **DAC Variable** from the list, and click **OK**.
9. Click **Save**.

Note: You can inactivate the parameter by selecting the Inactive check box.

Defining a Database-Specific Text Type Parameter

Follow this procedure to define a parameter using the DB Specific Text data type. This procedure applies to parameters defined at both the source system and task levels.

To define a database-specific text data type parameter

1. Do one of the following:
   - To define a source system parameter, in the Design view, click the **Source System Parameters** tab, and then click **New** in the toolbar.
   - To define a task parameter, in the Design view, click the **Tasks** tab, then click the **Parameters** subtab, and then click **New** in the bottom pane toolbar.
2. In the new record field, enter a parameter name in the Name field.
3. Select **DB Specific Text** as the Data Type.
4. Click **Save** to save the record.
5. Click in the **Value** field to open the Enter Parameter Value dialog.
6. Select one of the following Connection Type options:
   - **@DAC_SOURCE_DBTYPE.** Specifies a source database connection.
   - **@DAC_TARGET_DBTYPE**. Specifies a target database connection.
7. To define a parameter specific to all database types:
   a. Click in the **Default** field to open the Default text box.
   b. Enter the parameter definition, and click **OK**.
8. To define a parameter specific to a particular database type:
   a. Click in the appropriate database type field to open the text box.
   b. Enter the parameter definition, and click **OK**.
9. Click **OK** to close the Enter Parameter Value dialog.
10. Click **Save**.

Note: You can inactivate the parameter by selecting the Inactive check box.

Defining a Timestamp Type Parameter

Follow this procedure to define a parameter using the Timestamp data type. This procedure applies to parameters defined at both the source system and task levels.

To define a Timestamp data type parameter

1. Do one of the following:
   - To define a source system parameter, in the Design view, click the **Source System Parameters** tab, and then click New in the toolbar.

- To define a task parameter, in the Design view, click the Tasks tab, then click the Parameters subtab, and then click New in the bottom pane toolbar.

2. In the new record field, enter a parameter name in the Name field.
3. Select Timestamp as the Data Type.
4. Click Save to save the record.
5. Click in the Value field to open the Enter Parameter Value dialog.
6. Select one of the following options:
   - **Static.** Specifies a value that remains constant for all ETL runs.
   - **Runtime.** Specifies the value will be provided by DAC during the execution of the task.
   - **SQL**. Enables you to define the parameter using SQL.
7. If you selected the Static option:
   a. Click in the Date field to open the Date dialog.
   b. Enter a data and time, click OK.
8. If you selected the Runtime option:
   a. Click in the Value field to open the Enter Parameter Value dialog.
   b. Select a Variable from the list.
   c. From the Function list, select a format to which DAC will convert the date. If you select Custom, enter a custom date format.

      If you select SQL Syntax or SQL Syntax (Date Only), select a Connection Type.
9. If you selected the SQL option:
   a. Click in the SQL field to open the Enter Parameter Value dialog.
   b. Select a Logical Data Source from the list.
   c. Enter the parameter definition and click OK.
10. Click OK to close the Enter Parameter Value dialog.
11. Click Save.

Note: You can inactivate the parameter by selecting the Inactive check box.

Defining a SQL Type Parameter

Follow this procedure to define a parameter using the SQL data type. This procedure applies to parameters defined at both the source system and task levels.

To define a SQL data type parameter

1. Do one of the following:
   - To define a source system parameter, in the Design view, click the Source System Parameters tab, and then click New in the toolbar.
   - To define a task parameter, in the Design view, click the Tasks tab, then click the Parameters subtab, and then click New in the bottom pane toolbar.
2. In the new record field, enter a parameter name in the Name field.
3. Select the SQL as the Data Type.

4. Click Save to save the record.
5. Click in the Value field to open the Enter Parameter Value dialog.
6. Select a Logical Data Source.
7. Enter the parameter definition as a SQL statement, and click OK.
8. Click Save.

Note: You can inactivate the parameter by selecting the Inactive check box.

Defining an External Type Parameter

The External parameter data types enable you to define parameters using Java APIs to retrieve text or timestamp values from a source or target database. You implement the parameter producer interfaces by creating libraries and classes. During an ETL process, parameters received from the API are added to the DAC metadata.

Note: Producer code examples are available in the DAC\documentation\public directory.

Text Interface

The External-Parameter Text data type uses the DACTextParameterProducer interface to return string values. The interface maps data source number IDs, database types, JDBC connections, and initialization parameters.

The DACTextParameterProducer interface is as follows:

```
public interface DACTextParameterProducer {

     /**
      * Populating external parameters
      *
      * @param runtimeContext - runtime properties like DNS id to type map
      * @param jdbcConnection
      * @throws  DACParameterProducerException
      */
     public void init(Map<Object, Object> runtimeContext
          ,java.sql.Connection jdbcConnection) throws
          DACParameterProducerException;
     /**
      * @param dsnId
      * @param taskName
      * @return all parameters (global, dsn-specific and dsn-task specific)
        that apply to a given task
      * @throws DACParameterProducerException
      */
     public Map<String, String> getParameters(int dsnID
          ,String taskName) throws DACParameterProducerException;

     /**
      * Forcing implementing classes to explicitly implement finalize method.
      * All parameter stores inside are expected to be cleared as a result.
      *
      * @throws Throwable
      */
      public void finalize() throws Throwable;
}
```

Note: DAC forces DACTextParameterProducer to implement the finalize method.

Defining an External-Parameter Text Type Parameter Follow this procedure to define an External-Parameter Text data type parameter.

Note: Producer code examples are available in the DAC\documentation\public directory.

To define an External-Parameter Text data type parameter

1. Create a valid Java archive with one or more classes that will implement the DACTextParameterProducer interface.

   In this procedure, "abc.paramproducer.jar" is used as an example for the archive name.

2. Add the archive to the DAC\lib directory.

3. For DAC installations on Windows, add the archive to the set DACLIB entry in the config.bat file.

   For example, the set DACLIB entry would look similar to the following after adding the archive named abc.paramproducer.jar:

   ```
   set DACLIB=.\DAWSystem.jar;.;.\lib\wsclient_
   extended.jar;.\lib\wls-jse-client-wsm-dependencies.jar;.\lib\biacm.paramproduce
   r.jar; .\lib\abc.paramproducer.jar;
   ```

4. In DAC installations on UNIX, add the archive to the export DACLIB entry in the config.sh file.

   For example, the export DACLIB entry would look similar to the following after adding the archive named abc.paramproducer.jar:

   ```
   export DACLIB=./DAWSystem.jar:.:./lib/wsclient_
   extended.jar:./lib/wls-jse-client-wsm-dependencies.jar:./lib/biacm.paramproduce
   r.jar:./lib/ abc.paramproducer.jar:
   ```

5. If the DAC Server is deployed in WebLogic, copy the archive to the domain's library directory.

6. Register the producer in the DAC Client.

   a. In the Design view, click the Source System Parameters tab.

   b. In the top pane toolbar, click New.

   c. In the Edit subtab, enter a name for the parameter.

   d. Select External-Parameter Text as the Data Type.

   e. Select the appropriate Load Type.

      - **Full** indicates the parameter will apply to tasks that are mapped to full load workflows.

      - **Incremental** indicates the parameter will apply to tasks that are mapped to incremental load workflows.

      - **Both** indicates the parameter will apply to tasks that are mapped to both full and incremental load workflows.

   f. Click in the Value field to open the Enter Parameter Value dialog.

   g. Select Logical Data Source from the list.

   h. Enter the full class name for the interface in the External API field.

   i. Click OK.

7. Click Save in the Edit subtab.

Note: You can inactivate the parameter by selecting the Inactive check box.

Timestamp Interface

The External-Parameter Timestamp data type uses the DACTimestampParameterProducer interface to return timestamp values. DAC formats the timestamp during the ETL process according to your specifications.

The DACTimestampParameterProducer interface is as follows:

```
public interface DACTimestampParameterProducer {

     /**
      * Populating external parameters
      *
      * @param runtimeContext - runtime properties like DNS id to type map
      * @param jdbcConnection
      * @throws DACParameterProducerException
      */
     public void init(Map<Object, Object> runtimeContext
          ,java.sql.Connection jdbcConnection) throws
          DACParameterProducerException;
     /**
      * @param dsnId
      * @param taskName
      * @return all parameters (global, dsn-specific and dsn-task specific) that
        apply to a given task
      * @throws DACParameterProducerException
      */
     public Map<String, Timestamp> getParameters(int dsnId
          ,String taskName) throws DACParameterProducerException;

     /**
      * Forcing implementing classes to explicitly implement finalize method.
      * All parameter stores inside are expected to be cleared as a result.
      *
      * @throws Throwable
      */
     public void finalize() throws Throwable;
}
```

Note: DAC forces DACTimestampParameterProducer to implement the finalize method.

Defining an External-Parameter Timestamp Type Parameter Follow this procedure to define an External-Parameter Timestamp type parameter.

Note: Producer code examples are available in the DAC\documentation\public directory.

To define an External-Parameter Timestamp data type parameter

1. Create a valid Java archive with one or more classes that will implement the DACTimestampParameterProducer interface.

   In this procedure, "abc.paramproducer.jar" is used as an example for the archive name.

2. Add the archive to the DAC\lib directory.

3. For DAC installations on Windows, add the archive to the set DACLIB entry in the config.bat file.

   For example, the set DACLIB entry would look similar to the following after adding the archive named abc.paramproducer.jar:

   ```
   set DACLIB=.\DAWSystem.jar;.;.\lib\wsclient_
   extended.jar;.\lib\wls-jse-client-wsm-dependencies.jar;.\lib\biacm.paramproduce
   r.jar; .\lib\abc.paramproducer.jar;
   ```

4. In DAC installations on UNIX, add the archive to the export DACLIB entry in the config.sh file.

   For example, the export DACLIB entry would look similar to the following after adding the archive named abc.paramproducer.jar:

   ```
   export DACLIB=./DAWSystem.jar:.:./lib/wsclient_
   extended.jar:./lib/wls-jse-client-wsm-dependencies.jar:./lib/biacm.paramproduce
   r.jar:./lib/ abc.paramproducer.jar:
   ```

5. If the DAC Server is deployed in WebLogic, copy the archive to the domain's library directory.
6. Register the producer in the DAC Client.
   a. In the Design view, click the Source System Parameters tab.
   b. In the top pane toolbar, click New.
   c. In the Edit subtab, enter a name for the parameter.
   d. Select the External-Parameter Text data type.
   e. Select the appropriate Load Type.

      - **Full** indicates the parameter will apply to tasks that are mapped to full load workflows.

      - **Incremental** indicates the parameter will apply to tasks that are mapped to incremental load workflows.

      - **Both** indicates the parameter will apply to tasks that are mapped to both full and incremental load workflows.

   f. Click in the Value field to open the Enter Parameter Value dialog.
   g. Select a Logical Data Source from the list.
   h. Enter the full class name for the interface in the External API field.
   i. Select the appropriate Function for the timestamp.
   j. Enter a Format for the timestamp.
   k. Click OK.
7. Click Save in the Edit subtab.

Note: You can inactivate the parameter by selecting the Inactive check box.

Defining a Multi-Parameter Type Parameter

The Multi-Parameter data types enable you to define parameters that return string or timestamp values as name-value pairs.

This procedure applies to parameters defined at both the source system and task levels.

To define a Multi-Parameter Text or Timestamp data type parameter

1. Do one of the following:
   - To define a source system parameter, in the Design view, click the Source System Parameters tab, and then click New in the toolbar.
   - To define a task parameter, in the Design view, click the Tasks tab, then click the Parameters subtab, and then click New in the bottom pane toolbar.
2. In the new record field, enter a parameter name in the Name field.
3. Select one of the following options as the Data Type.
   - **Multi-Parameter Text.** This option returns string values as name-value pairs.
   - **Multi-Parameter Timestamp**. This option returns timestamp values as name-value pairs.
4. Click Save to save the record.
5. Click in the Value field to open the Enter Parameter Value dialog.
6. Select a Logical Data Source.
7. Enter the parameter definition as a SQL statement, and click OK.
8. Click Save.

Note: You can inactivate the parameter by selecting the Inactive check box.

Defining a JavaScript Type Parameter

Follow this procedure to define a parameter using the JavaScript data type. This procedure applies to parameters defined at both the source system and task levels.

To define a JavaScript data type parameter

1. Do one of the following:
   - To define a source system parameter, in the Design view, click the Source System Parameters tab, and then click New in the toolbar.
   - To define a task parameter, in the Design view, click the Tasks tab, then click the Parameters subtab, and then click New in the bottom pane toolbar.
2. In the new record field, enter a parameter name in the Name field.
3. Select the JavaScript as the Data Type.
4. Click Save to save the record.
5. Click in the Value field to open the Enter Parameter Value dialog.
6. Enter the parameter definition as a JavaScript statement, and click OK.
7. Click Save.

Note: You can inactivate the parameter by selecting the Inactive check box.

Defining a Global External Parameter

The Global External Multi-Parameter data types enable you to define parameters that return string or timestamp values as name-value pairs.

To define a parameter using the Global Multi-Parameter Text data type

1. In the DAC Client, on the **Tools** menu, select **Seed Data**, **Global External Parameters**.

   The Global External Parameters dialog is displayed.
2. In the toolbar, click **New**.
3. In the new record field, enter a name for the parameter in the **Name** field.
4. Select one of the following Parameter Type options:
   - **Global**. Returns name-value pairs based on data in the entire repository.
   - **By DSN**. Returns name-value pairs for a specific data source.
   - **By DSN-Task**. Returns name-value pairs for a specific data source and task.
5. Select one of the following options as the Data Type.
   - **Global Multi-Parameter Text.** This option returns string values as name-value pairs.
   - **Global Multi-Parameter Timestamp**. This option returns timestamp values as name-value pairs.
6. Click in the **Value** field to open the Enter Parameter Value dialog.
7. Select a **Logical Data Source Connection** from the drop-down list.
8. If you selected the Global Multi-Parameter Text data type, enter the parameter definition as a SQL statement.
9. If you selected the Global Multi-Parameter Timestamp data type, from the Function list, select a format to which DAC will convert the date. If you select Custom, enter a custom date format.

8

Designing Subject Areas

This chapter provides information about working with subject areas.

A subject area is a logical grouping of tables related to a particular subject or application context. A subject area includes the tasks that are associated with the tables, as well as the tasks required to load the tables.

Oracle BI Applications provides predefined subject areas. You can change these predefined subject areas or create new subject areas to correspond to your particular business processes.

> **Note:** To change a predefined subject area or to create a new subject area, you must first make a copy of an existing source system container or create a new container. For instructions, see "Creating or Copying a Source System Container".

This chapter contains the following topics:

About Designing a Subject Area

In designing a subject area, you should consider the following questions:

- **Tables.** Which tables need to be populated in the data warehouse? From which tables does the organization source data? What tables will create the star schemas?
- **Subject areas.** Do the subject areas cover all the relevant tables?
- **Tasks.** Are the tasks that load this table defined?
- **Indexes.** Do the target tables have the correct indexes defined?

Previewing Subject Areas

You can preview a subject area to determine whether it contains the appropriate tables and tasks to suit your business needs. When you assign a fact table to a subject area, the fact table's related tables are automatically assigned to the subject area. If you determine you do not need one or more of the related tables, you can remove the

tables during the subject area Assembly process. You can also add individual tables and tasks.

To preview a subject area, follow the procedure "Creating a Subject Area" through the assembly and review stages, but do not complete the process (by clicking Accept in the Subject Area Assembly dialog) unless the subject area is suitable for your needs.

How DAC Determines Tasks Required for Subject Areas

You define a subject area by specifying a fact table or set of fact tables to be the central table or tables in the subject area. When a subject area is defined, DAC performs the following process to determine the relevant tasks:

1. DAC identifies the dimension tables associated with the facts and adds these tables to the subject area.
2. DAC identifies the related tables, such as aggregates, associated with the fact or dimension tables and adds them to the subject area definition.
3. DAC identifies the tasks for which the dimension and fact tables listed in the two processes above are targets tables and adds these tasks into the subject area.

   Tasks that DAC automatically assigns to a subject area are indicated with the Autogenerated flag (in the Tasks subtab of the Subject Areas tab).

   You can inactivate a task from participating in a subject area by selecting the Inactive check box (in the Tasks subtab of the Subject Areas tab). When the Inactive check box is selected, the task remains inactive even if you reassemble the subject area.

   You can also remove a task from a subject area using the Add/Remove command in the Tasks subtab of the subject Areas tab, but when you remove a task it is only removed from the subject area until you reassemble the subject area.
4. DAC identifies the source tables for the tasks identified in the previous process and adds these tables to the subject area.

DAC performs this process recursively until all necessary tasks have been added to the subject area. A task is added to the subject area only once, even if it is associated with several tables in the subject area. DAC then expands or trims the total number of tasks based on the configuration rules, which are defined as configuration tags. This process can be resource intensive because DAC loads all of the objects in the source system container into memory before parsing.

Creating a Subject Area

When you create a new subject area, you assign one or more fact tables to the subject area. DAC then determines which dimension and other related tables are required as well as the tasks and their order of execution.

To create a new subject area

1. In the Design view, select the appropriate source system container from the drop-down list in the toolbar.
2. Click the **Subject Areas** tab.
3. In the top pane toolbar, click **New**.
4. In the Edit subtab, do the following:
   a. Enter a name for the subject area.

b. If you want to assign only configuration tag tasks to the subject area and exclude the subject area's autogenerated tasks, select the **Configuration Tag Tasks Only** check box.

c. Click **Save**.

5. Make sure the new subject area name is highlighted in the top pane, and click the Tables subtab.

6. Click **Add/Remove** in the subtab toolbar.

   The Choose Tables dialog is displayed. The left-hand window lists all the tables held in the selected container.

7. Query for one or more fact tables.

8. Select the fact table (use Shift+click to select more than one table), and click **Add**.

   The tables are added to the right-hand window, which represents the subject area.

9. Click **OK** to close the Choose Tables dialog.

10. In the top pane toolbar, click **Assemble**.

11. In the Assembling... dialog, select **Selected Record Only**.

    If you select the option **All Records in the List**, DAC will assemble all the subject areas listed in the top pane.

    DAC assembles the selected subject area by determining what dimensions and other related tables are required.

    The Subject Area Assembly dialog is displayed. The tree view on the left side shows the fact tables that belong to the subject area. You can expand the fact table node to view its related tables.

12. Click **Calculate Task List** to assemble the tasks needed to load the tables displayed in the tree view.

    If configuration tags have been assigned to any of the tasks, an information message will appear.

    A list of tasks is displayed in the Task List tab on the right side of the window. Also, the Table List tab displays the tables included in the subject area.

13. Click **Accept** to complete the subject area assembly process.

14. (Optional) Modify the subject area to better suit your business needs by removing one or more tables from the subject area.

    a. In the tree view, remove one or more tables from the subject area by deselecting the check box to the left of the table name.

    b. Click **Calculate List**.

    c. Review the changes to the subject area:

       The Task List tab displays the new set of tasks required to load the pruned subject area.

       The Task Difference Report tab displays the tasks that were added and the tasks that were deleted during the last assembly process.

       The Table List tab displays the tables that are included in the pruned subject area as well as the tables that were excluded during the pruning process.

    d. To accept the changes made to the subject area, click **Accept**.

e. Click **OK** in the message box stating the subject area was successfully assembled.

15. (Optional) Click the **Tasks** subtab to view which tasks DAC has determined are required for this subject area.

    Tasks that are automatically assigned to the subject area by DAC are indicated with the Autogenerated check mark.

    You can inactivate a task from participating in the subject area by selecting the Inactive check box. When the Inactive check box is selected, the task remains inactive even if you reassemble the subject area.

    You can also remove a task from the subject area using the Add/Remove command, but when you remove a task it is only removed from the subject area until you reassemble the subject area.

Modifying an Existing Subject Area

You can modify an existing subject area by doing any of the of the following steps:

- **Add or remove additional fact tables.** You can add or remove fact tables using the Add/Remove command in the Tables subtab of the Subject Areas tab.
- **Add or remove tasks.** You can add or remove individual tasks using the Add/Remove command in the Tasks subtab of the Subject Areas tab. You can also remove tasks by assigning a task to a configuration tag. See "Working with Configuration Tags" for more information.

After you modify an existing subject area, you need to reassemble it (by selecting it and clicking Assemble in the Subject Areas tab). This action opens the Subject Area Assembly dialog, where you can calculate the new task list and view the tasks that were added and removed in the Task Difference Report tab.

When to Reassemble a Subject Area

After the first assembly of a subject area, you will need to reassemble the subject area in the following situations:

- A new task is added to the data warehouse that extracts data from or loads data into any of the tables associated with the subject area.
- A task associated with the subject area is deleted, activated or inactivated.
- A configuration tag is added to the subject area or to a table or task associated with the subject area.

Note: You do not need to reassemble a subject area if an attribute of a task changes but the task is already associated with a subject area.

9

Managing Data Warehouse Schemas

This chapter provides information about managing data warehouse schemas.

DAC enables you to manage data warehouse schemas by creating, upgrading, and dropping data warehouse tables.

This chapter contains the following topics:

- Managing Data Warehouse Schemas for Oracle Databases.

  This section contains instructions for creating, upgrading, or dropping data warehouse tables when the Oracle Business Analytics Warehouse is on an Oracle database.

- Managing Data Warehouse Schemas for Non-Oracle Databases

  This section contains instructions for creating, upgrading, or dropping data warehouse tables when the Oracle Business Analytics Warehouse is on a SQL Server, DB2, DB2-390, or Teradata database.

> **Caution:** Before you make any changes to the data warehouse schema in a production environment, you should first test the changes in a non-production environment.

Managing Data Warehouse Schemas for Oracle Databases

For Oracle databases, DAC provides several methods for managing data warehouse schemas.

- **Creating, upgrading or dropping an entire schema**. This option uses the Data Warehouse Configuration Wizard to do a mass update of the schema by creating, upgrading, or dropping all tables at once. This option also enables you to create delete triggers on a transactional database for Siebel sources. See "Creating, Upgrading or Dropping an Entire Schema for Oracle Databases" for instructions.
- **Creating, upgrading or dropping subsets of tables**. This option uses the Generate DW Table Scripts right-click menu command to create, upgrade, or drop subsets of tables in the schema by generating SQL scripts in the Oracle database format. See "Creating, Upgrading or Dropping Subsets of Tables in the Schema for Oracle Databases" for instructions.

You can also parameterize default values for table columns before you create or upgrade the schema. For more information, see "Parameterizing Default Values for Table Columns".

In addition, customizing the schema creation and upgrade process is possible for advanced DAC users. See "Advanced Usage of the Schema Creation and Upgrade Process for Oracle Databases" for more information about customization options.

Note: Before you perform the procedures in this section, make sure you have done the following:

- Created an ODBC connection to the Oracle Business Analytics Warehouse database.
- Created an SSE role for the Oracle Business Analytics Warehouse and associated the database user with the role.

Creating, Upgrading or Dropping an Entire Schema for Oracle Databases

This procedure uses the Data Warehouse Configuration Wizard to create, upgrade, or drop all tables in the schema. You can also follow this procedure to create delete triggers on a transactional database for Siebel sources.

To create, upgrade or drop the data warehouse schema

1. From the DAC menu bar, select **Tools**, **ETL Management**, **Configure**.
2. In the Sources dialog, select **Oracle** as the target and source database platform.
3. Click **OK** to display the Data Warehouse Configuration Wizard.
4. Select the appropriate option to generate a SQL script to create, upgrade, or drop data warehouse tables. For Siebel sources you can also select the option to create delete triggers in the transactional data warehouse.

   Note: If you select the option to upgrade the data warehouse schema, the SQL scripts that DAC generates depend on what data warehouse objects need to be upgraded. See "About the Upgrade Schema SQL Scripts" for more information.
5. Click **Next**.

   The Data Warehouse tab is active.
6. Enter the following information:

Field	Description
Container	The name of the source system containers for which you want to create, upgrade or drop the data warehouse tables. Separate multiple names with commas. If you leave this field blank, DAC performs the action specified for all source system containers. If there are tables that are common to multiple containers, then only one table will be created. If columns are different for the same table across containers, DAC will create a table that has all the columns in the same table. When you specify a container name, you must enter the name exactly as it appears in the container drop-down list.
Is Unicode	Select this check box if you need a Unicode data warehouse. **Note**: The database must be set to support the creation of a Unicode schema.

Field	Description
Execute	Select this option of you want DAC to execute the SQL script automatically after it is generated. If you do not select this option, you can manually execute the script at a later time. DAC stores the SQL scripts in <Domain_Home>\dac\conf\sqlgen\sql\oracle. **Note:** If you are upgrading the data warehouse schema, depending on what objects need to be updated, DAC may generate a SQL script named upgrade-questionable.sql. DAC will not automatically execute this script. You must execute it manually after you have reviewed and corrected it as necessary. For more information about the upgrade scripts, see "About the Upgrade Schema SQL Scripts".
Physical Data Source	Select the appropriate target data source.
Change default parameter file	The default location for the default_parameter.properties file is displayed. This file stores parameter names and values for the parameterization of default values for not null table columns. You can select a different properties file by clicking **Change default parameter file**, and navigating to the appropriate file. See "Parameterizing Default Values for Table Columns" for more information.

7. Click **Start**.

   The Run Status tab displays information about the process, as follows:

   - If a 'Success' message is displayed, the data warehouse tables have been created. To review log information about the process, see the following log files:

     <Domain_Home>\dac\log\config\generate_ctl.log - A log of the schema definition process, including details of any conflicts between containers.

     <Domain_Home>\dac\log\config\createtables.log - A log of the DDL Import Utility process.

   - If a 'Failure' message is displayed, the data warehouse tables have not been created. Use the log information in <Domain_Home>\dac\log\config\generate_ctl.log to diagnose the error. The createtables.log is not generated.

Creating, Upgrading or Dropping Subsets of Tables in the Schema for Oracle Databases

To create, upgrade or drop subsets of tables, you use the Generate DW Table Scripts right-click menu command to generate SQL scripts in the Oracle database format.

You can access the right-click menu command from the tabs described in Table 9–1, depending on the context for the schema update:

Table 9–1 How to Access the Generate DW Table Scripts Right-Click Menu Command

Tab	Purpose
Tables tab (Design view)	Use when you have one or several tables to update in the schema.

Table 9–1 (Cont.) How to Access the Generate DW Table Scripts Right-Click Menu

Tab	Purpose
Task Source Tables (RO) and **Task Target Tables (RO) subtabs** of Subject Areas tab (Design view)	Use when you are working with a subject area and need to update a subset of tables.
Applied Patches tab (Setup view)	Use when you have applied a patch to the DAC repository and need to update the data warehouse schema.

To create, upgrade or drop subsets of tables

1. Go to the appropriate tab (as described in Table 9–1) and query for the tables with which you want to update the schema.
2. Right-click in the list of tables, and then select **Generate DW Table Scripts for Oracle**.
3. In the "Perform Operations for" area, select one of the following, and then click OK.
   - **Selected record only**
   - **All records in the list**
4. In the Generate DW Table Scripts for Oracle dialog, specify the following options:

Field	Description
Create New	Select this option to generate a SQL script to create new data warehouse tables.
Upgrade Existing	Select this option to generate a SQL script to upgrade existing data warehouse tables. **Note:** Depending on what objects need to be updated, DAC may generate a SQL script named upgrade-questionable.sql. DAC will not automatically execute this script. You must execute it manually after you have reviewed and corrected it as necessary. For more information about the upgrade scripts, see "About the Upgrade Schema SQL Scripts".
Drop Tables	Select this option to generate a SQL script to drop tables from the data warehouse schema.
Unicode	Select this check box if you need a Unicode data warehouse. **Note**: The database must be set to support the creation of a Unicode schema.
Execute	Select this option of you want DAC to execute the SQL script automatically after it is generated. If you do not select this option, you can manually execute the script at a later time. DAC stores the SQL scripts in <Domain_ Home>\dac\conf\sqlgen\sql\oracle.
Physical Data Source	Select the appropriate target data source. **Note**: If you are creating new data warehouse tables, and you did not select the Execute check box, you do not have to specify a physical data source.

Field	Description
Change default parameter file	The default location for the default_parameter.properties file is displayed. This file stores parameter names and values for the parameterization of default values for not null table columns. You can select a different properties file by clicking **Change default parameter file**, and navigating to the appropriate file. See "Parameterizing Default Values for Table Columns" for more information.

5. Click **Start**.

   The Run Status tab displays information about the process, as follows:

   - If a 'Success' message is displayed, the data warehouse tables have been created. To review log information about the process, see the following log files:

     <Domain_Home>\dac\log\config\generate_ctl.log - A log of the schema definition process, including details of any conflicts between containers.

     <Domain_Home>\dac\log\config\createtables.log - A log of the DDL Import Utility process.

   - If a 'Failure' message is displayed, the data warehouse tables have not been created. Use the log information in <Domain_Home>\dac\log\config\generate_ctl.log to diagnose the error. The createtables.log is not generated.

Advanced Usage of the Schema Creation and Upgrade Process for Oracle Databases

If you are an advanced DAC user, you can customize the schema creation and upgrade process. This section provides information you need to know before customizing the SQL scripts.

This section contains the following topics:

- About the Create Schema SQL Script
- About the Upgrade Schema SQL Scripts
- Schema Characteristics You Need to Consider
- Error Handling
- Creating or Upgrading the Schema When You Have Multiple Source System Containers
- Customizing the Schema XML Templates

About the Create Schema SQL Script

When you use DAC to create a new data warehouse schema, DAC generates the create.sql file. This file contains SQL syntax for creating a schema for the source system container you are working with.

After the create.sql file is generated, DAC saves it in the <Domain_Home>\dac\conf\sqlgen\sql\<database type> directory. During the schema creation process, you can choose whether to execute this script upon generation or save the script to execute at a later time.

About the Upgrade Schema SQL Scripts

When you use DAC to upgrade the data warehouse schema, DAC generates one or both of the following SQL scripts, depending on what data warehouse objects need to be upgraded.

upgrade-regular.sql DAC generates the upgrade-regular.sql file when it determines the following data warehouse schema modifications are needed:

- Add a new table
- Add a new column
- Increase a column length
- Modify a column from NOT NULL to NULL
- Drop a nullable column default value
- Modify a column default value

DAC saves the upgrade-regular.sql script in the <Domain_Home>\dac\conf\sqlgen\sql\oracle directory. During the schema upgrade process, you can choose whether to execute this script upon generation, or save the script to execute at a later time.

upgrade-questionable.sql DAC generates the upgrade-questionable.sql file when it determines the following data warehouse modifications are needed.

- Modify a column data type
- Decrease a column length
- Modify a column precision
- Modify a column from NULL to NOT NULL

Because of the nature of these changes, the SQL scripts may fail when executed. Therefore, a DBA must review the upgrade-questionable.sql file before it is executed, and make any necessary corrections to the SQL statements. DAC saves this file in the directory <Domain_Home>\dac\conf\sqlgen\sql\oracle directory. You cannot choose to have this script executed upon generation. You must execute it manually after it has been reviewed and corrected as necessary.

Schema Characteristics You Need to Consider

This section provides information about the most important schema characteristics specific to DAC that you need to consider.

Data Types Data types are specified as part of the column definition. The following data types are supported:

- **CHAR**. The CHAR data type specifies a fixed-length character string. The maximum length is enforced by the destination database. When you create a CHAR column, you must specify the maximum number of bytes or characters of data the column can hold in the Length field (in the Columns subtab of the Tables tab in DAC). If you try to insert a value that is shorter than the specified length, trailing spaces are added to the column value at run time.
- **VARCHAR**. The VARCHAR data type specifies a variable-length character string. The maximum length is enforced by the destination database. When you create a VARCHAR column, you must specify the maximum number of bytes or

characters of data it can hold in the Length field (in the Columns subtab of the Tables tab in DAC).

- **DATE**. The DATE data type specifies a valid date. The valid date is enforced by the destination database.
- **NUMBER**. The NUMBER data type stores positive and negative fixed and floating-point numbers. The maximum length for this value is enforced by the destination database. When you create a number column, you must specify a Length and Precision in the Columns subtab of the Tables tab in DAC. The Length value is the maximum number of bytes or characters of data the column can hold. The Precision value is the total number of significant decimal digits, where the most significant digit is the left-most nonzero digit, and the least significant digit is the right-most known digit.
- **TIMESTAMP**. The TIMESTAMP data type stores the date and timestamp. The maximum length for this value is enforced by the destination database.

NULL, NOT NULL and Default Values A column can have a NULL value if, in DAC, the Nullable check box is selected in the Columns subtab of the Tables tab. If the Nullable check box is not selected, the column is declared NOT NULL. NOT NULL columns must contain a value. If a NOT NULL column does not have an explicit value, it will get the default value if one is specified in the Default Value field in the Columns subtab. If a NOT NULL column does not have an explicit value and a default value is *not* specified, then it will get the value specified in the **defaults.properties** file only when this column is an existing column that is being upgraded.

For the upgrade process, the defaults.properties file enables you to specify missing default values for data warehouse tables with NOT NULL columns for which default values are not specified in DAC. This file also specifies the equivalents of CURRENT TIMESTAMP based on database type. This file is located in the <DAC_Config_Location>\CustomSQLs\Schema_Templates directory.

In the defaults.properties file, the default values specified are grouped by database type. The entries for the Oracle database type look similar to the following:

```
ORACLE_VARCHAR=' '
ORACLE_CHAR=' '
ORACLE_NUMBER=0
ORACLE_DATE=SYSDATE
ORACLE_TIMESTAMP=SYSDATE
ORACLE_%NOW%=SYSDATE
```

For the default values you enter in DAC, you need to enclose in single quotes the values for CHAR, VARCHAR, DATE, and TIMESTAMP data types.

Column Ordering In DAC, the column order is specified in the Position field of the Columns subtab in the Tables tab. Each column must have a unique position, and the column position specified in DAC must match the column position specified in the Informatica repository.

Unicode Considerations When generating SQL scripts to create or upgrade the schema, you can indicate in DAC whether the schema should be created as Unicode. When you create a schema as Unicode, you can still define specific table columns as non-Unicode in DAC by going to the Columns subtab in the Tables tab.

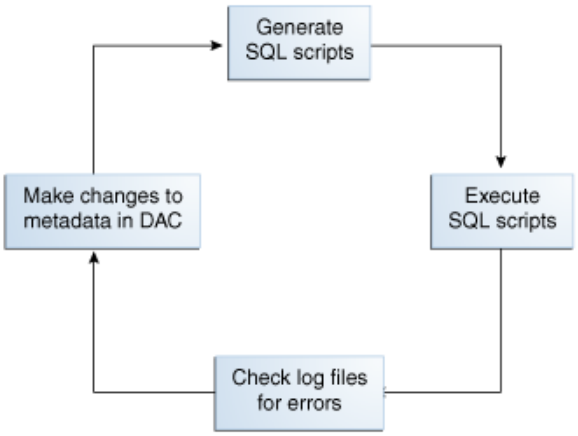
Index Creation Indexes are not created during the schema creation process. Indexes are created at runtime during the first full load.

Error Handling

Errors should be handled using an iterative process. Figure 9–1 illustrates the general process for handling errors during the schema creation and upgrade processes. The first step is to generate the SQL scripts for creating or upgrading the data warehouse schema. Next, you can review the SQL scripts before executing and also review the SQL generation log file, which is located in the directory <Domain_Home>\dac\log\config. You then execute the scripts, and check the SQL execution log file for errors. This file is located in the directory <Domain_Home>\dac\log\config.

If errors occurred during the script execution, you need to interpret the error messages and determine the best way to alter the metadata in DAC. Once you have made the necessary changes to the metadata, you can then re-execute the scripts, check the log files, and make any additional changes to the metadata. You need to repeat this process until no errors appear in the execution log file.

Figure 9–1 Error Handling Iterative Process



Creating or Upgrading the Schema When You Have Multiple Source System Containers

If your environment uses multiple source system containers, when you upgrade the schema, the upgrade-questionable.sql file will log discrepancies among the containers, such as discrepancies with data types. The process of finding errors will be iterative. You can execute the scripts, review them to find errors, and fix the errors before executing the scripts.

For the upgrade scripts to execute successfully, the containers must be synchronized. If you have made changes in one container, you need to replicate the changes in the other containers. You can do so by referencing the new or changed object or by recreating the new or changed object across all containers. For information about how to reference repository objects, see "About Object Ownership in DAC".

Customizing the Schema XML Templates

The schema creation and upgrade processes are controlled by XML templates, which contain database-dependent SQL syntax that specifies how data warehouse objects are created or upgraded.

This section includes the following topics:

- About XML Template Files Used for Creating the Data Warehouse Schema
- Customizing the createschema_template_designation.xml File
- Customizing the createschema_<database type>.xml File
- About the Upgrade XML Template

About XML Template Files Used for Creating the Data Warehouse Schema There are two XML templates that control the schema creation process: createschema_template_designation.xml and createschema_<database type>.xml.

When you create the data warehouse schema, you can use the default settings in these XML templates or you can customize them to suit your needs. If you use the default settings, you do not need to change any information in the files before following the procedure "Creating, Upgrading or Dropping Subsets of Tables in the Schema for Oracle Databases".

The XML templates are located in the directory <DAC_Config_Location>\CustomSQLs\Schema_Templates directory.

About createschema_template_designation.xml The createschema_template_designation.xml file designates which schema creation template or templates will be used to create the data warehouse schema. By default, the preconfigured createschema_<database type>.xml file is designated as the template for creating all data warehouse tables. If necessary, you can modify the createschema_<database type>.xml file or you can create your own schema creation templates in XML format.

The createschema_template_designation.xml file enables you to specify schema creation templates to do the following:

- Specify a single schema creation template for creating all the data warehouse tables or all tables other than those created by the override templates.
- Specify an override schema creation template for specific tables by table name.
- Specify an override schema creation template for specific tables by table type.

The XML example below shows the structure of the createschema_template_designation.xml file. The tag <ORACLE> indicates this block of XML applies to the Oracle database type. The XML elements available for customizing are TABLE_NAMES, TABLE_TYPES, and ALL_OTHER_TABLES. Templates specified in the TABLE_NAMES and TABLE_TYPES tags will override the template specified in the ALL_OTHER_TABLES tag.

```
<CREATE_SCHEMA_TEMPLATE_DESIGNATION>
   <ORACLE>
     <TEMPLATE NAME = "">
       <TABLE_NAMES></TABLE_NAMES>
     </TEMPLATE>
     <TEMPLATE NAME = "">
       <TABLE_TYPES></TABLE_TYPES>
     </TEMPLATE>
     <TEMPLATE NAME = "createschema_oracle.xml">
       <ALL_OTHER_TABLES/>
     </TEMPLATE>
</ORACLE>
```

As an example, if you wanted to specify a particular, custom schema creation template for creating fact and dimension tables, called createschema_dimension.xml, and use the default template (createschema_oracle.xml) for creating all other tables, the XML block would look like the following:

```
<CREATE_SCHEMA_TEMPLATE_DESIGNATION>
   <ORACLE>
     <TEMPLATE NAME = "">
       <TABLE_NAMES></TABLE_NAMES>
     </TEMPLATE>
     <TEMPLATE NAME = "createschema_dimension.xml">
       <TABLE_TYPES>FACT, DIMENSION</TABLE_TYPES>
     </TEMPLATE>
     <TEMPLATE NAME = "createschema_oracle.xml">
       <ALL_OTHER_TABLES/>
     </TEMPLATE>
</ORACLE>
```

About createschema_<database type>.xml The createschema_<database type>.xml file contains the database-dependent SQL syntax used to create the schema. It also specifies the tablsespace. You can customize this file to specify tablespaces for specific tables by table name and table type. You can also add additional SQL at the end of the file.

However, you should not change the SQL syntax in the file.

The code example below shows the XML elements available for customizing the tablespace. The tablespace specified in the TABLE_NAMES and TABLE_TYPES tags will override the tablespace specified in the ALL_OTHER_TABLES tag.

```
<CREATE_SCHEMA>
  <TABLESPACE DEFINITION>
     <TABLESPACE NAME = "">
       <TABLE_NAMES></TABLE_NAMES>
     </TABLESPACE>
     <TABLESPACE NAME = "">
       <TABLE_TYPES></TABLE_TYPES>
     </TABLESPACE>
     <TABLESPACE NAME = "">
       <ALL_OTHER_TABLES/>
     </TABLESPACE>
   </TABLESPACE DEFINITION>
```

Customizing the createschema_template_designation.xml File Follow this procedure to customize the createschema_template_designation.xml file.

Note: Templates specified for tables by name or type override the template specified by the ALL_OTHER_TABLES XML element.

To customize the createschema_template_designation.xml file

1. Go to the directory <DAC_Config_Location>\CustomSQLs\Schema_Templates.
2. Open the createschema_template_designation.xml file in a text editor.
3. To designate a template for a specific table name:
   a. Find the `TABLE_NAMES` XML element.

      For example:

      ```
      <TEMPLATE NAME = "">
        <TABLE_NAMES></TABLE_NAMES>
      </TEMPLATE>
      ```

   b. Enter the template name as the value for `TEMPLATE NAME` attribute. The value must be enclosed within quotes.
   c. Enter the table name as text content for the `TABLE_NAMES` element.

4. To designate a template for a table type:
   a. Find the TABLE_TYPES XML element.

      For example:

      ```
      <TEMPLATE NAME = "">
        <TABLE_TYPES></TABLE_TYPES>
      </TEMPLATE>
      ```

   b. Enter the template name as the value for the in the `TEMPLATE NAME` attribute. The value must be enclosed within quotes.
   c. Enter the table type as text content for the `TABLE_TYPES` element.
5. To designate a template for all tables or for all tables other than those created by templates specified in the steps above:
   a. Find the ALL_OTHER_TABLES XML element.

      For example:

      ```
      <TEMPLATE NAME = "">
        <ALL OTHER TABLES/>
      </TEMPLATE>
      ```

   b. Enter the template name as the value for the in the `TEMPLATE NAME` attribute. The value must be enclosed within quotes.
6. Save and close the file.

Customization Example

As an example, if you wanted to designate custom templates for the following:

- A custom template named createschema_employee_table.xml to create the table named W_EMPLOYEE_D
- A custom template named createschema_fact.xml to create all fact tables
- The default template createschema_oracle.xml to create all other tables

The XML block would look similar to the following:

```
<CREATE_SCHEMA_TEMPLATE_DESIGNATION>
   <ORACLE>
     <TEMPLATE NAME = "createschema_employee_table.xml">
       <TABLE_NAMES>W_EMPLOYEE_D</TABLE_NAMES>
     </TEMPLATE>
     <TEMPLATE NAME = "createschema_fact.xml">
       <TABLE_TYPES>FACT</TABLE_TYPES>
     </TEMPLATE>
     <TEMPLATE NAME = "createschema_oracle.xml">
       <ALL_OTHER_TABLES/>
     </TEMPLATE>
</ORACLE>
```

Customizing the createschema_<database type>.xml File You can customize the createschema_oracle.xml file by specifying tablespaces for tables by table name and table type. You can also add additional SQL statements to the end of the file. You should not change any of the SQL syntax in the file.

To customize the createschema_oracle.xml file

1. Go to the directory <DAC_Config_Location>\CustomSQLs\Schema_Templates.
2. Open the createschema_oracle.xml file in a text editor.

3. To designate a tablespace for a specific table name:
   a. Find the TABLE_NAMES XML element.

      For example:

      ```
      <TABLESPACE NAME = "">
        <TABLE_NAMES></TABLE_NAMES>
      </TABLESPACE>
      ```

   b. Enter the tablespace name as the value for the `TABLESPACE NAME` attribute. The value must be enclosed within quotes.
   c. Enter the table name as text content for the `TABLE_NAMES` element.
4. To designate a tablespace for a table type:
   a. Find the TABLE_TYPES XML element.

      For example:

      ```
      <TABLESPACE NAME = "">
        <TABLE_TYPES></TABLE_TYPES>
      </TABLESPACE>
      ```

   b. Enter the tablespace name as the value for the in the `TABLESPACE NAME` attribute. The value must be enclosed within quotes.
   c. Enter the table type as text content for the `TABLE_TYPES` element.
5. To designate a tablespace for all tables or for all tables other than those created by templates specified in the steps above:
   a. Find the XML element for the ALL_OTHER_TABLES element.

      For example:

      ```
      <TABLESPACE NAME = "">
        <ALL OTHER TABLES/>
      </TABLESPACE>
      ```

   b. Enter the tablespace name as the value for the in the `TABLESPACE NAME` attribute. The value must be enclosed within quotes.
6. (Optional) Add additional SQL statements as the text content of the element <SUPPLEMENT> towards the end of the file.
7. Save and close the file.

About the Upgrade XML Template The **upgradeschema_oracle.xml** file contains the database-dependent SQL syntax used to upgrade the data warehouse schema with changes that were made to tables in DAC. This file is located in the <DAC_Config_Location>\CustomSQLs\Schema_Templates directory. You should not change any of the SQL syntax in this file.

Managing Data Warehouse Schemas for Non-Oracle Databases

This section contains instructions for creating, upgrading, and dropping data warehouse tables when the Oracle Business Analytics Warehouse is on a SQL Server, DB2, DB2-390, or Teradata database.

You can also follow this procedure to create delete triggers in the transactional database for Siebel sources.

Note: Before you perform the procedure in this section, make sure you have done the following:

- Created an ODBC connection to the Oracle Business Analytics Warehouse database.
- Created an SSE role for the Oracle Business Analytics Warehouse and associated the database user with the role.

To create, upgrade or drop the data warehouse schema

1. From the DAC menu bar, select **Tools**, **ETL Management**, **Configure**.
2. In the Sources dialog, select the database platform for the target data warehouse and source transactional database.
3. Click **OK** to display the Data Warehouse Configuration Wizard.
4. Select the appropriate option to create, upgrade, or drop data warehouse tables. For Siebel sources you can also select the option to create delete triggers in the transactional data warehouse. Then, click **Next**.

   The Data Warehouse tab is active.
5. Enter the appropriate information for the database in which you want to store the data warehouse tables. The information that you need to enter is dependent on the type of target database.

Field	Description
Container	The name of the source system containers for which you want to create, upgrade or drop the data warehouse tables. Separate multiple names with commas. If you leave this field blank, DAC performs the action specified for all source system containers. If there are tables that are common to multiple containers, then only one table will be created. If columns are different for the same table across containers, DAC will create a table that has all the columns in the same table. If you only want to deploy a subset of the source business applications for which you imported seed data earlier, then use this field to specify a container name. When you specify a container name, you must enter the name exactly as it appears in the container drop-down list.
Table Owner	Valid database owner, username, or account that you set up to hold the data warehouse.
Password	Valid database user password for the database owner, username, or account that you specified in the Table Owner field.
ODBC Data Source	Data Source Name (DSN) for the Oracle Business Analytics Warehouse. You must specify the name of the ODBC connection that you created for the data warehouse.
Tablespace	(Optional) Tablespace where data warehouse tables are created.
Is Unicode	Specifies whether the data warehouse database is Unicode. The database must be set to support the creation of a Unicode schema.
Change default parameter file	(For Teradata databases only.) The default location for the default_parameter.properties file is displayed. This file stores parameter names and values for the parameterization of default values for not null table columns. You can select a different properties file by clicking **Change default parameter file**, and navigating to the appropriate file. See "Parameterizing Default Values for Table Columns" for more information.

6. Click **Start**.

The Run Status tab displays information about the process, as follows:

- If a 'Success' message is displayed, the data warehouse tables have been created. To review log information about the process, see the following log files:

  <Domain_Home>\dac\log\config\generate_ctl.log - A log of the schema definition process, including details of any conflicts between containers.

  <Domain_Home>\dac\log\config\createtables.log - A log of the DDL Import Utility process.

- If a 'Failure' message is displayed, the data warehouse tables have not been created. Use the log information in <Domain_Home>\dac\log\config\generate_ctl.log to diagnose the error. The createtables.log is not generated.

- For Teradata databases, DAC generates a SQL script for the specified action. The script is saved in the <Domain_Home>\dac\conf\sqlgen\sql\teradata directory. You need to manually execute the script in the data warehouse to complete the specified process.

Parameterizing Default Values for Table Columns

Before creating or upgrading a data warehouse schema, you can parameterize values for not null table columns. This feature can be useful for defining default values for table columns that are shared across multiple tables, such as DATASOURCE_NUM_ID.

Note: When you upgrade a schema and use a parameter to define a table column default value, the actual value of the parameter will be used to compare with the default value in the target data warehouse.

Follow these instructions to define a parameter and value before you create or upgrade the data warehouse schema.

To parameterize a default value for table columns

1. Go to the directory dac\CustomSQLs\Schema_Templates, and open the file default_parameters.properties.

   The file that is installed with Oracle BI Applications will be empty.

   Note: Alternatively, you can create your own properties file and save it in a directory of your choosing.

2. Enter a parameter and value in the following format:

   ```
   @DAC_<parameter_name>=<parameter_value>
   ```

   Note the following conditions:

   - If the value is a string, it must be enclosed within single quotation marks.

     For example:

     ```
     @DAC_param1='mydefault'
     ```

   - Null values are not allowed. A null value will throw an error.

   - To define a parameter value as an empty string, you must use single quotes. For example:

     ```
     @DAC_param2=''
     ```

Otherwise, the value will be considered null and will throw an error.

- When creating or upgrading a schema for multiple containers, if a parameter is used in one container and an actual value is defined for the other, the process will compare the actual value and the parameterized value. If the values are not the same, an exception will be thrown.

10

Performance Tuning With DAC

This chapter provides information about various performance tuning capabilities in DAC related to indexes, tables, tasks, and workflows.

This chapter contains the following topics:

Managing Indexes

In general, during the ETL process, before a table is truncated, all the indexes, as defined in the repository, will be dropped before the data is loaded and recreated automatically after the data is loaded. The dropping of indexes before the data load and recreating them after the data load improves ETL performance.

If you create an index on the database and it is not registered in the DAC repository, the index will not be dropped and the load will fail.

For Teradata databases, only secondary indexes should be registered in DAC. You should not register primary indexes or the more complex indexes, such as single- and multi-table indexes, because they cannot be dropped and recreated.

In the DAC Design view, the Indices tab lists all the indexes associated with the selected source system container. It is recommended that you have a clear understanding of when and where an index will be used at the time you register it in DAC. It is also recommended that you do not register in DAC any indexes for source tables.

This section contains the following topics:

Index Behavior

The general sequence of actions related to indexes when a task runs is as follows:

1. Tables are truncated.
2. Indexes are dropped.
3. Data is loaded into target tables.
4. ETL indexes and unique indexes are recreated.
5. Successor tasks are notified that ETL indexes were created.
6. Query indexes and additional unique indexes are recreated.
7. Tables are analyzed.

Note: Successor tasks can start before query-related and unique indexes are created.

Specifying Index Spaces for Indexes by Table Type

You can specify index spaces for indexes by table type. If you do not specify an index space by table type, then DAC will create indexes in the index space specified by the Default Index Space property in the Physical Data Sources tab of the Setup view. If you do not specify a value for the Default Index Space property, then DAC will create indexes in the default tablespace for the table owner of the database.

Note: You must create the index space in the database before you can specify index spaces for indexes in DAC.

To specify an index space for indexes by table type

1. In the Physical Data Sources tab of the Setup view, click the **Index Spaces** subtab.
2. In the subtab toolbar, click **Generate**.
3. When asked if you want to generate database properties based on table type, click **Yes**.

   A list of available table types is displayed in the Table Type list.
4. Click **OK** to close the Generating Properties dialog.
5. In the **Index Space** column, enter an index space for each table type, and click **Save**.

Specifying How Many Indexes Can Be Created in Parallel

You can specify how many indexes can be created in parallel for a specific table or how many indexes can be created in parallel for all tables associated with a specific physical data source connection.

To specify how many indexes can be created in parallel for a specific table

1. In the Physical Data Sources tab of the Setup view, select the appropriate physical data source in the top pane list.
2. Click the **Parallel Indexes** subtab.
3. Click **New** in the Parallel Indexes toolbar.
4. In the **Name** field, query for and enter the names of the table for which you want to specify how many indexes can be created in parallel.

5. For each table, in the **Number of Parallel Indexes** field, enter the number of indexes you want to create in parallel.
6. Click **Save**.

To specify how many indexes can be created in parallel for all tables associated with a physical data source

1. In the Physical Data Sources tab of the Setup view, select the appropriate physical data source in the top pane list.
2. Click the **Edit** subtab.
3. In the **Num Parallel Indexes Per Table** field, enter a numeric value to specify how many indexes can be created in parallel for a table on the specified physical data source.
4. Click **Save**.

Defining a During ETL Index

You can override the point at which an index is created during an ETL process by defining an index with the During ETL usage type. You assign the During ETL index to the task that populates the table for which the index was defined. The During ETL index will then be created after the task to which you assign it runs. In this manner, using During ETL indexes enables you to control when the index is created during the ETL process.

Note: If you define a During ETL index, you need to make sure that the index is assigned to a task, or else the index will not be created. You should only assign a During ETL index to one task per execution plan. Only advanced DAC users with a thorough understanding of their data warehouse and ETL processes should define During ETL indexes.

To define a During ETL Index

1. Create a new index in DAC:
   a. In the Design view, go to the **Indices** tab, and click **New** in the top pane toolbar.
   b. In the Edit subtab, define the index with the appropriate properties. For a description of the properties, see "Indices Tab".

      Make sure you select During ETL as the Index Usage type.
   c. Click **Save.**
2. Assign the index you created in step 1 to a task:
   a. Go to the **Tasks** tab, and query for the task to which you want to assign the During ETL index.
   b. Click the **During ETL** subtab.
   c. In the bottom pane toolbar, click **Add/Remove.**
   d. In the Name field, enter the name of the During ETL index, and click **Go**.
   e. Make sure the index is selected, and then click **Add**.

      The index appears in the list on the right side of the window.
   f. Click **Refresh** to populate the Table Name and Owner fields.
   g. Click **OK** to close the dialog.

The During ETL index will be created on the table associated with the task after the task runs.

Note: You can also modify an existing index to change the usage type to During ETL.

Defining Join Indexes on a Teradata Database

This section provides instructions for defining indexes on a Teradata database.

To define a join index on a Teradata database

1. Define a task action to drop indexes and a second task action to create indexes.

   For instructions on defining a task action, see "Defining a SQL Script for an Action".

2. Create a task for dropping indexes. Select the following attributes:
   - **SQL File** as the Execution Type.
   - **Pre ETL Process** as the Task Phase.

   For instructions on creating tasks in DAC, see "Creating Tasks in DAC for New or Modified Informatica Workflows".

3. Create a task for creating indexes. Select the following attributes:
   - **SQL File** as the Execution Type.
   - **Post ETL Process** as the Task Phase.

4. Assign the task for dropping indexes as a preceding task on the appropriate execution plan.
   a. In the Execution Plans tab, select the appropriate execution plan.
   b. Click the **Preceding Tasks** subtab.
   c. In the subtab toolbar, click **Add/Remove**.
   d. In the Choose Preceding Tasks dialog, query for the task you created in step 2.
   e. Click **Add**.
   f. Click **OK** to close the Choose Preceding Tasks dialog.

5. Assign the task for creating indexes as a following task on the appropriate execution plan.
   a. In the Execution Plans tab, select the appropriate execution plan.
   b. Click the **Following Tasks** subtab.
   c. In the subtab toolbar, click **Add/Remove**.
   d. In the Choose Following Tasks dialog, query for the task you created in step 3.
   e. Click **Add**.
   f. Click **OK** to close the Choose Following Tasks dialog.

6. Rebuild the execution plan.

Using Actions to Optimize Indexes and Collect Statistics on Tables

This section applies to incremental loads only.

When an index is marked in DAC as Drop & Create Always or Drop & Create Bitmap Always (for Oracle databases), DAC drops the indexes before the data load and creates them after the data load. Performance can be improved by customizing and tuning how indexes are dropped and created during incremental loads.

When large fact tables are involved in an incremental load, dropping and creating indexes on the whole table can impact performance. Table partitioning can be a performance enhancing alternative to dropping and creating indexes. For example, if the data being loaded is mostly based on updates to recent data, you can partition the table based on time. If the incremental data affects the current year, it may be more beneficial to disable the current year partition than to drop the index on the whole table; and, similarly, it may be more beneficial to rebuild the indexes for the partition rather than create the indexes on the whole table. This can be achieved by using different syntaxes for full and incremental loads, as described in the following example.

This example illustrates, for incremental runs, how to create a parameter that can be used to override the drop and create index syntaxes. It also describes how to override the analyze table behavior so that only the current year partition is analyzed rather than the whole table.

This example assumes the table in the underlying schema has already been partitioned, with the partitioned names as PART_2001, PART_2002, and so on.

1. Create a source system parameter called CURRENT_YEAR, using the Timestamp data type and the runtime variable @DAC_ETL_START_TIME_FOR_TARGET.

   For instructions on creating a source system parameter, see "Defining a Timestamp Type Parameter".

2. Create a custom index action to override the drop index behavior.

   a. Create an index action called Drop Partitioned Index.

   b. Enter the following SQL:

   ```
   alter index getIndexName()
   modify partition PART_@DAC_$$CURRENT_YEAR
   unusable
   ```

   For instructions on defining an index action, see "Defining a SQL Script for an Action".

3. Assign the Drop Partitioned Index action to indexes.

   a. In the Indices tab, query for the table name and indexes that need to be dropped and created during the incremental runs.

   b. For each record in the list of query results, right-click and select **Add Actions**.

   c. Select **Drop Index** as the Action Type, **Incremental** as the Load Type, and **Drop Partitioned Index** as the Action.

4. Create a custom index action to override the create index behavior.

   a. Create an index action called Enable Partitioned Index.

   b. Enter the following SQL:

   ```
   alter index getIndexName()
   rebuild partition PART_@DAC_$$CURRENT_YEAR
   nologging
   ```

5. Assign the Enable Partitioned Index action to indexes.

a. In the Indices tab, query for the table name and indexes that need to be dropped and created during the incremental runs.

b. For each record in the list of query results, right-click and select **Add Actions**.

c. Select **Drop Index** as the Action Type, **Incremental** as the Load Type, and **Enable Partitioned Index** as the Action.

6. Create a custom index action to override the analyze table behavior.

   a. Create an index action called Analyze Current Partition.

   b. Enter the following SQL:

```
DBMS_STATS.GATHER_TABLE_STATS(
NULL,
TABNAME => 'getTableName()'
CASCADE => FALSE,
PARTNAME => PART_@DAC_CURRENT_YEAR,
ESTIMATE_PERCENT => DBMS_STATS.AUTO_SAMPLE_SIZE,
GRANULARITY => 'FOR ALL INDEXED COLUMNS SIZE AUTO',
DEGREE => DBMS_STATS.DEFAULT_DEGREE)
```

   c. Select **Stored Procedure** as the Type.

7. Assign the Analyze Current Partition action to tables.

   a. In the Tables tab, query for the table names that need to be analyzed during the incremental runs.

   b. For each record in the list of query results, right-click and select **Add Actions**.

   c. Select **Analyze Table** as the Action Type, **Incremental** as the Load Type, and **Analyze Current Partition** as the Action.

For a more detailed example of using partitions, see "Oracle Business Intelligence Applications Version 7.9.6.x Performance Recommendations [ID 870314.1]" on My Oracle Support.

Using Heuristics to Manage Tasks, Tables and Indexes

The DAC heuristics functionality enables you to gather intelligence about the amount of incremental data that will be processed during an ETL execution. To optimize performance, you can use this intelligence to decide whether a task will run, whether tables are analyzed, and whether indexes are dropped and created.

This chapter contains the following topics:

Note: DAC heuristics applies only to incremental data loads and not full loads.

About DAC Heuristics

Typically, during an incremental ETL process, DAC drops indexes on target tables before a data load, recreates the indexes after the data load, and then analyzes the

tables. These processes can impact overall ETL process times. DAC heuristics enables you to optimize performance by deciding whether tasks will run, tables are analyzed, and indexes dropped and created.

There are three parts to defining heuristics:

1. Defining the heuristics rule.
2. Associating the heuristics rule with a task.
3. Providing the means for computing the source and target counts to be used in the computation of heuristics.

Using Heuristics to Determine Whether a Task Should Run

You can determine whether or not you want a task to run as part of an ETL process. DAC heuristics looks at the primary source tables (or other tables you specify by writing custom SQL) before a data load to see if there are any incremental data records. The decision about whether a task will run is based on a threshold value you set. If the incremental record count is equal or less than the threshold value, DAC will not execute the task.

For example, assume you use DAC heuristics to count the records in a primary staging table for a dimension load task, and you set a threshold value of zero. If DAC heuristics finds no records, DAC will not execute this task when the ETL process runs. If DAC heuristics finds one or more records, the task will be executed.

Using Heuristics to Determine Whether Indexes Are Dropped and Created

You can determine whether indexes are dropped and created during an ETL process. For this action, you configure a heuristics rule that either counts the number of incremental records in the primary staging table (or other tables you specify by writing custom SQL) or that provides a ratio of the incremental records in the primary source table and the total records in the target table. This value is then compared to the threshold value you set. If the count or ratio is equal or less than the threshold value, DAC will not drop and create indexes.

To determine an appropriate threshold value, you need to have an understanding of the impact that dropping and recreating indexes has on your database.

- **Example Using the Count Option:** Assume you configure DAC heuristics to count the incremental records in the primary source table. You have determined performance is only impacted during an incremental load with indexes in place if there are more than ten records to be loaded, so you set the threshold value at 10. If DAC heuristics finds between zero and nine incremental records in the primary source table, DAC will not drop and recreate indexes when the data is loaded into the target tables. The data will be loaded with indexes in place. If DAC heuristics finds ten or more incremental records in the primary staging table, DAC will drop and recreate indexes during the load process.
- **Example Using the Ratio Option:** Assume you configure DAC heuristics to provide a ratio of the number of incremental records in the primary source table and the total number of records in the target table. You have determined that if incoming records are 5% or more of the total target table record count, then performance can be impacted with indexes in place. Therefore, you set the threshold value at 5. If DAC heuristics finds the ratio of incremental records in the primary source table to be 5% or more of the total records in the target table, then DAC will drop and recreate indexes during the load process. If the percentage of incremental records is equal or less than 5%, DAC will load the new records with the indexes in place.

Using Heuristics to Determine Whether Tables Are Analyzed

You can determine whether or not you want tables to be analyzed after the ETL process runs. For this action, you configure a heuristics rule that either counts the number of incremental records in the primary staging table (or other tables you specify by writing custom SQL) or that provides a ratio of the incremental records in the primary source table and the total records in the target table. This value is then compared to the threshold value you set. If the count or ratio is equal or less than the threshold value, DAC will not analyze tables.

To determine an appropriate threshold value, you need to have an understanding of the impact that analyzing tables has on your database.

The count and ratio options work the same way for the action of analyzing tables as for dropping and creating indexes. For examples of these two options as they relate to dropping and creating indexes, see "Using Heuristics to Determine Whether Indexes Are Dropped and Created".

Note: This DAC heuristics setting overrides the Analyze Frequency (in Days) system property.

DAC Heuristics and Task Groups

If a child task of a task group has heuristics defined, the task can optionally be specified as a *heuristics driver*. Any number of child tasks within a task group can be specified as a heuristics driver. Before a task group is executed, DAC computes the task heuristics of each of the child tasks. If the computation of any of the heuristics driver tasks falls under the threshold value, none of the child tasks are executed. The status of these tasks in the Current Run tab of the Execute view is "Not Executed." The Status Description is updated with the cumulative summary of all the driver children as to why the task was not executed.

About Heuristics Rules and the Heuristics Dialog

A heuristics rule defines how the DAC heuristics functionality will manage the behavior of tasks, tables and indexes. Heuristics rules can apply to any DAC heuristics action; that is, you can use the same rule to indicate whether a task should run, whether tables should be analyzed, or indexes dropped and recreated. Heuristics rules apply to all containers. However, if you write custom SQL ("Writing Custom SQL for a Heuristics Rule") to specify tables other than the primary tables, the SQL is specific to the container in which you write it.

You use the Heuristics dialog to define heuristics rules. The Heuristics dialog contains the following fields:

- **Heuristic**

  The possible values are **Count** and **Ratio**. These options provide flexibility in how you gather intelligence. The Count option provides a count of the number of incremental records in the primary staging table that will be included in the ETL process. The Ratio option provides the number of incremental records in the primary source table *and* the total number of records in the target table. This value is expressed in the threshold field as a percentage.

  The Count option can be useful if you are deciding whether or not to run a task. For example, if you only want a task to run if there is one or more records in a staging table, you would select the Count option and set the Threshold value at 1. DAC heuristics counts the number of records in the staging table. If it finds one or more records, the task will run. If it finds no records, the task will not run.

The Ratio option can be useful if you are deciding whether to drop and create indexes or analyze tables. For examples of using the count and ratio options, see "Using Heuristics to Determine Whether Indexes Are Dropped and Created".

- **Metric**

  Provides options for how multiple primary source tables are handled (that is, when more than one primary source table is associated with the same task). If you are defining a heuristics rule that involves a single primary source table, the Metric field does not apply.

  The possible values are the following:

  – **Sum**

    Aggregates the total of new or changed records in all of the primary source tables.

  – **Average**

    Returns an average of the new or changed records in all of the primary source tables. That is, it will divide the sum of all the new or changed records in all of the primary tables by the number of primary tables.

  – **Max**

    Returns the largest count of new or changed records of all of the primary source tables.

  – **Min**

    Returns the smallest count of new or changed records of all of the primary source tables.

- **Source Table**

  The possible values are the following:

  – **Primary**

    Specifies that intelligence will be gathered on one or more primary tables associated with the task that this heuristics rule is assigned to.

  – **With SQL Overrides**

    Specifies that you will write custom SQL to include source tables other than the default primary source tables.

- **Threshold**

  The numerical value above which DAC heuristics will allow a particular action to take place.

  For example, if you are doing a count of a primary staging table to determine whether a task should run, and if you want the task to run if one or more records appears in the table, you would set the Threshold value at 1.

  If you use the Ratio option, the threshold value is a percentage. For example, suppose you were defining a heuristics rule for determining whether tables should be analyzed after an ETL process, and you knew you wanted to analyze tables if the incoming rows were more than 5% of the total target rows. In this case, you would set the threshold value at 5.

Creating a Heuristics Rule

Follow this procedure to create a heuristics rule. After you create the rule, you need to assign it to a task. For instructions, see "Associating a Heuristics Rule With a Task".

For an overview of DAC heuristics, see "Using Heuristics to Manage Tasks, Tables and Indexes". For detailed information about the Heuristics dialog, see "About Heuristics Rules and the Heuristics Dialog".

To create a heuristics rule

1. In the DAC Client, on the **Tools** menu, select **Seed Data**, **Heuristics**.

   The Heuristics dialog is displayed.

2. In the toolbar, click **New**.
3. In the Name field, enter a descriptive name for the heuristics rule.
4. In the Heuristics field, select one of the following:
   - **Count**

     Provides a count of the number of records in the primary source table that will be included in the ETL process (based on the task to which you assign the heuristics rule).

   - **Ratio**

     Provides a count of the number of records in the primary source table and the total number of records in the target table. If you select the Ratio option, the value of the Threshold field is a percentage.
5. If this heuristics rule applies to a task with multiple primary tables, select one of the following:
   - **Sum**. Aggregates the total of new or changed records in all of the primary source tables.
   - **Average**. Returns an average of the new or changed records in all of the primary source tables.
   - **Max**. Returns the largest count of new or changed records of all of the primary source tables.
   - **Min**. Returns the smallest count of new or changed records of all of the primary source tables.
6. In the Source Tables field, select one of the following:
   - **Primary**. Specifies that intelligence will be gathered on one or more primary tables associated with the task that this heuristics rule is assigned to.
   - **With SQL Overrides**. Specifies that you will write custom SQL to include tables other than the primary tables.

     For instructions on writing custom SQL to create a heuristics rule, see "Writing Custom SQL for a Heuristics Rule".
7. In the Threshold field, enter a numerical value above which DAC heuristics will allow a particular action to take place.

   For examples, see "About Heuristics Rules and the Heuristics Dialog".
8. (Optional) If you want to create this heuristics rule but leave it inactive, select **Inactive**.
9. Click **Save**.

After you create the heuristics rule, you need to assign the rule to a task. For instructions, see "Associating a Heuristics Rule With a Task".

Associating a Heuristics Rule With a Task

Before you perform this procedure, you need to have created a heuristics rule by following the instructions in "Creating a Heuristics Rule".

For an overview of DAC heuristics, see "Using Heuristics to Manage Tasks, Tables and Indexes". For detailed information about the Heuristics dialog, see "About Heuristics Rules and the Heuristics Dialog".

In this procedure, you will associate a heuristics rule with a task.

To assign a heuristics rule to a task

1. In the Design view, select the **Tasks** tab.
2. In the Tasks tab, select the task to which you want to assign the heuristics rule.
3. Select the **Extended Properties** subtab.
4. In the toolbar, click **New.**

   A new, empty record is created.
5. Click in the **Name** field to expose the drop-down list, and then select **Heuristics**.
6. Double-click in the **Value** field to open the Property Value dialog.

   The Property Value dialog enables you to select a heuristics rule that you created in the procedure "Creating a Heuristics Rule".
7. Assign the heuristics rule to a task by doing one of the following:
   - For a rule that determines whether a task should run, click in the **Task Heuristic** field, and then select the appropriate heuristics rule.
   - For a rule that determines whether tables should be analyzed, click in the **Table Heuristic** field, and then select the appropriate heuristics rule.
   - For a rule that determines whether indexes should be dropped and recreated, click in the **Index Heuristic** field, and then select the appropriate heuristics rule.
8. Click **OK** in the Select Value dialog, and then click **OK** in the Property Value dialog.

The heuristics rule is now assigned to a task and will be processed when the task to which it is assigned is called by an execution plan.

Heuristics rules are reflected in the Status Description field in the Task Details tab of the Details dialog, which is accessible by going to the Tasks subtab in the Current Runs or Run History tabs and clicking the Details button on subtab toolbar. The Status Description will indicate whether a step was skipped because of a heuristics rule, and the task status will be Not Executed.

Writing Custom SQL for a Heuristics Rule

When you create a heuristics rule, you have the option of gathering intelligence on primary source tables, which is the default option, or writing custom SQL to gather intelligence on tables other than primary source tables.

Note: When you write custom SQL for a heuristics rule, it applies only to the source system container you select before you write the SQL.

For an overview of DAC heuristics, see "Using Heuristics to Manage Tasks, Tables and Indexes". For detailed information about the Heuristics dialog, see "About Heuristics Rules and the Heuristics Dialog".

Note: Before you perform this procedure, you must have already created a heuristics rule, by following the procedure "Creating a Heuristics Rule", and selected the option **With SQL Overrides** in the Source Tables field.

To write custom SQL to create a heuristics rule

1. In the Design view, select the appropriate source system container from the drop-down list.
2. Select the **Container Specific Actions** tab.
3. Select **Heuristic** from the rightmost drop-down list on the toolbar.
4. In the toolbar, click **New**.

   A new, empty record is created.
5. In the **Name** field, enter a descriptive name for the SQL, and then click **Save**.
6. Click in the **Value** field to open the Value dialog.
7. Select a format for the tree view on the left side of the window.
   - **Flat view** displays the SQL entries in a list format in their order of execution.
   - **Category** view displays the entries by the categories Custom SQL and Stored Procedure.

     You can reorder the entries in the tree by dragging and dropping them.
8. Click **Add** in the right-hand toolbar.

   A new, empty record is created in the top pane.
9. Enter or select the appropriate information.

Field	Description
Name	Enter a logical name for the SQL block.
Type	Select one of the following: ■ **Select SQL**. Indicates the SQL you enter will be a SELECT statement. ■ **Stored Procedure**. Indicates the SQL you enter will be a stored procedure.
Continue on Fail	Specifies whether an execution should proceed if a given SQL block fails.
Retries	Specifies how many retries are allowed. If the number is not positive, a default number of one (1) will be used.
Valid Database Platform	Specifies the valid database platforms against which the SQL will run. If this field is left empty, the SQL can be run against any database.

10. In the lower-right side text box, enter a SQL statement.

The SQL Statement tab to the left of the text box lists the supported SQL functions and DAC source system parameters that you can use in constructing custom SQL statements. Double-click a function or source system parameter to move it into the text box.

The source systems parameters list contains the applicable source system parameters defined in the DAC repository, with the prefix @DAC_. During runtime, the DAC Server resolves the source system parameter and replaces its name with the runtime value.

See Table 10–1 for a list of supported SQL functions.

Table 10–1 Functions for Heuristics SQL

Function	Description
getAnalyzeTableStatement()	Returns the default DAC analyze table statement.
getDBType()	Returns the physical data source connection type (Oracle OCI8, Oracle Thin, DB2, DB2-390, MSSQL, Teradata, or BI Server).
getImageSuffix()	Returns the table image suffix if one exists. Otherwise, the string is empty.
getNamedSource()	Returns the physical connection name in DAC.
getTableName()	Returns the table name.
getTableOwner()	Returns the table owner.
getTableSpace()	Returns the table space name if one exists. Otherwise, the string is empty.
getTruncateTableStatement()	Returns the default DAC truncate table statement.
toString()	Returns a string value.

11. (Optional) Enter a comment about the SQL in the Comment tab.
12. Click **OK**.
13. If the SQL statement contains source tables, assign the heuristics SQL to the source tables.
    a. In the Design view, click the **Tasks** tab.
    b. Select the task whose source table is the one to which you want to assign the heuristics SQL.
    c. Click the **Source Tables** subtab.
    d. Select or query for the appropriate source table.
    e. Click in the **Task Count Override** field.

       The Choose SQL Override dialog is displayed.
    f. Select or query for the SQL heuristics rule you created in the preceding steps, then click **OK**.
14. Assign the heuristics SQL to a target table.
    a. In the Design view, click the **Tasks** tab.
    b. Select the task whose target table is the one to which you want to assign the heuristics SQL.
    c. Click the **Target Tables** subtab.

d. Select or query for the appropriate target table.

e. Click in the **Count Override** field.

   The Choose SQL Override dialog is displayed.

f. Select or query for the SQL heuristics rule you created in the preceding steps, then click **OK**.

   The heuristics SQL is now assigned to a task and will be processed when the task to which it is assigned is called by an execution plan.

Looping of Workflows

You can configure the full and incremental load commands for tasks to repeat (or loop) multiple times during the execution of an ETL process. This feature can be useful to enhance performance in the following ways:

- To configure workflows to read and write in parallel.
- To repeat application logic that applies to all repetitions of a task, for example the flattening of hierarchies.

The process for setting up workflows for looping involves two steps: First you define the looping properties, and then you create parameters to access the looping properties you defined.

This section includes the following topics:

Defining a Looping Property

Follow this procedure to define the properties for looping a workflow.

To define a workflow for looping

1. Enable the Informatica workflow property **Configure Concurrent Execution**:
   a. In Informatica Workflow Manager, open the appropriate workflow in the Workflow Designer.
   b. On the menu bar, select **Workflows**, and then select **Edit**.
   c. In the Edit Workflow dialog, select the **Enabled** check box for the **Configure Concurrent Execution** property.
   d. Click the **Configure Concurrent Execution** button.
   e. In the Configure Concurrent Execution dialog, select **Allow concurrent run with same instance name**.
   f. Click **OK**.
2. In the DAC Design view, select the appropriate container from the drop-down list.
3. In the Tasks tab, query for the task for which you want to configure looping.
4. With the appropriate task selected in the top pane, click the **Extended Properties** subtab.

5. In the bottom pane toolbar, click **New**.

   A new record is created with the name Loops. You cannot change the name of this record.

6. Click in the **Value** field of the new record.

   The Property Value dialog is displayed.

7. In the Loops field, enter a numeric value to specify how many times you want the workflow to repeat.

   The number of loops that you specify is applicable to all task execution types, that is, Informatica, SQL File, Stored Procedure, and External Program.

8. (Optional) Select the **Parallel** check box to have the run instances of the same task execute in parallel. If you do not select the Parallel check box, the run instances of the same task will be executed serially.

   Note:

   - Run instances will be executed in parallel only if resources are available. For example, if your environment is set up to run 10 workflows in parallel, and a task's Number of Loops property is set to 20, only the first 10 instances will be executed. These 10 instances will have a Current Runs status of Running. The remaining 10 instances will have a status of Runnable. For a description of all Current Runs statuses, see "Current Runs Tab".
   - When the instances are executed in serial, the first failure encountered stops the rest of the executions.
   - When the instances are executed in parallel, any failure in the running batch stops the rest of the executions.

9. (Optional) Select the **Restart All** check box to do the following:

   - Reissue the truncate command (if it exists as a task detail).
   - Mark all task details pertaining to the run instances as Queued. The Queued status enables them to be executed again.

     If you do not select the Restart All check box, the execution of the task will resume from the point of failure. If any runtime instances of the workflow fails, the task itself is marked as Failed. If you enable the Restart All property, the truncate command (if it exists as a task detail) is reissued, and all the task details pertaining to the run instances are marked as Queued and get re-executed. If you do not enable the Restart All property, the execution resumes from the point of failure

10. Click **OK** in the Property Value dialog.

11. Click **Save** in the subtab toolbar.

When the task executes, DAC assigns an instance number to each instance of the workflow. You can view the instance number in the Task Details subtab of the Current Runs tab.

The instance number is accessible as a predefined DAC variable named @DAC_TASK_RUN_INSTANCE_NUMBER. You can use this DAC variable as a runtime value for parameters that you create. The DAC variable @DAC_TASK_RUN_INSTANCE_NUMBER is available for use with parameters at the source system and task level.

Accessing the Loops Properties Using Parameters

To access the information defined in the looping properties, you need to create parameters in DAC that will be consumed by Informatica at runtime. You can create static or runtime parameters at either the task or source system level. The parameter values can be any of the following:

- Total number of loops
- Current instance number
- Current instance number in reverse
- Conditional statement

  For example, if the loop number is one, provide value x, or else provide the value y.

For more information about how parameters are used in DAC, see "Defining and Managing Parameters".

To create a parameter with a static value

1. In the DAC Design view, select the appropriate container from the drop-down list.
2. Do one of the following:
   - To set up a parameter at the task level, go to the Task tab, and click the Parameters subtab.
   - To set up a parameter at the source system level, go to the Source System Parameters tab.
3. Click **New** in the toolbar.
4. In the Name field, enter a name for the parameter.
5. In the Data Type field, select **Text**.
6. In the Load Type field, select one of the following.
   - **Full.** To specify a full load.
   - **Incremental.** To specify an incremental load.
   - **Both.** To specify both full and incremental loads.
7. Click **Save** to save the record.

   You must save the record before proceeding.
8. Click in the **Value** field to open the Enter Parameter Value dialog.
9. Select **Static**.
10. Enter the appropriate value.
11. Click **Save** in the toolbar.

To parameterize the Loops property as a runtime value

1. In the Design view, select the appropriate container from the drop-down list.
2. Do one of the following:
   - To set up a parameter at the task level, go to the Task tab, and click the Parameters subtab.
   - To set up a parameter at the source system level, go to the Source System Parameters tab.

3. Click **New** in the toolbar.
4. In the Name field, enter a name for the parameter.
5. In the Data Type field, select **Text**.
6. In the Load Type field, select one of the following.
   - **Full.** To specify a full load.
   - **Incremental.** To specify an incremental load.
   - **Both.** To specify both full and incremental loads.
7. Click **Save** to save the record.

   You must save the record before proceeding.
8. Click in the **Value** field to open the Enter Parameter Value dialog.
9. Select **Runtime**.
10. Select the appropriate runtime variable.
11. Click **OK**.
12. Click **Save** in the toolbar.

Parallelizing the Load Process on a Fact Table

Typically, a long running task is constrained by single reader and writer threads on Informatica. The looping of workflows parallelizes the reads and writes such that mutually exclusive records can be read from and written to without having overlaps.

For example, if a fact table load takes a long time to run because the volume of incremental data is high, then you can parallelize the load process by doing the following:

1. Modify the schema for the staging table and introduce a numerical column called LOOP_NUMBER.
2. In DAC, create a task level parameter for the extract process called NUMBER_OF_LOOPS, and set the value to 5.
3. Modify the extract mapping for the Informatica task so that each of the records loaded in the staging table gets a value of from 1 to 5.
4. Create a task level parameter in DAC for the load process called NUMBER_OF_LOOPS, and set it to the same value as the extract mapping.
5. Create a new parameter at the task level called CURRENT_LOOP_NUMBER, and set the runtime variable as @DAC_CURRENT_INSTANCE_NUMBER. This value will be populated with a different number, from 1 to 5, for each invocation of the task.
6. In the extended properties for the task, create a new record for "Loops" and set the value as @DAC_NUMBER_OF_LOOPS.
7. Modify the Informatica mapping to include a where clause for the load logic:

   ```
   WHERE LOOP_NUMBER = $$current_LOOP
   ```

   When DAC runs the load process, it will spawn off five processes in parallel, and will send a specific value for the CURRENT_LOOP_NUMBER parameter to Informatica. Because DAC is reading and writing in parallel, the data load time can be reduced.

Creating Logical Partitions on Load Tasks to Increase Performance

Slow performance can be caused by a small number of tasks holding up many other tasks, which is often due to single reader and writer threads on Informatica. You can performance tune long running tasks that hold up other tasks by creating logical partitions on the task itself. The procedure below provides a typical way to achieve logical partitioning of the task.

To create logical partitions on load tasks

1. On the staging table, create a column to store the logical partition number. In this example, the column is named log_part_num.
2. Create a bitmap index on the log_part_num column.
3. In the Informatica SDE mapping, make the extract-into-staging task populate the logical partition column. This divides the data into 'n' number of buckets.

   For example, `mod(rownum, 4) +1` creates four buckets with the numbers 1, 2, 3, and 4.

   Now, the load mapping will run once in parallel for each logical partition. This can help improve slow performance caused by single reader and writer threads, and, thus, reduce the time taken by a mapping having huge incremental data.
4. In the Informatica SIL mapping, introduce a new mapping parameter called $$LOGICAL_PARTITION_NUM. Use this parameter in the SQL override.

   For example:

   ```
   select * from staging_table
   where ... and log_part_num = $$LOGICAL_PARTITION_NUM
   ```
5. In the Informatica SIL workflow, edit the workflow properties, and select the "Enable Concurrent Execution" property.
6. Define a looping property:
   a. In DAC, select the SIL task.
   b. Define a looping property, and assign 4 as the value for the number of loops.

      See "Defining a Looping Property" for instructions. Note that enabling the Informatica Workflow property "Configure Concurrent Execution" is a step in the procedure Defining a Looping Property, and you must complete this step.
7. In DAC, define a parameter at the task level called $$LOGICAL_PARTITION_NUM. Use the Text parameter type, and assign it the runtime value @DAC_TASK_RUN_INSTANCE_NUMBER.

   When DAC runs this task, it will run four instances of it in parallel. Each instance invocation will be called with the following convention:

   - The $$LOGICAL_PARTITION_NUM will get distinct values, one per instance.
   - The workflows will be executed with the instance name with the instance number suffix.

Customizing customsql.xml to Drop and Create Indexes and Analyze Tables

The customsql.xml file, located in the <DAC_Config_Location>\CustomSQLs directory, contains the default syntax DAC uses for dropping and creating indexes and

analyzing tables. You can edit the customsql.xml file to change the behavior of these operations.

Note: Modifying the customsql.xml file affects the syntax on a global level, for all execution plans. If you want to do customizations at the table or index level, you should consider using the Action functionality. See "Using Actions to Optimize Indexes and Collect Statistics on Tables" for more information.

To edit the Analyze Table syntax

1. Open the **customsql.xml** file located in the <DAC_Config_Location>\CustomSQLs directory.
2. Locate the Analyze Table syntax for the appropriate database type.

   For example, the syntax for an Oracle database is as follows:

   ```
   <SqlQuery name = "ORACLE_ANALYZE_TABLE" STORED_PROCEDURE = "TRUE"> DBMS_
   STATS.GATHER_TABLE_STATS(ownname => '@TABLEOWNER', tabname => '%1', estimate_
   percent => 30, method_opt => 'FOR ALL COLUMNS SIZE AUTO',cascade => true )
   </SqlQuery>
   ```

3. Edit the syntax.

   For example, to gather statistics for only the indexed columns, edit the syntax as follows:

   ```
   <SqlQuery name = "ORACLE_ANALYZE_TABLE" STORED_PROCEDURE = "TRUE"> DBMS_
   STATS.GATHER_TABLE_STATS(ownname => '@TABLEOWNER', tabname => '%1', estimate_
   percent => 30, method_opt => 'FOR ALL INDEXED COLUMNS',cascade => true )
   </SqlQuery>
   ```

> **Note:** The variables @TABLEOWNER, %1, %2, and so on, will be substituted appropriately by the DAC when the statement is executed.

To edit the Create Index syntax

1. Open the customsql.xml file located in the <DAC_Config_Location>\CustomSQLs directory.
2. Locate the Create Index syntax for the appropriate database type, and edit the syntax.

Performance Tuning the Siebel Change Capture Process

DAC performs the change capture process for Siebel source systems. This process has two components:

- The **change capture** process occurs before any task in an ETL process runs.
- The **change capture sync** process occurs after all of the tasks in an ETL process have completed successfully.

See "SQL for Change Capture and Change Capture Sync Processes" for the SQL blocks used for both of these processes.

Supporting Source Tables

The source tables that support the change capture process are as follows:

- **S_ETL_I_IMG.** Used to store the primary key (along with MODIFICATION_NUM and LAST_UPD) of the records that were either created or modified since the time of the last ETL.
- **S_ETL_R_IMG.** Used to store the primary key (along with MODIFICATION_NUM and LAST_UPD) of the records that were loaded into the data warehouse for the prune time period.
- **S_ETL_D_IMG.** Used to store the primary key records that are deleted on the source transactional system.

Full and Incremental Change Capture Processes

The full change capture process (for a first full load) does the following:

1. Inserts records into the S_ETL_R_IMG table, which has been created or modified for the prune time period.
2. Creates a view on the base table. For example, a view V_CONTACT would be created for the base table S_CONTACT.

The incremental change capture process (for subsequent incremental loads) does the following:

1. Queries for all records that have changed in the transactional tables since the last ETL date, filters them against the records from the R_IMG table, and inserts them into the S_ETL_I_IMG table.
2. Queries for all records that have been deleted from the S_ETL_D_IMG table and inserts them into the S_ETL_I_IMG table.
3. Removes the duplicates in the S_ETL_I_IMG table. This is essential for all the databases where "dirty reads" (queries returning uncommitted data from all transactions) are allowed.
4. Creates a view that joins the base table with the corresponding S_ETL_I_IMG table.

Performance Tips for Siebel Sources

This section contains the following performance tips:

Performance Tip: Reduce Prune Time Period

Reducing the prune time period (in the Connectivity Parameters subtab of the Execution Plans tab) can improve performance, because with a lower prune time period, the S_ETL_R_IMG table will contain a fewer number of rows. The default prune time period is 2 days. You can reduce it to a minimum of 1 day.

Note: If your organization has mobile users, when setting the prune time period, you must consider the lag time that may exist between the timestamp of the transactional system and the mobile users' local timestamp. You should interview your business users to determine the potential lag time, and then set the prune time period accordingly.

Performance Tip: Eliminate S_ETL_R_IMG From the Change Capture Process

If your Siebel implementation does not have any mobile users (which can cause inaccuracies in the values of the "LAST_UPD" attribute), you can simplify the change capture process by doing the following:

- Removing the S_ETL_R_IMG table.
- Using the LAST_REFRESH_DATE rather than PRUNED_LAST_REFRESH_DATE.

To override the default DAC behavior, add the following SQL to the customsql.xml file before the last line in the file, which reads as </SQL_Queries>. The customsql.xml file is located in the dac\CustomSQLs directory.

```
<SqlQuery name = "CHANGE_CAPTURE_FULL_LOAD">
TRUNCATE TABLE S_ETL_I_IMG_%SUFFIX
;
TRUNCATE TABLE S_ETL_D_IMG_%SUFFIX
;
</SqlQuery>

<SqlQuery name = "CHANGE_CAPTURE_INCREMENTAL_LOAD">
TRUNCATE TABLE S_ETL_I_IMG_%SUFFIX
;
INSERT %APPEND INTO S_ETL_I_IMG_%SUFFIX
     (ROW_ID, MODIFICATION_NUM, OPERATION, LAST_UPD)
     SELECT
          ROW_ID
          ,MODIFICATION_NUM
          ,'I'
          ,LAST_UPD
     FROM
          %SRC_TABLE
     WHERE
          %SRC_TABLE.LAST_UPD > %LAST_REFRESH_TIME
          %FILTER
INSERT %APPEND INTO S_ETL_I_IMG_%SUFFIX
     (ROW_ID, MODIFICATION_NUM, OPERATION, LAST_UPD)
     SELECT
          ROW_ID
          ,MODIFICATION_NUM
          ,'D'
          ,LAST_UPD
     FROM
          S_ETL_D_IMG_%SUFFIX
     WHERE NOT EXISTS
     (
          SELECT
                 'X'
          FROM
                 S_ETL_I_IMG_%SUFFIX
          WHERE
                 S_ETL_I_IMG_%SUFFIX.ROW_ID = S_ETL_D_IMG_%SUFFIX.ROW_ID
          )
;
</SqlQuery>

<SqlQuery name = "CHANGE_SYNC_INCREMENTAL_LOAD">
DELETE
FROM S_ETL_D_IMG_%SUFFIX
WHERE
     EXISTS
```

```
     (SELECT
                 'X'
     FROM
                 S_ETL_I_IMG_%SUFFIX
     WHERE
                 S_ETL_D_IMG_%SUFFIX .ROW_ID = S_ETL_I_IMG_%SUFFIX.ROW_ID
                 AND S_ETL_I_IMG_%SUFFIX.OPERATION = 'D'
     )
;
</SqlQuery>
```

Performance Tip: Omit the Process to Eliminate Duplicate Records

When the Siebel change capture process runs on live transactional systems, it can run into deadlock issues when DAC queries for the records that changed since the last ETL process. To alleviate this problem, you need to enable "dirty reads" on the machine where the ETL is run. If the transactional system is on a database that requires "dirty reads" for change capture, such as MSSQL, DB2, or DB2-390, it is possible that the record identifiers columns (ROW_WID) inserted in the S_ETL_I_IMG table may have duplicates. Before starting the ETL process, DAC eliminates such duplicate records so that only the record with the smallest MODIFICATION_NUM is kept. The SQL used by DAC is as follows:

```
<SqlQuery name = "FIND_DUPLICATE_I_IMG_ROWS">
SELECT
     ROW_ID, LAST_UPD, MODIFICATION_NUM
FROM
     S_ETL_I_IMG_%SUFFIX A
WHERE EXISTS
     (
     SELECT B.ROW_ID, COUNT(*)  FROM S_ETL_I_IMG_%SUFFIX B
               WHERE B.ROW_ID = A.ROW_ID
                 AND B.OPERATION = 'I'
                 AND A.OPERATION = 'I'
     GROUP BY
         B.ROW_ID
     HAVING COUNT(*) > 1
     )
AND A.OPERATION = 'I'
ORDER BY 1,2
</SqlQuery>
```

However, for situations where deadlocks and "dirty reads" are not an issue, you can omit the process that detects the duplicate records by using the following SQL block. Copy the SQL block into the customsql.xml file before the last line in the file, which reads as </SQL_Queries>. The customsql.xml file is located in the dac\CustomSQLs directory.

```
<SqlQuery name = "FIND_DUPLICATE_I_IMG_ROWS">
SELECT
     ROW_ID, LAST_UPD, MODIFICATION_NUM
FROM
     S_ETL_I_IMG_%SUFFIX A
WHERE 1=2
</SqlQuery>
```

Performance Tip: Manage Change Capture Views

DAC drops and creates the incremental views for every ETL process. This is done because DAC anticipates that the transactional system may add new columns on tables to track new attributes in the data warehouse. If you do not anticipate such changes in the production environment, you can set the DAC system property "Drop and Create Change Capture Views Always" to "false" so that DAC will not drop and create incremental views. On DB2 and DB2-390 databases, dropping and creating views can cause deadlock issues on the system catalog tables. Therefore, if your transactional database type is DB2 or DB2-390, you may want to consider setting the DAC system property "Drop and Create Change Capture Views Always" to "false." For other database types, this action may not enhance performance.

Note: If new columns are added to the transactional system and the ETL process is modified to extract data from those columns, and if views are not dropped and created, you will not see the new column definitions in the view, and the ETL process will fail.

Performance Tip: Determine Whether Informatica Filters on Additional Attributes

DAC populates the S_ETL_I_IMG tables by querying only for data that changed since the last ETL process. This may cause all of the records that were created or updated since the last refresh time to be extracted. However, the extract processes in Informatica may be filtering on additional attributes. Therefore, for long-running change capture tasks, you should inspect the Informatica mapping to see if it has additional WHERE clauses not present in the DAC change capture process. You can modify the DAC change capture process by adding a filter clause for a <named source>.<table name> combination in the ChangeCaptureFilter.xml file, which is located in the dac\CustomSQLs directory.

SQL for Change Capture and Change Capture Sync Processes

The SQL blocks used for the change capture and change capture sync processes are as follows:

```
<SqlQuery name = "CHANGE_CAPTURE_FULL_LOAD">
TRUNCATE TABLE S_ETL_I_IMG_%SUFFIX
;
TRUNCATE TABLE S_ETL_R_IMG_%SUFFIX
;
TRUNCATE TABLE S_ETL_D_IMG_%SUFFIX
;
INSERT %APPEND INTO S_ETL_R_IMG_%SUFFIX
     (ROW_ID, MODIFICATION_NUM, LAST_UPD)
     SELECT
          ROW_ID
          ,MODIFICATION_NUM
          ,LAST_UPD
     FROM
          %SRC_TABLE
     WHERE
          LAST_UPD > %PRUNED_ETL_START_TIME
          %FILTER
;
</SqlQuery>

<SqlQuery name = "CHANGE_CAPTURE_INCREMENTAL_LOAD">
TRUNCATE TABLE S_ETL_I_IMG_%SUFFIX
;
```

```
INSERT %APPEND INTO S_ETL_I_IMG_%SUFFIX
     (ROW_ID, MODIFICATION_NUM, OPERATION, LAST_UPD)
     SELECT
          ROW_ID
          ,MODIFICATION_NUM
          ,'I'
          ,LAST_UPD
     FROM
          %SRC_TABLE
     WHERE
          %SRC_TABLE.LAST_UPD > %PRUNED_LAST_REFRESH_TIME
          %FILTER
          AND NOT EXISTS
          (
          SELECT
                  ROW_ID
                  ,MODIFICATION_NUM
                  ,'I'
                  ,LAST_UPD
          FROM
                  S_ETL_R_IMG_%SUFFIX
          WHERE
                  S_ETL_R_IMG_%SUFFIX.ROW_ID = %SRC_TABLE.ROW_ID
                  AND S_ETL_R_IMG_%SUFFIX.MODIFICATION_NUM = %
SRC_TABLE.MODIFICATION_NUM
                  AND S_ETL_R_IMG_%SUFFIX.LAST_UPD = %
SRC_TABLE.LAST_UPD
                )
;
INSERT %APPEND INTO S_ETL_I_IMG_%SUFFIX
          (ROW_ID, MODIFICATION_NUM, OPERATION, LAST_UPD)
          SELECT
                  ROW_ID
                  ,MODIFICATION_NUM
                  ,'D'
                  ,LAST_UPD
          FROM
                  S_ETL_D_IMG_%SUFFIX
          WHERE NOT EXISTS
          (
                  SELECT
                          'X'
                  FROM
                          S_ETL_I_IMG_%SUFFIX
                  WHERE
                          S_ETL_I_IMG_%SUFFIX.ROW_ID = S_ETL_D_IMG_%
SUFFIX.ROW_ID
          )
;
</SqlQuery>

<SqlQuery name = "CHANGE_SYNC_INCREMENTAL_LOAD">
DELETE
FROM S_ETL_D_IMG_%SUFFIX
WHERE
          EXISTS
          (
          SELECT
                    'X'
          FROM
```

```
                    S_ETL_I_IMG_%SUFFIX
          WHERE
                    S_ETL_D_IMG_%SUFFIX.ROW_ID = S_ETL_I_IMG_%SUFFIX.ROW_ID
                    AND S_ETL_I_IMG_%SUFFIX.OPERATION = 'D'
          )
;
DELETE
FROM S_ETL_I_IMG_%SUFFIX
WHERE LAST_UPD &lt; %PRUNED_ETL_START_TIME
;
DELETE
FROM S_ETL_I_IMG_%SUFFIX
WHERE LAST_UPD &gt; %ETL_START_TIME
;
DELETE
FROM S_ETL_R_IMG_%SUFFIX
WHERE
          EXISTS
          (
           SELECT
                    'X'
          FROM
                    S_ETL_I_IMG_%SUFFIX
          WHERE
                    S_ETL_R_IMG_%SUFFIX.ROW_ID = S_ETL_I_IMG_%SUFFIX.ROW_ID
          )
;
INSERT %APPEND INTO S_ETL_R_IMG_%SUFFIX
          (ROW_ID, MODIFICATION_NUM, LAST_UPD)
          SELECT
                  ROW_ID
                  ,MODIFICATION_NUM
                  ,LAST_UPD
          FROM
                  S_ETL_I_IMG_%SUFFIX
;
DELETE FROM S_ETL_R_IMG_%SUFFIX WHERE LAST_UPD &lt; %PRUNED_ETL_START_TIME
;
</SqlQuery>
```

Performance Tuning the ETL Process Using Tasks by Depth Command

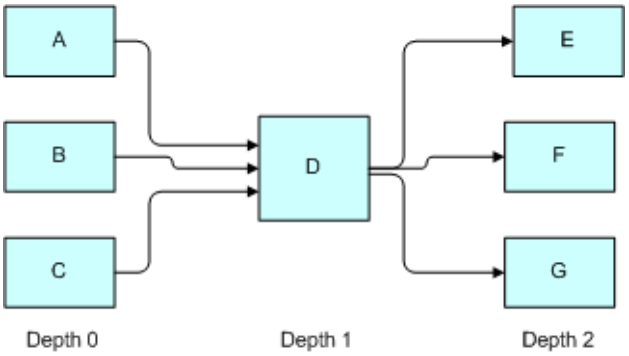
The Tasks by Depth right-click command is available in the Execution Plans tab of the Execute view. This command provides a data representation of the execution graph by showing how many tasks in an execution plan are at each depth level. The depth specifies the level of dependency.

For example, a task at depth 4 depends on one or more tasks at the level of depth 3, and a task at depth 3 depends one or more tasks at the level of depth 2, and so on. When an execution plan runs, DAC moves any task whose predecessors have completed into the execution queue. Therefore, tasks at a higher depth can execute at the same time as tasks with a lower depth.

Knowing how many tasks are at each depth level can help with performance tuning. A smaller number of tasks at any given depth can signify slow performance. For example, in Figure 10–1 task D is the only task at depth 1. This means that all of the tasks in depth 2 depend on task D. Situations such as this in which there is a small

number of tasks at a depth level can cause slow performance and elongate the ETL process. You may want to tune the tasks at depths that have a small number of tasks.

Figure 10–1 Tasks by Depth



11

Working With DAC Metadata Patches

The DAC metadata patch feature enables you to import and export subsets of DAC metadata at a fine grain. This allows you to move subsets of data from one environment to another. For example, when issues are fixed in a development environment, you can move the updated metadata to QA or production environments. This approach can be used with custom source system containers only and not with predefined ("out-of-the-box") containers

This chapter contains the following topics:

DAC Metadata Patching Life Cycle

Figure 11–1 shows the process of distributing metadata to other environments. The steps in the process are as follows:

- **Step 1**: **Create a new, empty patch.** DAC automatically assigns the patch version value as 1 and the status as Open. Open patches are listed in the Working Patches tab in the Setup view.

  For instructions on creating a patch, see "Creating a Patch".

- **Step 2: Add contents to the patch.** A patch must have an Open status in order for objects to be added or removed. The patch stores pointers to the repository objects. For information about what objects you can add to a patch, see "About Patch Contents".

  For instructions on adding contents to a patch, see "Adding Contents to a Patch".

- **Step 3: Close the patch.** When you change the patch status to Closed, the patch can no longer be edited. You can reopen a Closed patch in order to edit it. When you change the status from Closed to Open, the patch version is automatically incremented by 1. Closed patches are listed in the Working Patches tab.

  For instructions on closing a patch, see "Changing the Status of a Patch".

- **Step 4: Export the patch.** When you export the patch, the patch and its contents are stored in an XML file in a directory you specify.

  For instructions on exporting a patch, see "Exporting a DAC Metadata Patch".

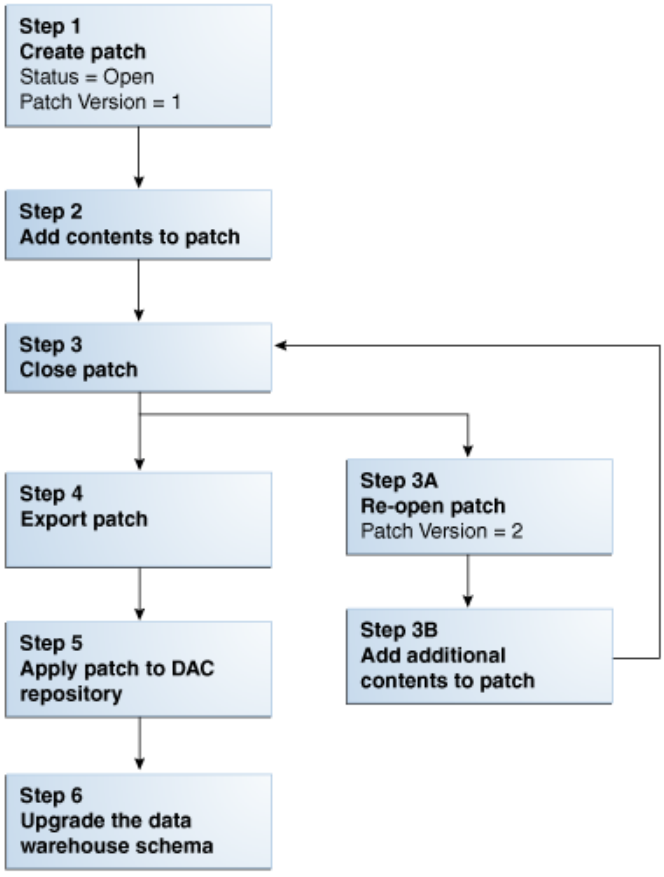
- **Step 5**: **Apply the patch to the DAC repository.** When you apply a patch, the DAC repository is upgraded with the patch contents stored in the XML file. After the patch process, the patch status will be one of the following:
  - **Completed**. All objects were applied successfully to the DAC repository.
  - **Incomplete.** The process of applying the patch completed with one or more errors.

  For information about why objects are not applied to the DAC repository, see "When Does a Patch Fail to be Applied to the DAC Repository?" Applied patches are listed in the Applied Patches tab in the Setup view.

  For instructions on applying a patch, see "Applying a DAC Metadata Patch to the DAC Repository".
- **Step 6:** Upgrade the data warehouse schema. If the patch contents included column-level changes, you need to upgrade the data warehouse schema with the same updates that were made to the DAC repository.

  For instructions on upgrading the data warehouse schema, see Chapter 9, "Managing Data Warehouse Schemas."

Figure 11–1 DAC Patch Process



Creating a DAC Metadata Patch

The process for creating a DAC metadata patch comprises the following steps:

- Creating a Patch
- About Patch Contents
- Adding Contents to a Patch
- Changing the Status of a Patch

For a description of the complete DAC metadata patching life cycle, see "DAC Metadata Patching Life Cycle".

Creating a Patch

You can create a patch in the following ways:

- Use the Working Patches tab to create a new, empty patch.
- Create a new patch at the time you add the first contents

  After you create the patch, you then add contents to it. See "About Patch Contents" for information about which objects can be added to a patch, and "Adding Contents to a Patch" for instructions on adding patch contents.

To create a new patch in the Working Patches tab

1. In the Setup view, select the **Working Patches** tab.
2. In the toolbar, click **New**.
3. Enter a descriptive name for the patch, and then click **Save**.

   DAC automatically sets the status as Open and the Patch Version as 1.

To create a new patch at the time you add the first contents

1. In the Design view, select the appropriate tab that contains the contents you want to add to the patch.
2. Query for and select the object you want to add to the patch.
3. Right-click the object, and select **Add Object to Patch**.
4. In the Patching... dialog, select one of the following:

   - **Selected record only**. To add only the selected record to the patch.

   - **All records in the list**. To add all the records in the list of query results to the patch.
5. In the Patches dialog, do the following:

   a. Click **Create a New Patch**.

   b. Enter a descriptive name for the patch.

   c. Click **OK**.

   The new patch will appear in the Working Patches tab. DAC automatically sets the status as Open and the Patch Version as 1.

About Patch Contents

The objects listed below are eligible to be included as contents in a patch. You can manually select any of these objects to be included in a patch. Alternatively, if you

choose to add contents based on a timestamp range, the objects from the list below that are new or changed in the timestamp range will be included in the patch.

- **Container-dependent objects** (objects visible in the DAC Design view)

  When you add a parent object to a patch, you can choose whether to have only the the immediate child objects added or the children of the child objects as well. Table 11–1 lists the parent objects you can add to a patch as well as the immediate child objects that will be added.

  When you add a container-dependent object to a patch, you have the option to define a timestamp range that acts as a filter so that only the child objects that are new or changed during the time period you specify will be added to the patch.

  Note: When you add a child object to a patch, the parent object is automatically included as well.

Table 11–1 Objects and Immediate Child Objects Added to Patches

Parent Object Added to Patch	Immediate Child Objects Also Added to Patch
Subject area	■ Configuration tags ■ Tables ■ Tasks **Note:** You need to reassemble the subject area after the patch has been applied to the target repository.
Tables	■ Actions ■ Columns ■ Indexes ■ Multi-Column Statistics ■ Related tables
Indexes	■ Actions ■ Columns
Tasks	■ Actions ■ Conditional tables ■ Configuration tags ■ Extended properties ■ Execution type ■ Phase dependencies ■ Parameters ■ Refresh date tables ■ Source tables ■ Target tables
Task groups	Child tasks
Configuration tags	■ Subject areas ■ Tasks
Source system folders	■ Logical folders ■ Physical folders
Source system parameters	None

Table 11–1 (Cont.) Objects and Immediate Child Objects Added to Patches

Parent Object Added to Patch	Immediate Child Objects Also Added to Patch
Source system actions	None

- **Execution plans and execution instances**

  You can add execution plans and execution instances to patches. When you do so, only the execution plan parent object or execution instance parent object is listed in the Contents subtab of the Working Patches and Applied Patches tabs. However, the following child objects will be exported to the patch XML file even though they do not appear in the Contents subtab:

  – Connectivity Parameters
  – Tables
  – Concurrent Dependencies
  – Preceding Tasks
  – Following Tasks
  – Ordered Tasks
  – Subject Areas
  – Immediate Dependencies

    Note: The data under the All Dependencies subtab is not exported to the patch XML file. The data under the All Dependencies subtab can be computed in the target repository by selecting the All Dependencies subtab, and then clicking **Go** in the subtab toolbar. After the query results appear, click **Generate**.

  Note the following points:

  – If an execution plan or execution instance contains tasks or subject areas from source system containers that are not in the DAC repository, then the execution plan or execution instance will not be applied in the target repository.

    For example, if an execution plan has subject areas and tasks from container 1 and container 2 but the target DAC repository only contains metadata for container 3, the execution plan will not be applied to the target DAC repository.

- **Seed data objects** (objects defined or viewed by clicking Tools on the menu bar, and then selecting Seed Data)

  – Actions (index, task, table)
  – Execution types
  – Global external parameters
  – Heuristics
  – Logical data sources
  – Task logical folders
  – Task phases
  – Task physical folders

- **Patches**

From the Working Patches tab in the Setup view, you can:

- Add a patch to another patch. A patch added to another patch becomes the child patch. The parent patch is referred to as a *cumulative patch*. A cumulative patch can include both objects and other patches (including the patch contents).
- Add objects to a patch based on a timestamp range and a distinct set of containers. Only objects that are new or changed during a timestamp range you specify and that reside within one or more containers that you specify will be added to the patch.

Adding Contents to a Patch

You can select any of the objects listed in the section "About Patch Contents" to be included in a patch.

Follow one of the procedures below to add to an object to a patch:

- To add container-specific contents to a patch
- To add an execution plan to a patch
- To add a seed data object to a patch
- To add a patch as a child patch to another patch
- To add contents to a patch based on a timestamp range and distinct containers

After you have added contents to a patch, you can verify the results by following the steps in "To view the contents of a patch".

To add container-specific contents to a patch

1. Review the section "About Patch Contents" for information about which repository objects you can add to a patch.
2. Make sure the patch that you want to add contents to has the status Open. You can only add contents to an Open patch. If the patch is Closed, you can reopen it. For instructions, see "Changing the Status of a Patch".
3. Select the appropriate source system container from the drop-down list.
4. Go to the appropriate tab, and query for the objects you want to add to the patch.
5. In the list of query results, right-click, and select **Add Object(s) to Patch**.
6. In the Patching... dialog, select one of the following:

   - **Selected record only**. To add only the selected record to the patch.

   - **All records in the list**. To add all the records in the list of query results to the patch.
7. In the Patches dialog, do one of the following:

   - Add the object to an existing patch by selecting **Select a patch**, and then clicking in the field below to open a dialog from which you can select the appropriate patch.

   - Create a new patch to which the object will be added by selecting **Create a new patch**, and entering a descriptive name for the patch in the text field.
8. In the Add to Patch Set area, select one of the following:

   - **Selected parent object(s) and immediate child object(s)**

An immediate child object is the object represented by the object subtab in the DAC user interface. For example, if you select a subject area, its immediate child objects are task links, table links, and configuration tag links.

- **Selected parent object(s) and all child object(s)**

When you select to have *all* child objects added to a patch, then the children of the child objects are also added. For example, a subject area is a parent object. The tasks, tables and configuration tag links are its immediate child objects. Actual tasks, tables and configuration tags are pulled into the patch because of the links. The child objects have child objects of their own. For example, a table child object has columns as its child objects.

9. (Optional) In the Select Time Stamps area, specify a timestamp range to act as a filter to so that only the child objects that are new or changed during the time period you specify will be added to the patch:

   - Click in the **From Timestamp** field to open the Date dialog, and configure the beginning timestamp.

   - Click in the **To Timestamp** field to open the Date dialog, and configure the ending timestamp.

Note: If you added a subject area to a patch, you need to reassemble the subject area after the patch has been applied to the target repository.

To add an execution plan to a patch

1. Review the section "About Patch Contents" for information about which repository objects you can add to a patch.
2. Make sure the patch that you want to add contents to has the status Open. You can only add contents to an Open patch. If the patch is Closed, you can reopen it. For instructions, see "Changing the Status of a Patch".
3. In the Execute view, go to the **Execution Plans** tab.
4. Query for the execution plan you want to add to the patch.
5. Right-click the execution plan, and then select **Add Object(s) to Patch**.
6. In the Patching... dialog, select one of the following:

   - **Selected record only**. To add only the selected record to the patch.

   - **All records in the list**. To add all the records in the list of query results to the patch.

7. In the Patches dialog, do one of the following:

   - Add the object to an existing patch by selecting **Select a patch**, and then clicking in the field below to open a dialog from which you can select the appropriate patch.

   - Create a new patch to which the object will be added by selecting **Create a new patch**, and entering a descriptive name for the patch in the text field.

To add a seed data object to a patch

1. Review the section "About Patch Contents" for information about which repository objects you can add to a patch.
2. Make sure the patch that you want to add contents to has the status Open. You can only add contents to an Open patch. If the patch is Closed, you can reopen it. For instructions, see "Changing the Status of a Patch".

3. On the **Tools** menu, select **Seed Data**, and then select the appropriate menu item.
4. In the dialog that opens, right-click the object you want to add to the patch, and then select **Add Object(s) to Patch**.
5. In the Patching... dialog, select one of the following:

   - **Selected record only**. To add only the selected record to the patch.

   - **All records in the list**. To add all the records in the list of query results to the patch.
6. In the Patches dialog, do one of the following:

   - Add the object to an existing patch by selecting **Select a patch**, and then clicking in the field below to open a dialog from which you can select the appropriate patch.

   - Create a new patch to which the object will be added by selecting **Create a new patch**, and entering a descriptive name for the patch in the text field.

To add a patch as a child patch to another patch

1. Review the section "About Patch Contents" for information about which repository objects you can add to a patch.
2. Make sure the patch that you want to add contents to has the status Open. You can only add contents to an Open patch. If the patch is Closed, you can reopen it. For instructions, see "Changing the Status of a Patch".
3. In the Working Patches tab of the DAC Setup view, select the patch that you want to add to another patch.
4. Right-click, select **Patches**, and then select **Add as Child Patch(es)**.
5. Do one of the following:
   - Add the object to an existing patch by selecting **Select a patch**, and then clicking in the field below to open a dialog from which you can select the appropriate patch.
   - Create a new patch to which the object will be added by selecting **Create a new patch**, and entering a descriptive name for the patch in the text field.

To add contents to a patch based on a timestamp range and distinct containers

Follow this procedure to add objects to a patch based on a timestamp range and a distinct set of containers. Only objects that are new or changed during a timestamp range you specify and that reside within one or more containers that you specify will be added to the patch.

1. Review the section "About Patch Contents" for information about which repository objects you can add to a patch.
2. Make sure the patch that you want to add contents to has the status Open. You can only add contents to an Open patch. If the patch is Closed, you can reopen it. For instructions, see "Changing the Status of a Patch".
3. In the Working Patches tab of the DAC Setup view, select the patch to which you want to add contents.
4. Right-click, select **Patches**, and then select **Add Objects to Patch (Time Range)**.

   The Patching... dialog opens.

5. Set up a timestamp range so that only objects that are new or changed during this time period are added to the patch:
   a. Click in the **From Timestamp** field to open the Date dialog, and configure the beginning timestamp.
   b. Click in the **To Timestamp** field to open the Date dialog, and configure the ending timestamp.
6. Select one or more containers that hold the objects you want to add to the patch.
7. Click **OK**.

To view the contents of a patch

- Click the **Contents** subtab to view container-dependent and seed data objects. Both parent and child objects are listed in the Contents subtab.
- Click the **Child Patches** subtab to view child patches added to the patch. Only the patch (and not the patch contents) is listed in the Child Patches subtab. For nested patches—that is, a child patch that has its own child patch—only the immediate child patch is listed in the Child Patches subtab.

Changing the Status of a Patch

After you have added contents to a patch, you need to change the patch status to Closed in order to export it. You can re-open a Closed patch if you need to add or remove objects. Each time you change the status from Closed to Open, the Patch Version value is incremented by one.

To change the status of a patch

1. Go to the **Working Patches** tab in the Setup view.
2. Query for the patch whose status you want to change.
3. Right-click, select **Patches**, and then select one of the following:
   - **Re-Open Patch**. Changes the status of Closed patches to Open.
   - **Close Patch**. Changes the status of Open patches to closed.

   The new patch status is displayed on the Status column of the Working Patches tab.

Exporting a DAC Metadata Patch

When you export a patch, the patch and its contents are exported from the DAC repository and saved in an XML file in a directory you specify. The XML file contains identifying information about the patch and the patch contents, including the repository object name, the object's source system container, and the parent object of child objects.

Note: If you inactivate a patch content entry, the object will not be exported to the XML file.

To export a patch

1. In the DAC Setup view, select the **Working Patches** tab.
2. Query for the patch you want to export.

3. Make sure the patch status is Closed. If the patch is Open, you need to close it by following the instructions in "Changing the Status of a Patch".
4. Right-click the patch, select **Patches**, and then select **Export Patch**.
5. In the Save dialog:
   a. Select the directory to which you want to export the patch.
   b. In the File Name field, you can leave the default name of the file, which is in the format <patchName><version>.xml, or you can enter a new name.
   c. Click **Save**.

Applying a DAC Metadata Patch to the DAC Repository

When you apply a patch, the patch contents are applied to the DAC repository. You can re-apply a patch as many times as needed. The patch history is tracked by the patch version and timestamps for when the patch was closed, exported, and applied to the repository. This information appears in the Audit Trails subtab of the Working Patches tab.

To apply a patch to the DAC repository

1. On the **Tools** menu, select **DAC Repository Management**, then select **Apply Patch**.
2. In the Open dialog:
   a. Select the folder that contains the XML file you want to import.
   b. Select the XML file you want to import.
   c. Click **Open**.
3. A message dialog informs you whether the process was successful. Click **OK** to close the dialog.

   The applied patches are listed in the Applied Patches tab. The status of an applied patch can be one of the following:

   - **Completed.** All objects in the patch were successfully applied to the DAC repository
   - **Incomplete.** Some objects were successfully applied, but other objects failed to be applied.
   - **Failed.** No objects were applied to the DAC repository.
4. Update the data warehouse schema to reflect the patch applied to the DAC repository. For information about updating the schema, see "Creating, Upgrading or Dropping Subsets of Tables in the Schema for Oracle Databases".

   Note: You can only update the data warehouse schema with a subset of tables (tables included in a patch) if you are using an Oracle database for the data warehouse.

When Does a Patch Fail to be Applied to the DAC Repository?

Patches can fail to be applied to the DAC repository for the following reasons:

- The container name in the XML file does not match the container name in the DAC repository.

- An object in the XML file has a container name that does not exist in the DAC repository.

Object Ownership During the Patching Process

Objects included as patch contents will maintain their ownership state. For example, original objects will be applied to the target repository as originals, clones as clones, and references as references. However, if objects in a customized source system container are updated by a patch, they become cloned objects. If the objects do not already exist, they will be applied to the repository as original objects. The properties of objects in the XML file will take precedence over those in the target repository. That is, if objects on the repository have changed, their properties will be overwritten by those in the XML file.

For more information about object ownership, see "About Object Ownership in DAC".

Exporting and Applying Patches Using the Command Line

You can use a command line to export patches and apply patches to the DAC repository. The command line parameters are exposed in the AutomationUtils.bat file, which is located in the <Domain_Home>\dac directory. For information about how to use the parameters, see "DAC Repository Command Line Parameters".

12

Common Tasks Performed in the DAC

This chapter provides instructions for performing common DAC tasks.

This chapter contains the following topics:

Accessing the DAC Server Using the Command Line

You can access the DAC Server through a command line to start and stop execution plans and to get status information for servers, databases, and execution plans. This feature enables you to access the DAC Server using third-party administration or management tools, without using the DAC Client. Refer to the file dacCmdLine.bat/.sh for usage.

This section includes the following topics:

Setting Up Command Line Access to the DAC Server

The Command Line utility enables you to invoke commands on a DAC Server running on a remote or local machine.

To set up command line access to the DAC Server

1. Make sure you have installed the supported version of the Java SDK.
2. Copy the following files from the <Domain_Home>\dac directory to a local directory:

- DAWSystem.jar
- dac.properties
- dacCmdLine.bat or dacCmdLine.sh

3. In the dacCmdLine.bat file, do the following:
   a. Edit the JAVA_HOME variable to point to the directory where the Java SDK is installed.

      Make sure there are no spaces in the path reference.

   b. Edit the DAC_HOME variable to point to the directory where the DAC is installed.
4. In the dac.properties file, edit the following parameter values.

Parameter	Value
ServerHost=	Host name of the DAC Server.
ServerPort=	Port of the DAC Server. The default is 3141.
RepositoryStampVal=	Repository stamp that appears in the DAC Client Login Details screen. To find this value, in the DAC Client navigate to Help, then select Login Details.

The dac.properties file should look similar to the following:

```
ServerHost=<host name> ServerPort=3141
RepositoryStampVal=851E0677D5E1F6335242B49FCCd6519
```

Using the Command Line to Access the DAC Server

Follow this procedure to use the command line to access the DAC Server.

To use the command line to access the DAC Server

- At the command prompt, enter the following:

  ```
  dacCmdLine <method name> <optional execution plan name>
  ```

  where `method name` is one of the following:

Method Name	Description
StartServer	Starts the DAC Server (Web mode only).
StopServer	Stops the DAC Server (Web mode only).
StartETL	Starts an execution plan. You must specify an execution plan name. -wait option lets you start the execution plan in synchronous mode.
StopETL	Stops the operation of an execution plan. You must specify an execution plan name.
ETLStatus	If you do not specify an execution plan name, the status of the execution plan that last ran is returned. If you specify an execution plan name, the status of the specified execution plan is returned.
DatabaseStatus	Verifies whether the DAC Server can connect to all active database connections. You do not need to specify an execution plan name.
InformaticaStatus	Verifies whether the DAC Server is able to ping all active Informatica PowerCenter Services machines.

Note: The method names are case insensitive. Execution plan names are case sensitive. Also, if the execution plan name contains spaces, place beginning and ending double quotes around the name.

For example:

Command Line	Description
`dacCmdLine EtlStatus`	Returns the status of the execution plan that last ran.
`dacCmdLine EtlStatus Forecast`	Returns the status of the last instance of the Forecast execution plan.
`dacCmdLine StopEtl Forecast`	If the execution plan currently running is Forecast, the operation will be terminated. Otherwise, the request is ignored.
`dacCmdLine databasestatus`	Returns the health status of all the database connections as defined in the DAC repository from the DAC Server.
`dacCmdLine InformaticaStatus`	Returns the health status of all the Informatica PowerCenter Services connections as defined in the DAC Client on the Informatica Services tab.

Command Line Status Monitoring Queries

The command line feature enables you to get the following status information:

- Summary of the requested execution plan. If there are multiple instances of the same execution plan, a summary of the instance that last ran is returned. Below is an example of the information contained in the summary.

```
(c) 2009 Oracle
Oracle DAC Server comprising the etl execution-management,  scheduler, logger,
and network server.
ETL details for the last run:
ETL Process Id : 255 ETL Name : Complete ETL Run Name : DRY RUN OF Complete
ETL: ETL Run - 2004-06-17 18:30:13.201 DAC Server : (<host name>) DAC Port :
3141 Status: Stopped Log File Name: Complete_ETL.255.log Database Connection(s)
Used :


OLTP jdbc:microsoft:sqlserver://<host name>:1433;DatabaseName=OLTP Data
Warehouse jdbc:microsoft:sqlserver://<host name>:1433;DatabaseName=olap

Informatica Server(s) Used  :

InformaticaServer4-<host name>:(4)  InformaticaServer2-<host name>:(2)
InformaticaServer3-<host name>:(3)  InformaticaServer1-<host name>:(10)

Start Time: 2004-06-17 19:00:06.885 Message: ETL was interrupted Actual Start
Time: 2004-06-17 18:30:13.357 End Time: 2004-06-17 19:05:56.781 Total Time
Taken: 35 Minutes
Start Time For This Run: 2004-06-17 19:00:06.885 Total Time Taken For This Run:
5 Minutes
Total steps: 212 Running steps: 0 Complete steps: 142 Failed/Stopped steps:70
```

- Summary of connection status to all active databases and Informatica servers.

DAC Repository Command Line Parameters

This section describes the DAC repository command line properties that are exposed by the AutomationUtils.bat file, which is located in the <Domain_Home>\dac directory.

Note: Before you can use the command line properties, you must first configure DAC repository database and authentication properties in the automationUtils.properties file, which is located in the <Domain_Home>\dac directory. Alternatively, you can create your own connection file that contains the repository database and authentication properties. Many command line properties discussed in this section require the path to the connection file (either automationUtils.properties or a user-defined connection file). The path to the connection file must include the file name.

The following DAC repository command line parameters are available in the AutomationUtils.bat file:

- Analyze DAC Schema
- Apply Distributed Dev Patch
- Assemble Subject Area
- Build Execution Plan
- Change Encryption Key
- Clear Encrypted Data
- Command Credentials
- Create DAC Schema
- Create DAC User
- Create Data Warehouse Schema
- Create Patch of Objects Between Time Stamps
- Database Credentials
- Delete Objects Execution Plan
- Delete Objects Subject Area
- Drop DAC Schema
- Drop Data Warehouse Schema Using Schema Definition File
- Export DAC Metadata by Application
- Export DAC Metadata by Categories
- Export Patch
- JKS Password
- Generate DW Schema Definition File
- Import DAC Metadata by Application
- Import DAC Metadata by Categories
- Repository XML Snapshot
- Server Setup
- Set Password

- Upgrade DAC Schema
- Upgrade Data Warehouse Schema Using Schema Definition File

Analyze DAC Schema

The analyzeDACSchema property analyzes the DAC repository tables.

Syntax:

```
<path to connection file> analyzeDACSchema
```

Apply Distributed Dev Patch

The applyDistributedDevPatch property applies a patch to the DAC repository.

Syntax:

```
<path to connection file> applyDistributedDevPatch <file name>
```

where:

Parameter	Description
`file name`	Name of the XML file.

Assemble Subject Area

The assembleSubjectArea property assembles a subject areas.

Syntax:

```
<path to connection file> assembleSubjectArea <container name> <subject area name>
```

Build Execution Plan

The buildExecutionPlan property builds an execution plan.

Syntax:

```
<path to connection file> buildExecutionPlan <execution plan name>
```

Change Encryption Key

The changeEncryptionKey property changes the encryption key in the credentials file and re-encrypts all encrypted information in the DAC repository.

Syntax to change encryption key with a randomly generated key:

```
<path to connection file> changeEncryptionKey -randomkey
```

Syntax to change encryption key by explicitly specifying the key from command line input:

```
<path to connection file> changeEncryptionKey
```

Clear Encrypted Data

The clearEncryptedData property removes all encrypted data from the DAC repository.

Syntax:

```
<path to connection file> clearEncryptedData <schema owner>
```

Note: An authentication file is not required for this operation. The repository password will be taken from command line input.

> **Caution:** This operation will clear all database, Informatica, and external executor passwords as well as all DAC user data. Use this command only when the authentication file with the encryption key was lost and cannot be recovered. After removing the encrypted data from the DAc repository, generate the authentication file with a new encryption key, and log into the DAC repository using the DAC Client. When prompted to add the encryption key, update all passwords in the DAC repository and distribute the new authentication file to all required DAC users.

Command Credentials

The cmdCredentials parameter stores credentials for the DAC command line utility for both the Fusion Middleware and DAC standalone authentication modes.

Syntax:

```
cmdCredentials <path to cwallet.sso file> <user name>
```

Note: The password will be taken from command line input. Also, make sure to include the cwallet.sso file name in the path.

Create DAC Schema

The createDACSchema property creates the schema of a new DAC repository.

Syntax:

```
<path to connection file> createDACSchema <unicode> <tablespace>
```

where:

Parameter	Description
`unicode`	Specifies whether the schema is created as unicode. Possible values are `true` and `false`.
`tablespace`	The name of the default tablespace in which the schema is created. Surround the name with double quotation marks. **Note:** If the value for the AuthenticationType property in the connection file is `FMW`, then user credentials will be used for authentication.

Create DAC User

The createDACUser property creates a DAC schema user.

Syntax for FMW authentication:

```
<path to connection file> createDACUser <user name> [role1] [role2] ...
```

Syntax for database authentication:

```
<path to connection file> createDACUser <user name> [role1] [role2] ...
[-dbVerification <schema owner>]
```

Note: The user password will be taken from command line input.

Create Data Warehouse Schema

The createDWSchema property creates the data warehouse schema using the schema definition file.

Syntax:

```
<path to connection file> createDWSchema <dwSchemaDefinitionFileName> <unicode>
<tablespace>
```

where:

Parameter	Description
dwSchemaDefinitionaFileName	Specifies the data warehouse schema definition file location and name. This file is generated by the generateDWSchemaDefinitionFile command line parameter.
tablespace	The name of the default tablespace in which the schema is created. Surround the name with double quotation marks.
unicode	Specifies whether the schema is created as unicode. Possible values are `true` and `false`.

Note: The file create.sql is generated and stored in the <Domain_Home>/dac/conf/sqlgen/sql/oracle directory of an enterprise installation, and the dac/conf/sqlgen/sql/oracle directory in a standalone or client installation. User authentication is not required for this command.

Create Patch of Objects Between Time Stamps

The createPatchOfObjectsBetweenTimeStamps property creates a DAC repository metadata patch containing all eligible objects that are new or changed between two timestamps.

Syntax:

```
<path to connection file> createPatchOfObjectsBetweenTimeStamps <patch name> <from
timestamp> <to timestamp>
```

where:

Parameter	Description
from timestamp	The beginning timestamp for the timestamp range in format yyyymmdd-hh:mm:ss:ss. The hh value must be between 00-23.
to timestamp	The ending timestamp for the timestamp range. The format is the same as that of the "from timestamp."
-now	A special string to specify the current timestamp.

Database Credentials

The dbCredentials property generates an authentication file (cwallet.sso) with the DAC repository credentials and encryption key.

Syntax for explicit key and credentials:

```
dbCredentials <path to cwallet.sso file> <user name> -withKey
```

Syntax for randomly generated encryption key and credentials:

```
dbCredentials <path to cwallet.sso file> <user name> -randomKey
```

Syntax to update credentials in existing file without modifying the encryption key:

```
dbCredentials <path to cwallet.sso file> <user name> -noKey
```

Note: The user password will be taken from command line input.

Delete Objects Execution Plan

The deleteObjects ExecutionPlan property deletes executions plan from the DAC repository.

Syntax:

```
<path to connection file> deleteObjects ExecutionPlan <name 1> [name 2] ...
```

Delete Objects Subject Area

The deleteObjects SubjectArea property deletes subject areas from the DAC repository.

Syntax:

```
<path to connection file> deleteObjects SubjectArea <container name> <subject area
name 1> [subject area name 2] ...
```

Drop DAC Schema

The dropDACSchema property drops the schema of the DAC repository.

Syntax:

```
<path to connection file> dropDACSchema
```

Drop Data Warehouse Schema Using Schema Definition File

The dropDWSchemaFromSchemaDefinitionFile property drops the data warehouse schema using the schema definition file.

Syntax:

```
<path to connection file> dropDWSchemaFromSchemaDefinitionFile
<dwSchemaDefinitionFileName> <execute>
```

where:

Parameter	Description
`dwSchemaDefinitionaFileName`	Specifies the data warehouse schema definition file location and name. This file is generated by the generateDWSchemaDefinitionFile command line parameter.
execute	Possible values are `true` and `false`. If set to `true` DAC will generate and execute the SQL statements that drop the data warehouse schema. If set to `false`, the SQL statements will be generated but not executed.

Note: The file drop.sql is generated and stored in the <Domain_ Home>/dac/conf/sqlgen/sql/oracle directory of an enterprise installation, and the dac/conf/sqlgen/sql/oracle directory in a standalone or client installation. User authentication is not required for this command.

Export DAC Metadata by Application

The Export property exports DAC metadata from the DAC repository for specified source system containers.

Syntax:

```
<path to connection file> export <folder name> <container name 1> <container name 2> ...
```

where:

Parameter	Description
`folder name`	Full path to the location of the export file structure.
`container name`	(Optional) Name of the source system container for which you want to export DAC metadata. If no container is named, all containers that are found in the file structure will be exported.

Export DAC Metadata by Categories

The exportCategory property exports DAC metadata from the DAC repository based on the Logical, Run Time, or System categories.

Syntax:

```
<path to connection file> exportCategory <folder name> <category>
```

where:

Parameter	Description
`folder name`	Full path to the root of the export file structure.
`category`	Possible values are the following: ■ `logical` - Exports all data categorized as logical (information contained in the DAC Design view). ■ `runtime` - Exports all data categorized as run time (information contained in the DAC Execute view). ■ `system` - Exports all data categorized as run time (information contained in the DAC Setup view).

Export Patch

The exportPatch property exports a DAC metadata repository patch in XML format.

Syntax:

```
<path to connection file> exportPatch <file name> <patch name>
```

where:

Parameter	Description
`file name`	Name of the XML file.

JKS Password

The jksPassword parameter stores the password for the Java key store.

Syntax:

```
jksPassword <path to cwallet.sso file>
```

Note: The password will be taken from command line input. Also, make sure to include the cwallet.sso file name in the path.

Generate DW Schema Definition File

The generateDWSchemaDefinitionFile generates a data warehouse schema definition file.

Syntax:

```
<path to connection file> generateDWSchemaDefinitionfile
<dwSchemaDefinitionFileName> <container 1> <container 2> ...
```

Note: The definition file will be generated for the specified containers. If you do not specify a container, the file will be generated for all containers.

Import DAC Metadata by Application

The Import property imports DAC metadata into the DAC repository for specified source system containers.

Syntax:

```
<path to connection file> import <folder name> [-noTruncate] [-noUpdate]
<container name 1> <container name 2> ...
```

where:

Parameter	Description
`folder name`	Full path to the root of the import file structure.
`container name`	(Optional) Name of the source system container for which you want to import DAC metadata. If no container is named, all containers that are found in the file structure will be imported.
`-noTruncate`	By default the repository will be truncated upon import. Use the -noTruncate option for incremental import.
`-noUpdate`	By default existing records will be updated during incremental import. Use the -noUpdate option if you only want to insert new records.

Import DAC Metadata by Categories

The importCategory property imports DAC metadata from the DAC repository based on the Logical, Run Time, or System categories.

Syntax:

```
<path to connection file> exportCategory <folder name> <category>
```

where:

Parameter	Description
`folder name`	Full path to the root of the import file structure.

Parameter	Description
category	Possible values are the following: ■ logical - Imports all data categorized as logical (information contained in the DAC Design view). ■ runtime - Imports all data categorized as run time (information contained in the DAC Execute view). ■ system - Imports all data categorized as run time (information contained in the DAC Setup view).
-noTruncate	By default the repository will be truncated upon import. Use the -noTruncate option for incremental import.
-noUpdate	By default existing records will be updated during incremental import. Use the -noUpdate option if you only want to insert new records.

Repository XML Snapshot

The repositoryXMLSnapshot property exports a snapshot of the DAC repository in XML format.

Syntax:

```
<path to connection file> repositoryXMLSnapshot <file name>
```

where:

Parameter	Description
file name	Name of the XML file.

Server Setup

The serverSetup property sets up the DAC Server connection to the DAC repository.

Syntax:

```
<path to connection file> serverSetup <database type> <database name/instance
name/tns> <host> <port>
```

Note: If Fusion Middleware authentication is specified in the connection file, the DAC Server URL should be specified in the properties file and the DAC Server will be set to where the URL points (location can be a remote server). If DAC standalone authentication is specified in the connection file, you must configure the repository connection properties in the properties file. The DAC Server will be set up in the local installation. The user name, password, and encryption key will be taken from the database credentials (cwallet.sso) file.

Set Password

The setPassword parameter sets the password for the Informatica Integration Service, Informatica Repository Service, and physical data sources in the DAC repository.

Syntax:

```
<path to connection file> setPassword <type> <logical name>
```

where:

Parameter	Description
type	Possible values are the following: ■ server - use for Informatica Integration Service and Repository Service. ■ dbconn - use for physical data sources.
logical name	Logical name of the server or physical data source record in DAC.

Note: The password will be taken from command line input. If the logical name or password contains spaces, surround the entry in double quotes.

Upgrade DAC Schema

The upgradeDACSchema parameter upgrades the DAC repository.

Syntax:

```
<path to connection file> upgradeDACSchema
```

Upgrade Data Warehouse Schema Using Schema Definition File

The upgradeDWSchemaFromSchemaDefinitionFile property upgrades the data warehouse schema using the schema definition file.

Syntax:

```
<path to connection file> upgradeDWSchemaFromSchemaDefinitionFile
<dwSchemaDefinitionFileName> <unicode> <execute>
```

where:

Parameter	Description
dwSchemaDefinitionaFileName	Specifies the data warehouse schema definition file location and name. This file is generated by the generateDWSchemaDefinitionFile command line parameter.
unicode	Specifies whether the schema is created as unicode. Possible values are true and false.
execute	Possible values are true and false. If set to true DAC will generate and execute the SQL statements that drop the data warehouse schema. If set to false, the SQL statements will be generated but not executed.

Note: User authentication is not required for this command.

Running the DAC Server Automatically (Standalone Mode)

Follow this procedure to set up the DAC Server to be run automatically when your machine reboots.

Note: This procedure applies to the DAC Server in standalone mode only.

To set up the DAC Server to run automatically upon rebooting the machine

1. From the Start menu, select **Programs**, **Accessories**, **System Tools**, **Scheduled Tasks**.
2. Double-click **Add Scheduled Task**.
3. In the Scheduled Task Wizard, browse to **startserver.bat**, and click **Open**.

4. Select the option **When my computer starts**, and click **Next**.
5. Enter the domain user account to start the DAC Server and a password, and click **Finish**.

   The startserver task is displayed in the Scheduled Task window.
6. Right-click the task and select **Properties**.
7. In the Settings tab, deselect the **Stop the task if it runs for 72 hours** check box.

To start the DAC Server as a scheduled task

1. From the Programs menu, select **Accessories**, **System Tools**, **Scheduled Tasks**.
2. Right-click **startserver**, and then click **Run**.

To stop the DAC Server as a scheduled task

1. From the Programs menu, select **Accessories**, **System Tools**, **Scheduled Tasks**.
2. Right-click **startserver**, and then click **End Task**.

To check if the DAC Server is running

1. From the Programs menu, select **Accessories**, **System Tools**, **Scheduled Tasks**.
2. Select the **startserver** task.
3. In the Windows menu bar, select **View**, **Details**.

Running Two DAC Servers on the Same Machine

In the DAC standalone mode, you can run two DAC Servers on the same machine as long as they are listening on different ports and pointing to two different repositories.

Note: This procedure does not apply when the DAC Server is running as a service of the DACServer application on WebLogic Server.

To run two DAC Servers on the same machine

1. Copy the <Domain_Home>\dac directory to a different directory on the same machine.

   For example, you might copy the C:\<Domain_Home>\dac directory to C:\<Domain_Home>\DAC_SERVER2\dac.
2. Edit the config.bat file to set the DAC_HOME variable appropriately for each instance.

   For example if you copy the C:\<Domain_Home>\dac directory to C:\<Domain_Home>\DAC_SERVER2\dac, make sure that the C:\<Domain_Home>\DAC_SERVER2\dac config.bat file is configured correctly.
3. Launch each of the DAC Clients by navigating to the DAC directories and double-clicking the startclient.bat file.
4. For each instance, configure the DAC Server connection.
   a. Navigate to **Tools**, **DAC Server Management**, **DAC Server Setup**.

      A message dialog states this operation should be performed on the machine running the DAC Server. It asks whether you want to continue.
   b. Click **Yes**.

c. In the Repository Connection Information tab, enter the appropriate information for each instance. The Database Host should be the same for each instance, and the Database Port should be different.

d. Click **Save**.

5. For each instance, configure the DAC repository connection.

   a. Navigate to **Tools**, **DAC Server Management**, **Repository Configuration**.

   b. In the Repository Configuration dialog, enter the following information:

Field	Description
Mode	Select **Standalone**.
Standalone Mode Configuration:	If you are using standalone mode configuration, enter the following information:
Host	Name of the host machine where the DAC Server resides.
Alternative Hosts	(Optional) Names of additional machines hosting DAC Servers that can be used to connect to the DAC repository if the main DAC Server host machine fails. Separate multiple hosts with commas.
Port	DAC Server port.

6. Start each DAC Server from its directory.

Accessing Multiple DAC Servers Using One DAC Client

You can use one DAC Client to access multiple DAC Servers. The DAC Servers can reside on the same machine, or they can each reside on a different machine. Each DAC Server must listen on its own port and point to a single DAC repository.

Pointing Multiple Informatica Integration Services to a Single Informatica Repository

You can install multiple Informatica Integration Services and point them to a single Informatica repository. You need to register each instance of the Integration Service in DAC and specify a unique machine name and service name. For instructions on registering Integration Services, see "Registering Informatica Services in DAC".

Resetting the Data Warehouse

When you reset a data warehouse all tables are truncated and refresh dates are deleted. This allows you to perform a full load of data into the data warehouse.

To reset the data warehouse

1. On the **Tools** menu, select **ETL Management**, and then select **Reset Data Warehouse**.

   The Reset Data Warehouse dialog is displayed and requests you to confirm you want to reset the data warehouse.

2. Enter the confirmation text in the text box.

3. Click **Yes**.

Viewing DAC Metrics Using Fusion Middleware Control MBean Browser

Fusion Middleware Control MBean Browser is an Oracle Web application (based on JMX MBean containers) that you can use to view information about running, failed and queued execution plans and the status of the DAC Server.

Information about execution plans is available through the MBean Browser only if the DAC Server is running.

To display DAC metrics using Fusion Middleware Control MBean Browser

1. Display Fusion Middleware Control.
2. In the Navigator window, expand the WebLogic Domain folder and the bifoundation_domain node, and select the **AdminServer** node.
3. Display the WebLogic Server menu and select **System MBean Browser** from the menu to display the Fusion Middleware Control System MBean Browser.
4. Display the DACMetrics MBean by selecting the following: ApplicationDefinedMBeans, com.oracle.dac, Server:<server name>, Application: DACServer, EMIntegration, and DACMetrics MBean.

   The attributes and operations that are available to be invoked are displayed.

Monitoring the DAC Server Using WebLogic Server

The DAC Server runs as a service of the DACServer application on WebLogic Server. The WebLogic Server Administration Console enables you to monitor the status of the DAC Server.

Note: The DAC Server can also run in standalone mode. For information about the differences between the standalone and WebLogic Server modes, see "About DAC Authentication Modes".

To monitor the DAC Server using the WebLogic Server Administration Console

1. Display the WebLogic Server Administration Console in one of two ways:
   - On the Overview page in Fusion Middleware Control, expand the WebLogic Domain node and select the bifoundation_domain. Locate the WebLogic Server Administration Console link in the Summary region.
   - Enter the following URL into a Web browser:

     ```
     http://<host>:<port>/console
     ```

     where host is the DNS name or IP address of the DAC Server and port is the listen port on which the WebLogic Server is listening for requests. The default port is 7001.
2. Enter the system administrator user name and password and click **Login**.

   This system wide administration user name and password was specified during the installation process, and you can use it to log in to WebLogic Server Administration Console and Fusion Middleware Control. Alternatively, enter a user name that belongs to one of the following security groups:
   - Administrators
   - Operators
   - Deployers
   - Monitors

3. Click on the Deployments link in the navigation tree on the left to display the list of deployed applications.
4. Select the DACServer application.

   The DAC Server status is displayed.

13

Integrating DAC With Other ETL Tools

This chapter provides information about how to integrate DAC with ETL tools other than Informatica.

The external executor framework enables you to integrate DAC with ETL engines other than Informatica. The general process for integrating DAC and an ETL engine is as follows:

- **Step 1:** Create a Java library for the required interfaces and classes.
- **Step 2:** Add the library to the Oracle_Home\dac\lib directory.
- **Step 3:** Register the ETL engine in DAC.

The required interfaces are defined in dac-external-executors.jar, which is stored in the Oracle_Home\dac\lib directory. Javadocs provide information about the interfaces and methods, and are located in the Oracle_Home\dac\documentation\External_Executors\Javadocs directory.

In addition, a code sample is provided that demonstrates how to integrate DAC with an ETL engine. The sample code files are stored in Oracle_Home\dac\documentation\External_Executors\SampleSQLExecutor.

This chapter contains the following topics:

Interfaces That Need to Be Implemented for the External Executor Framework

To integrate DAC with an ETL engine, you need to create classes to implement the following interfaces. Information about the methods used in these interfaces is included in the Javadocs, which are located in Oracle_Home\dac\documentation\External_Executors\Javadocs directory.

- DACExecutorDescriptor
- DACExecutor
- DACExecutorJob

In addition, the interface DACExecutorProperty is provided, which defines the properties of the DACExecutor interface. You can use the default class for this interface or create your own. The default package for DACExecutorProperty is `com.oracle.dac.thirdparty.executor.DACExecutorProperty`.

DACExecutorDescriptor

The DACExecutorDescriptor interface is a starting point for integrating an ETL engine with DAC. It describes what the ETL engine does and the properties it requires. An instance of DACExecutorDescriptor is used by the DAC code to generate the required executors during an ETL process.

The properties for this interface need to implement `com.oracle.dac.thirdparty.executor.DACExecutorProperty`. You can customize the implementation by using `com.oracle.dac.thirdparty.executor.DACExecutorPropertyImpl`.

In the sample SQL Executor code (located in Oracle_Home\dac\documentation\External_Executors\SampleSQLExecutor), DACExecutorDescriptor is implemented by `com.oracle.dac.thirdparty.executor.samplesqlexecutor.SampleSQLExecutorDescriptor`. This interface produces the corresponding executor DACExecutor and requires the following properties:

- JDBC driver name
- JDBC URL
- Database user
- Database user password
- Number of connections to be created.

  Note: This property also controls the number of tasks that can run in parallel.

DACExecutor

The DACExecutor interface is an instance of an external executor. The DACExecutor object is instantiated during the ETL process and is used to communicate with the ETL engine. Multiple instances of DACExecutor can be created during an ETL process to communicate with multiple ETL engines of a specific type. DACExecutor is initialized with a set of properties (as defined in the DAC metadata) and is used to create DACExecutorJobs, one for each DAC task. A map of DAC parameters relevant for the task and DAC task definitions is provided. Task definition information includes the following:

- Task name
- Command name (For Informatica, this is the workflow name.)
- Source table names
- Target table names

For more information, see `com.oracle.dac.thirdparty.executor.arguments.DACTaskDefinition` in the Javadocs (located in Oracle_Home\dac\documentation\External_Executors\Javadocs).

In the sample SQL Executor code, DACExecutor is implemented by `com.oracle.dac.thirdparty.executor.samplesqlexecutor.SampleSQLExecutor`. At the start of an ETL process, each instance of SampleSQLExecutor used in the ETL will create a connection pool to the specified database and will create the table SAMPLE_SQL_TBL if it has not already been created. During the ETL process, SampleSQLExecutor instances create the corresponding jobs (see "DACExecutorJob").

DACExecutorJob

DACExecutorJob runs actual tasks on the external ETL engine. Jobs are created by executors during the ETL process. One job is responsible to run one task. Jobs are also initialized in the executor using task parameters. DACExecutorJob should implement a synchronous call to the ETL engine and report back the execution status.

In the sample SQL Executor code, DACExecutorJob is implemented by com.oracle.dac.thirdparty.executor.samplesqlexecutor.SampleSQLExecutorJob. For each task of the execution type that corresponds to SampleSQLExecutorDescriptor, one job is run and makes an entry with the task command (for full or incremental mode, depending what load type is appropriate) and the current timestamp in the table SAMPLE_SQL_TBL.

External Executor Utilities

You can use the external executor utility classes to simplify database operations and logging. The utility classes are included in the package com.oracle.dac.thirdparty.executor.utils. This package is not included in the dac-external-executors.jar library. It is part of DAWSystem.jar. If you want to use any class from this package, you must add DAWSystem.jar to your build path.

The following classes are provided to assist with implementing an external executor:

- DACExecutorConnectionHelper
- DACExecutorLoggingUtils

DACExecutorConnectionHelper

You can use the DACExecutorConnectionHelper utility to create and manage database connection pools and to execute SQL statements with or without bind parameters.

When using this utility, make sure to call the cleanUp() method at the end of the ETL process (when the method is called on the executor that owns an instance of DACExcecutorConnectionHelper).

DACExecutorLoggingUtils

You can use the DACExecutorLoggingUtils class to have messages and exceptions added to DAC Server and ETL log files. The ETL logs are saved in the <Domain_Home>\dac\log\<subdirectory specific to ETL run> directory.

Registering an External Executor in DAC

Follow this procedure to register an external executor in DAC.

1. Create the classes to implement the required interfaces.

   For information about the interfaces that need to be implemented, see "Interfaces That Need to Be Implemented for the External Executor Framework".

2. Create a library with the classes you created.
3. On the DAC client machine, do the following:
   a. Add the library to the Oracle_Home\dac\lib directory.
   b. Append the library file name and directory path to the end of the DACLIB entry in the config.bat or config.sh file:

For example, in the config.bat file, enter `%BI_ORACLE_HOME%\dac\lib\<file name>`. The DACLIB entry should look similar to:

```
set DACLIB=%BI_ORACLE_HOME%\dac\DAWSystem.jar;%BI_ORACLE_
HOME%\biacm\applications\biacm.paramproducer.jar;%BI_ORACLE_
HOME%\dac\lib\dac-external-parameters.jar;%BI_ORACLE_HOME%\dac\lib\<file
name>;
```

In the config.sh file, the DACLIB entry should look similar to:

```
export DACLIB=${BI_ORACLE_HOME}/dac/DAWSystem.jar:${BI_ORACLE_
HOME}/biacm/applications/biacm.paramproducer.jar:${BI_ORACLE_
HOME}/dac/lib/dac-external-parameters.jar:{BI_ORACLE_HOME}/dac/lib/<fila
name>
```

4. On the DAC Server machine, add the library to the WebLogic domain libraries folder.
5. Define an execution type for the tasks that will be executed by the external executor.
   a. On the **Tools** menu, select **Seed Data**, and then select **Execution Types**.

      The Execution Types dialog is displayed.
   b. Click **New** in the toolbar.
   c. Enter a descriptive name for the execution type.
   d. Enter the appropriate DACExecutorDescriptor implementation class, including the package.

      For example:
      `com.oracle.dac.thirdparty.executor.samplesqlexecutor.SampleSQLExecutorDescriptor`
   e. Click **Save**.
   f. Click **OK** to close the Execution Types dialog.
6. Declare the external executor.
   a. Go to the **External Executors** tab in the Setup view.
   b. Click **New** in the toolbar.

      A new, blank record is displayed
   c. Enter a descriptive name for the external executor.
   d. From the **Type** drop-down list, select the execution type you defined in the previous step.
   e. Click **Save**.
   f. Click **Generate**.
   g. Go to the **Properties** subtab, and click **Refresh** in the bottom pane toolbar.

      The list of properties defined in DACExecutorDescriptor is displayed in the Properties subtab.
   h. For each property, enter the appropriate value.
   i. Click **Save**.
7. Create the DAC tasks that will need to run on the external executor.
   a. Go to the **Tasks** tab in the Design view.

b. Click **New** in the toolbar.

A new, blank record is displayed.

c. Enter the appropriate task definitions. For a description of each field, see "Tasks Tab". For the Execution Type, make sure you select the execution type you defined in step 6.

d. Click **Save**.

14

Upgrading, Comparing and Merging DAC Repositories

This chapter provides instructions for using the DAC Upgrade/Merge Wizard to upgrade and merge the content of DAC repositories.

This chapter contains the following topics:

Major Stages of the Upgrade/Merge Wizard

The Upgrade/Merge Wizard enables you to upgrade and merge the content of DAC repositories. It contains dialogs that guide you through the stages of an upgrade or merge, as shown in Figure 14–1.

Note: Figure 14–1 does not apply to the Simplified Refresh From Base upgrade option. For more information about this option, see "About the Simplified Refresh From Base Option".

Figure 14–1 Major Stages of the Upgrade/Merge Wizard

The Upgrade/Merge Wizard enables you to suspend an upgrade or merge process before it is complete and return to it at a later time. You can also repeat a stage that you just completed or reset the process to start from the beginning.

Table 14–1 summarizes the possible actions you can take at each stage of the upgrade or merge process.

Table 14–1 Possible Upgrade and Merge Actions

Stage	Possible Actions
After you have completed Stage 1: Import repository data	You can perform one of the following actions: ■ Repeat Stage 1: Import repository data ■ Perform Stage 2: Create Difference Report ■ Reset the process
After you have completed Stage 2: Create Difference Report	You can perform one of the following actions: ■ Repeat Stage 2: Create Difference Report ■ Perform Stage 3: View Difference Report ■ Reset the process
After you have completed Stage 3: View Difference Report	You can perform one of the following actions: ■ Repeat Stage 3: View Difference Report ■ Perform Stage 4: Merge repositories ■ Reset the process

Resetting the Upgrade or Merge Process

The Upgrade/Merge Wizard enables you to reset an upgrade or merge process at any stage you are in until the final stage in which you merge the repositories. For a description of the stages of the Upgrade/Merge Wizard, see "Major Stages of the Upgrade/Merge Wizard".

When you reset an upgrade or merge process, the Upgrade/Merge Wizard reverts all the actions you have already taken in the upgrade or merge process.

To reset an upgrade or merge process

1. Open the Upgrade/Merge Wizard by selecting from the DAC toolbar Tools, then DAC Repository Management, and then Upgrade/Merge Wizard.

   The Upgrade/Merge Wizard dialog displays the type of upgrade or merge process you previously started.

2. From the Perform drop-down list, select Reset and click OK.
3. In the Reset Upgrade Process dialog, re-type the text in the text box to confirm you want to proceed, and click Yes.

   A message dialog tells you the reset process was successful.

4. Click OK.

Overview of Upgrade and Merge Options

The Upgrade/Merge Wizard includes the following options for upgrading and merging DAC repositories:

- **Repository Upgrade (DAC 784)**

Use this option to upgrade a DAC repository in the release 7.8.4 format, which has a *non-partitioned* DAC repository structure, to the new release, which has a *partitioned* DAC repository structure. For more information, see "About the Repository Upgrade (DAC 784) Option".

- **Refresh Base**

  Use this option to upgrade the DAC repository when you are upgrading from an older release of Oracle BI Applications to a new release. For more information, see "About the Refresh Base Option".

- **Simplified Refresh From Base**

  This option is similar to the Refresh Base option. It allows you to upgrade the DAC repository from an older release of Oracle BI Applications to a new release without comparing repositories and creating a Difference Report. For more information, see "About the Simplified Refresh From Base Option".

- **Replace Base**

  Use this option to upgrade the DAC repository when you are phasing out an older release of a transactional application and moving to a newer release, for example, phasing out Siebel 7.5.3 and moving to Siebel 7.8. For more information, see "About the Replace Base Option".

- **Peer to Peer Merge**

  Use this option to merge DAC repositories of different instances of the same release. For example, in a development environment you may have two instances of a DAC repository used with Oracle BI Applications release 7.9.5 that you want to merge. For more information see, "About the Peer to Peer Merge Option".

About the Repository Upgrade (DAC 784) Option

This section includes the following topics:

DAC versions before Oracle BI Applications 7.9 had a non-partitioned DAC repository structure and held metadata for a single application. DAC versions released with Oracle BI Applications 7.9 and higher and with the DAC 10.1.3.4 release and higher have partitioned DAC repository structures. The partitioned structure, also known as a container, can hold metadata for multiple applications.

The Repository Upgrade (DAC 784) option enables you to upgrade from the DAC repository release 7.8.4 (non-partitioned) to the new partitioned structure.

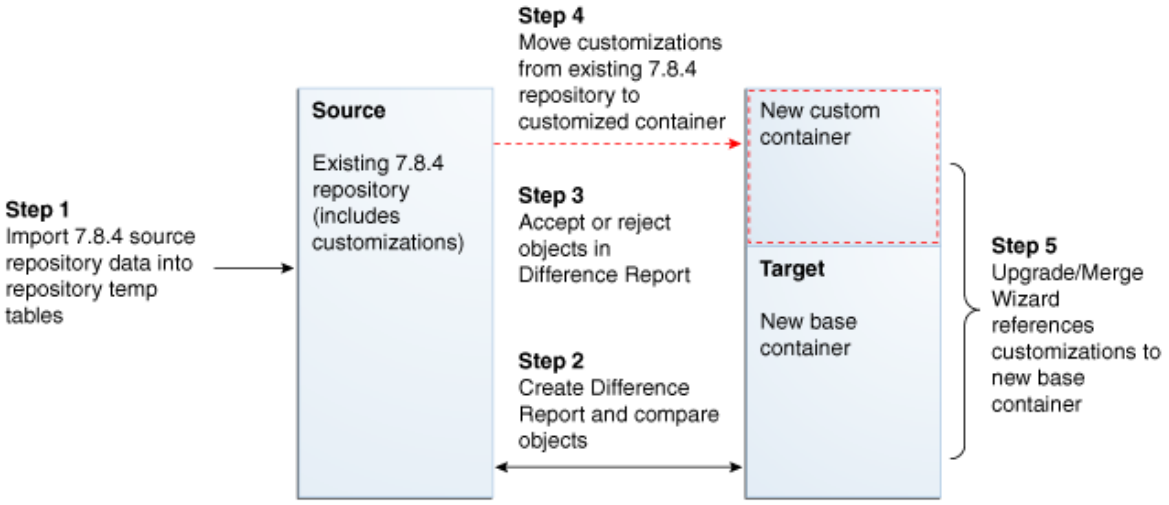
Note: If you want to upgrade a pre-7.8.4 release of a DAC repository, you must first upgrade the repository to the 7.8.4 release before you can use the Repository Upgrade (DAC 784) option to upgrade to the new release.

Repository Upgrade (784): High-Level Process Flow

Figure 14–2 shows a high-level process flow for using the Repository Upgrade (DAC 784) option to upgrade an existing release 7.8.4 DAC repository to the new release.

The term *base* is used to refer to a source or target container that you have not changed or customized in any way.

Figure 14–2 Upgrade Process for Repository Upgrade (DAC 784) Option



In Step 1 of the high-level process flow, you import the existing 7.8.4 DAC repository into the repository temporary tables. The existing 7.8.4 repository is referred to as the *source* in the Upgrade/Merge Wizard.

In Step 2, you create a Difference Report that compares the existing repository with the new base container, which is referred to as the *target*. (The new base container is the version to which you are upgrading and is the version you imported from the file system.)

In Step 3, you accept or reject the objects that the Difference Report shows as being either present or changed in the source but not the target. See Section , "Resolving Object Differences in the View Difference Report" for a description of how objects are merged based on the action you take.

In Step 4, after you have resolved the differences, you then execute the merge. In Step 5, the Upgrade/Merge Wizard references the customizations in the newly merged custom container with the new base container.

Repository Upgrade (784): Procedure for Upgrading

Follow this procedure to upgrade a DAC repository in the release 7.8.4 format to the new release.

> **Note:** You cannot use the Repository Upgrade (784) option with pre-7.8.4 releases of the DAC. To confirm the existing repository is in the release 7.8.4 format, open the 7.8.4 DAC Client and in the toolbar, select Help, and then select About DAC. The About DAC dialog displays the version of DAC you are running.

Before you begin this procedure, you need to have already installed the new release of Oracle BI Applications and imported the metadata that is the same version as the metadata in the existing 7.8.4 DAC repository. For example, if the existing 7.8.4 repository contains Siebel 6.3 metadata, you need to have imported the Siebel 6.3 source system container metadata from the new Oracle BI Applications release. If you

are using Oracle BI Applications 7.9.x with the DAC 7.9.x, then you should have the correct Siebel source system container in the 7.9.x DAC repository.

You should also review the section "Resolving Object Differences in the View Difference Report" to gain an understanding of the options for resolving object differences.

To upgrade a DAC repository in the release 7.8.4 format to the new release

1. Edit the datamapping.xml file to reflect the upgrade environment.
   a. Navigate to the directory <DAC_Config_Location>\conf-shared\upgrade.
   b. Open the datamapping.xml file for editing.
   c. Edit the entries for folder, database connection, and subject area to reflect the upgrade environment. See the comments in the datamapping.xml file for instructions.
2. Configure the connection between the release 7.8.4 DAC repository and the new DAC Client.
   a. In the Setup view, click Physical Data Sources.
   b. Click New to create a new record for the 7.8.4 DAC repository.
   c. Enter the appropriate information to connect to the release 7.8.4 DAC repository database. For information about the required fields, see the section "Physical Data Sources Tab".

      Note: The value for the Type field must be set to DAC repository.

   d. Click Test Connection to confirm the connection works.
   e. Click Save.
3. Navigate to the Upgrade/Merge Wizard by selecting Tools, then DAC Repository Management, and then Upgrade/Merge Wizard.
4. From the drop-down list, select Repository Upgrade (DAC 784), and then click OK.

   The Import 7.8.4 Repository dialog is displayed.
5. From the 784 Repository drop-down list, select the repository you configured in Step 1.

   Note: In order for a repository to appear in the drop-down list, it must be configured in the Physical Data Sources tab of the Setup view.

6. From the Source System Container drop-down list, select the new base container to which you are upgrading. This is the container you will compare against the existing release 7.8.4 DAC repository.

   Caution: Make sure you select the appropriate container. If you select a container different from that contained in the existing release 7.8.4 DAC repository, the Difference Report will compare dissimilar source system containers and will be inaccurate.

7. Select the categories of metadata from the existing release 7.8.4 DAC repository you want to import into the repository temporary tables for comparison with the new base container. The categories are as follows:

Categories Options	Description
Logical	Imports all information contained in the DAC Design view and the execution plan information for the DAC Execute view.
Run Time	Imports ETL Run History and the last refresh date information.
System	Imports all information contained in the DAC Setup view, except passwords for servers and database connections.

   The release 7.8.4 DAC repository tables are imported into the temporary tables.

8. Click OK in the Importing Tables dialog when the process is finished.

   The Create Difference dialog is displayed.

9. Create the Difference Report to compare the differences between the source and target DAC repositories.

   a. Enter a name for the Difference Report, or leave the default name.

   b. From the Source System Container drop-down list, select the new base container.

   c. In the New Custom Container ID/Name fields, enter an ID and a name for the custom container that will be created during the upgrade process.

      The ID and Name fields are alphanumeric. The Name field can contain spaces and must be at least five characters long. The ID field cannot contain spaces.

   d. (Optional) Enter a description for the Difference Report.

   e. Click OK.

      When the Difference Report is complete, the Creating Difference Report dialog tells you how long the process took.

   f. Click OK.

      The View Difference Report dialog displays the differences between the existing and new DAC repositories.

10. In the View Difference Report dialog, resolve the differences between the existing (source) and new (target) containers, that is, between the release 7.8.4 DAC repository and the repository that contains the new base container. To resolve the differences, you need to deselect the Accept Source check box for each object that appears in the difference report.

    For detailed information about the View Difference Report, see "Resolving Object Differences in the View Difference Report".

    a. In the navigation tree, select the repository object for which you want to view the differences between the existing and new repositories.

       If the object selected in the navigation tree is a hierarchical object, the subtabs for the child objects appear in the bottom pane of the Object Difference window.

    b. (Optional) Filter the objects that appear in the Object Difference window by selecting one of the options from the drop-down list in the toolbar.

c. For parent objects in the top pane and any child objects in the bottom pane, deselect the Accept Source check box.

   For detailed information about these options and the merge outcome, see "Possible Repository Merge Outcomes Based on Your Decisions"

   Note: If a child object has been changed but not the parent object, the parent object will still appear in the Object Difference window even though it has not been changed.

d. (Optional) After you have deselected the Accept Source check box for each repository object, select Resolved check box to indicate you have resolved the object.

e. Click Merge.

   The Merge dialog is displayed and lists the details of the merge.

11. Click Merge to begin the merge process.
12. Click OK in the Merging Repositories dialog when the merge process is complete.

About the Refresh Base Option

This section includes the following topics:

- Refresh Base: High-Level Process Flow
- Refresh Base: Procedure for Upgrading

The Refresh Base option enables you to upgrade an existing customized DAC repository. You should use this option if you are upgrading from a DAC release higher than 7.8.4. If you are upgrading a DAC repository in the release 7.8.4 format or lower-numbered releases, see "About the Repository Upgrade (DAC 784) Option".

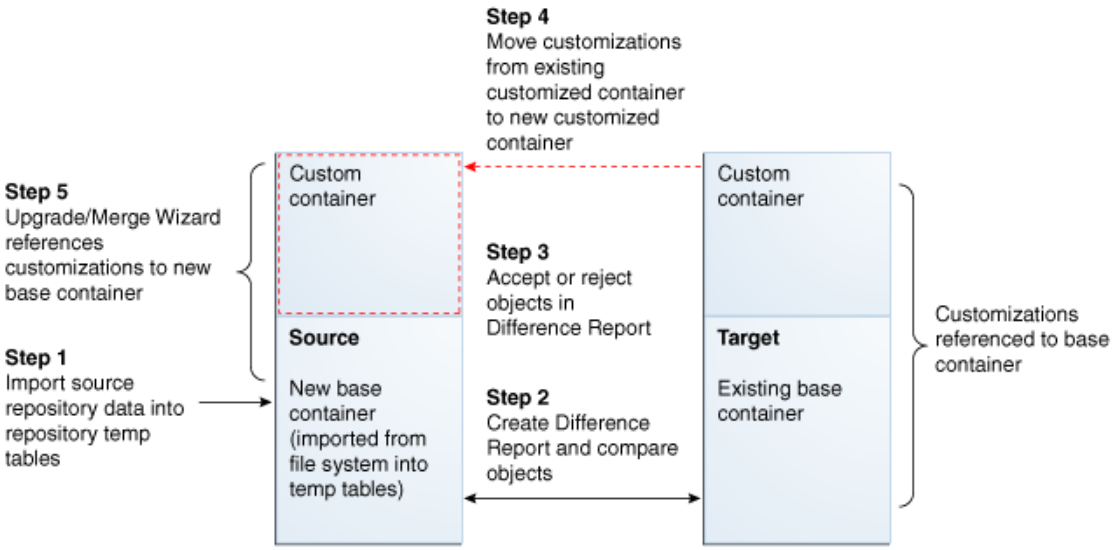
The Refresh Base option enables you to compare a new base container with the existing customized repository and to create a Difference Report. If you want to upgrade an existing customized DAC repository without comparing repositories and creating a Difference Report, you can use the Simplified Refresh From Base option. See "About the Simplified Refresh From Base Option" for more information.

Refresh Base: High-Level Process Flow

Figure 14–3 shows a high-level process flow for using the Refresh Base option to upgrade existing customized repositories from DAC releases higher than 7.8.4, such as Oracle BI Applications 7.9 and higher.

The term *base* is used to refer to a source or target container that you have not changed or customized in any way.

Figure 14–3 Upgrade Process for Refresh Base Option



In Step 1 of the high-level process flow, you import the repository data for the new base container from the file system into the repository temporary tables. This repository is referred to as the *source* in the Upgrade/Merge Wizard.

In Step 2, you create a Difference Report that compares the new base container with the existing repository (including customizations). The existing customized repository is referred to as the *target*.

In Step 3, you accept or reject the objects that the Difference Report shows as being present or changed in the source but not the target. See "Possible Repository Merge Outcomes Based on Your Decisions" for a description of how objects are merged based on the action you take.

In Step 4, after you have resolved the differences, you then execute the merge. In Step 5, the DAC references the customizations in the newly merged repository with the new base container.

Note: The repository data is imported from a file system and should be in the format of DAC 10.1.3.4 or higher. If it is in the 7.9.x format, you should do the following:

1. Restore the repository data into a database using the 7.9.x DAC (using the regular DAC import process).
2. Install DAC 10.1.3.4 or higher and configure it to the 7.9.x repository.
3. Export the relevant source system container to a file folder.

   The metadata in the folder in Step 3 above will become the source for this upgrade.

Refresh Base: Procedure for Upgrading

Follow this procedure to use the Refresh Base option to upgrade an existing customized repository from DAC releases higher than 7.8.4.

Before you begin this procedure you should review the section "Resolving Object Differences in the View Difference Report" to gain an understanding of the options for resolving object differences.

Note: If you are also upgrading your source system to a new version, after upgrading your existing repository, follow the steps in "To set up folder mappings and the physical data source connection".

To upgrade an existing repository using the Refresh Base option

1. Navigate to the Upgrade/Merge Wizard by selecting Tools, then DAC Repository Management, and then Upgrade/Merge Wizard.
2. From the drop-down list, select Refresh Base, and then click OK.

   The Import Source System Container dialog is displayed.
3. Click Change import/export folder to navigate to the directory that holds the metadata files for the new base container to which you are upgrading.
4. Select the container to which you are upgrading from the Source System Container drop-down list, and click OK.
5. In the Importing Tables dialog, re-type the text in the text box to confirm you want to proceed, and click Yes.

   When the import process is complete, the Importing Tables dialog tells you how long the process took.
6. Click OK.

   The Create Difference Report dialog is displayed.
7. Create the Difference Report to view the differences between the new and existing DAC repositories.
   a. Enter a name for the Difference Report, or leave the default name.
   b. Select the appropriate existing container.
   c. (Optional) Enter a description for the Difference Report.
   d. Click OK.

      When the Difference Report is complete, the Creating Difference Report dialog tells you how long the process took.
   e. Click OK.

      The View Difference Report dialog displays the differences between the new and existing DAC repositories.
8. In the View Difference Report dialog, resolve the differences between the new repository (source) and existing repository (target). For detailed information about the View Difference Report, see "Resolving Object Differences in the View Difference Report".

   To resolve the differences, you either accept or reject the objects that are listed as new or changed in the new repository (the version to which you are upgrading).
   a. In the navigation tree, select the repository object for which you want to view the differences between the new and existing repositories.

      If the object selected in the navigation tree is a hierarchical object, the subtabs for the child objects appear in the bottom pane of the Object Difference window.

b. (Optional) Filter the objects that appear in the top, right window by selecting one of the options from the drop-down list in the toolbar.

c. For parent objects in the top pane and any child objects in the bottom pane, accept or reject the object in the difference list by selecting or deselecting the Accept Source check box.

   For detailed information about these options and the merge outcome, see "Possible Repository Merge Outcomes Based on Your Decisions".

   Note: If a child object has been changed but not the parent object, the parent object will still appear in the Object Difference window even though it has not been changed.

d. (Optional) Once you have made a decision about whether to accept or reject the difference, select the Resolved check box to indicate you have resolved the object.

e. Repeat Steps a, b, and c, until you have resolved all object differences.

f. Click Merge.

   The Merge dialog is displayed and lists the details of the merge.

9. Click Merge to begin the merge process.

10. Click OK in the Merging Repositories dialog when the merge process is complete.

If you are also upgrading your source system to a new version, after upgrading the existing repository, follow the steps below.

To set up folder mappings and the physical data source connection

1. In the custom container, delete all of the mappings for logical folders to physical folders, which are listed in the Source System Folders tab in the DAC Design view.

2. In the custom container, reference the mappings for the logical folders to physical folders from the new base container.

   a. Click **Reference** in the upper pane toolbar.

   b. In the Reference dialog, select the new base container from the drop-down list.

   c. Select all the mapping records that appear in the list, and click **Add**.

      The Adding… dialog lists the mappings that were added to the custom container.

   d. Click **OK** to close the Add… dialog.

   e. Click **OK** to close the Reference dialog.

3. Change the name of the physical data source connection to reflect the name of the upgraded source system.

   a. Go to the **Physical Data Sources** tab in the Setup view.

   b. Locate the record for the source connection.

   c. Change the name of the source connection to reflect the name of the upgraded source system.

      For example, if you are upgrading from Oracle EBS R11 to R12, and the source connection name was Ora_R11, you would change it to Ora_R12. Do not change any other value in this record.

d. Click **Save**.

4. In Informatica Workflow Manager, open the Relational Connection Browser (in the menu bar, select **Connections**, and then select **Relational**), and edit the name of the connection to match the name you entered in step 3.
5. Rebuild all execution plans in the custom container.

About the Simplified Refresh From Base Option

The Simplified Refresh From Base option is similar to the Refresh Base option. It allows you to upgrade from a DAC repository from an older release of Oracle BI Applications to a new release, but you cannot compare repositories and create a Difference Report.

If you want to upgrade a DAC repository from an older release of Oracle BI Applications to a new release and you want to compare repositories and create a Difference Report before merging the repositories, you must use the Refresh Base option. See "About the Refresh Base Option" for more information.

Before you begin this procedure, do the following:

- Determine what customizations were made to the existing DAC repository.
- Make sure you have renamed and backed up the existing DAC repository into a different database. When you backup the DAC repository, you export the DAC metadata, in XML format (using the DAC's Export tool), into a folder other than the standard DAC export folder where backups are stored (DAC\export). For instructions on exporting DAC metadata, see "Exporting DAC Metadata".

To upgrade the DAC metadata repository

1. Upgrade the existing DAC repository tables to be compatible with the new DAC repository release.
   a. Configure the new DAC Client version to read the DAC metadata from the existing DAC repository.
   b. Log in to the DAC and select Yes if prompted to upgrade the repository tables.
2. Export the custom applications from the existing custom repository to a folder other than the standard DAC export folder (DAC\export) or the folder into which you backed up metadata in the previous format.
3. Import the new DAC metadata for the application from the standard DAC export folder (DAC\export). Select the Truncate Repository Tables check box.

   For instructions on importing DAC metadata, see "Importing DAC Metadata".
4. Import the customized DAC metadata that you exported in Step 2, and deselect the Truncate Repository Tables check box.

   This will append all the custom data to the repository, thus bringing in the customizations, which include the following:
   - All modified data.
   - All newly created custom data.
   - All deleted data.
5. Refresh the source system container to locate any missing objects in the customized application. The missing objects are any new objects that the

preconfigured applications may have that are not referenced in the custom applications.

a. Navigate to the Upgrade/Merge Wizard by selecting Tools, then DAC Repository Management, and then Upgrade/Merge Wizard.

b. From the drop-down list, select Simplified Refresh From Base, and then click OK.

c. Select the appropriate container from the Source System Container drop-down list, and click OK.

d. Confirm that you want to refresh the source system container.

6. Rebuild all the subject areas and execution plans in the customized application to include any new changes in the object dependency. For information about building subject areas and execution plans, see Chapter 6, "Customizing ETL Processes,", Chapter 8, "Designing Subject Areas," and Chapter 5, "Building and Running Execution Plans."

About the Replace Base Option

This section includes the following topics:

- Replace Base: High-Level Process Flow
- Replace Base: Procedure for Upgrading

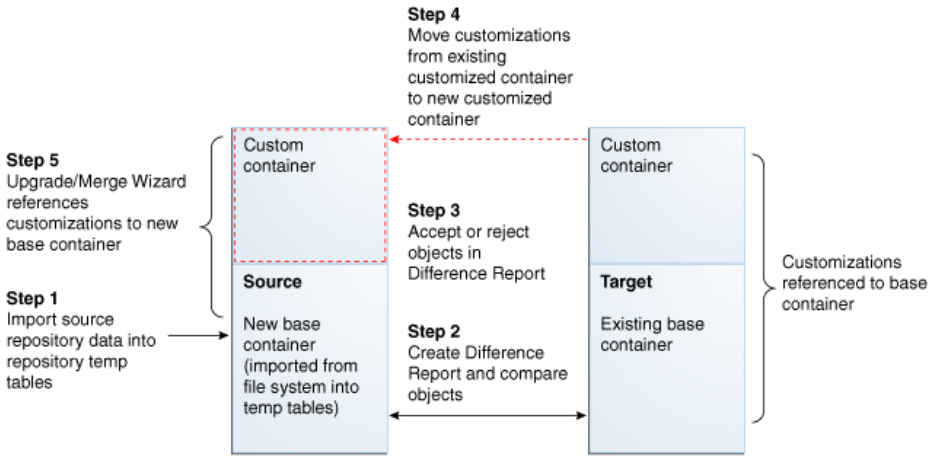
The Replace Base option enables you to upgrade the DAC repository when you are phasing out an older release of a transactional application and moving to a newer release, for example, phasing out Siebel 7.5.3 and moving to Siebel 7.8.

Replace Base: High-Level Process Flow

Figure 14–4 shows a high-level process flow for using the Replace Base option to upgrade the DAC repository when you are phasing out an older release of a transactional application and moving to a newer release.

The term *base* is used to refer to a source or target container that you have not changed or customized in any way.

Figure 14–4 Upgrade Process for Replace Base Option



In Step 1 of the high-level process flow, you import the repository data for the new base container into the repository temporary tables. This repository is referred to as the *source* in the Upgrade/Merge Wizard

In Step 2, you create a Difference Report that compares the new base container (source repository) with the existing base container (including customizations). The existing base container is referred to as the *target*

In Step 3, you accept or reject the objects that the Difference Report shows as being present in the source but not the target or changed in the source but not the target. See Section , "Possible Repository Merge Outcomes Based on Your Decisions" for a description of how objects are merged based on the action you take.

In Step 4, after you have resolved the differences, you then execute the merge. In Step 5, the DAC references the customizations in the newly merged repository with the new base container.

Note: The repository data is imported from a file system and should be in the format of DAC 10.1.3.4 or higher. If it is in the 7.9.x format, you should do the following:

1. Restore the repository data into a database using the 7.9.x DAC (using the regular DAC import process).
2. Install DAC 10.1.3.4 or higher and configure it to the 7.9.x repository.
3. Export the relevant source system container to a file folder.

   The metadata in the folder in Step 3 above will become the source for this upgrade.

Replace Base: Procedure for Upgrading

Follow this procedure to use the Replace Base option to upgrade the DAC repository when you are phasing out an older release of a transactional application and moving to a newer release.

Before you begin this procedure you should review the section "Resolving Object Differences in the View Difference Report" to gain an understanding of the options for resolving object differences.

To upgrade an existing DAC repository

1. Navigate to the Upgrade/Merge Wizard by selecting Tools, then DAC Repository Management, and then Upgrade/Merge Wizard.
2. From the drop-down list, select Replace Base, and then click OK.

   The Import Source System Container dialog is displayed.
3. Click Change import/export folder to navigate to the directory that holds the metadata files for the new base container to which you are upgrading.
4. Select the appropriate container from the Source System Container drop-down list, and click OK.
5. In the Importing Tables dialog, re-type the text in the text box to confirm you want to proceed, and click Yes.

   When the import process is complete, the Importing Tables dialog tells you how long the process took.
6. Click OK.

   The Create Difference Report dialog is displayed.
7. Create the Difference Report to view the differences between the new and existing DAC repositories.
   a. Enter a name for the Difference Report, or leave the default name.
   b. Select the appropriate existing container.
   c. (Optional) Enter a description for the Difference Report.
   d. Click OK.

      When the Difference Report is complete, the Creating Difference Report dialog tells you how long the process took.
   e. Click OK.

      The View Difference Report dialog displays the differences between the new and existing DAC repositories.
8. In the View Difference Report dialog, resolve the differences between the new repository (source) and existing repository (target). For detailed information about the View Difference Report, see "Resolving Object Differences in the View Difference Report".

   To resolve the differences, you either accept or reject the objects that are listed as new or changed in the new repository (the version to which you are upgrading).
   a. In the navigation tree, select the repository object for which you want to view the differences between the new and existing repositories.

      If the object selected in the navigation tree is a hierarchical object, the subtabs for the child objects appear in the bottom pane of the Object Difference window.
   b. (Optional) Filter the objects that appear in the top, right window by selecting one of the options from the drop-down list in the toolbar.

c. For parent objects in the top pane and any child objects in the bottom pane, accept or reject the object in the difference list by selecting or deselecting the Accept Source check box.

   For detailed information about these options and the merge outcome, see "Possible Repository Merge Outcomes Based on Your Decisions".

   Note: If a child object has been changed but not the parent object, the parent object will still appear in the Object Difference window even though it has not been changed.

d. (Optional) Once you have made a decision about whether to accept or reject the difference, select the Resolved check box to indicate you have resolved the object.

e. Repeat Steps a, b, and c, until you have resolved all object differences.

f. Click Merge.

   The Merge dialog is displayed and lists the details of the merge.

9. Click Merge to begin the merge process.
10. Click OK in the Merging Repositories dialog when the merge process is complete.
11. Rebuild all the subject areas and execution plans in the customized application to include any new changes in the object dependency. For information about building subject areas and execution plans, see Chapter 6, "Customizing ETL Processes,", Chapter 8, "Designing Subject Areas," and Chapter 5, "Building and Running Execution Plans."

About the Peer to Peer Merge Option

This section includes the following topics:

- Peer to Peer Merge: High-Level Process Flow
- Peer to Peer Merge: Procedure for Merging

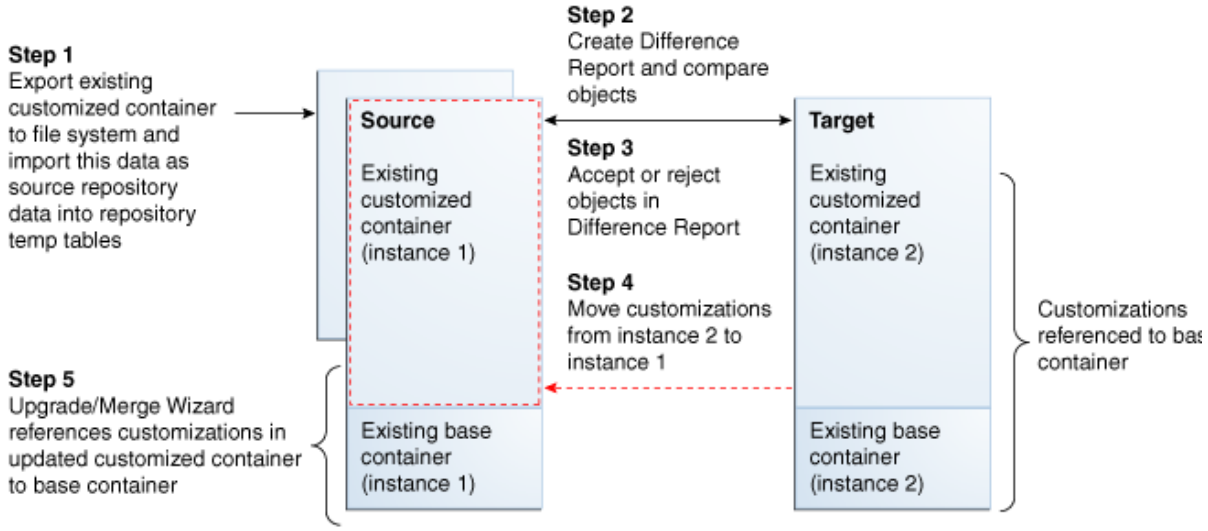
The Peer to Peer Merge option enables you to merge DAC repositories of different instances of the same release. For example, in a development environment you may have two instances of a DAC repository used with Oracle BI Applications release 7.9.5 that you want to merge.

Peer to Peer Merge: High-Level Process Flow

Figure 14–5 shows a high-level process flow for using the Peer to Peer Merge option to merge DAC repositories of different instances of the same release.

The term *base* is used to refer to a source or target container that you have not changed or customized in any way.

Figure 14–5 Merge Process for Peer to Peer Merge Option



In Step 1 of the high-level process flow, you export one instance of the existing customized source system container to the file system and then import this container into the repository temporary tables. This repository is referred to as the *source* in the Upgrade/Merge Wizard.

In Step 2, you create a Difference Report that compares the instance 1 container (including customizations) with the instance 2 container (including customizations). The instance 2 container is referred to as the *target*.

In Step 3, you accept or reject the objects that the Difference Report shows as being present or changed in the source but not the target. See Section , "Possible Repository Merge Outcomes Based on Your Decisions" for a description of how objects are merged based on the action you take.

In Step 4, after you have resolved the differences, you then execute the merge. In Step 5, the Upgrade/Merge Wizard references the customizations in the newly merged container with the instance 1 base container.

Peer to Peer Merge: Procedure for Merging

Follow this procedure to use the Peer to Peer Merge option to merge DAC repositories of different instances of the same release.

Before you begin this procedure you should review the section "Resolving Object Differences in the View Difference Report" to gain an understanding of the options for resolving object differences.

To merge two DAC repositories of different instances of the same release

1. Navigate to the Upgrade/Merge Wizard by selecting Tools, then DAC Repository Management, and then Upgrade/Merge Wizard.
2. From the drop-down list, select Replace Base, and then click OK.

   The Import Source System Container dialog is displayed.

3. Click Change import/export folder to navigate to the directory that holds the metadata files for instance 1 of the source system container you want to merge.
4. Select the appropriate container from the Source System Container drop-down list, and click OK.
5. In the Importing Tables dialog, re-type the text in the text box to confirm you want to proceed, and click Yes.

   When the import process is complete, the Importing Tables dialog tells you how long the process took.
6. Click OK.

   The Create Difference Report dialog is displayed.
7. Create the Difference Report to view the differences between the instance 1 container and the instance 2 container.
   a. Enter a name for the Difference Report, or leave the default name.
   b. In the Existing Container drop-down list, select the instance 2 container.
   c. (Optional) Enter a description for the Difference Report.
   d. Click OK.

      When the Difference Report is complete, the Creating Difference Report dialog tells you how long the process took.
   e. Click OK.

      The View Difference Report dialog displays the differences between the instance 1 and instance 2 containers.
8. In the View Difference Report dialog, resolve the differences between the instance 1 and instance 2 DAC repositories. The instance 1 repository is referred to as the source or existing container, and instance 2 as the target or new container. For detailed information about the View Difference Report, see "Resolving Object Differences in the View Difference Report".

   To resolve the differences, you either accept or reject the objects that appear as new or changed in the instance 1 container but do not appear in the instance 2 container.
   a. In the navigation tree, select the repository object for which you want to view the differences between the instance 1 and instance 2 containers.

      If the object selected in the navigation tree is a hierarchical object, the subtabs for the child objects appear in the bottom pane of the Object Difference window.
   b. (Optional) Filter the objects that appear in the top, right window by selecting one of the options from the drop-down list in the toolbar.
   c. For parent objects in the top pane and any child objects in the bottom pane, accept or reject the object in the difference list by selecting or deselecting the Accept Source check box.

      For detailed information about these options and the merge outcome, see "Possible Repository Merge Outcomes Based on Your Decisions".

Note: If a child object has been changed but not the parent object, the parent object will still appear in the Object Difference window even though it has not been changed.

d. (Optional) Once you have made a decision about whether to accept or reject the difference, select the Resolved check box to indicate you have resolved the object.

e. Repeat Steps a, b, and c, until you have resolved all object differences.

f. Click Merge.

The Merge dialog is displayed and lists the details of the merge.

9. Click Merge to begin the merge process.

10. Click OK in the Merging Repositories dialog when the merge process is complete.

Resolving Object Differences in the View Difference Report

This section includes the following topics:

- Overview of View Difference Report
- View Difference Report Interface
- Possible Repository Merge Outcomes Based on Your Decisions

The Upgrade/Merge Wizard generates a View Difference Report for the following upgrade and merge options:

- Repository Upgrade (DAC 784)
- Refresh Base
- Replace Base
- Peer to Peer Merge

A View Difference Report is not available if you select to upgrade using the Simplified Refresh From Base option. For more information about the upgrade and merge options, see "Overview of Upgrade and Merge Options".

Overview of View Difference Report

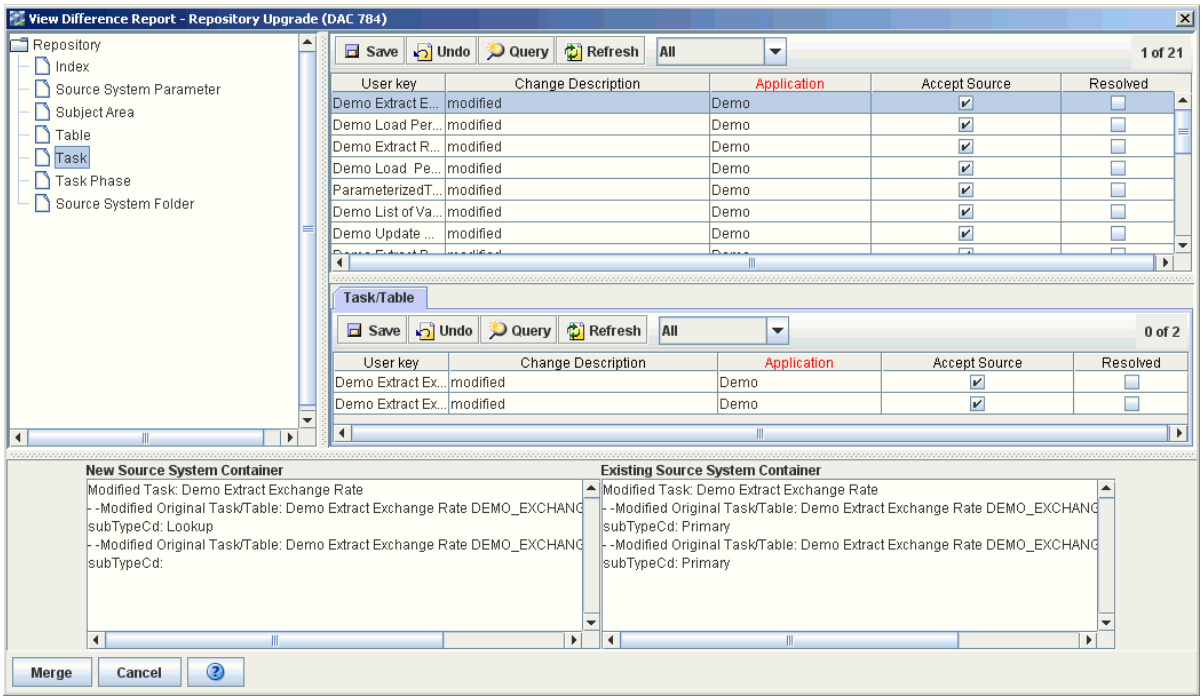
The View Difference Report dialog enables you to resolve the differences between the existing and new DAC repositories. To resolve the differences, you either accept or reject the objects that appear in the existing repository but do not appear in the new repository.

Objects in a source system container have an ownership property. The categories of ownership are original, reference, and clone. The ownership property is maintained when an object from the existing repository is merged with the new repository. For example, if an object in the existing source system container is a reference and you want to merge it into the new repository, it will be moved into the new container as a reference. For more information about object ownership properties, see "About Object Ownership in DAC".

Table 14–2 shows the merge outcomes based on object type.

View Difference Report Interface

This section describes the main features of the View Difference Report.



- **Navigation Tree**

  The navigation tree displays the repository object types that are compared in the existing and new repositories. When you select a repository object in the navigation tree, the differences are listed in the object difference window.

- **Object Difference Window**

  The Object Difference window lists the differences found between the existing and new containers for the object type selected in the navigation tree. If the object selected in the navigation tree is a hierarchical object, the subtabs for the child objects appear below the Object Difference window. The following columns appear in the Object Difference window and in the subtabs:

  – **User Key**

    Unique identifier for the object.

  – **Change Description**

    Indicates the type of difference that exists between the containers.

  – **Container**

    The new or target container.

  – **Accept Source**

    Selecting the Accept Source check box indicates you want to accept the change that appears in the source container. For detailed information about the

possible merge outcomes, see "Possible Repository Merge Outcomes Based on Your Decisions".

  - **Resolved**

    Optional field that indicates the object difference has been resolved. Note: This field is for user reference only. It does not have an affect on the merge process.

- **Subtabs for Child Objects**

  If the object selected in the navigation tree is a hierarchical object, the subtabs for the related child objects appear in the bottom pane of the Object Difference window.

- **Text Fields**

  The text fields in at the bottom of the dialog, labeled "New Source System Container "and "Existing Source System Container," display a textual and hierarchical representation of the object difference selected in the Object Difference window.

- **Query Functionality**

  You can query for objects in the Object Difference window and in the subtabs. For more information about the DAC query functionality, see "Using the DAC Query Functionality".

- **Difference Type Filter**

  The drop-down list in the Object Difference window toolbar enables you to filter the list of differences based on the Change Description. The following filter options are available:

  - **All**. Displays all changed objects.
  - **Added-Source**. Displays objects that were added in the source container.
  - **Added-Target**. Displays objects that were added in the target container.
  - **Cloned-Source**. Displays objects that were cloned in the source container.
  - **Cloned-Target**. Displays objects that were cloned in the target container.
  - **Deleted-Source**. Displays objects that were deleted from the source container.
  - **Deleted-Target**. Displays objects that were deleted from the target container.
  - **Modified**. Displays objects that were modified differently in the source and target containers.

Possible Repository Merge Outcomes Based on Your Decisions

The View Difference Report shows the changes between two containers based on object type.

For more information about object types and their ownership properties, see "About Object Ownership in DAC".

> **Tip:** Use the right-click command Record Info to view the lineage of an object.

The following tables list the merge outcomes based on the decision you make about the object differences listed in the View Difference Report.

- Table 14–2, " Merge Outcomes for Repository Upgrade (DAC 784) Option"

- Table 14–3, " Merge Outcomes for Replace Base and Refresh Base Options"
- Table 14–4, " Merge Outcomes for Peer to Peer Option"

Table 14–2 Merge Outcomes for Repository Upgrade (DAC 784) Option

Change	Object Ownership Type	Description	Merge Action with User Choice: Accept Source	Merge Action with User Choice: Reject Source
Added-Source	Original	An original object has been added in the source container.	Object will be merged into the custom container as an original object.	Object will be deleted from the custom container.
Added-Source	Clone	A cloned object has been added in the source container.	Not applicable	Not applicable
Added-Source	Reference	A referenced object has been added in the source container.	Not applicable	Not applicable
Added-Target	Original	An original object has been added in the target container.	Object will be deleted from the custom container.	Object will be merged into the custom container as a referenced object to the base container.
Added-Target	Clone	A cloned object has been added in the target container.	Object will be deleted from the custom container.	Object will be merged into the custom container as a referenced object to the base container.
Added-Target	Reference	A referenced object has been added in the target container.	Object will be deleted from the custom container.	Object will be merged into the custom container as a referenced object to the base container.
Deleted-Source	Original	An original object has been deleted from the source container.	Object will be deleted from the custom container.	Object will be retained in the custom container as a reference to the base container.
Deleted-Source	Clone	A cloned object has been deleted from the source container.	Object will be deleted from the custom container.	Object will be retained in the custom container as a reference to the base container.
Deleted-Source	Reference	A referenced object has been deleted from the source container.	Object will be deleted from the custom container.	Object will be retained in the custom container as a reference to the base container.
Deleted-Target	Original	An original object has been deleted from the target container.	Object will be merged into the custom container as an original object.	Object will be deleted from the custom container.
Deleted-Target	Clone	A cloned object has been deleted from the target container.	Object will be merged into the custom container as an original object.	Object will be deleted from the custom container.
Deleted-Target	Reference	A referenced object has been deleted from the target container.	Object will be merged into the custom container as an original object.	Object will be deleted from the custom container.

Table 14–2 (Cont.) Merge Outcomes for Repository Upgrade (DAC 784) Option

Change	Object Ownership Type	Description	Merge Action with User Choice: Accept Source	Merge Action with User Choice: Reject Source
Modified	Not applicable	An object has been modified differently in the source and target containers.	Object will be merged into the custom container as a cloned object.	Object will be merged into the custom container as a referenced object to the base container.
Modified-Cloned	Clone	The parent of a cloned object has been modified in the source container.	Not applicable	Not applicable
Modified-Referenced	Reference	The parent of a referenced object has been modified in the source container.	Not applicable	Not applicable
Cloned-Source	Not applicable	A referenced object has been cloned in the source container.	Not applicable	Not applicable
Cloned-Target	Not applicable	A referenced object has been cloned in the target container.	Not applicable	Not applicable

Table 14–3 Merge Outcomes for Replace Base and Refresh Base Options

Change	Object Ownership Type	Description	Merge Action with User Choice: Accept Source	Merge Action with User Choice: Reject Source
Added-Source	Original	An original object has been added in the source container.	Object will be added to the target custom container as an original object.	Not applicable
Added-Source	Clone	A cloned object has been added in the source container.	Object will be added to the target custom container as a cloned object. Automatically adjusts the lineage for objects in child containers if the object was referenced in them.	Not applicable
Added-Source	Reference	A referenced object has been added in the source container.	Object will be added to the target container as a referenced object.	Not applicable
Added-Target	Original	An original object has been added in the target container.	Object will be deleted from the target custom container. Automatically adjusts the lineage for objects in child containers.	Object will be retained in the target custom container.
Added-Target	Clone	A cloned object has been added in the target container.	Object will be deleted from the target custom container. Automatically adjusts the lineage for objects in child containers.	Object will be retained in the target custom container.

Table 14–3 (Cont.) Merge Outcomes for Replace Base and Refresh Base Options

Change	Object Ownership Type	Description	Merge Action with User Choice: Accept Source	Merge Action with User Choice: Reject Source
Added-Target	Reference	A referenced object has been added in the target container.	Object will be deleted from the target custom container. Automatically adjusts the lineage for objects in child containers.	Object will be retained in the target custom container.
Deleted-Source	Original	An original object has been deleted from the source container.	Object will be deleted from the target custom container. Automatically adjusts the lineage for objects in child containers if the object was referenced in them.	Object will be retained in the target custom container.
Deleted-Source	Clone	A cloned object has been deleted from the source container.	Object will be deleted from the target custom container. Automatically adjusts the lineage for objects in child containers if the object was referenced in them.	Object will be retained in the target custom container.
Deleted-Source	Reference	A referenced object has been deleted from the source container.	Object will be deleted from the target custom container. Automatically adjusts the lineage for objects in child containers if the object was referenced in them.	Object will be retained in the target custom container.
Deleted-Target	Original	An original object has been deleted from the target container.	Object will be added as an original object to the target custom container.	Not applicable
Deleted-Target	Clone	A cloned object has been deleted from the target container.	Object will be added as a cloned object to the target custom container.	Not applicable
Deleted-Target	Reference	A referenced object has been deleted from the target container.	Object will be added as a referenced object to the target custom container.	Not applicable
Modified	Not applicable	An object has been modified differently in the source and target containers. The right-click command Record Info shows the object's lineage.	Not applicable	Not applicable

Table 14–3 (Cont.) Merge Outcomes for Replace Base and Refresh Base Options

Change	Object Ownership Type	Description	Merge Action with User Choice: Accept Source	Merge Action with User Choice: Reject Source
Modified-Cloned	Clone	The parent of a cloned object has been modified in the source container.	De-clone the object and re-reference it. Automatically adjusts the lineage for objects in child containers if the object was referenced in them.	Not applicable
Modified-Referenced	Reference	The parent of a referenced object has been modified in the source container.	Re-reference the object. Automatically adjusts the lineage for objects in child containers if the object was referenced in them.	Not applicable
Cloned-Source	Not applicable	A referenced object has been cloned in the source container.	Not applicable	Not applicable
Cloned-Target	Not applicable	A referenced object has been cloned in the target container.	Not applicable	Not applicable

Table 14–4 Merge Outcomes for Peer to Peer Option

Change	Object Ownership Type	Description	Merge Action with User Choice: Accept Source	Merge Action with User Choice: Reject Source
Added-Source	Original	An original object has been added in the source container.	Object will be added to the target custom container as an original object.	Not applicable
Added-Source	Clone	A cloned object has been added in the source container.	Object will be added to the target custom container as a cloned object. Automatically adjusts the lineage for objects in child containers.	Not applicable
Added-Source	Reference	A referenced object has been added in the source container.	Object will be added to the target custom container as a referenced object.	Not applicable
Added-Target	Original	An original object has been added in the target container.	Object will be deleted from the target custom container. Automatically adjusts the lineage for objects in child containers.	Object will be retained in the target custom container.

Table 14–4 (Cont.) Merge Outcomes for Peer to Peer Option

Change	Object Ownership Type	Description	Merge Action with User Choice: Accept Source	Merge Action with User Choice: Reject Source
Added-Target	Clone	A cloned object has been added in the target container.	Object will be deleted from the target custom container. Automatically adjusts the lineage for objects in child containers.	Object will be retained in the target custom container.
Added-Target	Reference	A referenced object has been added in the target container.	Object will be deleted from the target custom container. Automatically adjusts the lineage for objects in child containers.	Object will be retained in the target custom container.
Deleted-Source	Original	An original object has been deleted from the source container.	Object will be deleted from the target custom container. Automatically adjusts the lineage for objects in child containers if the objects were referenced in them.	Object will be retained in the target custom container.
Deleted-Source	Clone	A cloned object has been deleted from the source container.	Object will be deleted from the target custom container. Automatically adjusts the lineage for objects in child containers if the objects were referenced in them.	Object will be retained in the target custom container.
Deleted-Source	Reference	A referenced object has been deleted from the source container.	Object will be deleted from the target custom container. Automatically adjusts the lineage for objects in child containers if the objects were referenced in them.	Object will be retained in the target custom container.
Deleted-Target	Original	An original object has been deleted from the target container.	Object will be added to the target custom container as an original object.	Not applicable
Deleted-Target	Clone	A cloned object has been deleted from the target container.	Object will be added to the target custom container as a cloned object.	Not applicable
Deleted-Target	Reference	A referenced object has been deleted from the target container.	Object will be added to the target custom container as a referenced object.	Not applicable

Table 14–4 (Cont.) Merge Outcomes for Peer to Peer Option

Change	Object Ownership Type	Description	Merge Action with User Choice: Accept Source	Merge Action with User Choice: Reject Source
Modified	Not applicable	An object has been modified differently in the source and target containers. The right-click command Record Info shows the object's lineage.	Not applicable	Not applicable
Modified-Cloned	Clone	The parent of a cloned object has been modified in the source container.	De-clone the object and re-reference it. Automatically adjusts the lineage for objects in child containers if the objects were referenced in them.	Not applicable
Modified-Referenced	Reference	The parent of a referenced object has been modified in the source container.	Re-reference the object. Automatically adjusts the lineage for objects in child containers if the objects were referenced in them.	Not applicable
Cloned-Source	Not applicable	A referenced object has been cloned in the source container.	Clone object in target custom container. Automatically adjusts the lineage for objects in child containers.	Not applicable
Cloned-Target	Not applicable	A referenced object has been cloned in the target container.	De-clone object in target custom container. Automatically adjusts the lineage for objects in child containers.	Not applicable

15

DAC Functional Reference

This chapter describes the functionality available in the DAC tabs, menu commands, and right-click commands.

This chapter contains the following main topics:

Menu Bar Commands

This section provides a description of the DAC menu bar commands. It includes the following topics:

File Menu Commands

Table 15–1 provides a description of the File menu commands.

Table 15–1 File Menu Commands

Command	Description
DAC User Management	Enables users with the Administrator role to create, delete, and inactivate user accounts. For more information, see "Managing DAC User Accounts" and "Creating, Deleting, Inactivating User Accounts".
Change DAC User Password	Enables users who are currently logged into DAC to change their password.
New Source System Container	Enables you to create a new, empty source system container or to make a copy of an existing container. **Note:** You cannot make any changes to a predefined source system container. You must make a copy of an existing container in order to make changes to it. For more information about managing source system containers, see "About Source System Containers". For instructions on creating or copying a container, see "Creating or Copying a Source System Container".
Rename Source System Container	Enables you to rename a source system container.
Delete Source System Container	Enables you to delete an existing source system container.
Close	Closes the DAC Client.

Views Menu Commands

Table 15–2 provides a description of the Views menu commands.

Table 15–2 Views Menu Commands

Command	Description
Design	The Design view provides access to functionality related to creating and managing subject areas. For more information, see "Design View Tabs". When the Design view is active, the Source System Container drop-down list appears to the right of the View buttons. It enables you to select the source system container that holds the metadata corresponding to a source system.
Setup	The Setup View provides access to functionality related to setting up DAC system properties, Informatica services, database connections, email notification, external executors, and patches. For more information, see "Setup View Tabs".
Execute	The Execute view provides access to functionality related to setting up, running, monitoring, and scheduling execution plans. For more information, see "Execute View Tabs".

Tools Menu Commands

This section provides a description of the Tools menu commands. It includes the following topics:

DAC Repository Management Menu Commands

This section provides a description of the DAC Repository Management menu commands on the Tools menu.

Export

Enables you to export the DAC metadata, in XML format, based on the source system container, in order to back up the metadata or to reproduce the environment elsewhere. In the Export dialog, you can specify a directory in which to store the XML file or accept the default directory, which is dac\export.

In the Export dialog, you can select the following category options:

- **Logical.** Exports all information contained in the Design view and metadata defined in the Seed Data menu.
- **Run Time.** Exports information about ETL runs and schedules (contained in the Execute view).
- **System.** Exports all information contained in the Setup view, except passwords for servers and database connections.
- **Update existing records.** (Applicable to DAC standalone authentication only) Exports the users, roles, and passwords.

Note: You can move small sets of data, without exporting the entire DAC repository, by using the DAC patching functionality. For more information, see Chapter 11, "Working With DAC Metadata Patches."

Import

Enables you to import the DAC metadata for the source system containers you specify. In the Import dialog, you can specify the following:

- **Import/Export folder.** A directory from which to import the data. The default directory is DAC\export.
- **Truncate repository tables.** Indicates whether you want to truncate the repository tables. If you select this option, the existing metadata is overwritten.
- **Enable bulk mode.** Indicates whether bulk mode is enabled. In bulk mode the imported metadata is inserted into the repository as an array insert. Using this option improves performance. If you do not select this option, DAC upserts records in the repository tables one row at a time.
- **Update existing records.** (Applicable to DAC standalone authentication only) Imports the users, roles, and passwords.

In the Import dialog, you can select the following category options:

- **Logical.** Exports all information contained in the Design view and metadata defined in the Seed Data menu.
- **Run Time.** Exports information about ETL runs and schedules (contained in the Execute view).
- **System.** Exports all information contained in the Setup view, except passwords for servers and database connections.
- **Update existing records.** (Applicable to DAC standalone authentication only) Exports the users, roles, and passwords.

Create Repository Report

Enables you to generate a DAC repository report based on the following criteria:

- Table Row Counts
- Object References by Entity
- Ownerless Objects
- Unreferenced Objects
- Dead References

The Clean Up command removes unused referenced objects.

Upgrade/Merge Wizard

The Upgrade/Merge Wizard enables you to upgrade and merge the content of DAC repositories. For more information, see "Upgrading, Comparing and Merging DAC Repositories".

Apply Patch

Enables you to apply a DAC metadata patch to the DAC repository. For more information, see "Working With DAC Metadata Patches".

Purge Run Details

Enables you to purge completed runs from the run history. The last run cannot be purged.

In the Purging Run History... dialog, the following options are available:

- **Delete all completed runs.** Purges all completed runs except for the last run.
- **Delete completed runs before specified date.** Enables you to select a date before which all runs except the last run will be purged.
- **Keep run definition.** Purges all related information about a run but leaves the run header information.

Analyze Repository Tables

Enables you to run analyze table commands for all the DAC repository tables.

Default Index Properties

Enables you to specify which databases will be associated with newly created indexes.

Repository Audit Trail

Enables you to access the Repository Audit Trail, which stores information about actions performed on the repository, users, machine IP addresses, and repository timestamps. You can also add user-defined audit trail records.

Drop DAC Repository

Enables you to drop all the DAC repository tables. This action deletes all data in the repository.

Change Encryption Key

Enables you to change the DAC repository encryption key. See "Changing the DAC Repository Encryption Key" for instructions.

DAC Server Management Menu Commands

Table 15–3 provides a description of the DAC Server Management menu commands on the Tools menu.

Table 15–3 DAC Server Management Menu Commands

Command	Description
DAC Server Setup	Enables you to configure the DAC Server connections and server email settings. This action should be performed on the machine where the DAC Server is running.
Start DAC Server	When running the DAC Server in Web mode, allows you to start the DAC Server.
Restart DAC Server	When running the DAC Server in Web mode, allows you to restart the DAC Server.
Stop DAC Server	When running the DAC Server in Web mode, allows you to stop the DAC Server.

ETL Management Menu Commands

Table 15–4 provides a description of the ETL Management menu commands on the Tools menu.

Table 15–4 ETL Management Menu Commands

Command	Description
Configure	Opens the Data Warehouse Configuration wizard, which enables you to create, upgrade and drop data warehouse tables and to create delete triggers. See Chapter 9, "Managing Data Warehouse Schemas," for more information.
Reset Data Sources	Clears the refresh dates for all source and target tables. This action forces a full load to occur during the next ETL process.

Seed Data Menu Commands

This section provides a description of the Seed Data menu commands on the Tools menu.

Actions, Index Actions

Enables you to set up index actions in order to trigger SQL scripts to create or drop indexes. See "Using Actions to Optimize Indexes and Collect Statistics on Tables" for more information.

Actions, Table Actions

Enables you to set up table actions in order to trigger SQL scripts to analyze and truncate tables. See "Using Actions to Optimize Indexes and Collect Statistics on Tables"

Actions, Task Actions

Enables you to set up task actions in order to trigger SQL scripts to perform various actions related to task behavior. See "Using Actions to Optimize Indexes and Collect Statistics on Tables" for more information.

Global External Parameters

Enables you to define a global external parameter. See "Overview of Parameters" and "Defining a Global External Parameter" for more information.

Heuristics

Enables you to define a heuristics rule. See "Using Heuristics to Manage Tasks, Tables and Indexes" for more information.

Logical Data Sources

Enables you to add, edit, or delete logical data sources.

Task Logical Folders

Enables you to add, edit, or delete task logical folders, which are used in the task definition (in the Tasks tab) so that task definitions do not have to be cloned

Task Phases

Enables you to add, edit, or delete task phases, and to assign a priority to the task phase. A task phase is the phase of the ETL process with which a task is associated. DAC uses the task phase primarily for task dependency generation. The DAC Server uses the task phase to prioritize tasks and to generate a summary of the time taken for each of the phases.

Task Physical Folders

Enables you to add, edit, or delete task physical folders, which correspond to the actual folder in the Informatica repository. The task physical folder name is used in the Ordered Tasks subtab of the Execution Plans tab.

UI Styles Menu Commands

Table 15–5 provides a description of the UI Styles menu commands on the tools menu.

Table 15–5 UI Styles Menu Commands

Command	Description
Windows (MFS)	Changes the user interface to the Windows style.
UNIX (MOTIF)	Changes the user interface to the UNIX style.
Java (METAL)	Changes the user interface to the Java style.

UI Preferences

The UI Preferences menu command displays the UI Preferences dialog, which contains the following options:

- **Always Show Last Updated**. Displays the Last Updated column in all tabs and subtabs. This column shows the timestamp for when the repository object was last updated.
- **Always show system columns**. Displays system columns in all tabs and subtabs.

- **Start in Query Mode.** Indicates a tab or subtab will start in Query Mode. Select a tab or subtab from the navigation window, and then select the Start in Query Mode check box.
- **Accessibility Mode Enabled.** Enables accessible keyboard shortcuts. For a description of the shortcuts, see "DAC Accessibility Keyboard Options".

Help Menu Commands

Table 15–6 provides a description of the Help menu commands.

Table 15–6 Help Menu Commands

Command	Description
Login Details	Provides login and connectivity information for the current session.
System Information	Provides system information for the DAC installation.
DAC Help	Opens the DAC online help.
About DAC	Provides DAC build, schema version, and repository version information.

Top Pane Toolbar Commands

Table 15–7 describes the commands available in the top pane toolbar that are common to all views.

Table 15–7 Tools in the Top Pane Toolbar Common to All Views

Tool	Description
New	Creates a placeholder for a new record in the selected list.
Save	Saves the current record.
Undo	Undoes changes made to the current record after the last save.
Delete	Deletes the selected record. If you delete a parent record, the child records are also deleted. When you delete a column from a table, the column is not automatically deleted from the index. The DAC does not display deleted objects. You must look at the database to figure out what objects were deleted.
Query	Opens a blank query field where you can enter query criteria.
Refresh	Retrieves the data from the repository with the last used query.
Help	Opens the context-sensitive online help system.

Table 15–8 describes the commands available in the top pane toolbar when the Design view is active.

Table 15–8 Tools in the Top Pane Toolbar Specific to the Design View

Command	Description
Reference	(Design view) Opens the Reference dialog, which enables you to copy objects from one container to another. See "About Object Ownership in DAC" for information about referencing objects.
Assemble	(Design view, Subject Areas tab) Assembles a subject area, with dimension and related tables as well as tasks. See Chapter 8, "Designing Subject Areas," for more information.
Drop-down list	(Design view) Enables you to filter the source system container objects that appear in the top pane list.

Table 15–9 describes the commands available in the top pane toolbar when the Setup view is active.

Table 15–9 Tools in the Top Pane Toolbar Specific to the Setup View

Tool	Description
Generate	(External Executors tab) Generates the external executor properties required for configuration. See Chapter 13, "Integrating DAC With Other ETL Tools," for more information.
Test	(External Executors tab only) Tests the configuration of the external executor properties.

Table 15–10 describes the commands available in the top pane toolbar when the Execute view is active.

Table 15–10 Tools in the Top Pane Toolbar Specific to the Execute View

Tool	Description
Build	(Execution Plans tab) Builds the execution plan, by assembling subject areas, tasks, indexes, tags, parameters, source system folders, and phases. See "Building and Running Execution Plans" for instructions.
R**un Now**	(Execution Plans tab) Starts a new ETL process.
Stop	(Current Runs and Run History tabs) Stops an ETL in progress. All currently running tasks will complete, and queued tasks will stop. The status of the ETL changes to Stopped.
Abort	(Current Runs and Run History tabs) Causes an ETL in progress to abort. All currently running tasks will be aborted. The status of queued tasks and the ETL itself will change to Stopped.
Restart	(Current Runs and Run History tabs) Restarts the selected ETL after the ETL has failed, stopped, or been aborted.
Auto Refresh(	Current Runs tab). Enables you to turn on and off the automatic screen refresh functionality and set the refresh interval.

Right-Click Menu Commands

The commands available in the right-click menus depend on the tab that is active. For descriptions of the commands, see the following topics:

- Common Right-Click Menu Commands
- Design View Right-Click Menu Commands
- Setup View Right-Click Menu Commands
- Execute View Right-Click Menu Commands

Common Right-Click Menu Commands

Table 15–11 describes right-click menu commands common to all tabs.

Table 15–11 Common Right-Click Menu Commands

Command	Description
Copy String	Copies the contents of a cell (editable and read-only) to the clipboard.
Paste String	Pastes a string from the clipboard into a selected cell that supports a string data type.
New	Creates a placeholder for a new record in the selected list.
Copy Record	Creates a copy of the selected record, with a unique record ID. The new record is committed to the DAC repository when you click the Save button or click outside the cell. In the Design view tabs (except for the Indices tab), Copy Record copies the selected record and the record's child records. When you copy a subject area, the tables are also copied but the tasks are not copied. You need to use the Assemble command to reassemble the subject area and add tasks to it. In the Design view Indices tab and Setup and Execute views, Copy Record copies only the selected record.
Save	Saves the current record.
Undo	Undoes the changes made to the current record after the last save.
Delete	Deletes the selected record. If you delete a parent record, the child records are also deleted. When you delete a column from a table, the column is not automatically deleted from the index. You must manually delete columns from indexes that were deleted from a table or else the ETL process will fail. The DAC does not display deleted objects. You must look at the database to figure out what objects were deleted.
Query	Opens a blank query.
Refresh	Retrieves the data from the repository with the last used query.
UI Preferences	Enables you to configure the active tab to start in query mode.
Output to File	Outputs to a text file in the DAC root directory the contents of the current tab's record list.
Record Info	Displays the record's unique ID, object type, current source system, owner source system, and the timestamp for when it was last updated. It also displays the source system lineage and the source systems that reference the object.
Update Records	For some columns, enables you to update the column value for each row to a single value.

Design View Right-Click Menu Commands

This section provides descriptions of the Design view right-click menu commands.

Ownership Right-Click Commands

The Ownership right-click menu contains the following commands:

- **Reference.** Opens the Reference dialog, which enables you to reference objects from one container to another. The reference function works like a symbolic link or shortcut.
- **Re-Reference.** If an object is a referenced object, that is, a reference to an object in another container, and a change is made to the original object's child objects, use this command to import the changes to the referenced object.
- **Push to References.** If an original object is changed, you can use this command to export the changes to all referenced objects' child objects.
- **De-Clone.** When you make changes to a referenced object, the new object is called a *clone*. This command enables you to revert a cloned object back to its state as a reference.
- **Compare Clone with Original**. Shows the difference between the clone and the base object that was modified.
- **Re-Assign Record.** This command enables you to reassign an objects ownership.

For more information about the ownership of objects, see "About Object Ownership in DAC".

Add Object(s) to Patch

Enables you to add the selected object or all objects in the list to a patch. You specify the patch to which you want to add the object or objects, and then specify whether you want to add to the patch only the parent object or the parent object, child object, and extended properties. For more information about patches, see "Working With DAC Metadata Patches".

Assemble

(Subject Areas tab) Assembles a specified subject area by adding to the subject area the dimension and related tables as well as the associated tasks. For more information, see "Creating a Subject Area".

Generate Index Scripts

(Tables and Indices tabs) Generates drop and create index scripts and analyze tables scripts for all tables that participate in the ETL process. The results are stored in the log\scripts directory.

Generate DW Table Scripts for Oracle

(Tables tab and Target For Tasks (RO) subtabs in the Tables tab) Enables you to generate scripts for creating, upgrading, and dropping data warehouse tables for Oracle database types. For more information, see "Creating, Upgrading or Dropping Subsets of Tables in the Schema for Oracle Databases".

Related Tasks

(Tables tab) Shows all tasks that directly or indirectly participate in populating the selected table.

Change Capture Scripts

(Tables tab) For Siebel sources only.

- **Image and Trigger Scripts.** Generates change capture scripts for tables with defined image suffixes. The scripts may include delete triggers, create and drop statements for delete triggers, and image tables and their indexes.
- **View Scripts.** Generates change capture view scripts for full or incremental mode for tables that participate in the change capture process. This command can be used for unit testing.
- **Change Capture SQL.** Generates change capture SQL scripts for full or incremental mode for tables that participate in the change capture process. This command can be used for unit testing.

Import from Database

(Tables tab)

- **Import Database Tables.** Imports table definitions from a selected database. This action does not import columns.
- **Import Indices.** Import index definitions from a selected database for one or more tables as listed in the result of the query.
- **Import Database Columns.** Import column definitions from a selected database.

See "Adding a New Table and Columns to the Data Warehouse" for more information.

Add Actions

(Tables, Indices, and Tasks tabs) Enables you to add previously defined actions to tables, indices, or tasks. For more information about actions, see "Using Actions to Optimize Indexes and Collect Statistics on Tables".

Import Foreign Keys

(Tables tab) Enables you to import foreign key columns from the data warehouse and associate the columns with foreign key tables. You can enter the table and column information in the text box or import the information from a file. If you select the Clone Referenced Records check box, DAC will make clones of any records that are referenced objects. If you do not select Clone Referenced Records, DAC will not import foreign keys for any referenced object.

Output Task Description

(Tasks tab) Saves to an HTML file the description for a selected task or for all tasks.

Synchronize Tasks

(Tasks tab) Imports the source and target table definitions for a task from Informatica into the DAC repository. You can also choose to activate tables that had previously been inactive or inactivate tables in DAC that are associated with the task but do not appear in Informatica.

Flat Views

Opens a dialog that enables you to query for various objects, modify data, and do mass updates.

Setup View Right-Click Menu Commands

Table 15–12 describes the Setup view right-click menu commands.

Table 15–12 Setup View Right-Click Menu Commands

Command	Description
Test Connection	In Physical Data Sources tab, this command tests the database connection. In the Informatica Servers tab, this command tests the connection to the Repository Service and Integration Service. The DAC Server performs this command if the DAC Client is connected to a server. If the DAC Client is not connected to a DAC Server, then the DAC Client performs the command.
Patches	(Working Patches tab) ▪ **Re-Open Patch** - Changes the status of Closed patches to Open. ▪ **Close Patch** - Changes the status of Open patches to closed. ▪ **Export Patch** - Exports a patch and its contents from the DAC repository and saves it in an XML file in a directory you specify. ▪ **Add as Child Patch(es)** - Adds the selected patch as a child patch to a patch you specify. ▪ **Add Objects to Patch (Time Range)** - Adds contents to a patch based on a period between two timestamps that you specify. See Chapter 11, "Working With DAC Metadata Patches," for more information about patches.

Execute View Right-Click Menu Commands

Table 15–13 describes the Execute view right-click menu commands.

Table 15–13 Execute View Right-Click Menu Commands

Command	Description
Run Now	(Execution Plans tab) Runs the selected execution plan.
Add Refresh Dates	(Execution Plans tab) Prepopulates tables associated with the selected execution plan in the Refresh Dates subtab of the Physical Data Sources tab. This feature enables you to set or reset refresh dates manually in the Refresh Dates subtab of the Physical Data Sources tab before running an execution plan. **Note:** The Micro ETL Refresh Dates subtab of the Execution Plans tab is reserved for micro ETL processes.
Reset source(s)	(Execution Plans tab) Resets the refresh dates to null for all tables relevant to the execution plan. You can choose to reset refresh dates for tables in one or more source and target data sources.
Add Object(s) to Patch	(Execution Plans tab) Enables you to add the selected execution plan or all execution plans in the list to a patch. When you add an execution plan to a patch, only the execution plan parent object is listed in the Contents subtab of the Working Patches and Applied Patches tabs. However, the execution plan child objects will be exported to the patch XML file even though they do not appear in the Contents subtab. For information about which child objects are added to the patch XML file, see "About Patch Contents".
Number of Tasks by Depth	(Execution Plans tab) Shows how many tasks are at each depth level for each task execution graph. For more information, see "Performance Tuning the ETL Process Using Tasks by Depth Command".
Show Concurrent ETL Dependency	(Execution Plans tab) Shows if two execution plans are sharing any common source or target tables. When there are such instances, DAC will serialize their execution. If there is no commonality, they can be run in parallel.
Build	(Execution Plans tab) Builds the execution plan, by assembling subject areas, tasks, indexes, tags, parameters, source system folders, and phases.
Auto Refresh	(Current Runs tab) Enables you to set an automatic refresh frequency for the selected execution plan.
Restart	(Current Runs and Run History tab) Restarts the selected execution plan.
Abort	(Current Runs and Run History tab) Stops the execution of the selected execution plan and changes the status to Failed.
Mark as Completed	(Current Runs and Run History tabs) Changes the status of a stopped or failed ETL to Completed. In the audit trail for this ETL, the status is Marked as Completed. Use this command with caution. It can cause the data warehouse to be inconsistent.
Get Run Information, Get Log File	(Current Runs and Run History tabs) Fetches the log file for this run from the DAC Server and saves it in the \ServerLog directory.
Get Run Information, Analyze Run	(Current Runs and Run History tabs) Saves a description of the run as an HTML file in the Log\Statistics directory.
Get Run Information, Get Chart	(Current Runs and Run History tabs) Displays a chart showing changes in task statuses over time in a separate window.

Table 15–13 (Cont.) Execute View Right-Click Menu Commands

Command	Description
Get Run Information, Get Graph	(Current Runs and Run History tabs) Displays a graph showing changes in task statuses over time in a separate window.
Flat Views	Opens a dialog that enables you to query for various objects, modify data, and do mass updates.

Common Elements of Interface Tabs

Some of the DAC interface tabs have common elements, which are displayed as columns or subtabs. Descriptions of the common elements follow.

Name

The Name column in a tab specifies the name of the database object.

Inactive

The Inactive column indicates whether a database object is active or inactive. You can inactivate a repository object by selecting the Inactive check box. Inactive objects do not participate in the ETL process.

Owner

The Owner column specifies the source system container in which the database object was created. For more information about object ownership, see "About Object Ownership in DAC".

Edit

The Edit subtab enables you to enter definition information for a new object or to edit the existing definition of an object that is selected in the top pane.

Description

The Description subtab displays a description of the object selected in the top pane. You can also edit the existing description of an object or add a new description.

Design View Tabs

The Design view provides access to functionality related to creating and managing subject areas.

This section describes the Design view interface tabs.

- Configuration Tags Tab
- Indices Tab
- Container Specific SQLs Tab
- Source System Folders Tab
- Source System Parameters Tab
- Subject Areas Tab
- Tables Tab
- Task Groups Tab
- Tasks Tab

Configuration Tags Tab

A configuration tag is an object that controls the inclusion of tasks in subject areas. When a task is "tagged" with a configuration tag, it is not eligible to be included in the collection of tasks for any subject area, unless the tag is part of the subject area definition "Include Task" property.

For more information, see:

- Working with Configuration Tags
- Configuration Tag Behavior and Best Practices

Include Tasks Column

If this check box is selected, the tasks that are assigned to a configuration tag will participate in the ETL process for the subject area to which this configuration tag is assigned.

For example, suppose Configuration Tag 1 is made up of Task 1 and Task 2, and Configuration Tag 1 is assigned to Subject Area 1. Task 1 and Task 2 will be executed when the execution plan for Subject Area 1 is executed, whether or not Task 1 and Task 2 relate to the tables that make up the subject area.

Configuration Tags Tab: Subject Areas Subtab

Lists the subject areas that belong to a configuration tag and enables you to add subject areas to a configuration tag.

Configuration Tag Tasks Only Column

This read-only field indicates whether configuration tag tasks are the only tasks associated with this subject area that will participate in the ETL process. If this check box is selected, only the tasks associated with the configuration tag will be chosen by DAC when the subject area is assembled.

See "Working with Configuration Tags" for more information.

Configuration Tags Tab: Tasks Subtab

Enables you to add or remove tasks from the configuration tag selected in the top pane.

For more information, see:

- Working with Configuration Tags
- Configuration Tag Behavior and Best Practices

Indices Tab

The Indices tab lists all the indexes associated with the selected source system container.

For more information, see:

- Index Behavior and Best Practices
- Managing Indexes
- Specifying Index Spaces for Indexes by Table Type
- Specifying How Many Indexes Can Be Created in Parallel
- Using Actions to Optimize Indexes and Collect Statistics on Tables
- Using Heuristics to Manage Tasks, Tables and Indexes

Note: It is recommended that you do not register any indexes for source tables.

For Teradata databases, only secondary indexes should be registered in DAC. You should not register primary indexes or the more complex indexes, such as single- and multi-table indexes, because they cannot be dropped and recreated. You can use SQL commands to drop and create such tasks in DAC.

Table Name Column

The table for which an index is created.

Index Usage Column

Specifies the index usage. Possible values are:

- **ETL**. An ETL index is used during the ETL process. DAC creates ETL indexes after the first task runs that writes to the table on which the indexes are created.
- **Query**. A query index is used during the reporting process and not during the ETL process. The last task that reads from or writes to a table creates the query indexes.
- **During ETL.** Like an ETL index, a During ETL index is used during the ETL process. The difference between an ETL index and a During ETL index is that with During ETL indexes you can override when the index is created by assigning it to a specific task. The During ETL index will be created after the task to which you assigned it runs.

  If you define a During ETL index, you need to make sure that the index is assigned to a task, or else the index will not be created. You should only assign one During ETL index to one task per execution plan. Only advanced DAC users with a thorough understanding of the execution plans should define During ETL indexes.

  For instructions on defining a During ETL Index, see "Defining a During ETL Index".

Databases Column

Lists the databases associated with the selected index.

Unique Columns Column

For unique indexes, the number of columns that will be unique.

Is Unique Column

Indicates whether the index is unique.

Is Clustered Column

Indicates whether the index is clustered. There can be only one clustered index per table.

Is Bitmap Column

Indicates whether the index is of the bitmap type.

Allow Reverse Scan Column

Applicable only for DB2-UDB databases. The index will be created with the Allow Reverse Scan option.

Always Drop & Create Column

Indicates whether the index will be dropped and created regardless of whether the table is being loaded using a full load or incremental load.

Always Drop & Create Bitmap Column

Indicates whether indexes of the bitmap type will be dropped and created regardless of whether the table is being loaded using a full load or incremental load.

Indices Tab: Actions Subtab

Lists the actions that have been set up for the selected index. For a description of index actions, see the following:

- Using Actions to Optimize Indexes and Collect Statistics on Tables
- Defining a SQL Script for an Action
- Assigning an Action to a Repository Object

Index actions can override the default index behavior for dropping and creating indexes by mode type (full load, incremental load, or both).

Action Type Column

The index action types are the following:

- Create Index
- Drop Index

Load Type Column

The load type specifies whether the SQL script is to be called for incremental runs, full runs, or both.

Action Column

The SQL statement or stored procedure that you define. You can define one or more SQL statements or stored procedures for each action.

Indices Tab: Columns Subtab

Displays a list of columns the index is made of.

Position Column

The ordinal position of the column in the index structure.

Sort Order Column

Indicates whether the index sort order is ascending or descending.

Container Specific SQLs Tab

The Container Specific SQLs tab lists custom SQL scripts that have been defined for container-specific heuristics rules and task actions.

You also use this tab to define new custom SQL scripts for heuristics rules and task actions.

The drop-down list on the rightmost side of the top pane toolbar enables you to switch between **Heuristic** custom SQL and **Task Action** custom SQL.

Heuristic Custom SQL

When you write custom SQL for a heuristics rule, it applies only to the specified source system container. For instructions on defining custom SQL, see "Writing Custom SQL for a Heuristics Rule".

For an overview of DAC heuristics, see "About DAC Heuristics".

Task Action Custom SQL

You define custom SQL for a task action by clicking in the Value field of the Edit subtab. Then, follow steps 4 through 10 in the procedure, "Defining a SQL Script for an Action". After you define a Task Action custom SQL, you assign it to a task using the Edit subtab in the Tasks tab. Select the Execution Type, "Container Specific Task Action," and then click in either the "Command for Incremental Load" or "Command for Full Load" field to display a list of Task Action custom SQLs. Select the appropriate custom SQL, and save the record.

Source System Folders Tab

The Source System Folders tab enables you to associate the logical folder in DAC with the physical Informatica folder.

Logical Folder Column

The name of the logical folder in DAC. This name is used in the task definition so that task definitions do not have to be cloned.

Physical Folder Column

The name of the physical Informatica folder. This name represents the actual folder in the Informatica repository. This name is used in the Ordered Tasks subtab of the Execution Plans tab.

Source System Parameters Tab

The Source Systems Parameters tab holds the source system parameters that apply to all tasks under a source system container. This tab enables you to view and edit existing source system parameters and to define new parameters.

For more information, see:

- Source System Parameter Behavior and Best Practices
- Defining and Managing Parameters

Data Type Column

The source system parameter data type. For a description of data types, see "Parameter Data Types".

Load Type Column

Specifies whether the parameter applies to full load commands, incremental load commands, or both.

Value Column

The parameter value.

Subject Areas Tab

A subject area is a logical grouping of tables related to a particular subject or application context. It also includes the tasks that are associated with the tables, as well as the tasks required to load the tables.

Subject areas are assigned to execution plans, which can be scheduled for full or incremental loads.

The Subject Areas tab lists all the subject areas associated with the selected source system container. It enables you to view and edit existing subjects areas and to create new ones.

For more information, see:

- Subject Area Behavior and Best Practices
- Designing Subject Areas

Configuration Tag Tasks Only Column

This column indicates whether configuration tag tasks are the only tasks associated with this subject area that will participate in the ETL process. If this check box is selected, only the tasks associated with the configuration tag will be chosen by DAC when the subject area is assembled.

For more information, see:

- Working with Configuration Tags
- Configuration Tags Tab

Last Designed Column

The timestamp indicates when the subject area was last designed.

Subject Areas Tab: Configuration Tags Subtab

The Configuration Tags subtab lists the configuration tags that are associated with this subject area and enables you to add new configuration tags to the subject area.

For more information, see:

- Working with Configuration Tags
- Configuration Tags Tab

Include Tasks Column

This read-only field indicates whether the configuration tag tasks will be executed.

Context Disabled Column

When this read-only check box is selected, the configuration tag is globally disabled.

Subject Areas Tab: Extended Tables (RO) Subtab

The Extended Tables (RO) subtab is a read-only tab that lists the extended tables associated with the selected subject area.

Included Column

Indicates whether the extended table is included in the selected subject area. If the check box is selected, the extended table will be included in the subject area assembly process.

Subject Areas Tab: Tables Subtab

Lists the tables that are associated with the selected subject area. It enables you to add tables to subject areas or to remove them.

For more information, see:

- Customizing ETL Processes
- Creating a Subject Area

Name Column

The name of the table associated with the selected subject area.

Subject Areas Tab: Tasks Subtab

Lists the tasks associated with the selected subject area. It enables you to add tasks to and remove tasks from a subject area and to inactivate tasks.

When you inactivate a task, it remains inactive even if you reassemble the subject area. When you remove a task from a subject area, it will be added back to the subject area upon reassembly.

For more information, see:

- Task Behavior and Best Practices
- Customizing ETL Processes

Parent Group Column

If the task belongs to a task group, this column displays the task group name.

Phase Column

The task phase of the ETL process.

Autogenerated Check Box

Indicates whether the task was automatically generated by the DAC's task generation process.

Is Group Check Box

Indicates whether the task belongs to a task group.

Primary Source Column

Connection for the primary source.

Primary Target Column

Connection for the primary target.

Subject Areas Tab: Task Source Tables (RO) Subtab

The Task Source Tables (RO) subtab opens in query mode and is read only. It enables you to query by task name, table name, table type or data source for the source tables for the tasks associated with the selected subject area.

Subject Areas Tab: Task Target Tables (RO) Subtab

The Task Target Tables (RO) subtab opens in query mode and is read only. It enables you to query by task name, table name, table type or data source for the target tables for the tasks associated with the selected subject area.

From this tab, you can access the Generate DW Table Scripts for Oracle right-click command, which enables you to update a subset of tables in the data warehouse schema for Oracle database types. For more information, see "Creating, Upgrading or Dropping Subsets of Tables in the Schema for Oracle Databases".

Tables Tab

The Tables tab lists the physical database tables defined in the database schema that are associated with the selected source system container. It enables you to view and edit existing tables and to create new ones.

For more information, see:

- Table Behavior and Best Practices
- Designing Subject Areas
- Adding a New Table and Columns to the Data Warehouse

Warehouse Column

Indicates whether the table is a warehouse table. If this option is not selected, the schema creation process will not include this table.

Image Suffix Column

Suffix for image tables. Applicable only to Siebel source tables. For more information about image tables, see the description of the Change Capture Scripts command in the section "Design View Right-Click Menu Commands".

Is MultiSet Column

Indicates whether the table is a MultiSet table. Applicable only to Teradata databases.

Has Unique Primary Index Column

Indicates whether the table has a Unique Primary Index. Applicable only to Teradata databases.

Tables Tab: Actions Subtab

Lists the actions that have been set up for the selected table. For a description of task actions, see the following:

- Using Actions to Optimize Indexes and Collect Statistics on Tables
- Defining a SQL Script for an Action
- Assigning an Action to a Repository Object

The table action types enable you to trigger SQL scripts to analyze or truncate tables. Table actions for analyzing or truncating tables override all other table properties.

Action Type Column

The default actions on a table during an ETL process are truncating a table and analyzing a table. If you want to override any of these syntaxes, you can define actions by mode (full or incremental). Once an action is defined, it overrides the default behavior.

Note: You can associate an action to multiple objects at the same time. First, identify the objects that you want to associate an action with by using the query functionality. Then, right-click on the results displayed, and select Associate Actions. Next, select an action and the mode you want to associate it with.

The table action types are the following:

- **Analyze Table**

Use this type to analyze tables.

- **Truncate Table**

  Use this type to truncate tables.

Load Type Column

The load type specifies whether the SQL script is to be called for incremental runs, full runs, or both.

Action Column

Actions are the SQL statement or stored procedure that you define. You can define one or more SQL statements or stored procedures for each action.

Tables Tab: Conditional for Tasks (RO) Subtab

Displays a read-only list of tasks that have defined the selected table as one of the conditional tables.

Task Phase Column

Task phase of the ETL process. This information is primarily used for dependency generation. Certain phases, such as Change Capture and Query Index Creation, are not available for you to assign to a task. The DAC Server uses the task phase to prioritize tasks and to generate a summary of the time taken for each of the phases.

Tables Tab: Columns Subtab

The Columns subtab displays the columns that belong to the selected table.

Position Column

An integer that represents the ordinal position of the column in the table structure.

Data Type Column

The data type of the column. Possible values are the following:

- **NUMBER.** Indicates the data is an integer.
- **CHAR.** Indicates a fixed-length character string. All values stored in a CHAR column have the length specified in the Length column (in the Columns subtab). If you insert a value that is shorter than the column length, blank spaces are added to the value up to the specified column length. If you try to insert a value that is too long for the column, an error is returned.
- **VARCHAR.** Indicates the data is a variable-length string, which can be from 0 to 4000 characters long. You must specify a string length in the Length column.
- **DATE.** A string that represents a date and time.
- **TIMESTAMP.** A string that represents a data, time, and fractional seconds.

Length Column

An integer value that indicates how many characters a column can hold.

Precision Column

An integer value that indicates the total number of digits to the left and right of the decimal point.

Nullable Column

Indicates the column can have a NULL value. "NULL" stands for the absence of a known value. If the Nullable check box is not selected, the column is declared NOT NULL, which means the column must contain a value. When inserting a row without providing an explicit value for a column, that column will get the default value if the column has one or NULL if the column is not specified as NOT NULL. Otherwise, an error is generated

Unicode Column

Indicates whether the column is Unicode.

Foreign Key to Table Column

Indicates the table in which the column is a foreign key.

Foreign Key to Column

Indicates the column that is a foreign key.

Default Value Column

The default value must be the same data type as the column. If the value is a CHAR or VARCHAR type, the value must be enclosed within quotes ("<value>"). If the value is a number, it does not need to be enclosed within quotes.

Teradata Length Column

Applicable to Teradata databases only. Indicates the column length.

Teradata Precision Column

Applicable to Teradata databases only. Indicates the column precision.

Teradata Primary Index Order Column

Applicable to Teradata databases only. On Teradata databases, it is important to have a primary index for every table. This column indicates the position that a given column will have in the primary index. If the value is 0 or empty, the column is not part of a primary index.

For Teradata Statistics Column

Applicable to Teradata databases only. If the check box is selected, DAC will collect statistics on the specified column.

Tables Tab: Indices (RO) Subtab

The Indices (RO) subtab displays a read-only list of indexes that belong to the selected table.

Index Usage Column

Specifies the index usage type as ETL, Query, or During ETL. An ETL index is typically used during the ETL process. A Query index is an index used only during the reporting process. It is recommended that you have a clear understanding of when and where the index will be used at the time you register the index. For a description of the possible values see, "Index Usage Column".

Unique Columns Column

For unique indexes, the number of columns that will be unique.

Is Unique Column

Indicates whether the index is unique.

Is Clustered Column

Indicates whether the index is clustered. There can be only one clustered index for each table.

Is Bitmap Column

Indicates whether the index is of the bitmap type.

Allow Reverse Scan Column

Applicable only for DB2-UDB databases. The index will be created with the Allow Reverse Scan option.

Tables Tab: Multi-Column Statistics Subtab

Applicable to Teradata databases only. Lists the columns for which multi-column statistics are collected.

Tables Tab: Related Tables Subtab

The Related Tables subtab lists tables that are related to the selected table. Related tables participate in the ETL process in addition to the tables that are associated with this table.

Tables Tab: Source for Tasks (RO) Subtab

The Source for Tasks (RO) subtab displays a read-only list of tasks that use the selected table as a source.

Task Phase Column

Task phase of the ETL process. This information is primarily used for dependency generation. Certain phases, such as Change Capture and Query Index Creation, are not available for you to assign to a task. The DAC Server uses the task phase to prioritize tasks and to generate a summary of the time taken for each of the phases.

Build Image Column

Applicable for Siebel transactional sources only. Indicates change capture for the primary/auxiliary source tables will be executed.

Type Column

Table type.

Data Source Column

Data source for the task.

Tables Tab: Target for Tasks (RO) Subtab

The Target for Tasks (RO) subtab displays a read-only list of tasks that use the selected table as a target.

Task Phase Column

Task phase of the ETL process. This information is primarily used for dependency generation. Certain phases, such as Change Capture and Query Index Creation, are

not available for you to assign to a task. The DAC Server uses the task phase to prioritize tasks and to generate a summary of the time taken for each of the phases.

Data Source Column

Data source for the task.

Truncate Always Column

Indicates the target table will be truncated regardless of whether a full or incremental load is occurring.

Truncate for Full Load Column

Indicates the target tables will be truncated only when a full load is occurring.

Analyze Tables Column

Indicates the tables will be analyzed during the ETL process.

Task Groups Tab

The Task Groups tab lists all the task groups associated with the selected source system container. It also enables you create new task groups. **Note:** A task can belong to only one task group.

For more information, see:

- Task Group Behavior and Best Practices
- Creating a Task Group

Restart All on Failure Column

Indicates the tasks in this task group will be restarted if one or more tasks fails during an ETL process.

Execute Serially Column

Indicates the tasks in this task group will be executed sequentially. This property overrides the execution order.

Truncate Always Column

Indicates the target tables will be truncated regardless of whether a full or incremental load is occurring. Any indexes registered for the target table are dropped before the command is executed and recreated after the command completes successfully. When indexes are dropped and created, the table is analyzed so that the index statistics are up-to-date. Make sure if you select this option that all the tasks write to the same data source.

Truncate for Full Load Column

Indicates the target tables will be truncated only when a full load is occurring. Any indexes registered for the target table are dropped before the command is executed and recreated after the command completes successfully. When indexes are dropped and created, the table is analyzed so that the index statistics are up-to-date.

Task Groups Tab: Child Tasks Subtab

The Child Tasks subtab lists the tasks that belong to the selected task group. It also enables you to add child tasks to a task group selected in the top pane.

Task Phase Column

Task phase of the ETL process.

Primary Logical Source Column

Logical database connection for the primary source database.

Primary Logical Target Column

Logical database connection for the primary target database.

Dependency Order Column

Order among the tasks in the task group in which this task will be executed. If two or more tasks have the same execution order and the Execute Serially flag is not checked, DAC will run the tasks in parallel.

Heuristic Driver Column

Indicates this task will specify whether the entire task group is executed. You can specify multiple tasks as heuristic drivers. For more information, see "DAC Heuristics and Task Groups".

Task Groups Tab: Source Tables (RO) Subtab

The Source Tables (RO) subtab lists the tables used for extracting data by the selected task group.

Table Column

Name of source table.

Task Column

Task that extracts data from the table.

Type Column

Source table type. If a table is marked as Primary or Auxiliary and the Build Image property of the task is selected, the change capture process is invoked. There are special tasks that force the base table data to be extracted when data in auxiliary tables changes. A table can be neither Primary nor Auxiliary but still be used for getting some attributes to populate a dimension or fact table. The changes in these kinds of source tables are not reflected in the dimension or fact table once the data is populated.

Task Groups Tab: Target Tables (RO) Subtab

The Target Tables (RO) subtab is a read-only tab that lists the tables into which the task group loads data.

Table Column

Name of the target table.

Task Column

Task that loads data into the target table.

Tasks Tab

The Tasks tab lists all the tasks associated with the selected source system container. For more information, see:

- Task Behavior and Best Practices
- Creating Tasks in DAC for New or Modified Informatica Workflows

Parent Group Column

If the task is a member of a group, this field lists the task group name.

Group Order Column

The order in which the task is defined to execute in a certain group.

Command for Incremental Load Column

A table can be loaded in full mode or incremental mode. Full mode refers to data loaded for the first time or data that is truncated and then loaded. Incremental mode refers to new or changed data being added to the existing data.

DAC maintains a last refresh timestamp whenever a table is changed during the ETL process. (You can view this timestamp by selecting the Refresh Dates subtab on the Physical Data Sources tab.) If a table has a timestamp, the command appearing in this column is executed. If a table does not have a timestamp, the command for a full load is executed. If the execution type is Informatica, the workflow name is used as the command.

Command for Full Load Column

If a table has no last refresh timestamp, this command is executed.

Logical Folder Column

Only for execution type of Informatica. The folder in which the workflow resides. Note: The name cannot contain spaces.

Primary Source Column

Logical database connection for the primary source database.

Primary Target Column

Logical database connection for the primary target database.

Task Phase Column

Task phase of the ETL process. This information is primarily used for dependency generation. Certain phases, such as Change Capture and Query Index Creation, are not available for you to assign to a task. The DAC Server uses the task phase to prioritize tasks and to generate a summary of the time taken for each of the phases.

Execution Type Column

Tasks are executed based on their execution type. Possible values are the following:

- **Informatica**. Task is invoked on an Informatica Integration Service using pmcmd.
- **External Program.** Task is an operable program on the operating system where the DAC Server is running. This program can be a batch file, shell script, or any other program that can be run like a bulk loader.

- **SQL File**. Task is a SQL script in XML or SQL format.
- **Stored Procedures**. Task is a stored procedure that has been defined on the database.

In addition, there are several internal execution types that you will not be able to select when creating new tasks. Tasks of these types are categorized as either internal change capture tasks or internal data warehouse tasks. All of these tasks are color-coded in pink in the Tasks tab.

- **IMG_BUILD**

  Used for internal change capture. If you are using multiple Siebel transactional sources, you cannot change the behavior of the change capture process. This task requires change capture tables to be created on the other sources also. When adding additional Siebel sources, on the Tables tab in the Design view, right-click and select **Change Capture Tasks**. This action generates change capture tasks. Use this same action to disable or delete change capture tasks.

- **IMG_SYNC**

  Used for internal change capture. If you are using multiple Siebel transactional sources, you can create this task for the additional tasks for doing similar change capture sync processes. You cannot change the behavior of the change capture sync process. This task requires change capture tables to be created on the other sources also. This task should be used with discretion for Siebel sources only.

Execution Priority Column

Indicates the order in which the task is executed. If two or more similar tasks (tasks having that same phase, similar truncate properties, the same number of successors, and the same number of source tables) have the same priority, the order occurs randomly.

Build Image Column

Applicable for Siebel transactional sources only. Indicates the change capture for the primary/auxiliary source tables will be executed.

Continue on Error Column

When this check box is selected, if the command fails, the dependent tasks are not stopped. However, if any autogenerated tasks fail, the dependent tasks are stopped.

Tasks Tab: Actions Subtab

The Actions subtab lists the actions that have been set up for the selected task. For a description of task actions, see the following:

- Defining a SQL Script for an Action
- Assigning an Action to a Repository Object
- Using Actions to Optimize Indexes and Collect Statistics on Tables

Action Type Column

Action types are predefined categories of task behaviors that trigger the execution of a SQL script. The following types are available:

- **Preceding Action**. Use to execute a SQL script before a task runs.
- **Success Action**. Use to execute a SQL script after a task runs successfully.

- **Failure Action**. Use to execute a SQL script if a task fails during its execution.
- **Restart Action**. Use to execute a SQL script when a task that previously failed is restarted.
- **Upon Failure Restart Action**. Use to execute a SQL script to restart a task that fails.

Load Type Column

The load type specifies whether the SQL script is to be called for incremental runs, full runs, or both.

Action Column

You define the task action in the form of a SQL statement or stored procedure. You can define one or more SQL statements for each action. Double-click in the Action field to open the Choose Action dialog, where you can select the appropriate action.

You define the SQL statement or stored procedure for an action in the Task Actions dialog, which you access by selecting Tools, then Seed Data, then Actions, and then Task Actions. For instructions, see "Defining a SQL Script for an Action".

Tasks Tab: Conditional Tables Subtab

Lists the tables that, if included in an execution plan, cause the optional task selected in the top pane to be executed.

For example, the Order Item fact table is a conditional table associated with the optional task called UpdateRecencyCat in Person Dimension. The UpdateRecencyCat in Person Dimension task is executed only when the Order Item fact table is included in an execution plan.

Tasks Tab: Configuration Tags Subtab

Lists the configuration tags to which the selected task belongs. It also enables you to associate the selected task with a configuration tag.

For more information, see:

- Configuration Tag Behavior and Best Practices
- Working with Configuration Tags

Include Tasks Column

This read-only field indicates whether the configuration tag tasks will be executed.

Context Disabled Column

If this check box is selected, the configuration tag is globally disabled.

Tasks Tab: Extended Properties Subtab

Enables you to specify the following extended properties for the task selected in the top pane:

- **Looping of workflows**. Enables you to configure the full and incremental load commands for tasks to repeat (or loop) multiple times during the execution of an ETL process. For instructions, see "Looping of Workflows".
- **Heuristics.** Enables you to assign a heuristics rule to a task. For instructions, see "Associating a Heuristics Rule With a Task".

Tasks Tab: During ETL Indices Subtab

Lists the During ETL index that is assigned to the selected task. Also enables you to assign a During ETL index to the selected task.

Tasks Tab: Parameters Subtab

Lists the parameters associated with the selected task and enables you to configure task level parameters. This parameter takes precedence over source system parameters when the name is the same. For more information about parameters, see "Defining and Managing Parameters".

Name Column

Name of the parameter.

Data Type Column

Parameter data type. For a description of data types, see "Parameter Data Types".

Load Type Column

Indicate whether the parameter applies to full load commands, incremental load commands, or both.

Value Column

The parameter value.

Comments Column

A text field in which you can add comments.

Tasks Tab: Phase Dependency Subtab

Lists the task phase dependencies for the selected task. Also enables you to define task phase dependencies. The DAC Server uses the task phase dependency to prioritize tasks. By changing the phase dependency properties of a task, you change the task's execution order. For instructions on setting a task phase dependency, see "Setting a Task Phase Dependency".

Action Column

The action to be taken in relation to the phase dependency. Possible values are the following:

- **Wait**. Indicates the selected task will wait to be executed until the tasks of a specified phase have been executed.
- **Block**. Indicates the selected task will block all tasks of the specified phase from being executed until is has been executed.

Grain Column

Applicable only for blocks. Enables you to specify whether the action you choose affects all tasks of a specified phase or related tasks. Possible values are the following:

- **All**. Indicates the action will affect all tasks.
- **Related**. Indicates the action will affect only related tasks. You can view a task's related tasks by navigating to the Execution Plans tab, All Dependencies subtab and viewing the specified task's predecessor tasks.

Scope Column

For multi-source execution plans only. Specifies how the Block action of the phase dependency behaves in relation to multi-source execution plans. Possible values are the following:

- Both

  Indicates the blocking action is active for tasks that have the same source and target physical data source connections.

- Source

  Indicates the blocking action is active for tasks that have the same source physical data source connection.

- Target

  Indicates the blocking action is active for tasks that have the same target physical data source connection.

- None

  Indicates the blocking action is active for all tasks regardless of the source and target physical data source connections.

Phase Column

The ETL phase that will apply to the Action and Grain properties.

Tasks Tab: Refresh Date Tables Subtab

Lists the refresh date tables for the selected task. Also enables you to add or remove refresh date tables.

When DAC executes a task, it determines the read and write mode based on the refresh dates that DAC stores for the database connection and the source and target table name combination of that task. However, if you want to determine the mode of both read and write operations based on a table that is not a source or target table, you can define the table as a refresh date table.

Tasks Tab: Refresh Dates (RO)

The Refresh Dates (RO) subtab is a read-only tab that lists the refresh dates by table for the selected task. For more information about refresh dates, see "About Refresh Dates and DAC's Incremental Load Strategy".

Tasks Tab: Source Tables Subtab

The Source Tables subtab lists the tables from which the selected task extracts data.

Type Column

Table type. Possible values are the following:

- **Primary**

  Indicates the table is a primary source of data.

- **Auxiliary**

  Indicates the table is a secondary source of data.

- **Lookup**

  Indicates the table is a lookup table.

Note: If a table is marked as Primary or Auxiliary and the Build Image property of the task is selected, the change capture process is invoked. There are special tasks that force the base table data to be extracted when data in auxiliary tables change.

A table can be neither Primary nor Auxiliary but still be used for getting some attributes to populate a dimension or fact table. The changes in these kinds of source tables are not reflected in the dimension or fact table once the data is populated.

Data Source Column

Data source for the table. When a data source is not specified, the default is the task's primary source.

Alias Column

An alternative name for a source table.

When multiple tasks from different execution plans read from the same source table and write to the same target table, it is necessary to define an alias for one of the source tables in order for DAC to track refresh dates accurately.

DAC stores refresh dates by the combination of data source and table name. If an alias is not defined in situations in which tasks from two different execution plans read from the same source table and write to the same target table, issues with data inconsistency can arise.

Tasks Tab: Subject Areas (RO) Subtab

This read only tab lists the subject areas that are associated with the task selected in the top pane.

Tasks Tab: Target Tables Subtab

The Target Tables subtab lists the tables into which the selected task loads data.

Type Column

Table type.

Data Source Column

Data source for the target table. If no data source is specified, this value defaults to the task's primary target.

Truncate Always Column

Indicates the target tables will be truncated regardless of whether a full or incremental load is occurring. Any indexes registered for this table are dropped before the command is executed and recreated after the command completes successfully. When indexes are dropped and created, the table is analyzed so that the index statistics are up-to-date.

Truncate for Full Load Column

Indicates the target tables will be truncated only when a full load is occurring. Any indexes registered for this table are dropped before the command is executed and recreated after the command completes successfully. When indexes are dropped and

created, the table is analyzed so that the index statistics are up-to-date. When the Truncate Always option is selected, this option is unnecessary.

Analyze Column

Indicates the tables will be analyzed during the ETL process.

Setup View Tabs

The Setup View provides access to functionality related to setting up DAC system properties, Informatica servers, database connections, and email notification.

This section describes the Setup view interface tabs.

DAC System Properties Tab

The DAC System Properties tab enables you to configure various properties that determine the behavior of the DAC Server.

Allow Clients to Remember User Password

When set to True, this property enables the DAC clients to remember the user's password when the user logs into the DAC repository. User passwords are encrypted and stored in the client machine's file system

Analyze Frequency (in days)

For DAC repository tables, this property specifies the frequency (in days) the DAC Client automatically updates the table and index statistics for the DAC repository. The value must be numerical.

The DAC repository tables need to have table and index statistics kept up to date to maintain the health of the repository.

Analyze Table After Query Index Creation

Possible values are True and False.

If this property is set to True, tables will be analyzed after the Query type indexes have been created (even if the Analyze check box for the target table is deselected).

If this property is set to False, the table is not analyzed (as long as the Analyze check box for the target table is deselected).

Auto Restart ETL

Possible values are True and False.

When this property is set to True, an ETL process that is running when the DAC Server abnormally terminates will continue running when the DAC Server is restarted.

When set to False, an ETL process that is running when the DAC Server abnormally terminates will not automatically restart when the DAC Server restarts. The ETL status will be updated to Failed. An administrator will have to manually restart the ETL.

Concurrency Level

The maximum number of ETL processes that can run in parallel. When this property is set to 1 (the default), the ETL process runs in the same process space with the DAC Server. When this property is set to more than 1, the ETL processes run as separate OS processes. DAC interprets invalid values as 1.

Drop and Create Change Capture Views Always

Possible values are True and False.

When set to True (the default value), the DAC Server drops and creates change capture views every time it performs a change capture process, including for both full and incremental loads.

When set to False, the DAC Server will drop and create views selectively, using the following rules:

- In full mode:
  - During the change capture phase, views will be dropped and created as full views.

  - During the change capture sync process, incremental views will be generated.
- In incremental mode:
  - If the view exists, it will not be dropped and created.
  - If the view does not exist, the incremental view will be created.

Dryrun

Possible values are True and False.

Indicates whether tasks are executed without invoking Informatica workflows. The following processes are executed: change capture, truncation of tables, drop and creation of indexes, and analyze statements.

This option should be used for debugging purposes only and not used in a production environment.

Generic SQL Concurrency Limit

When the execution plan includes several tasks with the execution type of SQL or Stored Procedure, this property controls how many of such tasks will be run concurrently at any given time. The value should be numerical. The changes will be effective for the next ETL. Restarting the DAC Server is not required.

Generic Task Concurrency Limit

Determines how many tasks with execution types other than Informatica can be run concurrently. The value must be numerical.

To set this value, you should consider what the external tasks do. For example, if the tasks open connections to a database, you should consider how this would affect the predefined tasks. The value must be numerical.

HeartBeatInterval

The frequency (in seconds) the DAC Server polls all subsystems, including database connections, to check whether they are in a consistent state. If the subsystems are not in a consistent state, recovery and correction mechanisms are triggered. The value must be numerical. For example, a value of 900 (the default value) indicates the system will perform subsystem diagnostics and recovery procedures every 900 seconds.

InformaticaFileParameterLocation

Directory where the Informatica parameter files are stored. This is the directory in which the DAC Server creates the Informatica parameter files and from which Informatica reads the files.

No Run

Possible values are True and False.

When this property is set to True, tasks are generated in the Task Details subtab of the Current Runs tab, but the tasks are not executed and the refresh dates are not updated. Use this property for debugging purposes only.

Output Redirect

Possible values are True and False.

When set to True, logging information and standard output and errors are redirected to files in the log directory. The file containing standard output starts with out_ and

ends with the .log extension. The standard error messages are in the file starting with err_ and ending with the .log extension.

If this property is set to False, the logging information is directed to the machine's standard output and error files, which typically defaults to the console from which the DAC Server was launched if the server was launched in a visible console mode. If the server is launched as a Windows service, the logging information is directed to the service log. If the DAC Server was launched with the command shell not visible, all logging information is deleted.

Repository DB Pool Size

Indicates the maximum number of connections to the DAC Repository that the DAC Server will maintain.

The idle DAC Server maintains a small spool of pre-existing database connections to the DAC Repository for efficiency and resource optimization. It increases the pool size to the value specified in this property when running a regular ETL process and reverts to the default on completion of the ETL process.

You should consult with the DBA for the DAC Repository database when setting this property.

Repository Name

Unique name for the DAC Repository. This name is displayed on the title bar.

Repository Upgrade Lockdown

Possible values are True and False.

When set to True, the DAC Repository cannot be upgraded to a newer version.

SQL Trace

Possible values are True and False.

When set to True, a hint is sent to the database connectivity layer of the DAC Server to enable SQL tracing. Thus, every SQL statement that is run by the DAC Server is spooled to the appropriate output log file.

It is recommended that you set this property to False, because enabling SQL tracing causes the log files to grow extremely large in a short period.

Scheduler.Poll.Interval

Frequency (in seconds) the DAC Server polls for changes in the schedule configuration.

Script After Every ETL

The name of the script or executable to be run after every execution plan. The file must be placed in the DAC\scripts directory. For example, after running an execution plan, you might want to run a process or perform certain tasks. These can be contained in a script or executable.

Script Before Every ETL

The name of the script or executable to be run before every execution plan. The file must be placed in the DAC\scripts directory. For example, before running an execution plan, you might want to run a process or perform certain tasks. These can be contained in a script or executable.

The execution plan will run only after the external process has finished. Therefore, it is important that the script or executable does not fail.

Server Log Level

The output logging level. Possible values are the following:

- FINEST
- FINER
- FINE
- CONFIG
- INFO
- WARNING
- SEVERE

The values are case sensitive. The values produce varying degrees of output reporting detail, with the Severe value producing minimal log details, and the Finest value producing the most extensive amount of reporting.

The recommended value for production environments is INFO.

Test Run

Possible values are True and False.

When set to True, the execution plan will not stop on errors.

Verify and Create Non-Existing Indices

Possible values are True and False.

When set to True, indexes defined in the DAC Repository will be automatically created in the data warehouse database during an incremental load.

Note: When this system property is set to True, the DAC Server verifies whether indexes defined in the DAC Repository are also defined in the data warehouse database. This verification process can delay the execution of an execution plan.

WebClient.Poll.Interval

Frequency (in seconds) the DAC Client polls the DAC Server for its status when the DAC Server is running in Web mode.

WorkerPoolSize

The number of worker threads available in the worker pool.

The worker threads do operations such as:

- drop and create indexes
- truncate and analyze tables
- execute SQL scripts and Informatica workflows

The value for this property should correspond to the number of task details that are anticipated to run in parallel.

Email Recipients Tab

This tab enables you to set up a list of email addresses that will be notified about the status of the ETL process. For instructions, see "Setting Up Email Notifications in the DAC Client and Server".

Name Column

Logical name of the user to be notified.

Email Address Column

Email address where the notification is sent.

Notification Level Column

The notification levels are as follows:

- **10**

  Notifies recipient of success or failure of each task.

- **5**

  Notifies recipient of success of failure of the entire ETL process.

- **1**

  Notifies recipient that ETL completed successfully.

External Executors Tab

You can configure DAC to integrate with external executors other than Informatica. This tab enables you to view, edit, and create external executor integrations with DAC. See Chapter 13, "Integrating DAC With Other ETL Tools" for more information.

Name Column

Name of the external executor you are integrating with DAC.

Type Column

Execution type for the tasks that will be executed by the external executor. You define the execution type by going to **Tools**, **Seed Data**, **Execution Type**. See "Registering an External Executor in DAC".

Properties Subtab

The Properties subtab contains the properties that you must configure to integrate DAC with other ETL tools. See Chapter 13, "Integrating DAC With Other ETL Tools" for more information.

Property Order Column

Used to sort the records in the subtab. Does not affect the functionality of the external executor.

Name Column

Name of the property.

Value Column

The value that defines the property.

Informatica Servers Tab

The Informatica Servers tab enables you to register one or more Informatica Integration Service services and one Informatica Repository Service and to specify how many workflows can be executed in parallel on each Integration Service. The DAC Server automatically load balances across the Integration Service services.

> **Note:** You can install multiple Informatica Integration Service services and point them to a single Informatica repository. You need to register each Informatica Service in DAC and specify a unique machine name and service name. For instructions on registering an Informatica Integration Server and Repository Service in DAC, see "Registering Informatica Services in DAC".

Name Column

Name of the Informatica Integration Service or Repository Service.

Type Column

Type of server.

- **Informatica**

  Specifies the Informatica Integration Service.

- **Repository**

  Specifies the Informatica Repository Service.

Service Column

The Informatica Integration Service name.

Server Port Column

Port number used by the Informatica Integration Service or Repository Service to listen to requests.

Domain Column

Informatica domain name (as seen in Informatica Administrator.

Login Column

Informatica repository user name who has Administrator privileges for the Informatica repository. Note that DAC must log in to Informatica as an Informatica repository Administrator user that is configured in the native security domain.

Password Column

Password for the user specified in the Login field.

Num Parallel Workflows per EP Column

The maximum number of workflows that can be executed in parallel on the Informatica Integration Service.

Repository Name Column

Name of the Informatica Repository Service that manages the Informatica repository for Oracle BI Applications.

Note: You deploy only one Informatica Repository Service, but you can deploy multiple Informatica Integration Service services.

Physical Data Sources Tab

The Physical Data Sources tab lists the connection pool information for the transactional and data warehouse databases. In this tab, you can view and edit existing physical data source connections and create new ones. For instructions on setting up connection pools, see "Setting Up Physical Data Sources".

Name Column

Logical name for the physical data source.

Type Column

Physical data source type. Possible values are the following:

- **Source**
- **Warehouse**
- **Informatica Repository**
- **DAC Repository**
- **Other**

Connection Type Column

Type of database connection. Possible values are the following:

- **BI Server**

  For Oracle Fusion Applications only. Indicates the BI Server is registered as a data source.

  For instructions on registering the BI Server in DAC, see "Integrating DAC and Oracle BI Server".

- **Oracle (OCI8)**

  Connects to an Oracle database using the tnsnames entry.

- **Oracle (Thin)**

  Connects to an Oracle database using thin driver.

- **DB2**

  DB2 UDB database.

- **DB2-390**

  DB2 390 database.

- **MSSQL**

  Microsoft SQL Server database.

- **Teradata**

  Teradata database.

- **Flat File**

Connection String Column

If you are using:

- **Oracle (OCI8)**: Use the tnsnames entry.

- **Oracle (Thin)**: Use the instance name.
- **SQL Server**: Use the database name.
- **DB2-UDB/DB2-390**: Use the connect string as defined in the DB2 configuration.
- **Teradata**: Use the database name.

Table Owner Column

Name of the table owner.

Num Connections per EP Column

Maximum number of database connections this connection pool can contain.

DBHost Column

Host machine where the database resides. This field is mandatory if you are using Oracle (Thin), MSSQL, or Teradata, but is not required if you are using Oracle (OCI8), DB2, or DB2-390.

Port Column

Port where the database receives requests. Required for Oracle (Thin) and MSSQL databases. Not required for Oracle (OCI8), DB2, or DB2-390, or Teradata databases.

Source Priority Column

Indicates the order in which DAC loads data from different sources participating in a multi-source execution plan. This property ensures that tasks attempting to write to the same target table will not be in conflict. If two sources are given the same priority, DAC will randomly stagger the tasks.

Data Source Number Column

User defined number of the data source.

Default Index Space Column

The Default Index Space property specifies the default index space for the physical data source. When indexes are dropped and created, they are created in this index space.

Num Parallel Indexes Per Table Column

Use this field to specify how many indexes are to be created in parallel for each table associated with the specified physical data source connection. For example, if a table with 30 indexes is the only task running in an execution plan at a given time, and you specify that 10 indexes are to be created in parallel, 10 of the 30 indexes will be created in parallel.

The number of indexes that can be created in parallel is limited by the value you set in the Num Connections per EP property.

The NumParallel Indexes Per Table property specifies the number of parallel indexes DAC will create for *all* tables associated with a specified physical data source connection. In contrast, you can use the Parallel Indexes subtab of the Physical Data Sources tab to specify the number of parallel indexes DAC will create at the table level.

JDBC Driver (Optional)

Specifies a JDBC driver for the data source connection. The value in this field must conform to the database specifications.

URL (Optional)

Specifies a JDBC URL for the data source connection. The value in this field must conform to the database specifications.

Physical Data Sources Tab: Analyze Frequencies Subtab

Lists the values for the Analyze Frequencies attribute by table type.

Physical Data Sources Tab: Extended Properties Subtab

Enables you to specify the following extended properties for the physical data source selected in the top pane:

- **Connection pool name** is used to register the BI Server as a data source (applies only to Oracle Fusion Applications). For instructions on integrating DAC and the BI Server, see "Integrating DAC and Oracle BI Server".
- **Event delay** is used to configure the extracts for the different data sources in a multi-source environment to occur independently. For instructions on setting up an event delay, see "Setting Up Extract Delays, Event Delays and Data Source Notifications".
- **Data source usage notifications** is used to initiate email notifications about data source usage and to define custom SQL to be executed based on the usage. For instructions on setting up notifications, see "Setting Up Data Source Usage Notifications".
- **Time Difference Override** is used to specify the time difference (in minutes) between the DAC Server machine and the physical data source database machine (if the two machines reside in different time zones). This value is required for certain data sources, such as Oracle Fusion Applications and other unknown data sources, in order for DAC to determine refresh dates.

  DAC updates refresh dates for all the source (primary and auxiliary) and target tables after an ETL process runs successfully, based on the ETL start time of the transactional system's database. If the DAC Server machine and the transactional system's database are in different time zones (adjusted to the timestamp of the transactional system's database), DAC automatically computes the time difference between the DAC Server machine and the transactional system database, except for Oracle Fusion Applications and other unknown data sources. For a detailed description of refresh dates, see "About Refresh Dates and DAC's Incremental Load Strategy".

  For Oracle Fusion Applications and other unknown data sources, if the DAC Server machine and the transactional system's database are in different time zones, you must set the Time Difference Override extended property to specify the time difference between the DAC Server machine and the transactional system's database. Once the override is provided, DAC does not issue a SQL to figure out the current timestamp; instead, DAC uses the value for this override to compute the timestamp for refresh date purposes.

  For instructions on setting the override extended property for Oracle Fusion Applications source system, see "Integrating DAC and Oracle BI Server".

  For unknown data sources (such as those for which DAC cannot figure out the current timestamp), in the Physical Data Sources tab of the Setup view, set the Connection Type to Flat File, and then define the Time Difference Override extended property in the Extended Properties subtab of the Physical Data Sources tab. Set the value for this property to specify the time difference between the DAC Server machine and the transactional system's database.

Name Column

The name of the DAC extended property.

Value Column

The definition for the extended property.

Physical Data Sources Tab: Index Spaces Subtab

Enables you to specify index spaces for indexes by table type. For instructions, see "Specifying Index Spaces for Indexes by Table Type".

Table Type Column

Table type for which you want to specify a tablespace.

Index Space Column

Specifies the name of the index space.

Note: You must create the index space on the database before you specify an index space in DAC.

Physical Data Sources Tab: Parallel Indexes Subtab

Enables you to specify how many indexes can be created in parallel for a given table.

Note: Use this property to specify the number of parallel indexes DAC will create for a specific table. Use the Num Parallel Indexes Per Table property in the Physical Data Sources tab to specify the number of parallel indexes DAC will create for all tables associated with a specified physical data source connection.

For more information, see "Specifying How Many Indexes Can Be Created in Parallel".

Name Column

Name of the table on which the indexes will be created.

Number of Parallel Indexes Column

Number of indexes that can be created in parallel for the specified table.

Physical Data Sources Tab: Refresh Dates Subtab

During an ETL process, this date is captured for all target tables and source tables of the type primary and auxiliary. DAC uses this date in the change capture process, during parameter generation, when choosing between full and incremental loads, and when deciding whether to truncate a table. (Does not apply to micro ETL processes.)

For more information about refresh dates, see "About Refresh Dates and DAC's Incremental Load Strategy".

Note: Refresh dates for micro ETL processes are captured in the Micro ETL Refresh Dates subtab of the Execution Plans tab.

Name Column

Name of source or target table.

Execution Plan Column

The name of the execution plan to which the source or target table belongs.

Refresh Date Column

The refresh date for the source or target table.

Analyze Date Column

Indicates when the table was analyzed.

Number of Rows Column

Valid for target tables only. Indicates the total number of rows in the table after the table has been loaded.

Working Patches Tab

The Working Patches tab lists patches that have the status Open and that have not been applied to the repository. For more information, see "Working With DAC Metadata Patches".

Status Column

The status of a patch listed in the Working Patches tab can be one of the following:

- **Open.** A patch must have an Open status in order for objects to be added or removed. Open patches cannot be exported.
- **Closed.** When the patch status is changed to Closed, the patch can no longer be edited. You can reopen a Closed patch in order to edit it. Closed patches can be exported.

Patch Version Column

The DAC automatically assigns the patch version value as 1 and the status as Open. When you change the status from Closed to Open, the patch version is automatically incremented by 1.

Repository Created Time Column

Local time of the machine that hosts the DAC repository when a patch is created.

Last Closed Column

Timestamp of when the patch was last closed.

Last Exported Column

Timestamp of when the patch was last exported.

Other Instructions Column

Provides a text box where you can enter information and instructions related to the patch.

Working Patches Tab: Child Patches Subtab

Lists the child patches that have been added to the patch selected in the top pane. For instructions on adding child patches to a patch, see "Creating a DAC Metadata Patch".

Added to Patch Version Column

The patch version of the patch to which the child patch was added.

Date Added Column

Timestamp of when the child patch was added to the patch.

Status Column

Status of the child patch. Possible values are:

- **Completed**. The patch was imported without any errors.
- **Failed**. An error occurred when the patch was imported, and the process failed.
- **Not Applied**. Either the container does not exist or the user does not have privilege to save the object in the container.

Comments Column

Provides a text box where you can enter comments related to the patch.

Working Patches Tab: Contents Subtab

Lists the parent and child objects that have been added to the patch selected in the top pane. For instructions on adding contents to a patch, see Creating a DAC Metadata Patch.

Type Column

Type of object added as contents to the patch.

Container Column

Parent container of the object added to the patch.

Added to Patch Version Column

The patch version of the patch to which the child patch was added.

Date Added Column

Timestamp of when the object was added to the patch.

Comments Column

Provides a text box where you can enter comments related to the patch.

Working Patches Tab: Audit Trails Subtab

Lists details of the history of the selected patch.

Patch Version Column

The patch version of the patch selected in the top pane.

User Column

The user who performed the action.

Client Start Time Column

Timestamp of the client machine when the action started.

Repository Start Time Column

Timestamp of the DAC repository when the action started.

Repository End Time Column

Timestamp of the DAC repository when the action ended.

Action Column

Action that was performed.

Client Host Column

Name of the client host machine.

Client IP Column

IP address of the client machine.

Applied Patches Tab

The Applied Patches tab lists patches that have been applied to the repository.

For more information, see

- Working With DAC Metadata Patches
- Exporting a DAC Metadata Patch
- Applying a DAC Metadata Patch to the DAC Repository
- Exporting and Applying Patches Using the Command Line

Status Column

The status of an applied patch can be one of the following:

- **Completed.** All objects in the patch were successfully applied to the DAC repository, without errors.
- **Incomplete.** The patch application process completed with one or more errors. Some objects may have been applied successfully to the DAC repository.

Patch Version Column

The DAC automatically assigns the patch version value as 1 and the status as Open. When you change the status from Closed to Open, the patch version is automatically incremented by 1.

Repository Created Time Column

Local time of the machine that hosts the DAC repository when a patch is created.

Last Closed Column

Timestamp of when the patch was last closed.

Last Applied Column

Timestamp of when the patch was last applied.

Last Exported Column

Timestamp of when the patch was last exported.

Other Instructions Column

Provides a text box where you can enter information and instructions related to the patch.

Applied Patches Tab: Child Patches Subtab

Lists the child patches of the applied patch that is selected in the top pane.

Added to Patch Version Column

The patch version of the patch to which the child patch was added.

Date Added Column

Timestamp of when the child patch was added to the patch.

Status Column

Status of the child patch. Possible values are:

- **Completed**. The patch was imported without any errors.
- **Failed**. An error occurred when the patch was imported, and the process failed.
- **Not Applied**. Either the container does not exist or the user does not have privilege to save the object in the container.

Comments Column

Provides a text box where you can enter comments related to the patch.

Applied Patches Tab: Contents Subtab

Lists the parent and child objects that make up the contents of the applied patch selected in the top pane.

Type Column

Type of object added as contents to the patch.

Container Column

Parent container of the object added to the patch.

Added to Patch Version Column

The patch version of the patch to which the child patch was added.

Date Added Column

Timestamp of when the object was added to the patch.

Comments Column

Provides a text box where you can enter comments related to the patch.

Applied Patches Tab: Audit Trails Subtab

Lists details of actions related to the selected patch.

Status Column

Status of the action performed. Possible values are:

- Completed
- Incomplete
- Canceled
- Interrupted

Patch Version Column

The patch version of the patch selected in the top pane.

User Column

The user who performed the action.

Client Start Time Column

Timestamp of the client machine when the action started.

Repository Start Time Column
Timestamp of the DAC repository when the action started.

Repository End Time Column
Timestamp of the DAC repository when the action ended.

Action Column
Action that was performed.

Client Host Column
Name of the client host machine.

Client IP Column
IP address of the client machine.

Execute View Tabs

The Execute View provides access to functionality that enables you to run, schedule, and monitor execution plans.

This section describes the Execute view interface tabs.

Current Runs Tab

The Current Runs tab displays a list of queued, running, and failed current ETL processes. This list includes comprehensive information about each process. Once an ETL process completes, it is accessible from the Run History tab.

Execution Plan Name Column

The execution plan whose runtime instance is this record. This field is read only.

Run Status Column

The status of the run. The possible values are the following.

Value	Description
Completed	All tasks have completed without errors.
Failed	Tasks were executed but encountered a problem or were killed or aborted.
Not Executed	Tasks were not able to run before the DAC Server stopped.
Paused	Task group members are waiting for the other tasks in the group to be executed.
Queued	Tasks for which the Depends On tasks are not yet completed.
Requeued	Tasks with a previous Failed status are submitted again. Tasks will start running as soon as an Informatica slot is available.
Runnable	Tasks for which the Depends On tasks have completed and are ready to be run but are waiting for an Informatica slot to be available.
Running	Tasks for which the Depends On tasks have been completed, have gotten an Informatica slot, and are being executed.
Stopped	Tasks for which one or more Depends On tasks have failed.

Start Timestamp Column

Start time of the ETL process. Reflects the start time of every ETL attempt. For example, if the ETL fails and is run again, it gets a new start timestamp. The history of attempted runs is maintained in the audit trail for the run. This field is read only.

End Timestamp Column

End time of the ETL process. Reflects the end time of every ETL attempt. For example, if the ETL fails and is run again, it gets a new start timestamp. The history of attempted runs is maintained in the audit trail for the run. This field is read only.

Duration Column

A calculated field that shows the difference between start and end time stamps.

Status Description Column

Displays messages generated during run time. You can add notes to this field for Completed runs.

Process ID Column

ID for the process. This value is generated by DAC and is calculated based on the timestamp of the DAC repository database. This field is read-only.

Total Number of Tasks Column

The total number of tasks for this run. This field is read only.

Number of Failed Tasks Column

The sum total of tasks that have failed and that have stopped. This field is read only.

Number of Successful Tasks Column

The number of tasks whose status is Completed. This field is read only.

Number of Tasks Still in Queue Column

The number of tasks whose prerequisite tasks have not completed, and the number of tasks whose prerequisite tasks are completed and are waiting for resources. This field is read only.

Schedule Name Column

The name of the scheduled ETL process.

OS Process Name Column

The system dependent process name for tasks that run as separate processes, such as occurs when execution plans run concurrently.

Current Runs Tab: Audit Trail (RO) Subtab

The Audit Trail (RO) subtab is a read-only tab that provides the history of the selected run.

Last Updated Column

The date the selected run was last updated.

Start Timestamp Column

Start time of the selected run.

End Timestamp Column

End time of the selected run.

Duration Column

The difference between the start timestamp and the end timestamp of the selected run.

Status Column

Status of the selected run.

Current Runs Tab: Phase Summary (RO) Subtab

The Summary (RO) subtab provides a summary (based on dynamic SQL) of the selected ETL run.

Task Phase Column

The task phase of the selected ETL run.

Start Time Column

Start time of the selected phase of the run.

End Time Column

End time of the selected phase of the run.

Duration Column

The difference between the start timestamp and the end timestamp of the selected phase of the run.

Source System Column

The source system associated with the ETL process.

Current Runs Tab: Run Type Summary (RO)

The Run Type Summary (RO) subtab is a read-only tab that indicates the number of task details by the execution type.

Current Runs Tab: Tasks Subtab

The Tasks subtab displays runtime instances of the tasks. As the execution proceeds, the tasks are executed based on the dependency rules and some prioritization.

As tasks complete, the tasks that depend on the completed tasks are notified and once their dependencies are completed, they become eligible to run. If a task fails, the administrator can address the failure and then requeue the task or mark it as Completed. The DAC Server polls for any changes in the failed task's detail status. If a failed task detail is queued, the task itself gets back into the ready-to-run queue and all its dependent tasks get into the queued status.

The rules of the prioritization are as follows:

- Tasks with no dependencies are executed first.
- If a task has failed and has been requeued, it gets the maximum priority.
- Tasks with greater phase priorities are executed next. When several tasks of the same phase are eligible to run, the tasks with greater task priorities are executed next. The prioritization is also based on the number of dependent tasks, the number of source tables, and the average time taken by a task.

Depth Column

The level of the task's dependency. Tasks that have no dependencies are depth 0. Tasks that depend on other tasks of depth 0 are depth 1, and so on

Name Column

The task name.

Mode Column

Indicates whether the task will load in full or incremental load.

Group Column

Indicates the name of the task group, if the task belongs to one.

Task Status Column

Possible task statuses are the following:

- **Queued.** Task is waiting for one or more predecessor tasks to complete. Appears as light yellow.

- **Runnable.** Task is waiting on a resource token to be available. All predecessor tasks have completed. Appears as yellow.
- **Waiting.** Task is eligible to run but is waiting because a time delay for the source was defined in the connectivity parameters or because an event delay was defined at the data source level (see "Setting Up Extract Delays, Event Delays and Data Source Notifications"). Appears as white.
- **Running**. Task obtained a resource token and has started running. Appears as blue.
- **Paused.** Task is a member of a task group and has paused until the child tasks of the task group are completed. Appears as blue.
- **Completed.** All task details of the task have executed successfully. Appears as green.
- **Failed.** One or more of the task details of the task have failed. Appears as red.
- **Stopped.** Task has stopped because one or more predecessor tasks failed. Appears as ochre.
- **Not Executed.** Task has a heuristics definition that prevents it from running (see "Using Heuristics to Manage Tasks, Tables and Indexes"). Appears as white.

Start Timestamp Column

Timestamp of when the task started.

End Timestamp Column

Timestamp of when the task ended.

Duration Column

Indicates how long the task ran.

Status Description Column

Description of the status of the task. For example, if the task failed, this field will indicate the cause of the failure.

Source Database Column

Name of the source database.

Target Database Column

Name of the target database.

Folder Name Column

Name of the Informatica folder in which the task resides

Task Phase Column

Task phase of the ETL process. The DAC Server uses the task phase to prioritize tasks and to generate a summary of the time taken for each of the phases

Execution Type Column

Execution type of the task.

Rows Successfully Passed Column

Number of rows that were successfully entered into the data warehouse.

Failed Rows Column

Number of rows that failed.

Source System Column

Source system container from which the task extracts data.

Current Runs Tab: Task Details Subtab

The Task Details subtab opens in query mode. It enables you to query for tasks associated with the selected ETL run in order to view execution details.

Execution Instances Tab

The Execution Instances tab enables you to create multiple instances (or copies) of an execution plan. You can then run the parent execution plan and the additional child instances concurrently. For more information about execution instances, see "Running Multiple Instances of an Execution Plan".

The Execution Instances tab shares the same top pane properties and subtabs as the Execution Plans tab. For a description of these properties and subtabs, see "Execution Plans Tab".

Execution Plans Tab

The Execution Plans tab enables you to view, edit, and create execution plans. For more information, see "Building and Running Execution Plans".

Full Load Always Column

Indicates the specified execution plan will execute a full load regardless of whether the source and target tables have refresh dates.

Micro ETL Column

Indicates an execution plan is a micro ETL execution plan, which is an execution plan that you schedule at very frequent intervals, such as hourly or half-hourly. Refresh dates for micro ETL execution plans are kept separate from the refresh dates for regular execution plans. DAC saves refresh dates for micro ETL execution plans in the Micro ETL Refresh Dates subtab of the Execution Plans tab (Execute view) and refresh dates for regular execution plans in the Refresh Dates subtab of the Physical Data Sources tab (Setup view).

For more information about micro ETL processes, see "About Micro ETL Execution Plans".

Last Designed Column

Indicates the date and time this execution plan was last designed.

Analyze Column

Indicates the tables associated with this execution plan will be analyzed.

Analyze Truncated Tables Only Column

Indicates only truncated tables will be analyzed.

Drop/Create Indices Column

Indicates indexes of the tables associated with this execution plan will be dropped and created.

Run Now Button Column

The Run Now button submits a request to the DAC Server to execute the execution plan.

Build Button

The Build button does the following:

- Collects all the tasks from all the subject areas.
- Substitutes the logical values with the physical values as defined in the Connectivity Parameters subtab (primary source/target connection and folder information).
- Removes any redundancies based on the task name and primary source/target connection information.
- Creates multiple instances of the same task to accommodate the number of copies of the parameters.
- Comes up with the ordered list of tasks with the dependencies computed among them.

For more information about building an execution plan, see "Building and Running Execution Plans".

Execution Plans Tab: All Dependencies Subtab

The All Dependencies subtab opens in query mode. It enables you to query for tasks that have a dependent relationship. The columns in this tab are the same as those in the Immediate Dependencies subtab.

Note: The All Dependencies information is not subject to export and, therefore, not importable elsewhere. However, if you want to compute all dependencies, you can do so by right-clicking and selecting Compute All Dependencies.

Depth Column

The level of the task's dependency. Tasks that have no dependencies are depth 0. Tasks that depend on other tasks of depth 0 are depth 1, and so on.

Source Column

Source table for the task.

Target Column

Target table for the task.

Successor Depth Column

Depth of the successor task.

Successor Name Column

Name of the successor task.

Successor Source Column

Source table of the successor task.

Successor Target Column

Target table of the successor task.

Execution Plans Tab: Concurrent Dependency

You can configure DAC to run multiple execution plans concurrently if the execution plans are independent of one another, that is, as long as the execution plans do not load data into the same table on the same physical data source. To avoid data inconsistency, execution plans that have dependencies with other execution plans cannot run concurrently (unless you override the default behavior).

The Concurrent Dependency subtab enables you explicitly to define execution plans as dependent or independent in relation to other execution plans.

For more information, see "Explicitly Defining Execution Plans as Independent or Dependent".

Execution Plans Tab: Execution Parameters Subtab

The Execution Parameters subtab enables you to configure parameters at the execution plan level.

For information about how to configure parameters, see Chapter 7, "Defining and Managing Parameters."

Data Type Column

Parameter data type. See "Parameter Data Types" for a description of the different parameter data types.

Value Column

User-defined parameter value.

Comments Column

Text field for entering comments.

Execution Plans Tab: Following Tasks Subtab

Enables you to add tasks that will run before the ordered tasks of an execution plan run. Also lists all the user-specified preceding tasks for an execution plan.

Execution Priority Column

Indicates the order among the following tasks in which this task is executed.

Command for Incremental Load Column

Command associated with the task.

Source System Column

Source system container from which the task extracts data.

Execution Plans Tab: Immediate Dependencies Subtab

The Immediate Dependencies subtab opens in query mode. It enables you to query for tasks that have an immediate dependent relationship between tasks that are generated during the automatic task generation process.

Depth Column

The level of the task's dependency. Tasks that have no dependencies are depth 0. Tasks that depend on other tasks of depth 0 are depth 1, and so on.

Task Name Column

Name of immediate dependent task.

Source Column

Source table from which the task extracts data.

Target Column

Target table into which the task loads data.

Successor Depth Column

Depth of the successor task.

Successor Name Column

Name of the successor task.

Successor Source Column

Source table of the successor task.

Successor Target Column

Target table of the successor task.

Execution Plans Tab: Ordered Tasks Subtab

The Ordered Tasks subtab lists tasks associated with the selected execution plan and the order in which they can be executed.

Depth Column

The level of the task's dependency. Tasks that have no dependencies are depth 0. Tasks that depend on other tasks of depth 0 are depth 1, and so on.

Group Column

Indicates the name of the task group, if the task belongs to one.

Primary Source Column

Primary source table from which the task extracts data.

Primary Target Column

Primary target table into which data is loaded.

Folder Name Column

Name of the Informatica folder in which the task resides.

Task Phase Column

Task phase of the ETL process. The DAC Server uses the task phase to prioritize tasks and to generate a summary of the time taken for each of the phases.

Source System Column

Source system container from which the task extracts data.

Details Button on Subtab Toolbar

The Details command displays detailed information about the task, including predecessor and successor tasks, and source, target, conditional, and refresh date tables.

Unit Test Button on Subtab Toolbar

The Unit Test command provides a summary of the details of what gets executed when the particular task runs as part of the execution plan.

It includes the following:

- Pre-ETL actions
- Upon Failure Restart actions
- Truncate table information
- List of indexes to be dropped and created
- Full or incremental command based on the current situation
- Tables to be analyzed
- Upon Success action
- Upon Failure action

Execution Plans Tab: Connectivity Parameters Subtab

The Connectivity Parameters subtab lists the parameters of the selected execution plan for database connections and Informatica folders.

Copy Number Column

Number of the data source.

Type Column

Possible values are the following:

- **Folder**

  Indicates an Informatica folder.

- **Datasource**

  Indicates a database connection parameter.

Name Column

Logical name of the folder or database connection.

Value Column

Physical name of the folder or database connection.

Delay Column

Indicates how many minutes an extract of a data source will be delayed after the first extract of a multiple source extract process has started. For more information about multi-source execution plans, see "Setting Up Extract Delays".

Prune Time Column

The Prune Time setting subtracts the number of prune minutes from the LAST_REFRESH_DATE and supplies this value as the value for the $$LAST_EXTRACT_DATE parameter. This property must be set based on experience with processes, such as remote sync, that potentially can cause records to be missed.

Note: Setting the Prune Time period too high can impact performance.

Source System Column

Name of the source system associated with the parameter.

Execution Plans Tab: Preceding Tasks Subtab

Enables you to add tasks that will run before the ordered tasks of an execution plan run. Also lists all the user-specified preceding tasks for an execution plan. For instructions on adding preceding tasks to an execution plan, see "Building and Running Execution Plans".

Name Column

Name of task.

Execution Priority Column

Indicates the order in which the task is executed. If two or more tasks have the same priority, DAC will execute them in parallel.

Command for Incremental Load Column

Command for incremental load that is associated with the task.

Source System Column

Source system container from which the task extracts data.

Execution Plans Tab: Micro ETL Refresh Dates Subtab

Displays refresh dates for micro ETL execution plans (indicated by selecting the Micro ETL check box in the Execution Plans tab). For more information about micro ETL execution plans, see "About Micro ETL Execution Plans".

Connection Column

Logical name for the database connection.

Refresh Date Column

Last refresh time of the execution plan. This applies only when separate refresh dates are maintained. Used for micro ETL processing.

Execution Plans Tab: Subject Areas Subtab

The Subject Areas subtab lists the subject areas associated with the selected execution plan. You can also add subject areas to or remove subject areas from the selected execution plan.

For instructions on adding and removing subject areas, see:

- Running an Execution Plan
- Building and Running Execution Plans

Subject Area Column

Name of the subject area associated with the execution plan.

Source System Column

The source system container associated with the subject area.

Execution Plans Tab: Tables (RO) Subtab

The Tables (RO) subtab is a read-only tab that lists all the tables that are touched by the selected execution plan.

Type Column

Table type.

Connection Column

Database connection.

Table Column

Table name.

Run History Tab

The Run History tab displays information about completed ETL processes. The information displayed in the top and bottom windows is the same as that in the Current Runs tab.

For a description of the information in the Run History tab, see "Current Runs Tab".

Run History Tab: Audit Trail (RO) Subtab

See "Current Runs Tab: Audit Trail (RO) Subtab".

Run History Tab: Phase Summary (RO) Subtab

See "Current Runs Tab: Phase Summary (RO) Subtab".

Run History Tab: Run Type Summary (RO) Subtab

See "Current Runs Tab: Run Type Summary (RO)".

Run History Tab: Tasks Subtab

See "Current Runs Tab: Tasks Subtab".

Run History Tab: Task Details

The Task Details subtab opens in query mode. It enables you to query for tasks associated with the selected ETL run in order to view execution details.

Scheduler Tab

The Scheduler tab enables you to schedule ETL processes to be executed either once or periodically. When you schedule an ETL or make changes to a schedule, the DAC Server picks up the information from the DAC Client. The DAC Server polls the DAC repository for changes periodically at a frequency set in the DAC System Properties tab.

The top pane of the Scheduler tab lists ETL runs that have been scheduled. The bottom window enables you to schedule an ETL run.

For instructions on scheduling ETL processes, see "Scheduling an Execution Plan".

Execution Plan Column

The name of the scheduled execution plan.

Last Schedule Status Column

The last run status of the scheduled ETL process. Possible values are Running, Completed or Stopped.

Next Trigger Column

Time the scheduled ETL run will next be executed.

Status Description Column

Description of the last ETL run. Possible values are Running, Completed, or the reason the process stopped.

Recurrence Column

Indicates how often the schedule will be executed.

A

Troubleshooting DAC

This appendix describes common problems and how to resolve them.

This appendix contains the following topics:

Recovering From a Lost Encryption Key

The process of recovering from the loss of an encryption key involves clearing the encrypted data from the DAC repository, creating a new cwallet.sso file (with a new encryption key), and stamping the DAC repository with the new encryption key.

This procedure involves steps that use automation utility commands. Before you can use these commands, you need to configure the automation utility property file.

To recover from the loss of an encryption key

1. Remove all the encrypted data from the DAC repository by calling the automation utility command clearEncryptedData.
2. Call one of the following automation utility commands to generate a new cwallet.sso file:
   - dbCredentials <cwallet.sso file path> <user name> -withKey

Use this command to generate a cwallet.sso file with a encryption key that you specify.

- dbCredentials <cwallet.sso file path> <user name> -randomKey

  Use this command to generate a cwallet.sso file with a randomly generated encryption key.

3. Copy the new cwallet.sso file to the appropriate directory where the DAC repository can access it. The default directory is <DAC_Config_Location>\conf-shared\security\repository.
4. Stamp the DAC repository with the new encryption key:
   a. Log in to the DAC Client.

      The following message will be displayed:

      Encryption key validation value is missing from the DAC repository. Would like to add it now?

   b. Click **OK** to add the new encryption key to the DAC repository.
   c. Enter the table owner name and password, and click **OK**.

      A message informs you whether the operation was successful.

   d. Distribute the new cwallet.sso file to all DAC installations that connect to this repository.

Restarting an Execution Plan When the DAC Server Fails

If the DAC Server crashes while an execution plan is running, some workflows associated with the execution plan may have been submitted to Informatica. When the DAC Server is restarted and the failed ETL process resumes, the workflows that DAC submitted to Informatica before the crash may have completed or failed or may still be running. You do not have to perform any manual intervention. DAC manages the tasks submitted for execution prior to the crash in the following manner:

- **Running. If a workflow is still running in Informatica, DAC waits for the running workflow to finish execution**.
  - If the workflow has successfully completed, DAC will change the status of the task detail to Completed and will resume execution of the successors.
  - If the workflow has failed, DAC will change the status of the task detail to Failed and will eventually halt the ETL. This is analogous to a normal failure encountered during ETL execution.
- **Completed.** If the workflow had completed in Informatica, the task will not be re-run. DAC will change the status of the task detail to Completed and will resume execution of the successors.
- **Failed.** If the workflow in Informatica had failed, DAC will resubmit the workflow for execution when the ETL process restarts.

  Note: If a failed task includes truncate statements, the truncate action will occur again when the task is restarted. This will also trigger the Upon Failure Restart action if one is defined for the task.

Discarding a Failed Execution Plan

You can discard an execution plan that failed by navigating to the Current Run tab, right-clicking on the execution plan and changing its status to Mark as Completed. This will force the run status to be updated as Completed. When you submit a request for another run, the DAC Server creates another instance of it.

Caution: Perform this procedure in a development or test environment only, because it might leave the data in an inconsistent state, causing you to have to reload all of the data.

Failure of Aggregator Transformation Tasks with Sorted Input

Tasks that use Informatica Aggregator Transformation can fail when the Sorted Input option is active. The tasks SDE_DTLFORECASTFACT and SDE_COSTLIST are examples of tasks that can fail in such a situation.

To prevent such tasks from failing, in Informatica PowerCenter Designer, navigate to Mapping Designer, open the corresponding mapping, and in the Aggregator transformation, deselect the Sorted Input check box.

In Case of Abnormal Termination of the DAC Server

If the DAC Server fails during the execution of an execution plan, the status of the execution plan will remain as Running. When the DAC Server is started again, it will automatically run the execution plan if the Auto Restart ETL system property is set to True. If the same system property is set to False, when the server restarts, it will set the correct status as Failed. In order to execute the execution plan from the point of failure, submit the request to the server again.

The DAC Server will automatically terminate if it looses connection to the DAC repository.

DAC Task Failing on Non-English Operating System

DAC Server uses pmcmd to initiate the workflows on Informatica Server. In the English-based operating systems, DAC issues the commands in the non-blocking mode (asynchronously), and polls Informatica for the status of the workflow. The output of the pmcmd getWorkFlowDetails is spooled to the Domain_Home\dac\log directory, and then gets parsed to determine whether the workflow is still running, completed successfully, or failed.

However, for non-English-based operating systems, DAC issues commands in the waiting mode (synchronously). This means that when the process completes the exit code tells DAC whether or not the workflow succeeded.

The commands used by DAC to communicate with Informatica are externalized in a file called infa_commands.xml. The DAC command template does not have a place holder for specifying the wait mode. Without this wait mode configuration, on a non-English- based operating system, DAC proceeds with the execution even before the workflow completes executing. This might result in errors, such as Informatica's bulk loader failing because of indexes are present or fact tables are being loaded without foreign key references.

To fix the problem, go to Domain_Home\dac\conf folder and edit the file called infa_commands.xml. Depending upon the version of informatica you are using, edit either

the block called START_WORKFLOW_7 or START_WORKFLOW_8 and verify whether %WAITMODE is in the syntax. If it is not, add it as follows:

1. For START_WORKFLOW_7 replace the following line:

   ```
   pmcmd startworkflow -u %USER -p %PASSWORD  -s %SERVER:%PORT -f %FOLDER -lpf
   %PARAMFILE %WORKFLOW
   ```

   With:

   ```
   pmcmd startworkflow -u %USER -p %PASSWORD  %WAITMODE -s %SERVER:%PORT -f
   %FOLDER -lpf %PARAMFILE %WORKFLOW
   ```

2. For START_WORKFLOW_8 replace the following line:

   ```
   pmcmd startworkflow -sv %SERVER -d %DOMAIN -u %USER -p %PASSWORD -f %FOLDER
   -lpf %PARAMFILE %WORKFLOW
   ```

   With:

   ```
   pmcmd startworkflow -sv %SERVER -d %DOMAIN -u %USER -p %PASSWORD %WAITMODE -f
   %FOLDER -lpf %PARAMFILE %WORKFLOW
   ```

3. Once you modify this file (the modifications should be done both on the DAC client and the server machines), restart the DAC server and client for the changes to take effect.

DAC Task Failing in a Multi-Source Environment

When you have a multi-source environment, you can use a single execution plan to extract data for a given subject area for all the sources. When you build the execution plan, DAC creates as many instances of the extract tasks as there are sources, without having the need to duplicate the workflows or DAC metadata.

At runtime, DAC creates an individual parameter file for each session, which contains the evaluated name-value pairs for all parameters. DAC stores the parameter file at the location specified by the DAC system property InformaticaParameterFileLocation. The default location is <Domain_Home>\dac\Informatica\parameters.

If a task in a multi-source environment fails and you want to restart it, you can do so from DAC or from Informatica. See the following instructions.

To restart a failed task in a multi-source environment using DAC

1. Go to the Execution Plans tab in the Execute view, and select the appropriate execution plan.
2. Click the **Ordered Tasks** subtab.
3. Query for the task that you want to restart.
4. Click **Unit Test** in the toolbar.

   The Unit Test dialog displays the steps that DAC will carry out if the task is executed.

5. Click **Execute** in the Unit Test dialog.

   Note: The DAC system property Dryrun must be set to False for the task to run.

To restart a failed task in a multi-source environment using Informatica

1. In DAC, locate the parameter file for the failed task.

The parameter file is stored in the location specified by the DAC system property InformaticaParameterFileLocation. The default location is <Domain_Home>\dac\Informatica\parameters.

2. Copy the file to the location specified by the Informatica parameter $PMSourceFileDir. This is the location where Informatica looks for parameter files.

   Note: If DAC is configured correctly, the location specified by $PMSourceFileDir and the location specified by the DAC system property InformaticaParameterFileLocation should be the same. If the locations are the same, the parameter file will already be in the correct location.

3. Change the parameter file name to match the naming convention Informatica uses for parameter files:

   ```
   <Informatica directory name>.<Informatica session name>.txt
   ```

4. Run the workflow in Informatica.

Restoring the DAC Repository on Unicode Oracle Databases

When the DAC repository resides on a non-Unicode Oracle database, the Informatica workflow names may not fit into the corresponding DAC fields, which causes tasks to fail during ETL processes.

To work properly in multi-byte character environments, the DAC repository should be created with the Unicode option selected.

Perform the following procedure to fix an existing DAC repository with this problem. You can use the DAC Client or DAC command line parameters to perform the procedure.

1. Connect to the existing DAC repository.
2. Export the entire repository (logical, system, and run time categories).
3. Stop the DAC Server and all DAC Clients except for the one you are using.
4. Drop the current repository.
5. If you are using a DAC Client, restart it.
6. Create a new repository with the Unicode option selected.
7. Import the DAC metadata that you exported in step 2.
8. Re-enter the passwords for all Informatica services and all physical data source connections.

Handling Parameter Files with Multi-Line Parameters

Informatica workflows initiated by DAC fail with error code 17 and the error message "Parameter file does not exist" when the parameter file has multi-line parameters. See the below text for an example. Note that in the following example, an error occurs because DAC issues `pmcmd` with -lpf in the syntax.

```
$$SYND_DS_PARTITION_TRUNCATE_SQL_TEXT=SELECT
LTRIM(MAX(SYS_CONNECT_BY_PATH('execute immediate ''ALTER TABLE
getTableName() TRUNCATE PARTITION ' || '"' ||COUNTRY_REGION_NAME|| '"' ||''''

,';')) KEEP (DENSE_RANK LAST ORDER BY curr),';') AS SQL_TXTFROM
(SELECT SOURCE,COUNTRY_REGION_NAME,ROW_NUMBER() OVER (PARTITION BY SOURCE
```

```
 ORDER BY COUNTRY_REGION_NAME) AS curr, ROW_NUMBER() OVER (PARTITION BY SOURCE

ORDER BY COUNTRY_REGION_NAME) -1 AS prev FROM W_SYND_PARTITION_TMP

WHERE SOURCE='W_SYNDD_DS_FS')
CONNECT BY prev = PRIOR curr START WITH curr = 1
```

To prevent this issue, edit <DAC_Config_Location>\conf-shared\infa_command.xml and replace all instances of <-lpf> with <-paramfile>.

This workaround will ensure that DAC uses -paramfile in the `pmcmd` syntax and Informatica can recognize the multi-line parameters.

Resetting Refresh Dates at the Task Level

In DAC, you can reset refresh dates at the task level rather than the table or subject area level by using one of the following options:

- Create an execution plan with the appropriate subject areas and select the Full Load Always property. Run the execution plan once; this will cause the tasks in the execution plan to run in full mode. After the execution plan successfully completes, delete the execution plan.
- Create an execution plan with the appropriate subject areas. Right-click the execution plan and select **Reset source(s).** Select all the sources, and then click **OK**.

  Note: With this option, given that the execution plan will run in full mode, there is the risk that source tables populated by this execution plan could also be included in other subject areas, and in such instances, could be populated in error. To work around this issue, you can set the DAC system property "No Run" to True, which will generate tasks but not execute them. You can review the Task Details in the Current Runs tab to verify if the correct tasks are running in full mode.

Error When Using Oracle(OCI8) Connection Type

If you are using the Oracle (OCI8) connection type when defining a physical data source and receive an error such as "no ocijdbc11," consider the following points:

- The JDBC calls go through the local database client installation libraries. This means that the client version and the database version should match. For example, you cannot use an Oracle 9i client to access an Oracle 10g database.
- Do not use multiple versions of the same library.
- Consider using the Oracle (Thin) connection, which has no overhead of going through the client libraries.

Tables Are Not Truncated When More Than One Task Writes to the Same Target Table

The DAC behavior for truncating tables when more than one DAC task writes to the same target table is as follows:

- The first DAC task truncates the target table. DAC does not truncate the target tables during the execution of the subsequent tasks even if the Truncate Always or Truncate for Full Load flags are checked on the subsequent tasks.
- DAC truncates a target table only once during the life span of one ETL execution.

- If the DAC task belongs to a group and the group truncate property is different from the individual task truncate properties, the target table will be not truncated.

If you want to override this default behavior, you can create a task action and use the Preceding Action action type to specify in a custom SQL statement that the table should be truncated. For more information about configuring task actions, see "Using Actions to Optimize Indexes and Collect Statistics on Tables".

Discrepancy Between DAC Task and Informatica Workflow

Discrepancies may exist between the status of a DAC task and the status of an Informatica workflow for the following reasons:

- **The Informatica pmcmd utility stops responding or "hangs."** DAC issues a command using the pmcmd utility to start the workflow. DAC waits on this process to finish. If the command "hangs," then DAC will also "hang." DAC issues this command in a polling mode, which means first DAC issues the command to start the workflow and then issues the getWorkflowStatus command to find out if the execution is completed. As long as the pmcmd output states the process is executing, DAC waits. The default polling interval is 30 seconds. Therefore, if a process runs for more than several minutes, you can assume it is "hanging."

  In Windows, you can open the Task Manager and observe the pmcmd executables. If an executable has the same process ID for more than a few minutes, then the process is "hanging."

- **Informatica loses connection to its own repository.** If Informatica loses the connection to the database, even if Informatica Workflow Monitor indicates the workflow ran to completion, Informatica will not be able to update the workflow status to completed. Therefore, when DAC issues a command to determine whether the workflow completed, DAC will always receive a response of "Running." To confirm if the workflow is in a perpetual "Running" status, review the log file with the naming convention folder.workflow.log in the DAC\log folder.

Making the DAC Server Visible to All Clients

If the DAC Server is not visible to all DAC Clients, make sure the DAC system property "DAC Server Host" has the correct value and not "localhost." Depending on your environment, you may need to use a fully qualified name. You can also try using the IP address of the DAC Server machine. You should be able to ping the DAC Server machine from the DAC Client machine. Make sure the ping requests do not time out.

You should also inspect the network settings to make sure the port number that is registered for the DAC Server Port system property is open and available.

Index

A

B

C

D

E

F

H

I

M

O

P

Q

R

S

T

U

W