Oracle® Business Intelligence Applications

Data Warehouse Administration Console Guide

Version 7.9.4

E10759-01

December 2007

ORACLE®

Oracle Business Intelligence Applications Data Warehouse Administration Console Guide Version 7.9.4

E10759-01

Contents

5 Customizing, Designing, Executing and Monitoring ETL Processes

6 Common Tasks Performed in the DAC

7 DAC Functional Reference

Index

Preface

Oracle Business Intelligence Applications consists of components that were formerly available from Siebel Systems as Siebel Business Analytics Applications (both CRM and Enterprise) with a number of significant enhancements.

The *Oracle Business Intelligence Applications Data Warehouse Administration Console Guide* contains information about using the Data Warehouse Administration Console (DAC), a centralized console for schema management as well as configuration, administration, loading, and monitoring of the Oracle Business Analytics Warehouse.

Oracle recommends reading the *Oracle Business Intelligence Applications Release Notes* before installing, using, or upgrading Oracle Business Intelligence Applications. The *Oracle Business Intelligence Applications Release Notes* are available:

- On the Oracle Business Intelligence Applications CD-ROM.
- On the Oracle Technology Network at http://www.oracle.com/technology/documentation/bi_apps.html (to register for a free account on the Oracle Technology Network, go to http://www.oracle.com/technology/about/index.html).

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at

http://www.oracle.com/accessibility

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, seven days a week. For TTY support, call 800.446.2398.

Related Documents

For more information, see the following documents in the Oracle Business Intelligence Applications Release 7.9.4 documentation set (available at `http://www.oracle.com/technology/documentation/bi_apps.html`):

- *Oracle Business Intelligence Applications Release Notes*
- *System Requirements and Supported Platforms for Oracle Business Intelligence Applications*
- *Oracle Business Intelligence Applications Installation and Configuration Guide*
- *Oracle Business Intelligence Infrastructure Installation and Configuration Guide*
- *Oracle Business Intelligence Applications Upgrade Guide*

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.

1

What's New in This Release

Oracle Business Intelligence Applications consists of components that were formerly available from Siebel Systems as Siebel Business Analytics Applications (both CRM and Enterprise) with a number of significant enhancements.

The *Oracle Business Intelligence Applications Data Warehouse Administration Console Guide* contains information about using the Data Warehouse Administration Console (DAC), a centralized console for schema management as well as configuration, administration, loading, and monitoring of the Oracle Business Analytics Warehouse.

Oracle recommends reading the Oracle Business Intelligence Applications Release Notes before installing, using, or upgrading Oracle Business Intelligence Applications. The *Oracle Business Intelligence Applications Release Notes* are available:

- On the Oracle Business Intelligence Applications CD-ROM.
- On the Oracle Technology Network at `http://www.oracle.com/technology/documentation/bi_apps.html` (to register for a free account on the Oracle Technology Network, go to `http://www.oracle.com/technology/about/index.html`).

What's New in Oracle Business Intelligence Data Warehouse Administration Console Guide, Version 7.9.4

This section lists changes described in this version of the documentation to support Release 7.9.4 of the software.

- The topic "About Parameter Management" on page 5-8 was added to describe the functionality available in the Parameter Management feature.

  In this release, the Parameter Management feature includes the following enhancements:

  - Ability to use database-specific text as a data type
  - Ability to nest parameters within other parameters
  - Ability to use SQL within timestamp parameters
- The new Parallel Index Creation feature enables you to create indexes in parallel. This feature is described in the sections "Parallel Index Creation" on page 7-30 and "Parallel Table Indexes" on page 7-30.
- The task "Specifying Tablespaces for Indexes by Table Type" on page 5-13 was added to describe this new feature.

- The Add Refresh Dates command was added to allow you to prepopulate table names in the Refresh Dates subtab of the Physical Data Sources tab. For information, see "Execute View Right-Click Menu Commands" on page 4-8.
- The Query mode feature was added to the following subtabs to enable you to query for various database objects.
  - "Current Run Tab: Task Details Subtab" on page 7-35
  - Task Details subtab on the "Run History Tab" on page 7-40
  - "Execution Plans Tab: All Dependencies Subtab" on page 7-37
  - "Execution Plans Tab: Immediate Dependencies Subtab" on page 7-37
  - "Subject Areas Tab: Task Source Tables (RO) Subtab" on page 7-11
  - "Subject Areas Tab: Task Target Tables (RO) Subtab" on page 7-11
- The task "Pointing Multiple Informatica Servers to a Single Informatica Repository" on page 6-22 was added.
- The section "Using SQL Files as an Execution Type in the DAC" on page 6-12 was enhanced to provide more detailed information about XML formatted files.

2

Overview of Oracle Business Analytics Warehouse

This chapter provides an overview of the Oracle Business Analytics Warehouse and the Data Warehouse Administration Console (DAC). It includes the following topics:

Oracle Business Analytics Warehouse Overview

The Oracle Business Analytics Warehouse is a unified data repository for all customer-centric data. The purpose of the Oracle Business Analytics Warehouse is to support the analytical requirements of Oracle Business Intelligence Applications.

The Oracle Business Analytics Warehouse includes the following:

- A data integration engine that combines data from multiple source systems to build a data warehouse.
- An open architecture to allow organizations to use third-party analytical tools in conjunction with the Oracle Business Analytics Warehouse using the Oracle Business Intelligence Server.
- Prebuilt data extractors to incorporate data from external applications into the Oracle Business Analytics Warehouse.
- A set of ETL (extract-transform-load) processes that takes data from multiple source systems and creates the Oracle Business Analytics Warehouse tables.
- The DAC, a centralized console for schema management as well as configuration, administration, loading, and monitoring of the Oracle Business Analytics Warehouse.

Oracle Business Analytics Warehouse Architecture

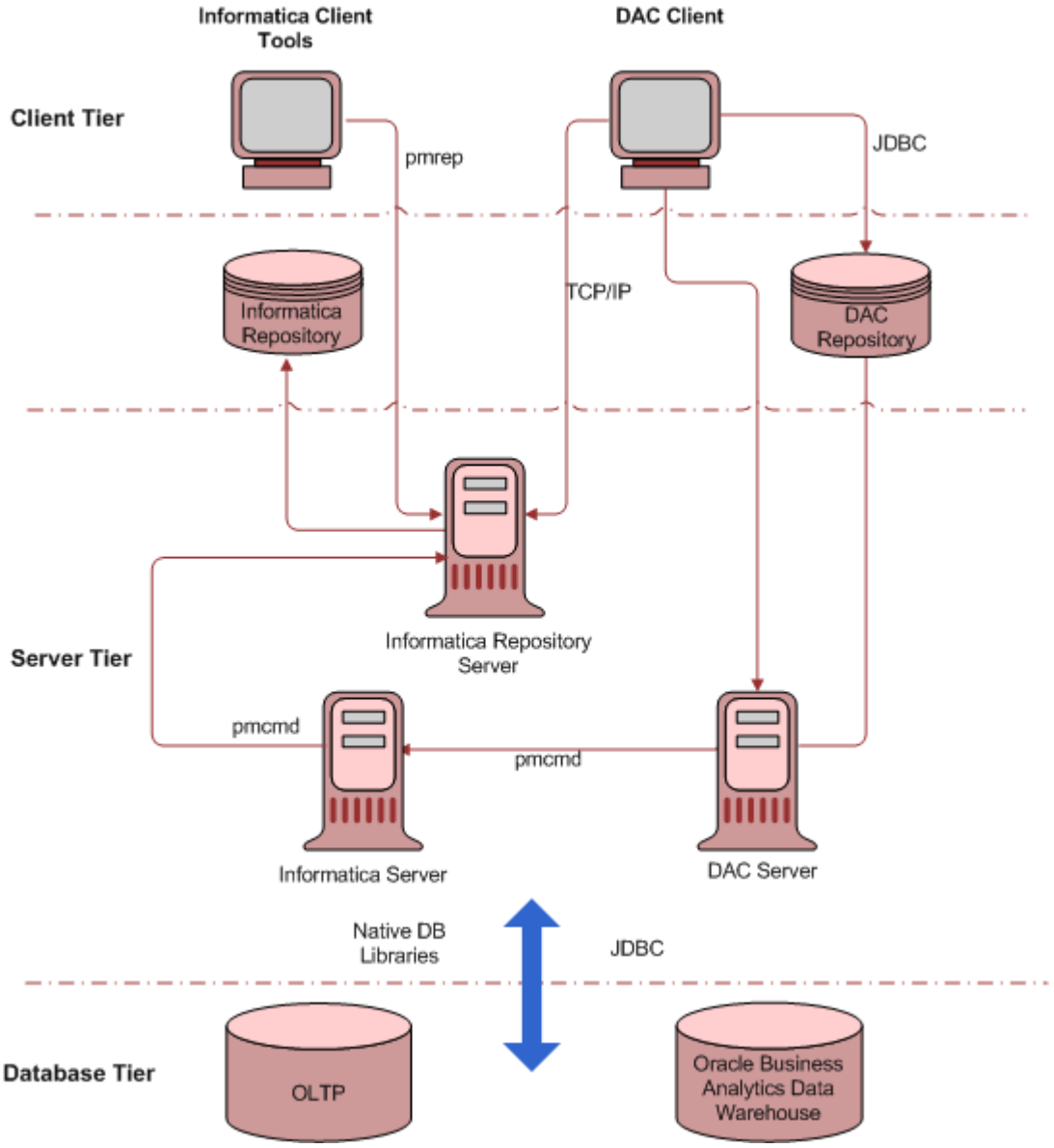
High-level analytical queries, like those commonly used in Oracle Business Analytics Warehouse, scan and analyze large volumes of data using complex formulas. This process can take a long time when querying a transactional database, which impacts overall system performance.

For this reason, the Oracle Business Analytics Warehouse was constructed using dimensional modeling techniques to allow for fast access to information required for

decision making. The Oracle Business Analytics Warehouse derives its data from operational applications, and uses Informatica's data integration technology to extract, transform, and load data from transactional databases into the Oracle Business Analytics Warehouse.

Figure 2–1 illustrates how the Oracle Business Analytics Warehouse interacts with the other components of Oracle BI Applications.

Figure 2–1 Oracle Business Intelligence Applications Architecture



Oracle Business Analytics Warehouse Architecture Components

The Oracle Business Analytics Warehouse architecture comprises the following components:

- **DAC client.** A command and control interface for the data warehouse to allow for schema management, and configuration, administration, and monitoring of data

warehouse processes. It also enables you to design subject areas and build execution plans.

- **DAC server.** Executes the instructions from the DAC client. The DAC server manages data warehouse processes, including loading of the ETL and scheduling execution plans. 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.** Stores the metadata (semantics of the Oracle Business Analytics Warehouse) that represents the data warehouse processes.
- **Informatica Server.** Loads and refreshes the Oracle Business Analytics Warehouse.
- **Informatica Repository Server.** Manages the Informatica repository.
- **Informatica Repository.** Stores the metadata related to Informatica workflows.
- **Informatica client utilities.** Tools that enable you to create and manage the Informatica repository.

About the Data Warehouse Administration Console

The DAC provides a framework for the entire life cycle of data warehouse implementations. It enables you to create, configure, execute, and monitor modular data warehouse applications in a parallel, high-performing environment. For information about the DAC process life cycle, see "About the DAC Process Life Cycle".

The DAC complements the Informatica ETL platform. 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. The DAC provides the following application capabilities at a layer of abstraction above the ETL execution platform:

- Dynamic generation of subject areas and execution plans
- Dynamic settings for parallelism and load balancing
- Intelligent task queue engine based on user- defined and computed scores
- Automatic full and incremental mode aware
- Index management for ETL and query performance
- Embedded high performance Siebel OLTP change capture techniques
- Ability to restart at any point of failure
- Phase-based analysis tools for isolating ETL bottlenecks

Important DAC Features

Important DAC features enable you to do the following:

Minimize installation, setup, and configuration time

- Create a physical data model in the data warehouse
- Set language, currency, and other settings
- Design subject areas and build execution plans

Manage metadata driven dependencies and relationships

- Generate custom ETL execution plans
- Automate change capture for the Siebel transactional database
- Capture deleted records
- Assist in index management
- Perform dry runs and test runs of execution plans

Provide reporting and monitoring to isolate bottlenecks

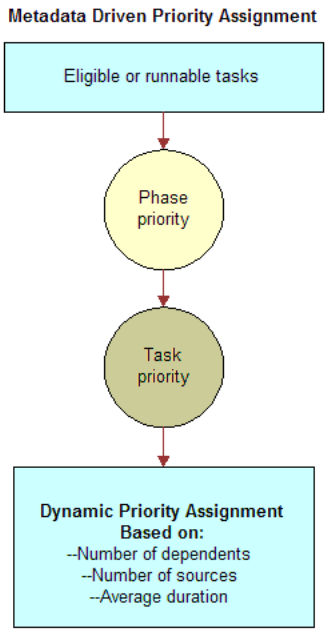
- Perform error monitoring and email alerting
- Perform structured ETL analysis and reporting

Utilize performance execution techniques

- Automate full and incremental mode optimization rules
- Set the level of Informatica session concurrency
- Load balance across multiple Informatica servers
- Restart from point of failure
- Queue execution tasks for performance (See Figure 2–2.)

  The 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.

Figure 2–2 Task Execution Queue



About Source System Containers

Source system containers hold repository objects that correspond to a specific source system. For information about the different kinds of repository objects, see "About DAC Repository Objects".

You can use the preconfigured source system containers to create your own source system container. You cannot modify objects in the preconfigured source system containers. You must make a copy of a preconfigured 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. For more information about source system containers, see "About Source System Containers" and "About Object Ownership in the DAC".

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

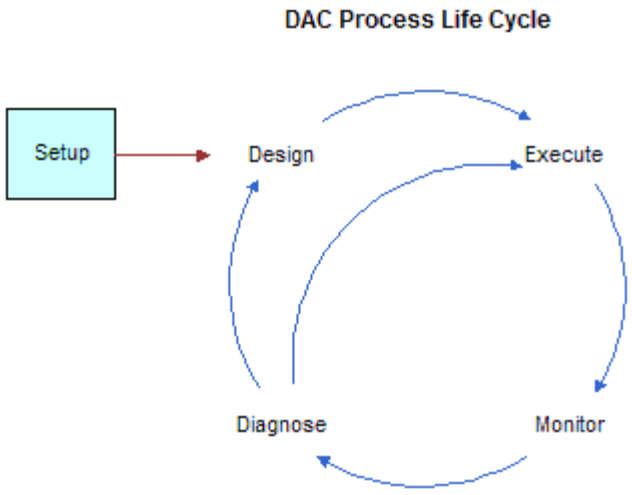
A data warehouse application comprises the following repository objects:

- **Subject area.** A logical grouping of tables related to a particular subject or application context, as well as the tasks that are associated with the tables. Subject areas are assigned to execution plans, which can be scheduled for full or incremental loads. A subject area also includes the tasks required to load the subject area tables.
- **Tables.** Physical database tables defined in the database schema. Can be transactional database tables or data warehouse tables. Table types can be fact, dimension, hierarchy, aggregate, and so on, as well as flat files that can be sources or targets.
- **Task.** A unit of work for loading one or more tables. A task comprises the following: source and target tables, phase, execution type, truncate properties, and commands for full or incremental loads. When you assemble a subject area, the DAC automatically assigns tasks to it. Tasks that are automatically assigned to the subject area by the DAC are indicated by the Autogenerated flag in the Tasks subtab of the Subject Areas tab.
- **Task Groups.** 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."
- **Execution plan.** A data transformation plan defined on subject areas that needs to be transformed at certain frequencies of time. An execution plan is defined based on business requirements for when the data warehouse needs to be loaded. An execution plan comprises the following: ordered tasks, indexes, tags, parameters, source system folders, and phases.
- **Schedule.** A schedule specifies when and how often an execution plan runs. An execution plan can be scheduled for different frequencies or recurrences by defining multiple schedules.

About the DAC Process Life Cycle

The 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 2–3.

Figure 2–3 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 (Informatica)
  - 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

3

DAC Quick Start

In order to start the DAC client, you must have completed the high-level steps listed below:

1. Install Oracle BI Infrastructure.
2. Install Oracle BI Applications.
3. Install Java SDK 1.5.x.
4. Copy Hibernate libraries to the DAC directory.
5. Install JDBC drivers.
6. Log into the DAC (create a connection).
7. Create a DAC repository.
8. Import DAC metadata (specific to one or more source system containers).

For instructions on installing the Oracle BI Infrastructure, see the *Oracle Business Intelligence Infrastructure Installation and Configuration Guide*.

For instructions on completing the remaining steps, see the *Oracle Business Intelligence Applications Installation and Configuration Guide*

4

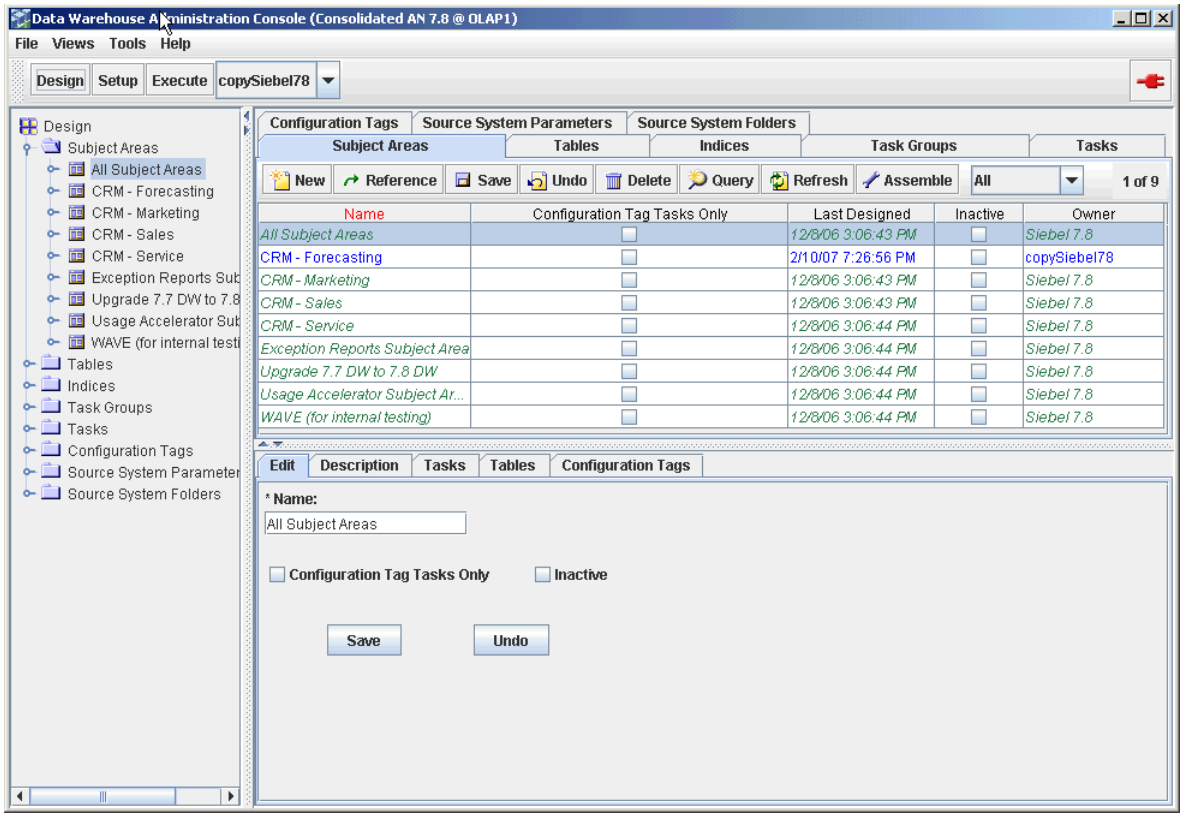
Overview of the DAC Interface

This chapter contains the following topics:

Navigating the DAC Interface

Figure 4–1 shows the main elements of the DAC window.

Figure 4–1 Main DAC Window



The DAC Menu Bar

Table 4–1 provides a description of the DAC menu bar options.

Table 4–1 DAC Menu Bar Options

Menu Names	Description
File	The File menu contains options to close the DAC client and to create, copy, or delete source system containers. For instructions on creating or copying a source system container, see "Creating or Copying a Source System Container".
Views	The Views menu allows you to navigate to the various tabs in the top pane window.
Tools	The Tools menu provides access to functionality related to the DAC and Informatica repositories. Table 4–2 provides a description of the Tools menu commands.
Help	The Help menu provides details about the current DAC login as well as the version of DAC metadata and software. There is no online help for the DAC.

Tools Menu Options

Table 4–2 provides a description of the Tools menu commands.

Table 4–2 DAC Tools Menu Commands

Tools Menu Command	Description
DAC Repository Management > Export	Allows 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 database connection information. ■ **Run Time.** Exports all information contained in the Execute view. ■ **System.** Exports all information contained in the Setup view, except passwords for servers and database connections.
DAC Repository Management > Import	Allows 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 batch mode.** Indicates whether batch mode is enabled. In Batch mode the imported metadata is inserted into the repository as an array insert. In the Import dialog, you can select the following category options: ■ **Logical.** Imports all information contained in the Design view and database connection information. ■ **Run Time.** Imports all information contained in the Execute view. ■ **System.** Imports all information contained in the Setup view, except passwords for servers and database connections.
DAC Repository Management > Create Repository Report	Allows 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.

Table 4–2 (Cont.) DAC Tools Menu Commands

Tools Menu Command	Description
DAC Repository Management > Purge Run Details	Allows you to purge completed runs from the run history. You can purge all runs (except the last run) or specify particular runs to be purged. The last run cannot be purged. In the Purging Runs... dialog, the following options are available: ■ **All.** Purges all completed runs except for the last run. ■ **By Execution Plan.** Allows you to select an execution plan whose associated runs you want to purge. ■ **By Run Name.** Allows you to select an individual run for purging. ■ **Before Specified Date.** Allows you to select a date before which all runs except the last run will be purged. ■ **Details Only.** Purges all related information about a run but leaves the run header information.
DAC Repository Management > Analyze Repository Tables	Allows you to run analyze table commands for all the DAC repository tables.
DAC Repository Management > Default Index Properties	Allows you to specify which databases will be associated with newly created indexes.
DAC Repository Management > Drop DAC Repository	Allows you to drop all the DAC repository tables. This action deletes all data in the repository.
DAC Server Management > Get Server Log	When the DAC server is running an ETL process, this command opens a text box that displays streaming data related to the process.
DAC Server Management > DAC Server Setup	Allows 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.
ETL Management > Configure	Opens the Data Warehouse Configuration wizard, which allows you to create and drop data warehouse tables and to create delete triggers.
ETL Management > Reset Data Warehouse	Clears the refresh dates for all source and target tables. This action forces a full load to occur.
Seed Data > Task Phases	Allows you to add, edit, or delete task phases.
Seed Data > Task Folders	Allows you to add, edit, or delete task folders.
Seed Data > Logical Data Sources	Allows you to add, edit, or delete logical data sources.
UI Styles > Windows (MFC)	Changes the user interface to the Windows style.
UI Styles > UNIX (MOTIF)	Changes the user interface to the UNIX style.
UI Styles > Java (METAL)	Changes the user interface to the Java style.

The DAC Views

The DAC View buttons are located directly under the menu bar. Table 4–3 provides a description of the different DAC views.

Table 4–3 DAC Views

View	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 allows 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 servers, database connections, and email notification. 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".

The DAC Top Pane Toolbar

Table 4–4 describes the commands available in the top pane toolbar.

Table 4–4 DAC Top Pane Toolbar

Command	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.
Refresh	Retrieves the data from the repository with the last used query.
Reference	Design view only. Opens the Reference dialog, which allows you to copy objects from one container to another. For more information about referencing objects, see "About Object Ownership in the DAC".
Assemble	Design view only. Assembles a subject area, with dimension and related tables as well as tasks.
Drop-down list	Design view only. Allows you to filter the source system container objects that appear in the top pane list.
Run Now	Execute view, Execution Plans tab only. Starts a new ETL process.
Start	Execute view, Current Run and Run History tabs only. Restarts the selected ETL, after the ETL has failed, stopped, or been aborted.
Stop	Execute view, Current Run and Run History tabs only. 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	Execute view, Current Run and Run History tabs only. 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.
Auto Refresh	Execute view, Current Run tab only. Allows you to turn on and off the automatic screen refresh functionality and set the refresh interval.

The DAC Right-Click Menus

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 4–5 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.
Copy Record	Creates a copy of the selected record, with a unique record ID. The new record is committed to the database 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.
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.
Output to File	Outputs to a text file in the DAC root folder 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, allows you to update the column value for each row to a single value.

Design View Right-Click Menu Commands

Table 4–6 Design View Right-Click Menu Commands

Command	Description
Ownership	■ **Reference.** Opens the Reference dialog, which allows 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, you can 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 allows you to revert a cloned object back to its state as a reference. ■ **Re-Assign Record.** This command allows you to reassign an objects ownership. For more information about the ownership of objects, see "About Object Ownership in the DAC".
Assemble	Assembles a subject area, with dimension and related tables as well as tasks.
Generate Index Scripts	Generates drop index, create index, and analyze table scripts for all tables that participate in the ETL process. The results are stored in the log\scripts directory.
Change Capture Scripts	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	■ **Import Database Tables.** Allows you to import table definitions from a selected database. This action does not import columns. ■ **Import Indices.** Allows you to import index definitions from a selected database for one or more tables as listed in the result of the query. ■ **Import Database Columns.** Allows you to import column definitions from a selected database.
Filter Indices	Allows you to filter by database type the indexes that are displayed in the top pane list.
Output Task Description	Saves to an HTML file the description for a selected task or for all tasks.
Synchronize Tasks	Synchronizes the information the DAC has for a task's source and target tables with the information in the Informatica repository.

Table 4–6 (Cont.) Design View Right-Click Menu Commands

Command	Description
Flat Views	Opens a dialog that allows you to query for various objects, modify data, and do mass updates. You can query for the following objects: **Tables tab:** ■ Related Tables ■ Table Columns **Indices tab:** Index columns **Tasks tab**: ■ Task Source Tables ■ Task Target Tables ■ Task Conditional Tables ■ Task Phase Dependencies ■ Task Parameters

Setup View Right-Click Menu Commands

Table 4–7 Setup View Right-Click Menu Commands

Command	Description
Test Connection	In the Physical Data Sources tab, it allows you to test the database connection. In the Informatica Servers tab, it allows you to test the connection to the Informatica Server and Repository Server. 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.

Execute View Right-Click Menu Commands

Table 4–8 Execute View Right-Click Menu Commands

Command	Description
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 Refresh Dates subtab of the Execution Plans tab is reserved for micro ETL processes.
Build	(Execution Plans tab) Builds the execution plan, by assembling subject areas, tasks, indices, tags, parameters, source system folders, and phases.
Mark as Completed	(Current Run 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 Run and Run History tabs) Fetches the log file for this run from the DAC server and saves it in the ServerLog folder.

Table 4–8 (Cont.) Execute View Right-Click Menu Commands

Command	Description
Get Run Information > Analyze Run	(Current Run and Run History tabs) Saves a description of the run as an HTML file in the Log/Statistics folder.
Get Run Information > Get Chart	(Current Run and Run History tabs) Displays a chart showing changes in task statuses over time in a separate window.
Get Run Information > Get Graph	(Current Run and Run History tabs) Displays a graph showing changes in task statuses over time in a separate window.

The 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 4–2. 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 4–3. Finally, 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 4–4. In addition, 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 4–2 DAC Server Down Icon

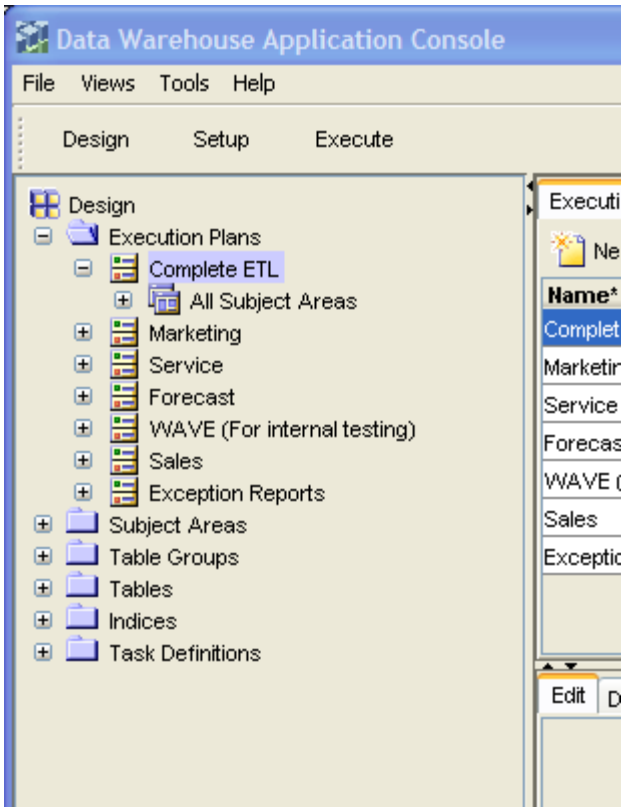
Figure 4–3 DAC Server Idle Icon

Figure 4–4 DAC Server Running Icon

The DAC Navigation Tree

The navigation tree appears on the left side of the DAC window, as shown in Figure 4–5. The tree root nodes correspond to the tabs in the top pane of the DAC window. When a plus sign (+) appears before a node, you can expand the node to view the records belonging to the node. You can double-click a record in the tree to have it display in the top pane in a single-record mode (New, Delete, Copy Record, and Query commands are unavailable), and double-click the root node to return to the list mode.

Figure 4–5 DAC Navigation Tree



The DAC 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 allows you to perform various tasks associated with a selected record in the list. For a description of the toolbar commands, see and "The DAC 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 "The DAC Right-Click Menus".

The list format allows 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.
- For fields that refer to values from other entities, use the query functionality in pop-up dialogs.
- Use Ctrl+C to copy an editable string to the clipboard (not available for read-only strings).
- Ctrl+V to paste a string from the clipboard into a selected cell that supports a string data type.

About Object Ownership in the DAC

The source system container in which an object originates is 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, but the object's ownership remains unchanged.

For example, W_INVOICE_F is a fact table whose owner is the Siebel 7.8 source system container. You can reuse W_INVOICE_F in any other container by referencing it.

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. Use the right-click command and select Ownership, then select Push to References to *push* the changes to the referenced object's child objects. 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 Table 4–6.

Font Variations of Objects Displayed in the DAC

The different categories of objects are represented in the DAC with differing fonts.

Table 4–9 Font Variations Displayed in the DAC

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

Object Behaviors To Consider

- 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 have to push the changes to the referenced objects. For example, if you add a column to a table registered in the DAC, the new column is not automatically added to the references.
- When you delete a referenced object, only the referenced object is deleted, not the original object.
- If you delete an object from the owner container, the object is deleted and all references are deleted. 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.

- 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.

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 child tabs in the bottom pane are not available.

This section includes the following topics:

DAC Query Commands and Operators

Table 4–10 describes the query commands and operators you can use to define your query criteria.

Table 4–10 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.
\	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.)

Table 4–10 (Cont.) DAC Query Commands and Operators

Operator	Description
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 surround 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, click Query on the toolbar or in right-click menu.

   A blank row in a list appears.
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 appears.
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 allows 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 DAC, 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.

   An informational message tells you which records were updated.
10. Click OK to close the window.

5

Customizing, Designing, Executing and Monitoring ETL Processes

This chapter provides information about customizing, designing, executing, and monitoring ETL processes.

This section includes the following topics:

Creating or Copying a Source System Container

The metadata for a source system is held in a container. You cannot change the metadata for preconfigured containers. If you want to customize the metadata in a preconfigured container, you must first make a copy of the container. The 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.

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

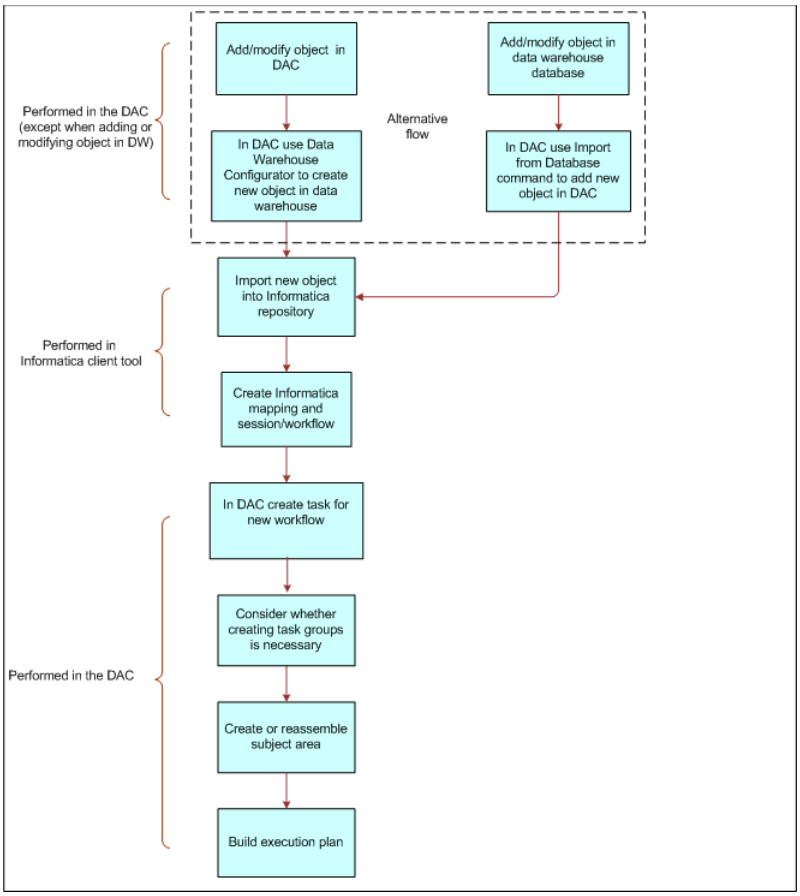
   The ID and Name fields are alphanumeric. The Name can contain spaces but the ID cannot.
3. Select one of the following:
   - Create Empty New Source System Container
   - Create as a Copy of Source System Container
4. If you are creating an empty, new container, click OK.
5. If you are making a copy of an existing container, select the existing container from the drop-down list, and then click OK.

About Customizing the Data Warehouse

You can add tables, columns, and indexes to the data warehouse, and you can modify these existing objects. Customizing the data warehouse in this way requires using the DAC and Informatica client tools. For more information about using Informatica client tools to customize the Oracle Business Analytics Warehouse, see the *Oracle Business Intelligence Applications Installation and Configuration Guide*.

Figure 5–1 shows the major steps required for adding a new object to the data warehouse or modifying existing objects. As shown in Figure 5–1, you can begin the customization process by adding or modifying the new data warehouse object in the DAC and then using the DAC's Data Warehouse Configurator to create or update the object in 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 the DAC.

Figure 5–1 Process Flow to Add New Object to Data Warehouse



Adding a New Table and Columns to the Data Warehouse

As shown in Figure 5–1, there are two alternative process flows for adding a new object to the data warehouse. You can enter the table and column definitions in the DAC and then use the DAC's Data Warehouse Configurator to create the table and columns in the data warehouse database; for this method, follow the procedure, "To add a new table and columns to the data warehouse using the DAC's Data Warehouse Configurator".

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 the DAC; for this method, follow the procedure, "To add a new table and columns using the DAC's Import command".

To add a new table and columns to the data warehouse using the DAC's Data Warehouse Configurator

1. In the DAC toolbar, select the appropriate source system container from the drop-down list in the toolbar.
2. In the Menu bar, select Views, then select Design, then select Tables.
3. Create the new table.
   a. In the Tables tab, click New.
   b. In the Edit child tab, enter the appropriate information about the table, and click Save.

For a description of the fields in this tab, see "Tables Tab".

4. Add the columns for the new table.
   a. In the Columns child tab, 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 of fact table.

5. Create the new tables and columns in the data warehouse database.
   a. Select Tools, then select ETL Management, then select Configure.
   b. Select the appropriate Source and Target database platforms, and then click OK.
   c. In the Data Warehouse Configuration Wizard, select Create Data Warehouse Tables, and then click Next.
   d. Enter the required information, and then click Start.

      An informational message reports whether the process was successful. For information about the process, you can review the createwtables.log file in the OracleBI\DAC\log\config folder.

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 toolbar, select the appropriate source system container from the drop-down list in the toolbar.
3. In the Menu bar, select Views, then select Design, then select Tables.
4. Import the new table definition.
   a. Right-click and select Import from Database, then select 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.

      An informational message indicates whether the process was successful.
5. Import the new column definitions.
   a. In the Tables tab, query for the table you imported in Step 4.
   b. With the table highlighted, right-click and select Import from Database, then select 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.

An informational message 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 toolbar, select the appropriate source system container from the drop-down list in the toolbar.
3. In the Menu bar, select Views, then select Design, then select Tables.
4. Query for the table for which you want to import index definitions.
5. Right-click and select Import from Database, then select Import Indices.
6. Choose to import indexes for a selected table or for all the records retrieved in your query, and click OK.
7. In the Import Indices dialog, select DataWarehouse from the Data Sources drop-down list.
8. 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.

   An informational message 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 *Oracle Business Intelligence Applications Installation and Configuration Guide*.

Creating Informatica Mappings and Workflows

This step requires using Informatica client tools to create new Informatica mappings and workflows for the data warehouse objects that you imported into the Informatica repository. For instructions on this step of customizing the data warehouse, see *Oracle Business Intelligence Applications Installation and Configuration Guide*.

Creating Tasks in the 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 toolbar, select the appropriate source system container from the drop-down list in the toolbar.
2. In the DAC, create custom logical and physical task folders for the custom folder you created in the Informatica repository.
   a. In the DAC, navigate to Tools, then select Seed Data, then select Task Folders.
   b. To create a custom logical folder, click New.
   c. In the Name field, enter a name for the custom logical folder, for example, Custom Logical.
   d. In the Type field, select Logical.
   e. To create a custom physical folder, click New.
   f. In the Name field, enter a name for the custom physical folder, for example, Custom Physical.
   g. In the Type field, select Physical.
3. Register the folders you created in Step 2 in the Source System Folders tab.
   a. Navigate to Design, then select Source System Folders.
   b. Click New.
   c. In the Edit child tab, 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.
4. Create new tasks for the workflows.
   a. Navigate to Design, then select Tasks, and click New in the top pane toolbar.
   b. In the Edit child tab, 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. Click OK in the informational message box.

      This command synchronizes the source and target table information between the DAC and Informatica.

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 change the order in which tasks are executed.

To set a task phase dependency

1. In the DAC toolbar, select the appropriate source system container from the drop-down list in the toolbar.
2. In the Menu bar, select Views, then Design, then select Tasks.
3. Query for the task for which you want to add a phase dependency, and make sure it is highlighted.
4. Click the Phase Dependency child tab.
5. Click Add/Remove.
6. In the Choose Phases dialog, select a phase and click Add.
7. Click OK in the message box that states the phase was added.
8. Select an Action and a Grain, click OK.

   For information about these fields, see "Task Tab: Phase Dependency Subtab".

   The task phase dependency appears in the Phase Dependency child tab.
9. Reassemble the appropriate subject area.
   a. In the Subject Areas tab, query for the appropriate subject area.
   b. Click Assemble.
10. Rebuild the execution plan.
    a. Navigate to the Execution Plans tab in the Execute view.
    b. Query for the appropriate execution plan.
    c. In the Parameters child tab, click Generate.
    d. In the top pane toolbar, click Build.

Creating a Task Group

The DAC automatically organizes tasks into a dependency structure based on dependency rules. For information about the DAC's dependency rules, see "How the DAC Determines the Order of Task Execution within an Execution Plan". The DAC assigns priority randomly to tasks that have the same properties. You can use the Task Group feature to group such tasks that share the same properties and enforce a priority of your choosing.

This feature can be useful for the following: truncation and restartability purposes; when more than one task with similar properties writes to the same table; and when there is a circular read/write relationship between tables; 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 DAC, select the appropriate source system container from the drop-down list in the toolbar.
2. In the Menu bar, select Views, then select Design, then select Task Groups.
3. Create a new task group.
   a. Click New in the top pane toolbar.
   b. In the Edit child tab, enter a name and select the appropriate properties.
4. Click the Child Tasks child tab, 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 right-hand window, enter an execution order.
8. Click Save, and then click OK to close the window.

About Parameter Management

This section describes how the DAC handles parameters and how you can define and manage parameters at the source system and task levels. It contains the following topics:

Overview of Parameters

The ETL logic in Oracle Business Intelligence Applications uses parameters in the Informatica mappings and sessions. You define and manage parameters using the DAC parameter management feature. A parameter can apply to all tasks under a source system container (referred to as a source system parameter) or it can apply to a particular task (referred to as a task level parameter). Parameters set up at the task level have priority over parameters set up at the source system level.

In the DAC, there are two types of parameters: static and runtime. The value of static parameters remains constant for all ETL runs. Examples of static parameters include language codes and currencies. The value of runtime parameters is dynamic, and the DAC updates this value for each ETL run. Examples of dynamic parameters include last refresh dates and last WID sequence numbers.

Parameter Data Types

Parameters can have one of the following data types.

Text

The value for the parameter is defined as text. You can use the Text data type for both static and runtime parameters.

DB Specific Text

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. The DAC evaluates the string based on the source or target database type. If you do not specify database-specific text, the DAC returns the default value.

Timestamp

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 the 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. The 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 specified for a given timestamp parameter can include nested DAC parameters. For information about nested parameters, see "Nesting Parameters within Other Parameters".

SQL

The DAC fetches the value for the parameter from a database using SQL.

Preconfigured Parameters

Oracle Business Intelligence Applications ships with preconfigured parameters. Some of these preconfigured parameters are held in text files named parameterfileDW.txt and parameterfileOLTP.txt, which are stored in the folder \OracleBI\DAC\Informatica\parameters\input. Other preconfigured parameters are held in the DAC. The parameters held in the DAC are specific to the different source system containers.

You can add new parameters in the DAC or change the existing parameters held in the DAC. However, Oracle recommends that you do not change the parameters held in the parameter text files. You can override the parameters in the text files by creating new parameters in the DAC.

If you do make changes to the parameter text files, however, make sure the files remain in the folder \OracleBI\DAC\Informatica\parameters\input.

How the DAC Handles Parameters at Runtime

During an ETL execution, the DAC reads and evaluates the parameters held in the text files parameterfileDW.txt and parameterfileOLTP.txt along with the parameters held in the DAC. The DAC then creates an individual parameter file for each session. This file contains the evaluated name-value pairs for all parameters, both static and runtime. The naming convention for this parameter file is <Informatica folder name>.<Informatica session name>.txt. The DAC copies this file to a location specified in the DAC system property InformaticaParameterFileLocation.

Note: The Informatica Server must be configured to read parameter files from the location specified in the DAC system property InformaticaParameterFileLocation. For instructions on setting this property, see *Oracle Business Intelligence Applications Installation and Configuration Guide*.

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_.

An example of a text parameter that returns the current run ID placed in a where clause of a SQL parameter would look similar to the following:

```
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, the DAC randomly picks one of the parameters in the circle and evaluates it as an empty string.

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 type parameter:

1. Do one of the following:
   - To define a source system parameter, from the Views menu, select Design, and then select Source System Parameters.
   - To define a task level parameter, from the Views menu, select Design, then select Tasks, and then click the Parameters subtab.
2. Click New.
3. Enter a parameter name.
4. Select the Text data type.
5. Click in the Value field to open the Enter Parameter Value dialog.
6. Select one of the following options:
   - **Static.** This option specifies a value that remains constant for all ETL runs.
   - **Runtime.** This option specifies a value will be updated by the DAC before each ETL run.
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. (Optional) To inactivate the parameter, select Inactive.
10. Click Save.

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 type parameter:

1. Do one of the following:
   - To define a source system parameter, from the Views menu, select Design, and then select Source System Parameters.
   - To define a task level parameter, from the Views menu, select Design, then select Tasks, and then click the Parameters subtab.
2. Click New.
3. Enter a parameter name.
4. Select the DB Specific Text data type.
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.** This option specifies a source database connection.
   - **@DAC_TARGET_DBTYPE.** This option 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. (Optional) To inactivate the parameter, select Inactive.
11. Click Save.

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 type parameter:

1. Do one of the following:
   - To define a source system parameter, from the Views menu, select Design, and then select Source System Parameters.
   - To define a task level parameter, from the Views menu, select Design, then select Tasks, and then click the Parameters subtab.

2. Click New.
3. Enter a parameter name.
4. Select the Timestamp data type.
5. Click in the Value field to open the Enter Parameter Value dialog.
6. Select one of the following options:
   - **Static.** This option specifies a value that remains constant for all ETL runs.
   - **Runtime.** This option specifies the value will be updated by the DAC before each ETL run.
   - SQL. This option
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 the 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. (Optional) To inactivate the parameter, select Inactive.
12. Click Save.

Defining a SQL 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 SQL type parameter:

1. Do one of the following:
   - To define a source system parameter, from the Views menu, select Design, and then select Source System Parameters.
   - To define a task level parameter, from the Views menu, select Design, then select Tasks, and then click the Parameters subtab.
2. Click New.
3. Enter a parameter name.
4. Select the SQL data type.
5. Click in the Value field to open the Enter Parameter Value dialog.

6. Select a Logical Data Source.
7. Enter a SQL statement, and click OK.
8. (Optional) To inactivate the parameter, select Inactive.
9. Click Save.

Specifying Tablespaces for Indexes by Table Type

You can specify tablespaces for indexes by table type in the Setup view in the DAC. For example, you could specify a tablespace for indexes of the dimension table type.

Note: You must create the tablespace in the database before you can specify tablespaces for indexes.

To specify tablespaces for indexes by table type

1. In the DAC toolbar, click Setup, then click the Physical Data Sources tab, then click the Index Spaces subtab.
2. In the Index Spaces toolbar, click Generate.

   A list of available table types appears in the Table Type list.
3. In the Index Space column, enter an index space for each table type, and click Save.

   If the Index Space property is left empty for a table type, the default index space for the default database connection will be used. The default index space is specified on the Edit subtab of the Physical Data Sources tab in the Setup view. If the Default Index Space property is also empty, the default tablespace assigned for the table owner will be used.

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 any subject area, unless the tag is part of the subject area definition "Include Task" property.

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 DAC toolbar, select the appropriate source system container from the drop-down list in the toolbar.
2. In the Menu bar, select Views, then select Design, then select Configuration Tags.
3. Create a new configuration tag.
   a. Click New in the top pane toolbar.
   b. In the Edit child tab, enter a name.
   c. Make sure the Include Tasks check box is not selected.
   d. Click Save.
4. Add tasks to the configuration tag.
   a. With the new configuration tag highlighted in the top pane, click the Tasks child tab.
   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 the DAC toolbar, select the appropriate source system container from the drop-down list in the toolbar.
2. In the Menu bar, select Views, then Design, then select Configuration Tags.
3. Query for the configuration tag that contains the tasks you want to reassign to a subject area.
4. Verify the configuration tag contains the appropriate tasks by clicking the Tasks child tab 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.

---

5. 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 child tab.
    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.
6. 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 Reassemble.

To add non-autogenerated tasks to a subject area

1. In the DAC toolbar, select the appropriate source system container from the drop-down list in the toolbar.
2. In the Menu bar, select Views, then select Design, then select Configuration Tags.
3. Create a new configuration tag.
    a. Click New in the top pane toolbar.
    b. In the Edit child tab, enter a name.
    c. Select the Include Tasks check box.
    d. Click Save.
4. Add the non-autogenerated tasks to the configuration tag.
    a. With the new configuration tag highlighted in the top pane, click the Tasks child tab.
    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.
5. 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 child tab.
    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.
6. 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 Reassemble.

To assign only configuration tag tasks to a subject area (excludes the subject area's autogenerated tasks)

1. In the DAC toolbar, select the appropriate source system container from the drop-down list in the toolbar.
2. In the Menu bar, select Views, then select Design, then select Subject Areas.
3. 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.

4. Select the Configuration Tag Tasks Only check box, and click Save.
5. Create a configuration tag.
   a. Navigate to the Configuration Tags tab.
   b. Click New in the top pane toolbar.
   c. In the Edit child tab, enter a name.
   d. Select the Include Tasks check box.
   e. Click Save.
6. Add the tasks to the configuration tag.
   a. With the new configuration tag highlighted in the top pane, click the Tasks child tab.
   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.
7. Associate the configuration tag with the subject area.
   a. With the configuration tag highlighted in the top pane, click the Subject Areas child tab.
   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.
8. 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 Reassemble.

Considerations in Designing a Subject Area

Oracle Business Intelligence Applications provides preconfigured subject areas. You can change these preconfigured subject areas or create new subject areas to correspond to your particular business processes.

Note: To change a preconfigured 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".

Designing a Subject Area

In designing a subject area, you should consider the following questions:

- **Tables.** Which tables need to be populated for the data warehouse? From which tables does your 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?

How the DAC Determines Tasks Required for Subject Areas

A subject area is a collection of tasks. When a subject area is defined, the DAC uses the following logic to assemble the collection of tasks:

1. Initial selection of tables.

   Find all the fact tables that belong to the subject areas.

2. Recursive selection of related tables.

   Recursively find all the tables directly related through foreign keys and all other logically related tables.

3. Initial selection of tasks.

   Find all the tasks that load into the tables selected above, that is, tasks whose target tables are one of the tables identified above.

4. Recursive selection of all tasks.

   Depending on the source and target table relationships, recursively figure out the prerequisite tasks.

How the DAC Determines the Order of Task Execution within an Execution Plan

An execution plan is a collection of subject areas and a unique collection of tasks. A task can have prerequisite tasks that need to be executed before its own execution. The DAC determines the order of tasks based on the following considerations:

- A task's source and target table

  The DAC server first looks at a task's source and target table. 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 DAC server would determine task T2 should be executed before T1.

The DAC server next considers the following:

- Task phase

  An ETL process typically goes through several 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

- A table's Truncate Always properties

  The order of execution based on Truncate Always properties is as follows:

  1. Insert
  2. Upsert
  3. Physical data source
  4. Priority

- The DAC randomly organizes tasks that are the same in all properties. If some tasks need to be executed in a particular order, the DAC enables you to create a task group in which you can specify an execution order.

Creating a Subject Area

When you create a new subject area, you assign one or more fact tables to the subject area. The 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 DAC, select the appropriate source system container from the drop-down list in the toolbar.
2. In the Menu bar, select Views, then select Design, then Subject Areas.
3. In the top pane toolbar, click New.
4. In the Edit child tab, enter a name for the subject area, and click Save.
5. Make sure the new subject area name is highlighted in the top pane, and click the Tables child tab.
6. Click Add/Remove in the child tab toolbar.

   The Choose Tables dialog opens. 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, the DAC will reassemble all the subject areas listed in the top pane.

    The DAC assembles the selected subject area by determining what dimensions and other related tables are required and what tasks are needed to load these tables.

    You will receive an informational message when the assemble process is completed.

12. Click the Tasks tab to view which tasks the DAC has determined are required for this subject area.

    Tasks that are automatically assigned to the subject area by the 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.

Building and Running an Execution Plan with the DAC

Execution plans are subject areas that are used to execute ETL processes. Before you attempt to run an execution plan, make sure you have completed the following:

- Set database connections to the transactional and data warehouse databases (in the Physical Data Sources tab).
- Registered the Informatica servers and repository server (in the Informatica Servers tab).

Note: All subject areas in an execution plan must belong to the same source system container.

To build and run an execution plan

1. Navigate to the Execute view, then select the Execution Plans tab.
2. Create a new execution plan.
   a. In the top pane toolbar, click New.
   b. In the Edit child tab, enter a name for the execution plan and other appropriate information.

      For a description of the fields in this tab, see "Execution Plans Tab".

   c. Click Save.
3. Associate one or more subject areas with the execution plan.
   a. Click the Subject Areas child tab.
   b. Click Add/Remove in the bottom pane toolbar.
   c. In the Choose Subject Areas dialog, select the appropriate source system container.
   d. Query for the subject area you want to associate with the execution plan.

e. Select the subject area and click Add.

   You can associate multiple subject areas with an execution plan, but all the subject areas must be from the same source system container.

f. Click OK to close the window.

4. Generate the runtime parameters.

   a. Click the Parameters child tab.

   b. Click Generate in the bottom pane toolbar.

   c. For each Datasource type, enter the appropriate name in the Value field.

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

5. In the top pane of the Execution Plans tab, make sure the new execution plan is highlighted, and click Build.

   The DAC builds the execution plan.

6. To run the execution plan, select the execution plan in the top pane, and click Run Now.

   Once the ETL process starts running you can monitor its progress in the Current Run tab.

   For information about how refresh dates are tracked, see "About Refresh Dates".

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

Creating a Micro ETL Execution Plan

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. The 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, the DAC populates refresh date values in the Refresh Dates child tab of the Execution Plans tab. If a subject area is used in a regular execution plan (an execution plan with the Keep Separate Refresh Dates option not selected) as well as a micro ETL execution plan, the DAC maintains refresh dates for the tables in the regular execution plan in the Refresh Dates child tab of the Physical Data Sources tab (Setup view).

In cases of a subject area being 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, the 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.

Caution: 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.
- 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, 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.

To create a micro ETL execution plan

1. In the DAC toolbar, select the appropriate source system container from the drop-down list in the toolbar.
2. In the Menu bar, select Views, then select Design, then select Subject Areas.
3. In the Subject Areas tab, assemble a small subject area.
4. In the Tasks child tab, inactivate all tasks that are not required for the execution plan.
5. Create a new execution plan.
   a. Navigate to the Execute view, then select the Execution Plans tab.
   b. Enter a name for the execution plan
   c. Select the Keep Separate Refresh Dates check box.
   d. Click Save.
6. Associate one or more subject areas with the execution plan.
   a. Click the Subject Areas child tab.
   b. Click Add/Remove in the bottom pane toolbar.
   c. In the Choose Subject Areas dialog, select the appropriate source system container.
   d. Query for the subject area you want to associate with the execution plan.
   e. Select the subject area and click Add.

      You can associate multiple subject areas with an execution plan, but all the subject areas must be from the same source system container.

f. Click OK to close the window.

7. Generate the runtime parameters.
   a. Click the Parameters child tab.
   b. Click Generate in the bottom pane toolbar.
   c. For each Datasource type, enter the appropriate name in the Value field.
8. In the top pane of the Execution Plans tab, make sure the new execution plan is highlighted, and click Build.

   The DAC builds the execution plan.
9. Navigate to the Ordered Tasks child tab and verify the sequence of tasks.

   The execution plan is now ready to run as a micro ETL execution plan.
10. Create a schedule for the micro ETL execution plan. For instructions, see "Scheduling an Execution Plan".

Scheduling an Execution Plan

Use this procedure to schedule an execution plan in the DAC.

To schedule an execution plan

1. In the DAC, navigate to the Scheduler tab.

   The current list of schedules appears 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.
5. If you want the schedule to run once, select the Run Only Once check box, and then select a start and end date and time.
6. To create a periodic schedule, select a recurrence pattern, and enter the appropriate date and time parameters.
7. Click Save.

About Refresh Dates

Refresh dates refer to the date of the last ETL process (the last time data was extracted from tables in a given database or loaded into tables in a given database). The DAC uses the refresh dates to determine whether to run the incremental load commands or to run full load commands and whether to truncate the target tables.

Refresh dates are tracked only for tables that are either a primary source or a primary target on tasks in a completed run of an execution plan. The DAC runs the full load command for tasks on which a table is a primary source or target if the refresh date against the table is null. When there are multiple primary sources, the earliest of the refresh dates will trigger a full load or an incremental load. If any one of the primary source tables has no refresh date, then the DAC will run the full load command.

Table 5–1 shows the possible scenarios regarding refresh dates.

Table 5–1 Refresh Date Scenarios

Scenario	Table Type (in Tasks child tabs)	Refresh Date	Command DAC Will Use	Truncate Target Table?
1	Primary Source	Null	Full Load	Yes
1	Primary Target	Null	Not applicable	Not applicable
2 (See note below)	Primary Source	Null	Full Load	No
2	Primary Target	Not Null	Not applicable	Not applicable
3 (See note below)	Primary Source	Not Null	Full Load	Yes
3	Primary Target	Null	Not applicable	Not applicable
4	Primary Source	Not Null	Incremental Load	No
4	Primary Target	Not Null	Not applicable	Not applicable

- **Scenario 2.** When two or more source tables load into the same target table as separate tasks, the source table in the second task may have refresh date as null while the target may have a refresh date.
- **Scenario 3.** When a source loads into more than one target table in separate tasks, the refresh date may be null for the second target table while the source table may have refresh dates.

Monitoring Execution Plan Processes

The Current Run tab in the Execute view provides predefined reports that enable you to monitor execution plan processes in order to isolate bottlenecks and enhance performance.

To monitor an execution plan process

1. In the DAC, navigate to the Current Run tab.
2. Right-click and select Get Run Information.
3. The following options are available:
   - Get log file
   - Analyze run
   - Get chart
   - Get phase chart
   - Get graph

6

Common Tasks Performed in the DAC

This chapter contains the following topics:

Importing DAC Metadata

The DAC's Import/Export feature enables you to import or export source system-specific DAC metadata into or 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.

To import DAC metadata

1. In the DAC menu bar, select Tools, then select DAC Repository Management, then select Import.
2. Select the directory from which you want to import DAC metadata, or accept the default directory.
3. Select the appropriate source system containers.

4. Select the appropriate categories of metadata you want to import:
   - **Logical.** Imports all information contained in the Design view and database connection information.
   - **System.** Imports all information contained in the Setup view, except passwords for servers and database connections.
   - **Run Time.** Imports information about ETL runs (contained in the Execute view)
5. If you are importing metadata into a blank repository or to completely replace the current metadata in the repository, select Truncate Repository Tables.

   This action overwrites the content in the current repository. Selecting the Truncate Repository Tables option greatly increases the speed of the import process.
6. (Optional) Select Enable Batch Mode to insert the imported metadata into the repository as an array insert.

   This action increases the speed of the import process.
7. Click OK.
8. Verify the import process by reviewing the log file \OracleBI\DAC\log\import.log.

Exporting DAC Metadata

The DAC's Import/Export feature enables you to import or export source system-specific DAC metadata into or 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.

To export DAC metadata

1. In the DAC menu bar, select Tools, then select DAC Repository Management, then select Export.
2. Select the directory to which you want to export DAC metadata, or accept the default directory.
3. Select the appropriate source system containers.
4. Select the appropriate categories of metadata you want to export:
   - **Logical.** Exports all information contained in the Design view and database connection information.
   - **System.** Exports all information contained in the Setup view, except passwords for servers and database connections.
   - **Run Time.** Exports information about ETL runs (contained in the Execute view).
5. Click OK.
6. Verify the export process by reviewing the log file \OracleBI\DAC\log\export.log.

Distributing DAC Metadata

Typically, you may have multiple environments, such as development, QA, production, and so on. When you make changes to the development environment, you

test it, and then deliver it, exporting the whole environment and distributing it to the other environments. The data is exported as XML files, which are stored in the DAC\export directory on the client machine where the export is done.

To apply changes from the development environment to any other, you copy all of the XML files into the DAC\export folder and then import the data. To export the DAC metadata, follow the instructions in the procedure, "Exporting DAC Metadata". To import the DAC metadata, follow the instructions in the procedure, "Importing DAC Metadata".

Running the DAC Server Automatically

Follow this procedure to set up the DAC server to be run automatically when your machine reboots.

To set up the DAC server to run automatically upon rebooting the machine

1. Navigate to Programs, then select Accessories, then select System Tools, then select Scheduled Tasks.
2. Double-click Add Scheduled Task.
3. In the Scheduled Task Wizard, browse to the startserver.bat file, 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 appears in the Scheduled Task window.
6. Right-click the task and select Properties.
7. In the Settings tab, remove the check from the "Stop the task if it runs for 72 hours" check box.

To start the DAC server as a scheduled task

1. Navigate to Programs, then select Accessories, then select System Tools, then select Scheduled Tasks.
2. Right-click startserver, and then click Run.

To stop the DAC server as a scheduled task

1. Navigate to Programs, then select Accessories, then select System Tools, then select Scheduled Tasks.
2. Right-click startserver, and then click End Task.

To check if the DAC server is running

1. Navigate to Programs, then select Accessories, then select System Tools, then select Scheduled Tasks.
2. Select the startserver task.
3. In the Windows menu bar, select View, then select Details.

Command Line Access to the DAC Server

This section covers accessing the DAC server through a command line. It includes the following topics:

- Setting Up Command Line Access to the DAC Server

- Using the Command Line to Access the DAC Server

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.

Command Line Operations

The command line feature enables you to start an execution plan and stop the operation of a running execution plan.

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 another execution plan, the request will be rejected. If an execution plan failed, a request to run the same execution plan again will be executed; however, a request to run a different execution plan will be rejected. If the execution plan that last ran completed successfully, a request to run a new execution plan will be executed.

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 (set in the system properties) is not a positive number.
- There are no active Informatica servers in the server list.
- One or more Informatica servers do not have the passwords defined.
- One or more Informatica servers do not have a maximum number of sessions properly set.
- One or more data sources do not have the passwords defined.
- One or more data sources do not have a maximum number of connections properly set.
- One or more data sources do not have a number defined (set in Physical Data Sources tab).

Stopping the Operation of a Running 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.

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) 2003 Siebel Systems, Inc.
Siebel 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 : (aqamarD510) DAC Port :
3141 Status: Stopped Log File Name: Complete_ETL.255.log Database Connection(s)
Used :


OLTP jdbc:microsoft:sqlserver://vranganaw8:1433;DatabaseName=OLTP Data
Warehouse jdbc:microsoft:sqlserver://vranganaw8:1433;DatabaseName=olap

Informatica Server(s) Used  :

InformaticaServer4-vranganaw8:(4)  InformaticaServer2-vranganaw8:(2)
InformaticaServer3-vranganaw8:(3)  InformaticaServer1-vranganaw8:(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.

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. The Command Line utility does not need the entire DAC environment. The machine on which you invoke the commands for the DAC server requires only the files DAWSystem.jar, dac.properties, and dacCmdLine.bat.

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 OracleBI\DAC directory to a local directory:
   - DAWSystem.jar
   - dac.properties
   - dacCmdLine.bat
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.

Your dac.properties file should look similar to the following:

```
ServerHost=vranganaw8 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
StartETL	Starts an execution plan. You must specify an execution plan name.
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 servers.

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.

Command Line	Description
`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 server connections as defined in the DAC repository from the DAC server.

DAC Repository Command Line Options

This section describes the DAC repository command line parameters that are exposed by the AutomationUtils.bat file, which is located in the OracleBI\DAC folder.

Import DAC Metadata by Application

The IMPORT option imports DAC metadata into the DAC repository for specified source system containers. The import process truncates all imported tables. You cannot perform an incremental import with this command.

Syntax:

```
IMPORT <folderName> <contName1> <contName2> ...
```

where:

Parameter	Description
`folderName`	Full path to the root of the import file structure.
`contName`	(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.

Export DAC Metadata by Application

The EXPORT option exports DAC metadata from the DAC repository for specified source system containers.

Syntax:

```
EXPORT <folderName> <contName1> <contName2> ...
```

where:

Parameter	Description
`folderName`	Full path to the root of the export file structure.
`contName`	(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.

Import DAC Metadata by Categories

The IMPORTCATEGORY option imports DAC metadata into the DAC repository based on the Logical, Run Time, or System categories. The import process truncates all imported tables. You cannot perform an incremental import with this command.

Syntax:

```
IMPORTCATEGORY <folderName> <logical> <runtime> <system>
```

where:

Parameter	Description
`folderName`	Full path to the root of the import file structure.
`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).

Export DAC Metadata by Categories

The EXPORTCATEGORY option exports DAC metadata from the DAC repository based on the Logical, Run Time, or System categories.

Syntax:

```
EXPORTCATEGORY <folderName> <logical> <runtime> <system>
```

where:

Parameter	Description
`folderName`	Full path to the root of the import file structure.
`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).

Create Schema

The CREATESCHEMA option creates the schema of a new DAC repository.

Syntax:

```
CREATESCHEMA <unicodeFlag> <workSpace name>
```

where:

Parameter	Description
`unicodeFlag`	If the value of this parameter is `true`, the schema is created as unicode. If the value is `false`, it is not created as unicode.
`workSpace name`	The name of the workspace in which the schema is created.

Drop Schema

The DROPSCHEMA option drops the schema of the DAC repository.

Syntax:

```
DROPSCHEMA
```

Analyze

The ANALYZE option analyzes the DAC repository tables.

Syntax:

```
ANALYZE
```

Upgrade

The UPGRADE option upgrades the DAC repository.

Syntax:

```
UPGRADE
```

Set Password

The SETPASSWORD option sets the passwords for the Informatica servers and physical data sources in the DAC repository.

Syntax:

```
SETPASSWORD <type> <logicalName> <password>
```

where:

Parameter	Description
`type`	Possible values are server or dbconn.
`logicalName`	Logical name of the server or data source record in the DAC.

Note: : If the logical name or password contains spaces, quotes are required.

Replacing an Informatica Workflow with a Custom SQL File

You can improve the performance of loads by replacing Informatica workflows with custom SQL files.

To replace an Informatica workflow with a custom SQL file

1. Create a SQL file to be used to load the table, and unit test it.
2. Create an XML or SQL file with one or more SQL statements in the format that the DAC can understand.

   For more information about creating an XML or SQL file, see "Using SQL Files as an Execution Type in the DAC".

   You can create one file for a full load and one for an incremental load, or you can use the same file for both full and incremental loads.
3. Save the file in the OracleBI\DAC\CustomSQLs directory.
4. In the Tasks tab of the DAC Design view, query for the task for which you want to replace the Informatica workflow.
5. Replace the workflow name in the Command for Incremental Load or Command for Full Load fields with the XML or SQL file.
6. Change the Execution Type to SQL.

Determining the Informatica Server Maximum Sessions Parameter Setting

You set the Maximum Sessions parameter value when you register the Informatica Server in the DAC client. This parameter specifies the maximum number of workflows that can be executed in parallel on the Informatica server. If the number of sessions is zero or is not specified, the DAC server assigns the default value of 10.

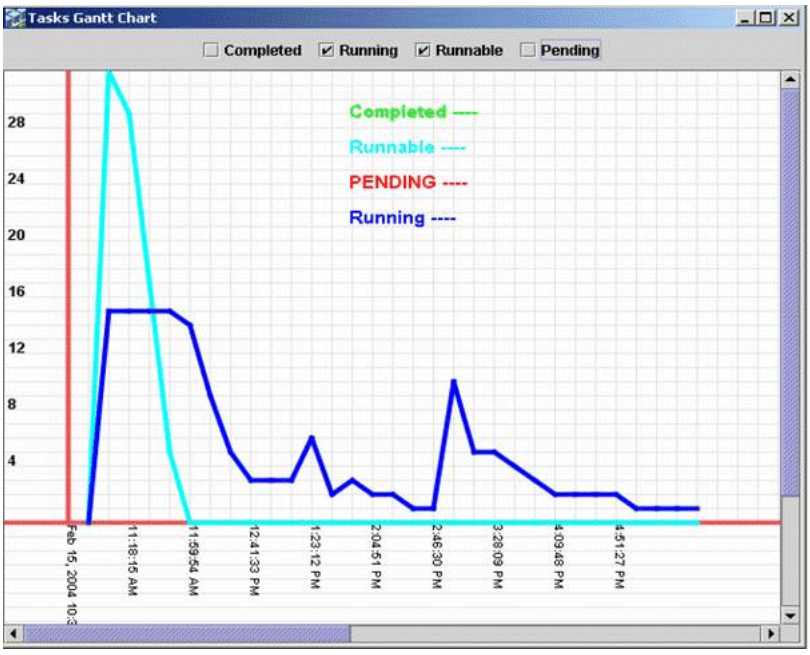
You should consider the following factors when determining the Maximum Sessions parameter value:

- How powerful the machine is that hosts the Informatica Server.
- How many Informatica Server servers 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 Run Tab".

  For an optimal run, the runnable queue should be at zero or should reach zero in a short time. For example, Figure 6–1 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 6–1 from the right-click menu (select Get Run Information, then select Get Graph) on the DAC Current Run and Run History tabs. If you find that the runnable curve does not reach zero in a short time, you should increase the Maximum Sessions parameter value to make more Informatica slots available.

Figure 6–1 Sample Performance Run



Determining the Number of Transactional and Data Warehouse Database Connections

This section describes how to determine the maximum number of database connections you need between the DAC server and the transactional database and the DAC server and the data warehouse database. You set the Max Num Connections parameter when you create the transactional and data warehouse database connections.

For the transactional database, the DAC server uses these connections to perform change capture. The number of connections you set for this connection pool 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 Max Num Connections setting to 15 or 20 (10 is the default). If you have a less powerful transactional database server, you should not overload the operational system with ETL processes. Therefore, you should set the value below 10.

For the data warehouse database, the DAC server uses these connections to perform processes such as truncate tables, drop and create indexes, and analyze tables. You should not set the Max Num Connections value higher than the Maximum Sessions parameter value (the maximum number of workflows that can be executed in parallel on the Informatica server) because these values have a one to one relationship.

Running Two DAC Servers on the Same Machine

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.

To run two DAC servers on the same machine

1. Copy the OracleBI\DAC folder to a different folder on the same machine.

   For example, you might copy the C:\OracleBI\DAC folder to C:\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:\OracleBI\DAC folder to C:\DAC_SERVER2\DAC, make sure that the C:\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 repository connection.

   a. Navigate to Tools, then select DAC Server Management, then select DAC Server Setup.

      An informational 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.

5. For each instance, set up the DAC server system properties.

a. Navigate to Setup, then select DAC System Properties.

b. Set the DAC Server Host, OS, and Port properties.

6. Start each DAC server from its directory.

Customizing Index and Analyze Table Syntaxes

The customsql.xml file, located in the OracleBI\DAC\CustomSQLs directory, contains the syntax for dropping and creating indexes and analyzing tables. You can edit the customsql.xml file to change the behavior of these operations.

To edit the Analyze Table syntax

1. Open the customsql.xml file located in the OracleBI\DAC\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 OracleBI\DAC\CustomSQLs directory.
2. Locate the Create Index syntax for the appropriate database type, and edit the syntax.

Using SQL Files as an Execution Type in the DAC

There are two types of custom SQL files that can be executed through the DAC: XML formatted .xml files, and plain text .sql files. For examples of XML and SQL files, see the files named samplesql.sql and samplesql.xml in the folder Oracle BI\DAC\CustomSQLs.

XML Formatted Files

An XML file consists of a set of SQL statements for which various options are defined using XML attributes. The tag names and values are case sensitive. The following tags are available:

name

The name of the SQL block. The DAC server uses this name to report errors.

type

The type of SQL block. Possible values are SQL and Stored Procedure.

- The value SQL indicates a typical SQL block that does not return resultsets. For example, you can use insert, update, and delete statements but not select statements.
- The value Stored Procedure indicates a stored procedure name with its appropriate parameters.

The default value is SQL.

ContinueOnFail

Indicates whether the ETL execution will continue if an error occurs. Possible values are true and false.

- If the value is set to true, the execution will continue with the next statement (if any exist). If there are no additional statements, the task will be marked as Completed.
- If the value is set to false, the execution will stop and additional statements will not be executed. The task will be marked as Failed.

The default value is false.

dbConnType

Indicates the connection type against which the statement will be executed. Possible values are source, target, and both.

- If the value is set to source, the statement will be executed against the source connection.
- If the value is set to target, the statement will be executed against the target connection.
- If the value is set to both, the statement will be executed against the source and target connections.

If no value is specified for dbConnType, the statement will be executed against the target. If there is a @DAC_TABLE keyword (see"Runtime Substitution of Keywords") specified in the SQL statement, then this statement will be executed once for each source and target table.

If the table specified by the @DAC_TABLE keyword has a connection override (indicated in the Data Source field on the Source Tables and Target Tables subtabs of the Tasks tab), then the override connection will be used.

validDBPlatforms

Indicates the database platform types against which the SQL block can be executed. Possible values are DB2, DB2-390, MSSQL, Oracle, and Teradata. If this value is left empty or is not specified, then the SQL block is eligible to run against all database types.

You enter multiple values separated by a comma.

retries

Indicates how many times the DAC server will attempt to execute the statement if an error occurs. The value must be an integer. If no value is specified or a non-integer is specified, the value defaults to 1. If the statement is successfully executed after an error occurred, then the loop specified by this value will break.

Use of CDATA Section

SQL statements appear in CDATA sections, which allows for special characters (like <, >, \) to be used without breaking the XML structure. There should be only one SQL statement per CDATA section.

Runtime Substitution of Keywords

Keywords allow you to substitute values in the tags at runtime. You can use any of the source system parameters as keywords by placing the prefix @DAC_ before the parameter name.

For example, if you have a source system parameter called $$LAST_REFRESH_DTyou can refer to the parameter by using @DAC_$$LAST_REFRESH_DT.

Some common keywords are the following:

@DAC_TABLE

Depending on the value of the dbConnType tag, the substitution for this keyword is either the source or target tables of the task. The statement will be executed as many times as there are source or target tables.

@DAC_DATASOURCE_ID

The substitution for this keyword is the Data Source Number that appears in the Physical Data Sources tab.

@DAC_SYSDATE

The substitution for this keyword is the current timestamp.

@DAC_TBL_REFRESH_DATE

The substitution for this keyword is the refresh date of the connection name and table combination. This value will be substituted only if the @DAC_TABLE keyword is also used.

@DAC_CURRENT_PROCESS_ID

The substitution for this keyword is the Process ID, located in the Process ID column of the Current Run tab in the Execute view.

Example of XML Formatted File

Below is an example of an XML formatted file.

```
<!--
The following statement will be executed once on the target connection.

-->
<sql name="Insert Statement" type="SQL">
 <![CDATA[

 insert into test_trgt (id) values (999)
```

```
 ]]>
</sql>

<!--
The following stored procedure will be executed once per target table
on the table connection or task's target connection if table doesn't have
it's own defined.
Even if there is an error, the execution will continue.
Database platform type does not matter.
-->

<sql name="Stored Procedure1" type="Stored Procedure" continueOnError="true">
 <![CDATA[
 test_procedure ('abc', 2, '@DAC_TABLE')
 ]]>
</sql>

<!--
The following statement will be executed once per target table on the table
connection or task's target connection
if table doesn't have it's own defined.
The statement will be executed as many times as there are target tables.
The statement will be executed only if target table connection is DB2-390 or
Oracle
-->
<sql name="insert new row" type="SQL" dbConnType="target" continueOnFail="true"
validDBPlatforms="DB2-390, Oracle">
<![CDATA[
insert into @DAC_TABLE (INTEGRATION_ID, DATASOURCE_NUM_ID, ETL_PROC_WID) values
('1', @DAC_DATASOURCE_NUM_ID, @DAC_CURRENT_PROCESS_ID)
]]>
</sql>

<!--
The following statement will be executed once per source table on the table
connection
or task's source connection if table doesn't have it's own defined.
Database platform type does not matter.
-->
<sql name="UPDATE ETL_PROCESS_ID" type="SQL" dbConnType="source"
continueOnFail="true">
<![CDATA[
UPDATE @DAC_TABLE SET DATASOURCE_NUM_ID= @DAC_DATASOURCE_NUM_ID
]]>
</sql>

<!--
The following statement uses one of the DAC Source System Parameters.
Assume there was a variable defined as $$LAST_REFRESH_DATE that need to reflect
the last refresh date of
the target table, here is how the sql will look like.
Database platform type does not matter.
-->
<sql name="UPDATE ETL_PROCESS_ID SINCE LAST REFRESH" type="SQL"
dbConnType="target" continueOnFail="true">
<![CDATA[
UPDATE @DAC_TABLE SET DATASOURCE_NUM_ID= @DAC_DATASOURCE_NUM_ID
WHERE LAST_UPD = @DAC_$$LAST_REFRESH_DATE
]]>
</sql>
```

```
</CustomSQLs>
```

Plain Text SQL Files

Plain text SQL files consist of a set of SQL statements (no stored procedure calls). The SQL statements are separated by a semicolon (;), and comment tags are allowed (//, /* comment */, --). If any of the SQL statements fail, the Task Run status will be Failed.

An example of a plain text SQL file follows:

```
CREATE TABLE w_etl_temp (name varchar(50))
;
UPDATE w_etl_temp
SET name = 'that's right' //this line demonstrates the use of ' in a text area
WHERE name LIKE 'gone fishing%';

/*
*some
*query
*statement
*/
 SELECT * FROM w_etl_temp
;
DROP TABLE w_etl_temp
;
/*end of file*/
```

Overview of Change Capture Process (Siebel Sources Only)

This section describes the change capture process used to extract data from the Siebel transactional database. It includes the following topics:

Initial Data Capture

For each Siebel transactional source table (S_) from which data is extracted, there is one S_ETL_I_IMG_ table and one S_ETL_R_IMG_ table.

The first time a staging table is extracted, rows that are in the specified period are selected and inserted into the appropriate S_ETL_R_IMG_ table. The specified period is set in the Prune Days parameter, in the Execution Plans tab. For more information about the Prune Days parameter, see "Execution Plans Tab".

Change Capture Mechanisms

There are two kinds of change capture mechanisms used:

- Change capture using tables

  This is the most common method. It uses S_ETL_I_IMG_ and S_ETL_R_IMG_ table types and the LAST_UPD column of the source tables.

- Change capture using the date column

  In some cases, a predefined date column is used to enable change capture (without use of S_ETL_I_IMG_ and S_ETL_R_IMG_ tables).

Change Capture Using Tables

When S_ tables are extracted, the process looks for row ID information (the combination of ROW_ID and MODIFICATION_NUM from the S_ table) that does not exist in the row image table (S_ETL_R_IMG_) and in columns where the value for LAST_UPD is more recent than that of LAST_REFRESH_DATE minus the Prune Days parameter setting. This information is inserted into S_ETL_I_IMG table. The S_ETL_I_IMG_ table is joined with the base table during the SDE extraction process to extract only the change capture rows during refresh.

S_ETL_R_IMG tables store the ROW_ID, MODIFICATION_NUM, and LAST_UPD for the rows in the S_ table that have the LAST_UPD in the defined Prune Days period. The LAST_UPD column is used to delete the records from the S_ETL_R_IMG table. Records are deleted as soon as they go beyond the Prune Days period. This table is used to make sure that records that fall in the Prune Days period are not captured as updates unless they have actually been updated. This guarantees an efficient and faster change capture refresh. For information about tracking deleted records, see

Once the ETL process is completed, the data from the change capture image tables (S_ETL_I_IMG_) is pushed into the row image (S_ETL_R_IMG) table. The S_ETL_R_IMG_ information is subsequently used in the next refresh.

Although the LAST_UPD column in Siebel transactional tables is used for change capture, the timestamp reflects the time the data is committed in the database, rather than the actual transaction event time. This may happen because of remote synchronization, handheld synchronization, UTC conversion, and other processes in which there may be a significant lag between the time of a transaction and its commitment to the database. It is possible, therefore, for a data row to get committed to the transactional database with a LAST_UPD date that is older than the date on which last refresh was executed. Consequently, if the extract is based purely on the date in LAST_UPD, some rows might be missed during extraction.

The LAST_UPD date column, however, still permits change capture process efficiency by limiting the number of rows that have to be compared. The rows from transactional tables are filtered based on the LAST_UPD date being more recent than the LAST_REFRESH_DATE, minus the prune days. Then the ROW_ID and MODIFICATION_NUM combination is compared with the row image table to discover the changed records.

The Prune Days parameter ensures that the rows having LAST_UPD values older than LAST_REFRESH_DATE are not missed. This is a parameter that customers can set based on experience with processes (such as remote synchronization) that may cause records to be missed.

Primary and Auxiliary Tables The DAC performs change capture for both primary and auxiliary tables. When more than one source table is involved, then both the auxiliary and primary table records need to be marked as changed. For auxiliary tables, you need to write auxiliary mappings to mark the primary tables as changed. The SQL queries that do this are part of the mapping SDEINC_FindAux_.

The extract logic sometimes requires that rows be considered as changed, even if the record has not been updated in the primary table (and therefore extracted during the SDE process). This situation occurs when child table rows have changed and the header/master rows need to be extracted so that data warehouse tables are loaded with a consistent set.

When the S_CONTACT_X row is changed, the corresponding S_CONTACT also needs to be extracted. In this case, rows in S_CONTACT are also marked as changed by inserting rows in the change capture row image table.

When the S_ORDERITEM row is changed, the corresponding S_DOC_ORDER also needs to be extracted. In this case, rows in S_DOC_ORDER are also marked as changed (by inserting rows in the change capture row image table).

These auxiliary changes capture processes are heavily dependent on the data warehouse data model and are required to support the ETL logic.

Example: Building S_ETL_I_IMG_ Table for Loading Account Dimension This section gives an extended example of the process of change capture using tables.

1. Load image tables for all relevant source tables.

   The content of this entity comes from the S_ORG_EXT and S_ORG_EXT_X tables. Whenever any of the rows change in either of these tables, the record is marked as changed.

   The image table for S_ORG_EXT is S_ETL_I_IMG_26. The image table prefix can be found using the DAC to view any source table. This table is truncated before loading with fresh data during every refresh.

   During the ETL, process rows are inserted into S_ETL_I_IMG_26 by selecting ROW_ID information from S_ORG_EXT, for rows (combined ROW_ID and MODIFICATION_NUM) that do not exist in the S_ETL_R_IMG_26 and for which LAST_UPD is more recent than LAST_REFRESH_DATE minus the Prune Days setting. This is done during the ETL execution by the DAC's internal image building tasks.

   Similarly, the image table S_ETL_I_IMG_27 for S_ORG_EXT_X is loaded.

2. Load the image table for auxiliary table-based changes.

   In addition to the basic change capture, extra processing might be required due to special ETL requirements. In this example, it happens that S_ORG_EXT needs to be extracted for processing even if only S_ORG_EXT_X changes. This is because both the tables are joined to form W_ORG_D, and the extract process of W_ORG_D (a SDE mapping) looks for a changed ROW_ID in the change capture row image table for the primary table S_ORG_EXT only. Therefore, the extract happens only when the ROW_ID for S_ORG_EXT exists in the row image table.

   In this case, the SDEINC_FindAux_ mapping is needed to insert corresponding rows of S_ORG_EXT.ROW_ID in the change capture row image table whenever S_ORG_EXT_X changes. The following logical statement shows the method:

   Identify the records that have changed in the S_ORG_EXT_X (rows in S_ETLI_IMG_27) table and then find the corresponding rows in S_ORG_EXT. Insert the ROW_ID and MODIFICATION_NUM of those corresponding rows from S_ORG_EXT into S_ETL_I_IMG_26 table.

   Using Informatica, the auxiliary mapping SDEINC_FindAux_ has to be written for each primary table that requires it, depending on data warehouse extract logic. Using the DAC, this auxiliary task has to be linked as a parent to the extract mapping for the base table (S_ORG_EXT in this case).

   This is the SQL override for the SDEINC_FindAux Informatica mapping:

```
SELECT
   S_ORG_EXT.ROW_ID,
   S_ORG_EXT.MODIFICATION_NUM,
   S_ORG_EXT.LAST_UPD
FROM
   S_ORG_EXT,
   S_ORG_EXT_X,
```

```
   S_ETL_I_IMG_27 IMG
WHERE
   (
   IMG.ROW_ID = S_ORG_EXT_X.ROW_ID
   AND
   S_ORG_EXT_X.PAR_ROW_ID = S_ORG_EXT.ROW_ID
   )
   AND NOT EXISTS
   (  SELECT 'X'
      FROM
      S_ETL_I_IMG_26 IMG1
      WHERE
      IMG1.ROW_ID = S_ORG_EXT.ROW_ID
   )
```

3. Extract source table information using change capture image information.

   After the records that are new or modified are identified, those rows are loaded into the staging tables. The Informatica mappings that load the staging tables use the ROW_ID information captured in the image tables.

   This example shows the loading of staging table W_ORG_DS. The main logic of populating this table lies in the SQL override of the mapping SDE_ OrganizationDimension.

   The DAC creates views on tables that are being extracted. The views are different, depending on whether a table is extracted the first time or is a change capture extract.

   - If extracting for the first time, the view is created as `SELECT * FROM S_ ORG_EXT`.
   - If it is a change capture extract, the view is created as `SELECT * FROM S_ ORG_EXT, S_ETL_I_IMG_26 IMG WHERE S_ORG_EXT.ROW_ID = IMG.ROW_ID`.

   The SQL override in the mapping uses the view to extract the data.

```
SELECT
   S_ORG_EXT.ROW_ID,
   S_ORG_EXT.NAME, …..
   ….
   ….
FROM
   V_ORG_EXT,
   S_ORG_EXT_X,
   …..
WHERE
   {
      V_ORG_EXT S_ORG_EXT
      LEFT OUTER JOIN S_ORG_EXT_X ON
      S_ORG_EXT.ROW_ID = S_ORG_EXT_X.PAR_ROW_ID
      …..
   }
AND
   S_ORG_EXT.ROW_ID <> 'INT_COMPANY_ID'
```

Change Capture Using the Date Column

Forecasts are extracted without using the image table. The value S_FCSTSER_DATE is tracked using the date column ARCHIVE_TS. The administrator sets the ARCHIVE_ TS for forecasts that are submitted, frozen, and ready to be loaded into the Oracle

Business Analytics Warehouse. S_ETL_RUN stores the previous ETL date when forecasts were extracted and the current ETL date when the forecasts are being extracted. All forecasts with ARCHIVE_TS values greater than that of the previous ETL date and ARCHIVE_TS (less than the current ETL date) are extracted in the current ETL. Both ETL date and ARCHIVE_TS (less than the current ETL date) are stored in S_ETL_CURR_RUN.

Note: Forecasts in the Oracle Business Analytics Warehouse are never updated. Once loaded, they are frozen.

```
SELECT
…..
FROM
   S_FCSTSER_DATE,
   S_FCSTSER,
   S_ETL_CURR_RUN,
   ……..
WHERE
   S_FCSTSER_DATE.FCSTSER_ID =  S_FCSTSER.ROW_ID
   AND S_FCSTSER_DATE.ARCHIVE_TS > S_ETL_CURR_RUN.PREV_LOAD_DT
   AND S_FCSTSER_DATE.ARCHIVE_TS <= S_ETL_CURR_RUN.LOAD_DT
```

Using the Change Capture Filter

The change capture filter enables you to selectively load data from the Siebel transactional database into the data warehouse. You can set a condition in the ChangeCaptureFilter.xml file to filter data from specific tables. This file is located in the OracleBI\DAC\CustomSQLs directory. It provides an XML sample that you can copy and alter to fit your needs. Detailed instructions for using this feature are included at the top of the file.

Tracking Deleted Records

The Oracle Business Analytics Warehouse change capture process uses delete triggers to identify records for deletion on the Siebel transactional database. The deleted records are stored in S_ETL_D_IMG tables. During the change capture process, the DAC server moves the data from the S_ETL_D_IMG tables to the S_ETL_I_IMG tables, where D appears in the OPERATION column to show the records were deleted. During the change capture sync process, the records in the S_ETL_D_IMG tables that were moved to the S_ETL_I_IMG tables are flushed. In the DAC, you can view the SQL that runs during the change capture and change capture sync processes by using the Output to File right-click command in the Tasks tab of the Design view.

The preconfigured ETL process captures deleted records for the target tables W_ORG_D and W_PERSON_D, the source tables for which are S_ORG_EXT, S_CONTACT, and S_PRSP_CONTACT. These source tables need to have delete triggers created in the Siebel transactional database in order for deleted records to be tracked.

For vertical applications, the preconfigured ETL process captures deleted records for W_FUND_F and W_ALIGNMT_DH. You need to create delete triggers in the transactional database for the following additional tables: S_MDF_TXN, S_ASGN_GRP_POSTN, S_ASGN_RULE_ITEM.

In the Oracle Business Analytics Warehouse, preconfigured visibility tables are inactivated. If you activate visibility tables, you should also create delete triggers on the optional tables.

The preconfigured SIA Account and Contact visibility tables are activated by default for vertical applications. If your organization is not going to use any of the visibility tables, you need to inactivate them in the DAC.

On the target tables for which deleted records are tracked, a D appears in the INACTIVE_FLG column to show the records as deleted when the source records are deleted. This method of flagging a record as deleted is known as a **soft delete,** as compared to a **hard delete** when the record is physically deleted. When deleted records are tracked on visibility-related data warehouse tables, the records are physically deleted. The general rule is that soft deletes should be used for tables that are referenced by other tables. If a table is not referenced by any other table, then you can use hard deletes.

Aggregate tables are rebuilt during each ETL process. Therefore, records can be physically deleted from the base tables without consequence. If you want to use the soft delete method, you should consider changing the aggregate building mappings so that the deleted records are omitted.

Note: The Oracle BI Server does not recognize soft deletes. Therefore, you have to modify the .rpd file so that it does not pick up soft-deleted records for reporting.

To create delete triggers for preconfigured ETL change capture

1. From the DAC menu bar, select Tools, then select ETL Management, then select Configure.
2. In the Sources dialog, select the database platform for the target and transactional databases, and click OK.
3. In the Data Warehouse Configuration Wizard, select the Create Delete Triggers in Transaction Database check box, and click Next.

   The Delete Triggers tab is active.
4. Select one of the following:

Option	Description
Create Triggers	Executes the trigger statements directly.
Write Script to File	Writes the trigger statements to a file, which can be executed by a database administrator.

5. Select the database type as defined in the DAC.
6. For DB2 zSeries databases, enter the base table owner.
7. (Optional) Select the Include Optional Triggers check box to create triggers for the optional tables.
8. Click Start.

To create delete triggers for new source tables

1. In the DAC, navigate to the Design view, then select Tables.
2. Select the table for which you want to track deleted records.

   Make sure the table has an image suffix.

3. Right-click the table and select Change Capture Scripts, then select Generate Image and Trigger Scripts.
4. In the Triggers and Image Tables dialog, select the database type of the source database.
5. Make sure the Generate Image Table Scripts and Generate Trigger Script(s) options are selected.
6. Execute the script on the database.

To track deleted records

1. Make sure the delete triggers are enabled for the appropriate tables.
2. Write custom Informatica workflows with a clause WHERE operation = 'D' to the appropriate I_IMG table to take them across to the dimension and fact tables.
3. In the DAC, register the workflows as tasks.
4. Define the appropriate dependencies.

   For an example of such a workflow, see the preconfigured task SDE_ OrganizationDimension_LoadDeletedRows.

Pointing Multiple Informatica Servers to a Single Informatica Repository

You can install multiple Informatica servers and point them to a single Informatica Repository. You need to register each Informatica Server in the DAC and specify a unique machine name and server name. For instructions on registering an Informatica Server in the DAC, see the *Oracle Business Intelligence Applications Installation and Configuration Guide*.

Handling ETL Failures with the DAC

This section includes the following topics:

When the Execution of an Execution Plan Fails

When an execution plan is executed, 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 execution plan's status is Running. When all the tasks have been run, and if one or more tasks have failed, the execution plan's status is changed to Failed.

You can check the tasks that have failed in the Current Run tab of the Execute view, fix the problems, and then requeue the failed tasks by changing the status to Queued. You can then restart the ETL. All the tasks will then be rerun. You can also manually run a task, change its status to Completed, and then restart the ETL. Tasks with a Completed status are skipped.

Caution: The DAC server does not validate tasks that have been run manually.

To restart a failed ETL, click Run Now from the Current Run tab of the Execute View.

In Case of Abnormal Termination of the DAC Server

If the DAC server fails during the execution of the ETL, the status of the ETL execution will remain as Running. When the DAC server is started again, it will automatically run the ETL if the Auto Restart ETL DAC 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 ETL 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.

Discarding the Current Run 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 testing environment only, since 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 Designer, navigate to Mapping Designer, open the corresponding mapping, and in the Aggregator transformation, remove the check from the Sorted Input check box.

7

DAC Functional Reference

This chapter describes the functionality available in the Data Warehouse Administration Console (DAC). It contains the following topics:

Common Elements of Interface Tabs

Some of the DAC interface tabs have common elements, such as columns or subtabs. The common elements are described below.

Name

The Name column in a tab specifies the name of the database object.

Inactive

The Inactive column indicates whether a database object is inactive. 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.

Edit

The Edit subtab enables you to edit an object that is selected in the top window.

Description

The Description subtab displays and enables you to edit a description of the object selected in the top window.

Design View Tabs

The Design view provides access to functionality related to creating and managing subject areas. The tabs in this view are listed in alphabetical order.

- Indices 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, 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.

A configuration tag can function in one of the following ways:

- **Remove tasks from all subject areas**

  When you assign a task to a configuration tag, the task will not be eligible to participate in any subject area.

- **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.

- **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.

- **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 on creating configuration tags, see "Working with Configuration Tags".

Include Tasks

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

Use this subtab to view the subject areas that belong to a configuration tag or to add subject areas to a configuration tag.

Configuration Tag Tasks Only

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 the DAC when the subject area is assembled.

Configuration Tags Tab: Tasks Subtab

Use this subtab to add or remove tasks from the configuration tab selected in the top window.

For instructions, see "Working with Configuration Tags".

Indices Tab

The Indices tab lists all the indexes associated with the selected source system container. It is recommended that you do not register any indexes for source tables. During the ETL process, when a table is going to be truncated, all the indexes as defined in the repository will be dropped before the data is loaded and will be created after the data is loaded automatically. While this improves the ETL performance, the preconfigured workflows have the bulk load option turned on. The bulk load will fail if there are indexes on the table. Therefore, it is important to keep the index definitions in sync with the database. For example, if you create an index on the database, and it is not registered in the repository, the index will not be dropped and the load will fail.

For Teradata databases, only secondary indexes should be registered in the 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 the DAC.

Table Name

The table for which an index is created.

Index Usage

Specifies the index usage: ETL or Query. 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 of registering the index in the DAC.

Unique Columns

For unique indexes, the number of columns that will be unique.

Is Unique

Indicates whether the index is unique.

Is Clustered

Indicates whether the index is clustered. There can be only one clustered index per table.

Is Bitmap

Indicates whether the index is of the bitmap type.

Allow Reverse Scan

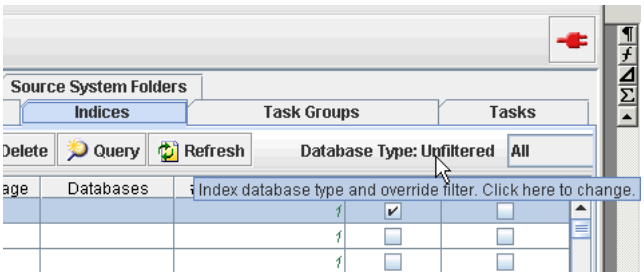
Applicable only for DB2-UDB databases. The index will be created with the Allow Reverse Scan option.

Always Drop & Create

Indicates whether the index will be dropped and created regardless of whether the table is being loaded using a full load or incremental load.

Database Filter

The database filter appears in the Indices tab to the right of the toolbar in the top window.



Click on the words "Database Type" to open the Index Filtering dialog. It enables you to filter the set of displayed indexes based on the database type. To display all indexes regardless of database type, select the option All.

About Advanced Custom Index Management

The DAC enables you to drop and recreate indexes during the load process, which reduces the overall load time during a full load. The DAC drops and recreates indexes based on the index definitions stored in the DAC metadata.

More advanced index management needs to be handled outside of the DAC, such as creating single-table and multi-table join indexes on Teradata databases. In such cases you can use DAC SQL tasks placed appropriately in the task dependencies, or you can use a pre-session or post-session script in Informatica.

If you need to modify the preconfigured indexes with extra options or syntax, you must inactivate them in the DAC metadata so that the DAC server does not try to drop and recreate them. You can then manage these indexes in the same manner as the advanced index management described above.

Indices Tab: Columns Subtab

The Columns subtab displays a list of columns the index is made of.

Position

The position of the column in the index.

Sort Order

Indicates whether the sort order is ascending or descending.

Indices Tab: Databases Subtab

The Databases subtab lists the database types that apply to the selected index. If no database type is indicated, the index will not be created.

Database Type

The type of database.

Index Override

Specifies an index space that overrides the default index space for the default database connection, which is specified in the Edit subtab of the Physical Data Sources tab in the Setup View.

Source System Folders Tab

The Source System Folders tab lists the Informatica folders associated with the selected source system container. It enables you to view existing folders and to create new ones.

Logical Folder

The name of the logical Informatica folder. This name is used in the task definition (in the Tasks tab) so that task definitions do not have to be cloned.

Physical Folder

The name of the physical Informatica folder. The physical Informatica folder corresponds to 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 parameters and to define new parameters.

For more information about managing parameters in the DAC, see:

- About Parameter Management
- Defining a Text Type Parameter
- Defining a Database Specific Text Type Parameter
- Defining a Timestamp Type Parameter
- Defining a SQL Type Parameter

Data Type

Parameter data type. For more information, see "Overview of Parameters".

Possible values are the following:

Data Type Option	Description
Text	The value for the parameter is defined as text.
DB Specific Text	Enables you to add database specific hints in Informatica mappings. When you select this option, in the Value field, you specify the logical connection where the parameter applies, and you specify a default value for the parameter. The DAC evaluates the parameter to this default value for all databases. If you enter text that is applicable to a particular database, you can specify a value for the database type, and the DAC will evaluate the parameter to this value at runtime as long as the logical connection matches the specified database.
Timestamp	The value for the parameter is a timestamp and can be static, runtime or SQL.
SQL	The value for the parameter is a SQL statement.

Value

The parameter value.

Subject Areas Tab

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.

Configuration Tag Tasks Only

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 the DAC when the subject area is assembled.

For more information, see "Working with Configuration Tags" and "Configuration Tags Tab".

Last Designed

Date and time the subject area was last assembled.

Subject Areas Tab: Configuration Tags Subtab

The Configuration Tags subtab lists the configuration tags that are associated with this subject area.

For more information, see "Working with Configuration Tags" and "Configuration Tags Tab".

Include Tasks

This read-only field indicates whether the configuration tag tasks will be executed.

Context Disabled

When this read-only check box is selected, the configuration tag is globally disabled.

Subject Areas Tab: Tables Subtab

The 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.

Subject Areas Tab: Tasks Subtab

The Tasks subtab lists the tasks associated with the selected subject area, and enables you to add tasks to a subject area, inactivate tasks, and remove tasks from a subject area.

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.

Task Name

Name of the task.

Parent Group

If the task belongs to a task group, this column displays the task group name.

Phase

Task phase of the ETL process.

Autogenerated

Indicates whether the task was automatically generated by the DAC's task generation process.

Is Group

Indicates whether the task is a task group.

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.

Tables Tab

The Tables tab lists all the tables associated with the selected source system container. It enables you to view and edit existing tables and to create new ones.

Table Type

Indicates the type of table.

Warehouse

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

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

Indicates whether the table is a MultiSet table. Applicable only to Teradata databases.

Has Unique Primary Index

Indicates whether the table has a Unique Primary Index. Applicable only to Teradata databases.

Tables Tab: Conditional for Tasks (RO)

The Conditional for Tasks (RO) subtab displays a read-only list of tasks that are optional tasks for the selected table.

Build Image

Applicable for Siebel transactional sources only. Indicates the change capture for the primary/auxiliary source tables will be executed.

Tables Tab: Indices (RO)

The Indices (RO) subtab displays a read-only list of indexes that belong to the selected table.

Index Usage

Specifies the index usage: ETL or Query. 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 of registering the index.

Unique Columns

For unique indexes, the number of columns that will be unique.

Is Unique

Indicates whether the index is unique.

Is Clustered

Indicates whether the index is clustered. There can be only one clustered index per table.

Is Bitmap

Indicates whether the index is of the bitmap type.

Allow Reverse Scan

Applicable only for DB2-UDB databases. The index will be created with the Allow Reverse Scan option.

Table Space Name

Name of the table space.

Tables Tab: Multi-Column Statistics Subtab

Applicable to Teradata databases only.

Table Tab: 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.

Table Type

Type of 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.

Build Image

Applicable for Siebel transactional sources only. Indicates the change

capture for the primary/auxiliary source tables will be executed

Type

Type of table.

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.

Build Image

Applicable for Siebel transactional sources only. Indicates the change

capture for the primary/auxiliary source tables will be executed.

Type

Type of table.

Task Groups Tab

The Task Groups tab lists all the task groups associated with the selected source system container. A task can belong to only one group.

Restart All on Failure

Indicates the tasks in this task group will be restarted if one or more tasks fails during an ETL process.

Execute Serially

Indicates the tasks in this task group will be executed sequentially. This property overrides the execution order.

Truncate Always

Indicates the target tables are 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 are 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

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 are 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.

Execution Order

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, the DAC will run the tasks in parallel.

Task Groups Tab: Source Tables (RO) Subtab

The Source Tables (RO) subtab lists the tables used for getting data by the selected task group.

Table

Name of source table.

Task

Task that extracts data from the table.

Type

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

Name of the target table.

Task

Task that loads data into the target table.

Type

Type of target table.

Tasks Tab

The Tasks tab lists all the tasks associated with the selected source system container.

Parent Group

If the task is a member of a group, this field lists the task group name.

Command for Incremental Load

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.

The DAC maintains a last refresh timestamp whenever a table is changed during the ETL process. (You can view this timestamp by selecting Setup, then selecting Physical Data Sources, and then selecting Refresh Dates.) 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

If a table has no last refresh timestamp, this command is executed.

Folder Name

Only for execution type of Informatica. The folder in which the workflow resides. Note: The name cannot contain spaces.

Primary Source

Logical database connection for the primary source database.

Primary Target

Logical database connection for the primary target database.

Task Phase

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

Tasks are executed based on their execution type. The following types are supported:

- **Informatica**

  Task is invoked on an Informatica Server 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 databases.

In addition, there are several internal execution types that you will not be able to select when creating new tasks. These tasks 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, go to Design > Tables, 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.

- **QUERY_INDEX**

  Used for internal data warehouse. This task enables you to alter when the Query indexes are created. The DAC server drops all indexes before loading when the CreateQueyIndexesAtTheEnd setting is set to True. When this setting is set to False, all the indexes, regardless of the index type, get created as part of the task that does the loading.

- **UPDATE_ETL_PARAM**

  Used for internal data warehouse. This task is used only to update W_PARAM_G from the DAC server. This task populates the system properties to the W_ PARAM_G table in the data warehouse by querying values defined in the DAC repository. Because only one data warehouse per DAC repository is supported, this execution type should not be chosen for any task.

Priority

Indicates the order in which the task is executed. If two or more tasks have the same priority, the order occurs randomly.

Pre-SQL for Full Load

The SQL script (derived from a SQL or XML file) that is executed before the specified task when the task is participating in a full load.

Pre-SQL for Incremental Load

The SQL script (derived from a SQL or XML file) that is executed before the specified task when the task is participating in an incremental load.

Post-SQL for Full Load

The SQL script (derived from a SQL or XML file) that is executed after the specified task when the specified task is participating in a full load.

Post-SQL for Incremental Load

The SQL script (derived from a SQL or XML file) that is executed after the specified task when the specified task is participating in an incremental load.

Build Image

Applicable for Siebel transactional sources only. Indicates the change capture for the primary/auxiliary source tables will be executed.

Analyze Tables

The DAC automatically analyzes tables when tasks truncate tables. By selecting this check box, however, you can force the DAC to analyze tables even when they are not truncated.

Continue on Error

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: Conditional Tables Subtab

The Conditional Tables subtab lists the tables that, if included in an execution plan, cause the optional task selected in the top window 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

The 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.

Include Tasks

This read-only field indicates whether the configuration tag tasks will be executed.

Context Disabled

If this check box is selected, the configuration tag is globally disabled.

Tasks Tab: Parameters Subtab

The Parameters subtab lists the parameters associated with the selected task. It 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 managing parameters in the DAC, see:

About Parameter Management.

Defining a Text Type Parameter

Defining a Database Specific Text Type Parameter

Defining a Timestamp Type Parameter

Defining a SQL Type Parameter

Name

Name of the parameter.

Data Type

Parameter data type. For more information, see "Overview of Parameters".

Possible values are the following:

- **Text**

  The value for the parameter is defined as text.

- **DB Specific Text**

  Enables you to add database specific hints in Informatica mappings.

  When you select this option, in the Value field, you specify the logical connection where the parameter applies, and you specify a default value for the parameter. The DAC evaluates the parameter to this default value for all databases. If you enter text that is applicable to a particular database, you can specify a value for the database type, and the DAC will evaluate the parameter to this value at runtime as long as the logical connection matches the specified database.

- **Timestamp**

  The value for the parameter is a timestamp and can be static, runtime or SQL.

- **SQL**

  The value for the parameter is a SQL statement.

Data Type

The parameter data type. Possible values are Text, Timestamp, and SQL.

Value

The parameter value.

Task Tab: Phase Dependency Subtab

The DAC server uses the ETL phase property to prioritize tasks. By changing the phase property of a task, you change the task's execution order.

Action

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

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.

Phase

The ETL phase that will apply to the Action and Grain properties.

Tasks Tab: Source Tables Subtab

The Source Tables subtab lists the tables from which the selected task extracts data.

Type

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

Data source for the table. When a data source is not specified, the default is the task's primary source.

Tasks Tab: Target Tables Subtab

The Target Tables subtab lists the tables into which the selected task loads data.

Type

Table type.

Data Source

Data source for the target table. If no data source is specified, this value defaults to the task's primary target.

Truncate Always

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 are 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

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 are 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.

Setup View Tabs

The Setup View provides access to functionality related to setting up DAC system properties, Informatica servers, database connections, and email notification. The tabs in this view are listed in alphabetical order.

- DAC System Properties Tab
- Email Recipients Tab
- Informatica Servers Tab
- Physical Data Sources Tab

DAC System Properties Tab

The DAC System Properties tab enables you to configure various properties that determine the behavior of the DAC server.

Analyze Frequency (in days)

For DAC metadata tables, the frequency (in days) the DAC client automatically updates the table and index statistics for the DAC repository. The value must be numerical.

Analyze Tables

The DAC automatically analyzes tables when tasks truncate tables. By selecting this check box, however, you can force the DAC to analyze tables even when they are not truncated.

Auto Restart ETL

Possible values are True and False.

When set to True: An ETL that is running when the DAC server abnormally terminates will continue running when the DAC server is restarted.

When set to False: An ETL 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.

CreateQueryIndexesAtTheEnd

Possible values are True and False.

During the ETL process, the DAC server automatically drops and creates indexes. When set to True, this property groups all indexes of the Query type and creates them after the ETL is complete.

The DropAndCreateIndexes property takes precedence over this property. Therefore, if the DropAndCreateIndexes property is set to False, you cannot set the property CreateQueryIndexesAtTheEnd to True to have indexes of the Query type created at the end of the ETL process.

Also, be aware that when this property is set to True, tables will be analyzed twice. If any indexes are marked as Query type indexes, and are used by ETL processes, it can adversely affect the performance of the ETL process.

DAC Alternate Server Hosts

Host name of the machine where the alternate DAC server resides. The alternate DAC server is used for failover purposes. The DAC client cannot talk to the alternate server unless the main DAC server is not running.

DAC Server Host

Host name of the machine where the DAC server resides. You cannot use an IP address for this property.

The DAC server and a given DAC repository have a one-to-one mapping. That is, you can only run one DAC server against any given DAC repository. Thus, in the repository you must specify the network host name of the machine where the DAC sever is to be run.

This property also takes the value localhost. However, this value is provided for development and testing purposes and should not be used in a production environment.

DAC Server OS

Operating system of the machine where the DAC server resides. Possible values are Windows, Solaris, HP, or AIX.

If you move the DAC server from another operating system to AIX, you need to do the following: change the DAC server host to the appropriate value; restart the DAC client; reenter all the password fields for the Informatica servers and database connections; and reconfigure the DAC server on the AIX machine by running serverSetupPrompt.sh.

DAC Server Port

Network port to which the DAC server binds in order to listen to client requests. The default value is 3141. If this port has been assigned to another process, you can enter any numerical port value greater than 1024.

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.

Setting this property to True can create system catalog lock up for DB2-UDB and DB2-390 databases. Therefore, by setting the property 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.

DropAndCreateIndexes

Possible values are True and False. Indicates whether, during the ETL process, the DAC server automatically drops and creates indexes.

This property takes precedence over the CreateQueryIndexesAtTheEnd. Therefore, if the DropAndCreateIndexes property is set to False, you cannot set the property CreateQueryIndexesAtTheEnd to True to have indexes of the Query type created at the end of the ETL process.

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 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 preconfigured tasks.

HeartBeatInterval

Frequency (in seconds) the DAC server checks on the health of the database connections. The value must be numerical. For example, a value of 300 (the default value) indicates the system will perform subsystem diagnostics and recovery procedures every 300 seconds.

InformaticaFileParameterLocation

Directory where the Informatica parameter file is stored.

Output Redirect

Indicates whether logging information and standard output and errors are redirected to files in the log directory (when property is set to True). 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 server is 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 server will maintain.

Repository Name

Unique name for the DAC repository.

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.

For more information, see the description of the property Script Before Every ETL.

Script Before Every ETL

The name of the script or executable to be run before every execution plan.

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. This file should be placed in the scripts subdirectory of the DAC server.

The execution plan runs only after the external process has finished. Therefore, it is important that the script or executable does not fail.

Server Log Level

Output logging level. Possible values are Finest, Finer, Fine, Config, Info, Warning, and Severe. The Severe value produces minimal log details, and Finest produces the most extensive amount of reporting.

SQL Trace

Possible values are True and False.

Indicates whether the SQL statements to the DAC repository and database connections are added to the log file. Possible values are True and False. The True value sends a hint 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.

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.

Indicates whether indexes defined in the DAC repository will be automatically created in the data warehouse database during an incremental load.

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.

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.

Name

Logical name of the user to be notified.

Email Address

Email address where the notification is sent.

Notification Level

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.

Informatica Servers Tab

The Informatica Servers tab enables you to register one or more Informatica servers and one Informatica Repository server and to specify how many workflows can be executed in parallel on each server. The DAC server automatically load balances across the servers.

> **Note:** You can install multiple Informatica servers and point them to a single Informatica Repository. You need to register each Informatica Server in the DAC and specify a unique machine name and server name. For instructions on registering an Informatica Server in the DAC, see the *Oracle Business Intelligence Applications Installation and Configuration Guide*.

Name

Name of Informatica Server or Informatica Repository Server.

Type

Type of server.

- **Informatica**

  Specifies the Informatica Server.

- **Repository**

  Specifies the Informatica Repository Server.

Server Hostname

The host machine name where the Informatica Server or Informatica Repository Server is installed.

Server Port

Port number used by the Informatica Server or Informatica Repository Server to listen to requests.

Login

Informatica Repository user login.

Password

Informatica Repository password.

Maximum Sessions

The maximum number of workflows that can be executed in parallel on the Informatica Server.

Repository Name

Informatica Repository name.

You deploy only one Informatica Repository Server, but you can deploy multiple Informatica Servers.

Physical Data Sources Tab

The Physical Data Sources tab provides access to the connection properties for the physical data sources. In this tab, you can view and edit existing physical data source connections and create new ones.

Name

Logical name for the physical data source.

Type

Physical data source type. Possible values are the following:

- **Source**
- **Warehouse**
- **Informatica Repository**
- **DAC Repository**
- **Other**

Connection Type

Type of database connection. Possible values are the following:

- **Oracle (OCI8)**

  Connects to Oracle using the tnsnames entry.

- **Oracle (Thin)**

  Connects to Oracle using thin driver.

- **DB2**

  DB2 UDB database.

- **DB2-390**

  DB2 390 database.

- **MSSQL**

  Microsoft SQL Server database.

- **Teradata**

  Teradata database.

- **Flat File**

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.

Table Owner
Name of the table owner.

Max Num Connections
Maximum number of database connections this connection pool can contain.

DBHost
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
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.

Priority
User-defined priority of the data source.

Data Source Number
User-defined number of the data source.

Default Index Space
Applicable to Oracle databases only. The default index space for the physical data source. When indexes are dropped and created, they are created in this index space.

Parallel Index Creation
Use this field to specify how many indexes are to be created in parallel. For example, if a table with thirty 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 Max Num Connections property.

You must select the Parallel Tables Indexes check box in order to specify a number in the Parallel Index Creation field.

Parallel Table Indexes
Use this check box to indicate that indexes within a table are to be created in parallel.

Note: All indexes are dropped in serial order.

Physical Data Sources Tab: Index Spaces Subtab

The Index Spaces subtab allows you to specify tablespaces for indexes by table type. For instructions, see "Specifying Tablespaces for Indexes by Table Type".

Table Type
Table type for which you want to specify a tablespace.

Index Space
Specifies the name of the index space.

Note: You must create the index space on the database before you specify an index space in the DAC.

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. The 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.)

Note: Refresh dates for micro ETL processes are captured in the Refresh Dates subtab of the Execution Plans tab.

Name

Name of source or target table.

Execution Plan

The name of the execution plan to which the source or target table belongs.

Refresh Date

The refresh date for the source or target table.

Number of Rows

Valid for target tables only. Indicates the total number of rows in the table after the table has been loaded.

Execute View Tabs

The Execute View provides access to functionality that enables you to run, schedule, and monitor execution plans. The tabs in this view are listed in alphabetical order.

- Current Run Tab
- Execution Plans Tab
- Run History Tab
- Scheduler Tab

Current Run Tab

The Current Run tab displays a list of queued, running, and failed current ETL processes in the top window. This list includes comprehensive information about each process. Once an ETL process completes, it is accessible from the Run History tab.

Execution Plan Name

The execution plan whose runtime instance is this record. This field is read only.

Run Status

The status of the run. The possible values are the following.

Value	Description
Queued	Tasks for which the Depends On tasks are not yet completed. Displayed in yellow in the Current Run list.
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. Displayed in blue.
Paused	Task group members that are waiting for the other tasks in the group to be executed.
Failed	Tasks that were executed but encountered a problem. Displayed in red.
Stopped	Tasks for which one or more Depends On tasks have failed.
Completed	All tasks have completed without errors. Displayed in green.

Start Timestamp

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

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

A calculated field that shows the difference between start and end time stamps.

Status Description

Displays messages generated during run time. You can add notes to this field for Completed runs.

Process ID

ID for the process. This value is an integer that is incremented by 1 for every run. This value is stored as ETL_PROC_WID in all the data warehouse tables. This field is read-only.

Total Number of Tasks

The total number of tasks for this run. This field is read only.

Number of Failed Tasks
The sum total of tasks that have failed and that have stopped. This field is read only.

Number of Successful Tasks
The number of tasks whose status is Completed. This field is read only.

Number of Tasks Still in Queue
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
The name of the scheduled ETL process.

Current Run 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
The date the selected run was last updated.

Start Timestamp
Start time of the selected run.

End Timestamp
End time of the selected run.

Duration
The difference between the start timestamp and the end timestamp of the selected run.

Status
Status of the selected run.

Current Run Tab: Summary (RO) Subtab
The Summary (RO) subtab provides a summary (based on dynamic SQL) of the selected ETL run.

Task Phase
The task phase of the selected ETL run.

Start Time
Start time of the selected ETL run.

End Time
End time of the selected ETL run.

Duration
The difference between the start timestamp and the end timestamp of the selected ETL run.

Source System

The name of the source system container associated with the selected ETL run.

Current Run 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.

Current Run 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 Plans Tab

The Execution Plans tab enables you to view and edit existing execution plans and to create new ones.

Full Load Always

Indicates the specified ETL process will always execute a full load.

Keep Separate Refresh Dates

Used for micro ETL processes. Indicates refresh dates are kept separate for each ETL run of the execution plan.

Prune Days

When the source system is Oracle's Siebel CRM applications, the LAST_UPD column in the transactional database tables is used for incremental change capture. This timestamp reflects the actual event time. It is therefore possible for a data row to be committed to the transactional database with a LAST_UPD date that is older than the date on which the last refresh was executed. This will cause the data row to be missed in the subsequent extract (if based purely on LAST_UPD date).

However, the LAST_UPD date column still provides an opportunity to improve the change capture process by overlapping the extraction date window by the number of days set in this parameter. The records extracted in the overlapped window are filtered by comparing this information with information in the Image table.

The Prune Days setting ensures that the rows that had values in LAST_UPD older than values in LAST_REFRESH_DATE are not missed. This is a parameter that can be set based on experience with processes, such as remote sync, that potentially can cause records to be missed. This parameter cannot be less than 1.

For example: Assume the table W_PERSON_D was refreshed on January 15th by querying the table S_CONTACT. And, the Prune Days setting was set to 5. The next time S_CONTACT is queried to load W_PERSON_D, the records that have a LAST_UPD value since January 10 are compared with the ROW_ID of the Image table to cover for any missing records between January 15 and January 10 (the overlap period).

For source systems other than Siebel, the Prune Days setting is used in the same way except that the DAC subtracts the number of prune days from the LAST_REFRESH_DATE of a given source and supplies this as the value for the $$LAST_EXTRACT_DATE parameter.

Last Designed

Date this execution plan was last designed.

Analyze

Indicates the tables associated with this execution plan will be analyzed.

Analyze Truncated Tables Only

Indicates only truncated tables will be analyzed.

Drop/Create Indices

Indicates indexes of the tables associated with this execution plan will be dropped and created.

Run Now Button

The Run Now button submits a request to the DAC server to execute the execution plan.

Build Button

Builds the execution plan, by assembling subject areas, tasks, task phases, indices, tags, parameters, and source system folders.

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. For a description of the columns, see "Execution Plans Tab: Immediate Dependencies Subtab".

Execution Plans Tab: Following Tasks Subtab

The Following Tasks subtab lists the tasks that must be completed after an ETL is executed. Also enables you to add tasks.

It includes the same properties as the Preceding Tasks subtab.

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.

Task (Calculated)

Shows the source and target of the named task.

Predecessor Name

Predecessor task for the named task.

Predecessor (Calculated)

Shows the source and target of the predecessor 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.

Primary Source

Primary source table from which the task extracts data.

Primary Target

Primary target table into which data is loaded.

Folder Name

Name of the Informatica folder in which the task resides.

Task Phase

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.

Command
Command associated with the task.

Source System
Source system container from which the task extracts data.

Details Button
The Details button in the subtab toolbar opens a dialog that lists the following details about a selected task:

- All Predecessors
- All Successors
- Immediate Predecessors
- Immediate Successors
- Source Tables
- Target Tables
- Conditional Tables

Execution Plans Tab: Parameters Subtab
The Parameters subtab lists the parameters of the selected execution plan for database connections and Informatica folders.

Type
Possible values are the following:

- **Folder**

  Indicates an Informatica folder.

- **Datasource**

  Indicates a database connection parameter.

Name
Logical name of the folder or database connection.

Value
Physical name of the folder or database connection.

Source System
Name of the source system associated with the parameter.

Execution Plans Tab: Preceding Tasks Subtab
The Preceding Tasks subtab lists the tasks that must be completed before an ETL process is executed. It also enables you to add preceding tasks.

Name
Name of task.

Priority

Indicates the order in which the task is executed. If two or more tasks have the same priority, the DAC will execute them in parallel.

Command

Command associated with the task.

Source System

Source system container from which the task extracts data.

Execution Plans Tab: Refresh Dates Subtab

Applies to micro ETL execution plans (indicated by selecting the Keep Separate Refresh Dates check box in the Execution Plans tab).

Connection

Logical name for the database connection.

Refresh Dates

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 the selected execution plan.

Subject Area

Name of the subject area associated with the execution plan.

Source System

The source system container associated with the subject area.

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 Run tab. For a description of the information in the Run History tab, see "Current Run Tab".

Scheduler Tab

The Scheduler tab enables you to schedule ETL processes to be executed either once at a later time 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.

The top window of the Scheduler tab lists ETL runs that have been scheduled. The bottom window enables you to schedule an ETL run.

Execution Plan

The name of the scheduled execution plan.

Last Schedule Status

The last run status of the scheduled ETL process. Possible values are Running, Completed or Stopped.

Next Trigger

Time the scheduled ETL run will next be executed.

Status Description

Description of the last ETL run. Possible values are Running, Completed, or the reason the process stopped.

Recurrence

Indicates how often the schedule will be executed.

Index

C

D

E

F

I

O

P

Q

R

S

T