Oracle® Business Intelligence

Data Warehouse Administration Console User's Guide

Version 10.1.3.4

E12652-02

October 2012

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

ORACLE®

Oracle Business Intelligence Data Warehouse Administration Console User's Guide Version 10.1.3.4

E12652-02

Primary Author: Jill Arehart

Contributing Author: Sam Myers

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

Contents

Preface xi

Audience xi
Documentation Accessibility xi
Related Documents xi
Conventions xii

1 What's New in This Release

What's New in Oracle Business Intelligence Data Warehouse Administration Console User's Guide, Release 10.1.3.4 1-1
What's New in Oracle Business Intelligence Data Warehouse Administration Console User's Guide, Release 10.1.3.4.1 1-2

2 Overview of Oracle Business Analytics Warehouse

Oracle Business Analytics Warehouse Overview 2-1
Oracle Business Analytics Warehouse Architecture 2-1
Oracle Business Analytics Warehouse Architecture Components 2-2

3 Basic Concepts About DAC

Introduction to DAC 3-1
Important DAC Features 3-1
About the DAC Process Life Cycle 3-3
About Source System Containers 3-4
About DAC Repository Objects Held in Source System Containers 3-4
About Object Ownership in DAC 3-5
About DAC Metadata Export Behavior 3-6

4 DAC Quick Start

About User Account Management 4-1
Creating, Deleting and Inactivating User Accounts 4-2
Logging into DAC for the First Time 4-3
About the DAC's Extract and Load Processes 4-5
Performing an Initial Full Load of Data into the Data Warehouse 4-6

5 Best Practices for Defining Repository Objects

Best Practices for Containers ... 5-1
Best Practices for Tasks ... 5-2
Best Practices for Task Groups ... 5-3
Best Practices for Tables ... 5-3
Best Practices for Indexes ... 5-3
Best Practices for Columns ... 5-4
Best Practices for Configuration Tags ... 5-5
Best Practices for Source System Parameters ... 5-5
Best Practices for Subject Areas ... 5-5
Best Practices for Execution Plans ... 5-5

6 Overview of the DAC Interface

Navigating the DAC Interface ... 6-1
The DAC Menu Bar ... 6-2
Tools Menu Options ... 6-3
The DAC Views ... 6-5
The DAC Top Pane Toolbar ... 6-5
The DAC Right-Click Menus ... 6-6
Common Right-Click Menu Commands ... 6-6
Design View Right-Click Menu Commands ... 6-7
Setup View Right-Click Menu Commands ... 6-9
Execute View Right-Click Menu Commands ... 6-9
The DAC Server Monitor Icons ... 6-9
The DAC Navigation Tree ... 6-10
The DAC Editable Lists ... 6-11
Font Variations of Objects Displayed in the DAC ... 6-11
Using the DAC Query Functionality ... 6-11
DAC Query Commands and Operators ... 6-12
DAC Query Examples ... 6-12
Common DAC Query Procedures ... 6-13
Using Flat Views Querying ... 6-13

7 About Index, Table and Task Actions

Overview of Index, Table and Task Actions ... 7-1
Defining a SQL Script for an Action ... 7-2
Assigning an Action to a Repository Object ... 7-5
Functions for Use with Actions ... 7-5
Example of How to Use a DAC Source System Parameter in an Action ... 7-7

8 Customizing DAC Objects and Designing Subject Areas

Creating or Copying a Source System Container ... 8-1
About Customizing the Data Warehouse ... 8-2
Adding a New Table and Columns to the Data Warehouse ... 8-3
Adding an Index to the Data Warehouse ... 8-5
Importing New Data Warehouse Objects into the Informatica Repository ... 8-5

Creating Informatica Mappings and Workflows 8-6
Creating Tasks in the DAC for New or Modified Informatica Workflows 8-6
Setting a Task Phase Dependency 8-7
Creating a Task Group 8-7
About Parameter Management 8-8
Overview of Parameters 8-8
Parameter Data Types 8-8
Preconfigured Parameters 8-9
How DAC Handles Parameters at Runtime 8-9
Nesting Parameters within Other Parameters 8-10
Defining a Text Type Parameter 8-10
Defining a Database Specific Text Type Parameter 8-11
Defining a Timestamp Type Parameter 8-11
Defining a SQL Type Parameter 8-12
Specifying Tablespaces for Indexes by Table Type 8-13
Working with Configuration Tags 8-13
Overview of Subject Areas 8-17
Designing a Subject Area 8-17
Previewing and Pruning Subject Areas 8-17
How DAC Determines Tasks Required for Subject Areas 8-17
Creating a Subject Area 8-18

9 Building, Running and Monitoring Execution Plans

Types of Execution Plans 9-1
About Single-Source Execution Plans 9-1
About Multi-Source Execution Plans 9-2
Multi-Source Order of Execution 9-2
Delay 9-2
Folder Level Priorities 9-2
Common Extract and Load Scenarios 9-3
Single Extract and Single Load Scenario 9-3
Truncate Table Behavior in a Single Extract Scenario 9-3
Multiple Extract and Single Load Scenario 9-4
Truncate Table Behavior in a Multiple Extract Scenario 9-4
Multiple Extract and Multiple Load Scenario 9-5
Best Practices for Multi-Source Execution Plans 9-5
How DAC Determines the Order of Task Execution within an Execution Plan 9-7
Building and Running Single-Source and Multi-Source Execution Plans 9-8
Unit Testing Execution Plan Tasks 9-10
Building and Running Micro ETL Execution Plans 9-10
Scheduling an Execution Plan 9-13
About Refresh Dates 9-13
Monitoring Execution Plan Processes 9-14

10 Upgrading, Comparing and Merging DAC Repositories

Overview of Upgrade/Merge Wizard 10-1

Major Stages of the Upgrade/Merge Wizard 10-1
Resetting the Upgrade or Merge Process 10-2
Overview of Upgrade and Merge Options 10-3
About the Repository Upgrade (DAC 784) Option 10-3
Repository Upgrade (784): High-Level Process Flow 10-4
Repository Upgrade (784): Procedure for Upgrading 10-5
About the Refresh Base Option 10-7
Refresh Base: High-Level Process Flow 10-8
Refresh Base: Procedure for Upgrading 10-9
About the Simplified Refresh From Base Option 10-10
About the Replace Base Option 10-11
Replace Base: High-Level Process Flow 10-12
Replace Base: Procedure for Upgrading 10-13
About the Peer to Peer Merge Option 10-14
Peer to Peer Merge: High-Level Process Flow 10-15
Peer to Peer Merge: Procedure for Merging 10-15
Resolving Object Differences in the View Difference Report 10-17
Overview of View Difference Report 10-17
View Difference Report Interface 10-18
Possible Repository Merge Outcomes Based on Your Decisions 10-19

11 Common Tasks Performed in the DAC

Importing DAC Metadata 11-2
Exporting DAC Metadata 11-2
Distributing DAC Metadata 11-3
Running the DAC Server Automatically 11-3
Command Line Access to the DAC Server 11-4
Command Line Operations 11-4
Starting an Execution Plan 11-4
Stopping the Operation of a Running Execution Plan 11-5
Command Line Status Monitoring Queries 11-5
Setting Up Command Line Access to the DAC Server 11-5
Using the Command Line to Access the DAC Server 11-6
DAC Repository Command Line Options 11-7
Import DAC Metadata by Application 11-7
Export DAC Metadata by Application 11-7
Import DAC Metadata by Categories 11-8
Export DAC Metadata by Categories 11-8
Create Schema 11-8
Drop Schema 11-9
Analyze 11-9
Upgrade 11-9
Set Password 11-9
Replacing an Informatica Workflow with a Custom SQL File 11-9
Determining the Informatica Server Maximum Sessions Parameter Setting 11-10
Determining the Number of Transactional and Data Warehouse Database Connections 11-11
Running Two DAC Servers on the Same Machine 11-11

Customizing Index and Analyze Table Syntaxes ... 11-12
Overview of Change Capture Process (Siebel Sources Only) ... 11-13
Initial Data Capture ... 11-13
Change Capture Mechanisms ... 11-13
Change Capture Using Tables ... 11-13
Primary and Auxiliary Tables ... 11-14
Example: Building S_ETL_I_IMG_ Table for Loading Account Dimension ... 11-14
Change Capture Using the Date Column ... 11-16
Using the Change Capture Filter ... 11-17
Tracking Deleted Records ... 11-17
Pointing Multiple Informatica PowerCenter Services to a Single Informatica Repository ... 11-19
Handling ETL Failures with the DAC ... 11-19
When the Execution of an Execution Plan Fails ... 11-19
In Case of Abnormal Termination of the DAC Server ... 11-19
Discarding the Current Run Execution Plan ... 11-20
Failure of Aggregator Transformation Tasks with Sorted Input ... 11-20
Forcing Password Encryption When DAC Interacts With Informatica ... 11-20
Upgrading the Data Warehouse Schema on Teradata ... 11-21
Using JDBC Connection URL Override to Handle Database Connections ... 11-22
Using Parameters to Specify Full or Incremental Loads ... 11-22
Mapping Multiple Database-Specific Informatica Workflows to the Same DAC Task ... 11-23
Connecting to the DAC Repository When Using Oracle RAC ... 11-24

12 DAC Functional Reference

Common Elements of Interface Tabs ... 12-2
Design View Tabs ... 12-3
Configuration Tags Tab ... 12-4
Configuration Tags Tab: Subject Areas Subtab ... 12-4
Configuration Tags Tab: Tasks Subtab ... 12-5
Indices Tab ... 12-6
Indices Tab: Actions Subtab ... 12-7
Indices Tab: Columns Subtab ... 12-7
Source System Folders Tab ... 12-8
Source System Parameters Tab ... 12-9
Subject Areas Tab ... 12-10
Subject Areas Tab: Configuration Tags Subtab ... 12-10
Subject Areas Tab: Extended Tables (RO) Subtab ... 12-10
Subject Areas Tab: Tables Subtab ... 12-10
Subject Areas Tab: Tasks Subtab ... 12-11
Subject Areas Tab: Task Source Tables (RO) Subtab ... 12-11
Subject Areas Tab: Task Target Tables (RO) Subtab ... 12-11
Tables Tab ... 12-12
Tables Tab: Actions Subtab ... 12-12
Tables Tab: Conditional for Tasks (RO) Subtab ... 12-13
Tables Tab: Columns Subtab ... 12-13
Tables Tab: Indices (RO) Subtab ... 12-13
Tables Tab: Multi-Column Statistics Subtab ... 12-14

Tables Tab: Related Tables Subtab 12-14
Tables Tab: Source for Tasks (RO) Subtab 12-14
Tables Tab: Target for Tasks (RO) Subtab 12-14
Task Groups Tab 12-15
Task Groups Tab: Child Tasks Subtab 12-15
Task Groups Tab: Source Tables (RO) Subtab 12-15
Task Groups Tab: Target Tables (RO) Subtab 12-16
Tasks Tab 12-17
Tasks Tab: Actions Subtab 12-18
Tasks Tab: Conditional Tables Subtab 12-19
Tasks Tab: Configuration Tags Subtab 12-19
Tasks Tab: Parameters Subtab 12-19
Tasks Tab: Phase Dependency Subtab 12-20
Tasks Tab: Refresh Date Tables Subtab 12-21
Tasks Tab: Source Tables Subtab 12-22
Tasks Tab: Target Tables Subtab 12-22
Setup View Tabs 12-23
DAC System Properties Tab 12-24
Email Recipients Tab 12-28
Informatica Servers Tab 12-29
Physical Data Sources Tab 12-31
Physical Data Sources Tab: Index Spaces Subtab 12-32
Physical Data Sources Tab: Parallel Indexes Subtab 12-33
Physical Data Sources Tab: Analyze Frequencies Subtab 12-33
Physical Data Sources Tab: Refresh Dates Subtab 12-33
Execute View Tabs 12-34
Current Run Tab 12-35
Current Run Tab: Audit Trail (RO) Subtab 12-36
Current Run Tab: Phase Summary (RO) Subtab 12-36
Current Run Tab: Run Type Summary (RO) 12-37
Current Run Tab: Tasks Subtab 12-37
Current Run Tab: Task Details Subtab 12-37
Execution Plans Tab 12-38
Execution Plans Tab: All Dependencies Subtab 12-39
Execution Plans Tab: Following Tasks Subtab 12-39
Execution Plans Tab: Immediate Dependencies Subtab 12-39
Execution Plans Tab: Ordered Tasks Subtab 12-40
Execution Plans Tab: Parameters Subtab 12-41
Execution Plans Tab: Preceding Tasks Subtab 12-41
Execution Plans Tab: Refresh Dates Subtab 12-42
Execution Plans Tab: Subject Areas Subtab 12-42
Execution Plans Tab: Tables (RO) Subtab 12-42
Run History Tab 12-43
Run History Tab: Audit Trail (RO) Subtab 12-43
Run History Tab: Phase Summary (RO) Subtab 12-43
Run History Tab: Run Type Summary (RO) Subtab 12-43
Scheduler Tab 12-44

Handling Parameter Files with Multi-Line Parameters A-1
Restoring the DAC Repository on Unicode Oracle Databases A-1

Index

Preface

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

Oracle recommends reading the *Oracle Business Intelligence Data Warehouse Administration Console Release Notes* before installing, using, or upgrading DAC. The *Oracle Business Intelligence Data Warehouse Administration Console Release Notes* are available:

- On the Oracle Business Intelligence Data Warehouse Administration Console CD-ROM.
- On the Oracle Technology Network at http://www.oracle.com/technetwork/middleware/bi-foundation/documentation/bi-dac-087220.html.

Audience

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

Documentation Accessibility

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

Access to Oracle Support

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

Related Documents

For more information, see the following documents in the Oracle Business Intelligence Applications documentation set (available at (http://www.oracle.com/technetwork/middleware/bi-foundation/documentation/bi-apps-098545.html):

- *Oracle Business Intelligence Data Warehouse Administration Console Release Notes*
- *System Requirements and Supported Platforms for Oracle Business Intelligence Data Warehouse Administration Console*
- *Oracle Business Intelligence Data Warehouse Administration Console Installation, Configuration, and 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

The first section lists changes described in this version of the documentation to support releases 10.1.3.4 of the software.

The second section lists changes described in this version of the documentation to support releases 10.1.3.4.1 of the software.

1.1 What's New in Oracle Business Intelligence Data Warehouse Administration Console User's Guide, Release 10.1.3.4

The *Oracle Business Intelligence Data Warehouse Administration Console User's Guide* includes the following new and changed topics:

- **User Management.** The User Management feature enables you to create and manage user accounts. See "About User Account Management".
- **DAC Repository Database Authentication File.** Upon logging into DAC for the first time, the DAC Repository Database Authentication File authenticates the database in which the repository resides. See "DAC Repository Database Authentication File".
- **Best Practices for Defining Repository Objects.** See "Best Practices for Defining Repository Objects".
- **Enhanced Subject Area Design.** See "Customizing DAC Objects and Designing Subject Areas".
- **Enhanced Execution Plan Design.** See "Building, Running and Monitoring Execution Plans".
- **Multi-Source Execution Plans.** See "About Multi-Source Execution Plans".
- **Repository Audit Trail.** See "Tools Menu Options".
- **Actions Feature.** Provides a flexible way to drop and create indexes, truncate and analyze tables, and define SQL scripts to enhance task behaviors. See "About Index, Table and Task Actions".
- **Upgrade/Merge Wizard.** Enables you to upgrade and merge DAC repositories by comparing repositories and creating a Difference Report. See "Upgrading, Comparing and Merging DAC Repositories"

1.2 What's New in Oracle Business Intelligence Data Warehouse Administration Console User's Guide, Release 10.1.3.4.1

The *Oracle Business Intelligence Data Warehouse Administration Console User's Guide* revision 1 includes the following topics originally published in the Oracle Business Intelligence Applications Release Notes for 10.1.3.4 and new features in version 10.1.3.4.1:

- **Informatica Password Encryption.** By default, passwords are unencrypted when DAC interacts with Informatica; however, you can force password encryption. See "Forcing Password Encryption When DAC Interacts With Informatica".
- **Creating Parallel Indexes.** DAC can create parallel indexes by table. See "Physical Data Sources Tab: Parallel Indexes Subtab".
- **Worker Pool Size.** This is a new system property that specifies the number of worker threads that perform various operations. See "Worker Pool Size".
- **Multi-Line Parameters.** You can configure the infa_command.xml file to enable the parameter file to contain multi-line parameters. See "Handling Parameter Files with Multi-Line Parameters".
- **Restoring DAC Repository on Unicode Oracle Database.** On a non-Unicode Oracle database the Informatica workflow names may not fit into the corresponding fields. See "Restoring the DAC Repository on Unicode Oracle Databases".
- **Password Management.** Users in all roles can change their passwords. See "About User Account Management".
- **Prune Days.** Prune days can be set by source for multi-source ETL. See "Prune Days".
- **JDBC Override.** JDBC connection "URL Override" handles specific database connections. See "Using JDBC Connection URL Override to Handle Database Connections".
- **Upgrading the Data Warehouse Schema on Teradata**. When the data warehouse schema is on a Teradata database and you use the Data Warehouse Configuration Wizard to upgrade the data warehouse schema, DAC generates SQL scripts and log files. For a description of these files, see "Upgrading the Data Warehouse Schema on Teradata".
- **DAC Parameter Load Specification.** DAC Parameters can be specified for full or incremental load ETLs. See "Using Parameters to Specify Full or Incremental Loads".
- **Mapping Multiple Workflows.** Multiple database-specific Informatica workflows can be mapped to the same DAC task. See "Mapping Multiple Database-Specific Informatica Workflows to the Same DAC Task".
- **Connecting to DAC When Using Oracle RAC.** Use a specific URL to connect to the DAC repository when using Oracle RAC. See "Connecting to the DAC Repository When Using Oracle RAC".

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
- Oracle Business Analytics Warehouse Architecture

2.1 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.

2.2 Oracle Business Analytics Warehouse Architecture

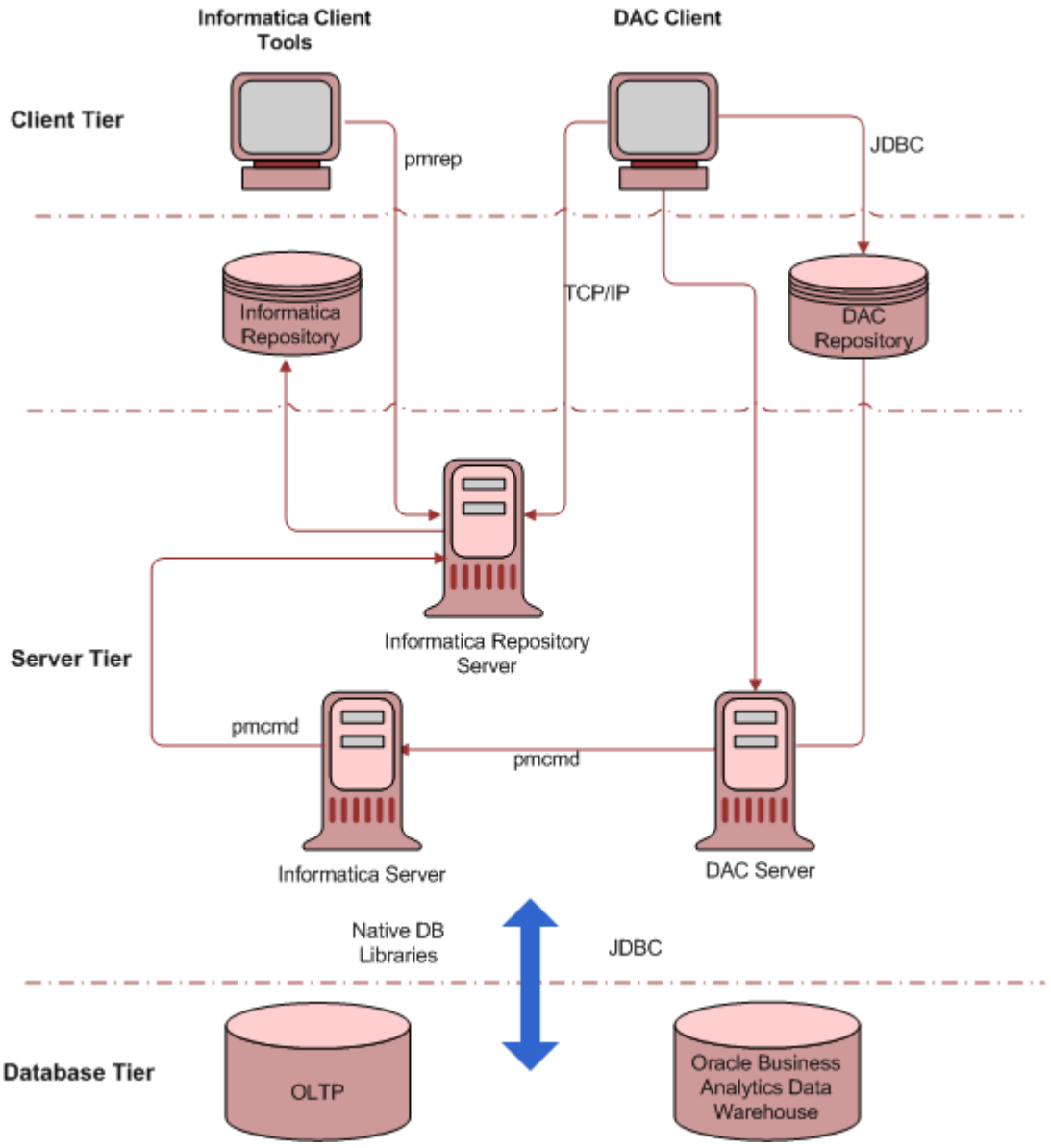
High-level analytical queries, like those commonly used in Oracle Business Analytics Warehouse, scan and analyze large volumes of data using complex formulas. This process can take a long time when querying a transactional database, which impacts overall system performance.

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



2.2.1 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.** Also referred to as PowerCenter Services. Loads and refreshes the Oracle Business Analytics Warehouse.
- **Informatica Repository Server.** Also referred to as Repository Services. 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.

3

Basic Concepts About DAC

This chapter describes basic concepts pertaining the Oracle Business Intelligence Data Warehouse Administration Console.

This chapter contains the following topics:

- Introduction to DAC
- About the DAC Process Life Cycle
- About Source System Containers
- About DAC Repository Objects Held in Source System Containers
- About Object Ownership in DAC
- About DAC Metadata Export Behavior

3.1 Introduction to DAC

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".

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. 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

3.1.1 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

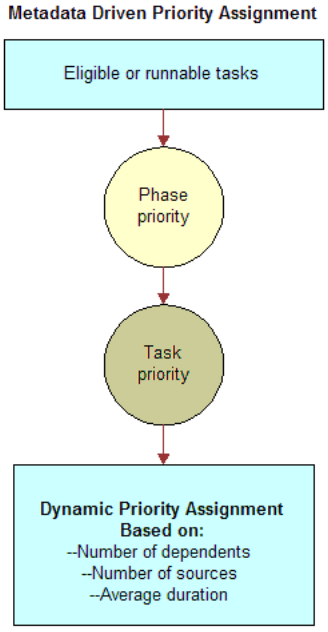
- Perform error monitoring and email alerting
- Perform structured ETL analysis and reporting

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 3–1.)

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

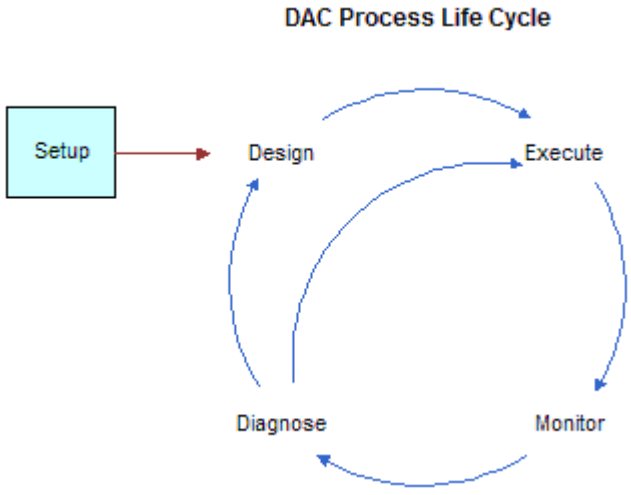
Figure 3–1 Task Execution Queue



3.2 About the DAC Process Life Cycle

DAC is used by different user groups to design, execute, monitor, and diagnose execution plans. These phases together make up the DAC process life cycle, as shown in Figure 3–2.

Figure 3–2 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

3.3 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 Held in Source System Containers".

You should use the preconfigured source system containers to create your own source system container. The DAC client lets you do modifications to objects in your own source system container built as a copy of the preconfigured containers. This enables the DAC client to track the customizations, such as newly created objects, modified objects, and those that are used as is.

Caution: You should not modify objects in the preconfigured source system containers either through the DAC client or directly through SQL statements to the DAC repository. 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".

3.4 About DAC Repository Objects Held in Source System Containers

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 DAC".

DAC repository stores application objects in a hierarchical framework that defines a data warehouse application. 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 includes but is not limited to the following repository objects:

- **Subject areas.** A logical grouping of tables related to a particular subject or application context. It also includes the tasks that are associated with the tables, as well as the tasks required to load the tables. Subject areas are assigned to execution plans, which can be scheduled for full or incremental loads.
- **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.

- **Indexes.** Physical database indexes to be defined in the database schema either to better the performance of the ETL processes or the queries for reporting purposes.
- **Tasks.** 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, DAC automatically assigns tasks to it. Tasks that are automatically assigned to the subject area by DAC are indicated by the Autogenerated flag in the Tasks subtab of the Subject Areas tab.
- **Task 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 plans.** 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.
- **Schedules.** 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.

3.5 About Object Ownership in 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 data warehouse 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 6–6, " Design View Right-Click Menu Commands".

3.6 About DAC Metadata Export Behavior

The DAC Import/Export feature enables you to import or export source system-specific DAC metadata into or out of the DAC repository.

Caution: When you export DAC metadata, DAC erases all of the files in the target folder.

DAC export behavior is as follows:

- If the target folder is empty, DAC exports without a warning.
- If the target folder contains DAC metadata, DAC exports after warning and when **OK** is clicked. The process replaces all content in the target folder with a new export.
- If the target folder has non-DAC metadata along with DAC Metadata, DAC exports after warning and when **OK** is clicked. The process replaces all content in the target folder with new export. All non-DAC metadata is deleted.
- If the target folder has only non-DAC metadata, DAC cannot export into that target folder.

4

DAC Quick Start

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

This chapter contains the following topics:

- About User Account Management
- Creating, Deleting and Inactivating User Accounts
- Logging into DAC for the First Time
- About the DAC's Extract and Load Processes
- Performing an Initial Full Load of Data into the Data Warehouse

4.1 About User Account Management

The User Management feature includes three roles: Administrator, Developer, and Operator. As shown in Table 4–1, each role has a set of permissions that determines what DAC functionality the role can access.

The User Management dialog box enables a user with the Administrator role to manage user accounts. A user account includes a unique identifier, password, and one or more roles. The Administrator can also inactivate a user account. For instructions on managing user accounts, see "Creating, Deleting and Inactivating User Accounts".

Upon the initial login to a new DAC installation, a user account with the Administrator role is automatically created. This default user account name is Administrator, and the default password is Administrator. It is recommended that after the initial login, the user change the default password. All DAC users can change their own passwords regardless of roles and without the help of their administrator.

Note: A user with the Administrator role must distribute the DAC Repository database authentication file to user accounts that need to access the DAC Repository. For information about the authentication file, see "DAC Repository Database Authentication File".

Table 4–1 User Account Roles and Permissions

Role	Permissions
Administrator	Read and write permission on all DAC tabs and dialog boxes.

Table 4–1 (Cont.) User Account Roles and Permissions

Role	Permissions
Developer	Read and write permission on the following: ▪ All Design view tabs ▪ All Execute view tabs ▪ Export dialog box ▪ New Source System Container dialog box ▪ Rename Source System Container dialog box ▪ Delete Source System Container dialog box ▪ Purge Run Details ▪ All functionality in the Seed Data menu
Operator	Read and write permission on all Execute view tabs

4.1.1 Creating, Deleting and Inactivating User Accounts

The User Management feature enables a user with the Administrator role to create, delete, and inactivate user accounts.

To create a user account

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

   For more information about authentication files, see "DAC Repository Database Authentication File".

To delete a user account

1. From the toolbar, select File, then User Management.
2. In the User Management dialog box, select the user account you want to delete.
3. Click Delete.
4. Click Close to exit the User Management dialog box.

To inactivate a user account

1. From the toolbar, select File, then User Management.
2. In the User Management dialog box, select the user account you want to inactivate.
3. Click the Inactive check box.
4. Click Save.

5. Click Close to exit the User Management dialog box.

The administrator generates the connection information and password files and distributes them to the users. Using the encrypted authentication file, the users will be able to use into the DAC Client using their own user name and password.

4.2 Logging into DAC for the First Time

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

DAC Repository Database Authentication File

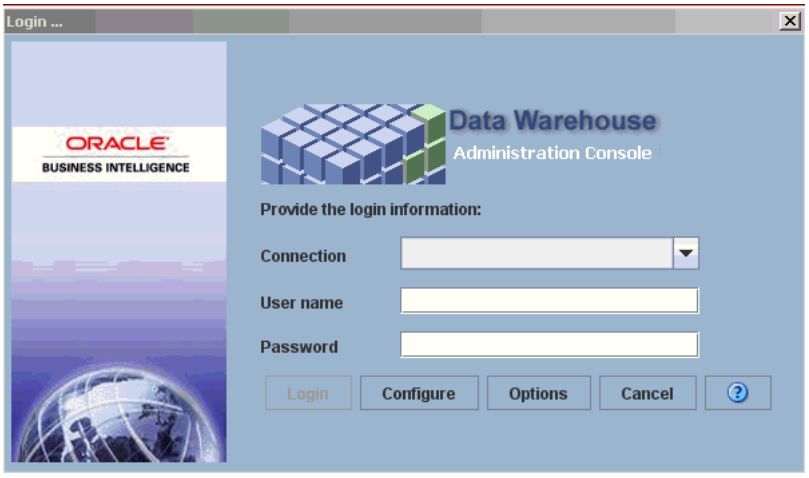
When you configure a connection to the DAC Repository, the configuration process includes creating a new authentication file or selecting an existing authentication file. The authentication file authenticates the database in which the repository resides. If you create a new authentication file, you will specify the table owner and password for the database.

A user with the Administrator role must distribute the authentication file to any user account that needs to access the specified DAC Repository.

To log into DAC for the first time

1. Start the DAC Client by navigating to the $ORACLE_HOME\bifoundation\dac directory and double-clicking the startclient.bat file.

   The Login ... dialog box appears.



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

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

Field	Required Value
Connection String, or Database name, or TNS Name, or Instance	Select the database name or database account name of the DAC Repository. If you are using: ■ Oracle (OCI8), use the tnsnames entry. ■ Oracle (Thin), use the instance name. ■ SQL Server, use the database name. ■ DB2-UDB, use the connect string as defined in the DB2 configuration.
Database Host	Enter the name of the machine where the DAC Repository will reside.
Database Port	Enter the port number on which the database listens. For example, for an Oracle database the default port is 1521, or for a SQL Server database the default port is 1433.
Optional URL	Can be used to override the standard URL for this connection.
Optional Driver	Can be used to override the standard driver for this connection.
Authentication File	Click in this field to do one of the following: ■ Select an existing authentication file: Navigate to the appropriate location, select the authentication file , and click OK. ■ Create a new authentication file: Navigate to the folder where you want to save the authentication file, and click OK. Proceed to the next step for detailed instructions.

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

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

6. To create a new authentication file, do the following:
   a. Click in the Authentication File field of the Configuring... dialog box.
   b. In the Authentication File dialog box, select Create authentication file.
   c. Navigate to the folder where you want to save the new authentication file, and click OK.
   d. In the Create Authentication File dialog box, enter a unique name for the authentication file, and click OK.

e. Enter the Table Owner Name and Password for the database where the repository will reside.

f. In the Configuring... dialog box, click Test Connection to confirm the connection works.

g. Click Apply, and then click Finish.

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

---

7. In the Login... dialog box, do the following:
   a. Select the appropriate Connection from the drop-down list.
   b. Enter Administrator as the User Name.
   c. Enter Administrator as the Password.
   d. Click Login.
8. If asked whether you want to create or upgrade the DAC Repository schema, click Yes.

4.3 About the DAC's Extract and Load Processes

To understand how the DAC handles extract and load processes, you must have an understanding of the DAC repository objects. For a description of these objects, see "About DAC Repository Objects Held in Source System Containers".

When you run an execution plan, data is extracted from one or more tables in the source system database, dumped into staging tables, and then loaded into tables in the data warehouse.

Each task in the DAC is mapped to a full load or an incremental load workflow in Informatica. The DAC uses refresh dates (indicated in the Refresh Dates child tab of the Physical Data Sources tab) to determine whether to invoke a full or incremental workflow.

If the source or target table for a task is null, then the DAC invokes a full load workflow (command). If both the source and target tables have a refresh date, then the DAC invokes the incremental workflow (command). For a detailed description of refresh dates, see "About Refresh Dates".

The DAC supports the following extract and load combinations:

- **Full extract and full load**

  This extract and load combination is used for the very first extract and load. All data is extracted from the source system and loaded into the data warehouse.

  The DAC performs a full extract for a task if the source and staging tables have null refresh dates. The DAC performs a full load for a task if the staging and target tables have null refresh dates.

- **Full extract and incremental load**

  This extract and load combination loads existing data warehouse tables with data from new sources. Data is extracted from the source system through a full extract command. When the source or staging table is null, the DAC invokes the full extract workflow. Data is loaded from the staging table into the target table through an incremental load command. When the staging and target tables have

refresh dates, the DAC invokes an incremental load command. This situation arises when data is loaded into an existing data warehouse from a new source connection.

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

- **Incremental extract and incremental load**

  This extract and load combination is used for regular nightly or weekly ETL processes. New or changed records are extracted from the source system and loaded into the data warehouse. The DAC performs an incremental extract for a task if the source and staging tables have refresh dates and performs an incremental load for a task if the staging and target table have refresh dates.

4.3.1 Performing an Initial Full Load of Data into the Data Warehouse

After you install and configure Oracle BI Applications, you need to perform a full load of data into the data warehouse. Chapter 8, "Customizing DAC Objects and Designing Subject Areas", contains detailed instructions concerning ETL processes.

To perform an initial full load, at a minimum you will need to perform the following procedures in Chapter 8:

1. Create a copy of your source system container.

   For instructions, see "Creating or Copying a Source System Container".

2. Create a subject area.

   For instructions, see "Creating a Subject Area". See also "Overview of Subject Areas".

3. Assign a subject area to an execution plan and run the execution plan.

   For instructions, see "Building and Running Single-Source and Multi-Source Execution Plans".

5

Best Practices for Defining Repository Objects

This chapter provides a description of the best practices for defining repository objects.

This chapter contains the following topics:

- Best Practices for Containers
- Best Practices for Tasks
- Best Practices for Task Groups
- Best Practices for Tables
- Best Practices for Indexes
- Best Practices for Columns
- Best Practices for Configuration Tags
- Best Practices for Source System Parameters
- Best Practices for Subject Areas
- Best Practices for Execution Plans

5.1 Best Practices for Containers

The following best practices apply to containers:

- When changes are made to objects in the container that owns them, the change is instantaneous.
- Changes made to parent objects in the owner container are automatically pushed to the parent referenced objects.
- When you add child objects to a parent object, you must you the Push to References right-click command (Design view) to push the changes to the child referenced objects. For example, if you add a column to a table registered in DAC, the new column is not automatically added to the references in the other containers referencing the parent object.
- 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 from the containers that may be referencing this object. 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 and subsequently from all the containers that may be referencing this object.

- If you delete a column from the owner table, the column is deleted in all the referenced objects.
- If you delete child objects from the owner object, the referenced child objects are automatically deleted.

5.2 Best Practices for Tasks

The following best practices apply to tasks:

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

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

    You can use the task synchronize functionality to import data from Informatica. You can also manually assign the source or target tables.
- Select at least one primary table because the incremental and full mode properties are determined based on the refresh dates of the primary table.
- Design tasks so that they load only one table at a time.
- Define granular tasks rather than tasks that include bigger blocks of processes. Granular tasks are more efficient and have better restartability.
- Do not truncate a table on the source system tables (for example, Oracle, Siebel or Peoplesoft).
- Make sure the truncate property for the target tables is set properly.
- For tables that need to get truncated regardless of the mode of the run (Full or Incremental), set the Truncate Always property to True.
- For tables that need to get incrementally loaded, set the Truncate for Full Load property to True.
- Select the Analyze Table option if the task should analyze the table. The default value for this option is True if either of the Truncate options are selected.
- Do not truncate a table more than once within the single life span of an ETL.
- If a task that writes to a target table is contingent upon another table being loaded, use conditional tables. This ensures that the task qualifies only if the conditional table is part of the subject area design.
- Assign an appropriate phase to the task. An understanding of task phases is essential to understanding ETL processes.
- If you want to force a relationship where none exists, consider using phase dependencies. For example, if you have an ETL process in which the extract facts and extract dimensions do not share any common source or target tables, but your design requires that the extract facts should run before extracting dimensions,

then, for the task that extracts facts, add extract dimension as the phase that waits. For more information about phase dependencies, see "Tasks Tab: Phase Dependency Subtab".

- Make sure you do not introduce conflicting phase dependencies. This can cause the DAC Server to hang.
- If the source qualifier needs to use a data parameter, always use the DAC date parameter that can be formatted to the database-specific syntax.

5.3 Best Practices for Task Groups

The following best practices apply to task groups:

- Do not create task groups unnecessarily, because this can impact a better way of ordering tasks. Create task groups only if you want the following scenarios and the auto-dependency (auto ordering of tasks for an execution plan) cannot solve the order in a proper way.
- Avoid circular relationships among tasks. For example, avoid situations in which Task 1 reads from Table A and writes to Table B, and Task 2 reads from Table B and writes to Table A. You can use task groups to avoid these situations. If the tasks belong to different phases, however, this situation is acceptable.
- If you have many tasks belonging to the same phase that write to the same table, you can use task groups to run the tasks in parallel. If the target tables need to be truncated before the tasks are run, select the properties Truncate Always and Truncate Full Load in the Task Group tab.
- Do not mix extracts and loads under a single table group.
- Do not make Task Groups for obvious ordering needs. DAC handles ordering in such cases.
- If a source system container uses a task group, make other containers that reference the task also include the task group.

5.4 Best Practices for Tables

The following best practices apply to tables:

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

5.5 Best Practices for Indexes

The following best practices apply to indexes:

- Always use all upper case characters for column names.

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

5.6 Best Practices for Columns

The following best practices apply to columns:

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

5.7 Best Practices for Configuration Tags

The following best practices apply to configuration tags:

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

5.8 Best Practices for Source System Parameters

The following best practices apply to source system parameters:

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

5.9 Best Practices for Subject Areas

The following best practices apply to subject areas:

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

5.10 Best Practices for Execution Plans

The following best practices apply to execution plans:

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

6

Overview of the DAC Interface

This chapter provides an overview about the DAC user interface.

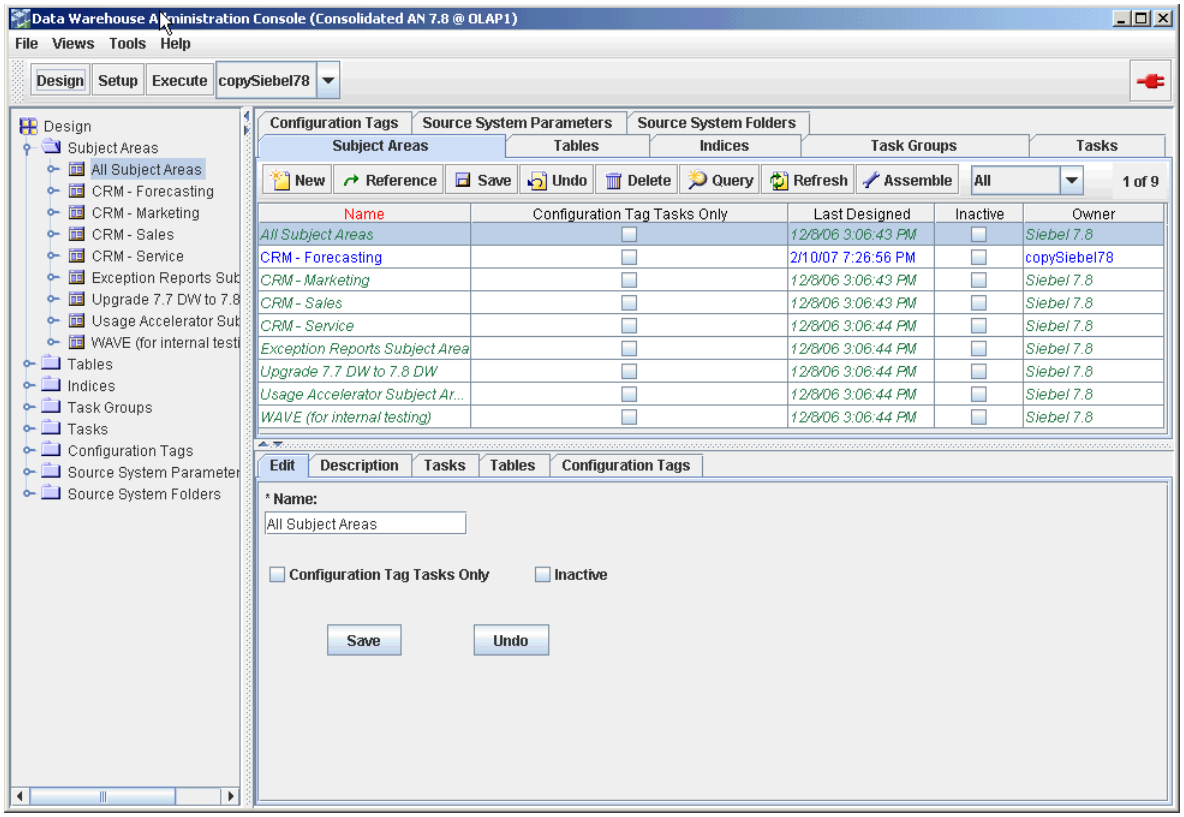
This chapter contains the following topics:

- Navigating the DAC Interface
- The DAC Menu Bar
- The DAC Views
- The DAC Top Pane Toolbar
- The DAC Right-Click Menus
- The DAC Server Monitor Icons
- The DAC Navigation Tree
- The DAC Editable Lists
- Font Variations of Objects Displayed in the DAC
- Using the DAC Query Functionality

6.1 Navigating the DAC Interface

Figure 6–1 shows the main elements of the DAC window.

Figure 6–1 Main DAC Window



6.2 The DAC Menu Bar

Table 6–1 provides a description of the DAC menu bar options.

Table 6–1 DAC Menu Bar Options

Menu Names	Description
File	The File menu provides access to the following: ■ **User Management** Enables users with the Administrator role to manage user accounts. ■ **New Source System Container** Enables you to create a new source system container or make a copy of an existing container. For instructions see, "Creating or Copying a Source System Container". ■ **Rename Source System Container** Enables you to rename an existing source system container. ■ **Delete Source System Container** Enables you to delete an existing source system container. ■ **Close** Closes the DAC client.
Views	The Views menu enables you to navigate to the various tabs in the top pane window.

Table 6–1 (Cont.) DAC Menu Bar Options

Menu Names	Description
Tools	The Tools menu provides access to functionality related to the DAC and Informatica repositories. Table 6–2 provides a description of the Tools menu commands.
Help	The Help menu provides access to the following: ■ **Login Details** Lists database connection information. ■ **System Information** ■ **DAC Help** ■ **About DAC** Lists information about the DAC version.

6.2.1 Tools Menu Options

Table 6–2 provides a description of the Tools menu commands.

Table 6–2 DAC Tools Menu Commands

Tools Menu Command	Description
DAC Repository Management > Export	Enables you to export the DAC metadata, in XML format, based on the source system container, in order to back up the metadata or to reproduce the environment elsewhere. In the Export dialog, you can specify a directory in which to store the XML file or accept the default directory, which is DAC\export. In the Export dialog, you can select the following category options: ■ **Logical.** Exports all information contained in the Design view and 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	Enables you to import the DAC metadata for the source system containers you specify. In the Import dialog, you can specify the following: ■ **Import/Export folder.** A directory from which to import the data. The default directory is DAC\export. ■ **Truncate repository tables.** Indicates whether you want to truncate the repository tables. If you select this option, the existing metadata is overwritten. ■ **Enable 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.

Table 6–2 (Cont.) DAC Tools Menu Commands

Tools Menu Command	Description
DAC Repository Management > Create Repository Report	Enables you to generate a DAC repository report based on the following criteria: ■ Table Row Counts ■ Object References by Entity ■ Ownerless Objects ■ Unreferenced Objects ■ Dead References The Clean Up command removes unused referenced objects.
DAC Repository > Upgrade/Merge Wizard	See "Upgrading, Comparing and Merging DAC Repositories"
DAC Repository Management > Purge Run Details	Enables 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.** Enables you to select an execution plan whose associated runs you want to purge. ■ **By Run Name.** Enables you to select an individual run for purging. ■ **Before Specified Date.** Enables 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	Enables you to run analyze table commands for all the DAC repository tables.
DAC Repository Management > Default Index Properties	Enables you to specify which databases will be associated with newly created indexes.
DAC Repository Management > Repository Audit Trail	Enables you to access the Repository Audit Trail, which stores information about users, machine IP addresses, and repository timestamps.
DAC Repository Management > Drop DAC Repository	Enables 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	Enables you to configure the DAC Server connections and server email settings. This action should be performed on the machine where the DAC Server is running.
ETL Management > Configure	Opens the Data Warehouse Configuration wizard, which enables you to create and drop data warehouse tables and to create delete triggers.
ETL Management > Reset Data Sources	Clears the refresh dates for all source and target tables. This action forces a full load to occur.
Seed Data > Task Phases	Enables you to add, edit, or delete task phases.

Table 6–2 (Cont.) DAC Tools Menu Commands

Tools Menu Command	Description
Seed Data > Task Logical Folders	Enables you to add, edit, or delete Task Logical folders.
Seed Data > Task Physical Folders	Enables you to add, edit, or delete Task Physical folders.
Seed Data > Logical Data Sources	Enables you to add, edit, or delete logical data sources.
UI Styles > Windows (MFC)	Changes the user interface to the Windows style.
Seed Data > Actions > Index	Enables you to set up index actions in order to trigger SQL scripts to create or drop indexes. For more information, see "About Index, Table and Task Actions".
Seed Data > Actions > Tables	Enables you to set up table actions in order to trigger SQL scripts to analyze and truncate tables. For more information, see "About Index, Table and Task Actions".
Seed Data > Actions > Tasks	Enables you to set up task actions in order to trigger SQL scripts to perform various actions related to task behavior. For more information, see "About Index, Table and Task Actions".
UI Styles > Windows (MFS)	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.
UI Preferences	Enables you to select use interface behaviors.

6.3 The DAC Views

The DAC View buttons are located directly under the menu bar. Table 6–3 provides a description of the different DAC views.

Table 6–3 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 enables you to select the source system container that holds the metadata corresponding to a source system.
Setup	The Setup View provides access to functionality related to setting up DAC system properties, Informatica 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".

6.4 The DAC Top Pane Toolbar

Table 6–4 describes the commands available in the top pane toolbar.

Table 6–4 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 enables you to copy objects from one container to another. For more information about referencing objects, see "About Object Ownership in DAC".
Assemble	Design view only. Assembles a subject area, with dimension and related tables as well as tasks.
Drop-down list	Design view only. Enables 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.
Restart	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. Enables you to turn on and off the automatic screen refresh functionality and set the refresh interval.

6.5 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

6.5.1 Common Right-Click Menu Commands

Table 6–5 Common Right-Click Menu Commands

Command	Description
Copy String	Copies the contents of a cell (editable and read-only) to the clipboard

Table 6–5 (Cont.) Common Right-Click Menu Commands

Command	Description
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, enables you to update the column value for each row to a single value.

6.5.2 Design View Right-Click Menu Commands

Table 6–6 Design View Right-Click Menu Commands

Command	Description
Ownership	■ **Reference.** Opens the Reference dialog, which enables you to reference objects from one container to another. The reference function works like a symbolic link or shortcut. ■ **Re-Reference.** If an object is a referenced object, that is, a reference to an object in another container and a change is made to the original object's child objects, 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 enables you to revert a cloned object back to its state as a reference. ■ **Re-Assign Record.** This command enables you to reassign an objects ownership. For more information about the ownership of objects, see "About Object Ownership in DAC".
Assemble	(Subject Areas tab) Assembles a subject area, with dimension and related tables as well as tasks.

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

Command	Description
Generate Index Scripts	(Tables and Indices tabs) 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	(Tables tab) For Siebel sources only. ■ **Image and Trigger Scripts.** Generates change capture scripts for tables with defined image suffixes. The scripts may include delete triggers, create and drop statements for delete triggers, and image tables and their indexes. ■ **View Scripts.** Generates change capture view scripts for full or incremental mode for tables that participate in the change capture process. This command can be used for unit testing. ■ **Change Capture SQL.** Generates change capture SQL scripts for full or incremental mode for tables that participate in the change capture process. This command can be used for unit testing.
Import from Database	(Tables tab) ■ **Import Database Tables.** Enables you to import table definitions from a selected database. This action does not import columns. ■ **Import Indices.** Enables 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.** Enables you to import column definitions from a selected database.
Add Actions	(Tables tab) See "About Index, Table and Task Actions".
Add Actions	(Indices tab) See "About Index, Table and Task Actions".
Output Task Description	(Tasks tab) Saves to an HTML file the description for a selected task or for all tasks.
Synchronize Tasks	(Tasks tab) Synchronizes the information the DAC has for a task's source and target tables with the information in the Informatica repository.
Flat Views	Opens a dialog that enables 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 **Run History tab** ■ Audit Trail

6.5.3 Setup View Right-Click Menu Commands

Table 6–7 Setup View Right-Click Menu Commands

Command	Description
Test Connection	In the Physical Data Sources tab, it enables you to test the database connection. In the Informatica Servers tab, it enables you to test the connection to the PowerCenter Services and Integration Service. The DAC server performs this command if the DAC client is connected to a server. If the DAC client is not connected to a DAC server, then the DAC client performs the command.

6.5.4 Execute View Right-Click Menu Commands

Table 6–8 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.
Reset source(s)	Resets the source or sources for this execution plan.
Show Tables	Shows execution plan source and target tables.
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.
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.
Auto Refresh	(Current Run tab) Enables you to set an automatic refresh frequency.

6.6 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 6–2. 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 6–3. 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 6–4. 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 6–2 DAC Server Down Icon

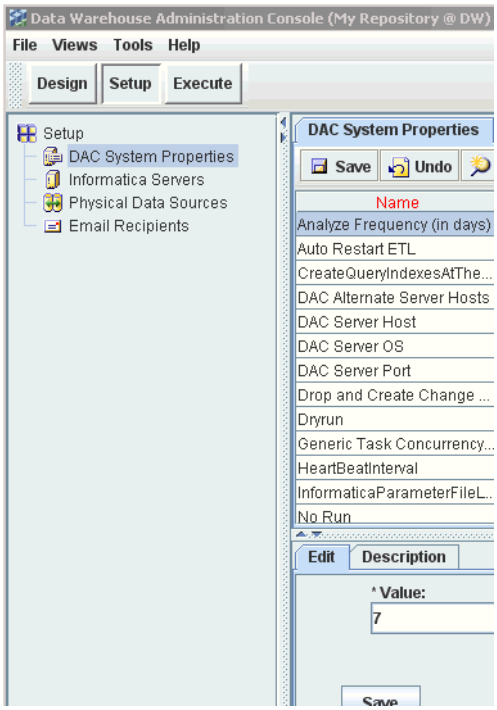
Figure 6–3 DAC Server Idle Icon

Figure 6–4 DAC Server Running Icon

6.7 The DAC Navigation Tree

The navigation tree appears on the left side of the DAC window, as shown in Figure 6–5, and displays the top-level tabs of the selected view.

Figure 6–5 DAC Navigation Tree



6.8 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 enables 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 enables you to do the following:

- Edit the data in place and save the record by either clicking another record in the list or clicking the Save button.
- Reorder the columns.
- Sort the data in the list by clicking on the column name.
- Select predefined values from picklists.
- 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.

6.9 Font Variations of Objects Displayed in the DAC

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

Table 6–9 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.

6.10 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
- Common DAC Query Procedures
- Common DAC Query Procedures
- Using Flat Views Querying

6.10.1 DAC Query Commands and Operators

Table 6–10 describes the query commands and operators you can use to define your query criteria.

Table 6–10 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.)
OR	Placed between values, returns records for which at least one condition is true. (Not case sensitive.)

6.10.2 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 ("").

6.10.3 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.

6.10.4 Using Flat Views Querying

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

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

To update multiple records using the Flat Views query feature

1. In the 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.

7

About Index, Table and Task Actions

This chapter describes how to use the actions feature, which enables you to enables you to define and run SQL scripts to carry out various actions in relation to indexes, tables, and tasks.

This chapter contains the following topics:

- Overview of Index, Table and Task Actions
- Defining a SQL Script for an Action
- Assigning an Action to a Repository Object
- Functions for Use with Actions
- Example of How to Use a DAC Source System Parameter in an Action

7.1 Overview of Index, Table and Task Actions

The Actions feature enables you to define and run SQL scripts to carry out various actions in relation to indexes, tables, and tasks.

- **Index actions** can override the default behavior for dropping and creating indexes by mode type (full load, incremental load, or both). The Index action type are Create Index and Drop Index.

  For example you can define an index action to create indexes for tasks with the commands defined for full loads, or incremental loads, or both types. Index actions override all other index properties.

- **Table actions** can override the default behavior for truncating and analyzing tables by mode type. The Table action types are Truncate Table and Analyze Table.

  For example you can define a table action to truncate tables with the commands defined for full loads, or incremental loads, or both . Table actions override all other index properties.

- **Task actions** can add new functionality based on various task behaviors. The following task action types are available:

  - Preceding Action

    Use this type to execute a SQL script before a task runs.

  - Success Action

    Use this type to execute a SQL script after a task runs successfully.

  - Failure Action

    Use this type to execute a SQL script if a task fails during its execution.

- Upon Failure Restart

  Use this type to execute a SQL script when a task that previously failed is restarted.

You can also use the Actions Template feature to:

- Combine SQL templates to do synchronized actions, such as create and analyze indexes.
- Combine object level properties with user-defined parameters in SQL statements and stored procedures.

To define an action and assign it to a repository object, you need to complete the following procedures:

- Defining a SQL Script for an Action
- Assigning an Action to a Repository Object

7.2 Defining a SQL Script for an Action

Follow this procedure to define a SQL statement for index, table, and task actions. After completing this procedure, proceed to "Assigning an Action to a Repository Object".

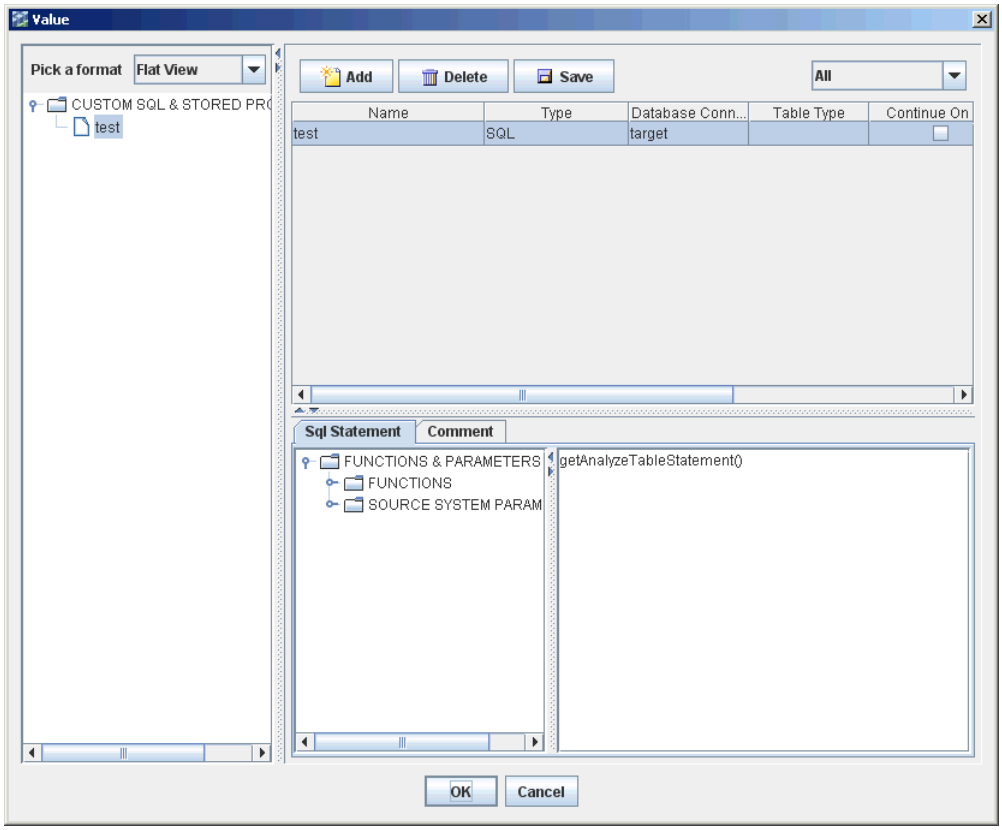
For more information about Index, Table and Task actions, see "Overview of Index, Table and Task Actions".

To define a SQL statement for an action

1. From the Tools menu, select Seed Data, then select one of the following:
   - Index Actions
   - Table Actions
   - Task Actions

   The Actions dialog box opens.
2. In the toolbar, click New.
3. In the new record field, enter a name for the action, and then click Save.
4. Double-click in the Value field.

   The Value dialog box appears.



5. Select a format for the tree view.
   - Flat view displays the SQL entries in a list format in their order of execution.
   - Category view displays the entries by the categories SQL and Stored Procedure.

     You can reorder the entries in the tree by dragging and dropping them.
6. Click Add.
7. In the new record field, enter or select the appropriate information.

Field	Description
Name	Logical name of the SQL block.
Type	SQL or Stored procedure

Field	Description
Database Connection Type	Should be used only for SQL types (not stored procedures). Defines which database the SQL statement will run against. Possible values are: Source - SQL runs against the source connection defined for the task. Target - SQL runs against the source connection defined for the task. Both - SQL runs against both the source and target connection. Table Connection - SQL runs against the table-specific connection if a separate table connection is available.
Table Type	Specifies the table type against which the SQL will run. Possible values are: All Source - SQL runs against all source tables defined for the task. All Target - SQL runs against all target tables defined for the task. Source Lookup - SQL runs against all the source lookup tables defined for the task. Source Primary - SQL runs against all the source primary tables defined for the task. Source Auxiliary - SQL runs against all the source auxiliary tables defined for the task.
Continue on Fail	Specifies whether an execution should proceed if a given SQL block fails.
Retries	Specifies how many retries are allowed. If the number is not positive, a default number of one (1) will be used.
Valid Database Platforms	Specifies the valid database platforms against which the SQL will run. If this field is left empty, the SQL can be run against any database.

8. In the lower-right side text box, enter a SQL statement.

   The SQL Statement tab to the left of the text box lists all the supported SQL functions and DAC source system parameters that you can use in constructing custom SQLs. Double-click a function or source system parameter to move it into the text box.

   For a description of the available functions, see "Functions for Use with Actions".

   The source systems parameters list contains the names of all source system parameters defined in the DAC Repository, with the prefix @DAC_. During runtime, the DAC Server resolves the source system parameter and replaces its name with the runtime value.

   For an example of how to use a source system parameter in a SQL statement, see "Example of How to Use a DAC Source System Parameter in an Action".

9. (Optional) Enter a comment about the SQL in the Comment tab.

10. Click OK.

> **Note:** You can add multiple SQL statements and stored procedures to a single action.

11. To assign this action to a repository object, proceed to "Assigning an Action to a Repository Object".

7.3 Assigning an Action to a Repository Object

Follow this procedure to assign an action to a DAC Repository object. Before you do this procedure, you must have defined a SQL script for an action. For instructions, see "Defining a SQL Script for an Action".

To assign an action to a repository object

1. In the Design view, navigate to one of the following tabs, depending on the object type for which you want to assign an action:
   - Indices tab
   - Tables tab
   - Tasks tab
2. Select or query for the object for which you want to assign an action.
3. With the appropriate object selected in the top window, select the Actions subtab.
4. Click New in the subtab toolbar.
5. In the new record field, do the following:
   a. Select an Action Type.

      For a description of the available Action Types, see the following: "Indices Tab: Actions Subtab", "Tables Tab: Actions Subtab", and "Tasks Tab: Actions Subtab".

   b. Select the Full, Incremental, or Both load type.

   c. Double-click the Action field to open the Choose Action dialog box, and select an action.

      > **Note:** For instructions on defining an action, see "Defining a SQL Script for an Action".

   d. Click OK to close the Choose Action dialog box.

   e. Click Save in the subtab toolbar.

7.4 Functions for Use with Actions

This section includes a list of functions that are available for Index, Table and Task actions.

- Table 7–1, " Functions for Index Actions"
- Table 7–2, " Functions for Table Actions"
- Table 7–3, " Functions for Task Actions"

Table 7–1 Functions for Index Actions

Function	Description
getAdditionalColumns()	Returns a comma separated list of included index columns. This function is related to the #Unique Columns index property. For DB2 and DB2-390 databases, the list may include a combination of unique and included columns.
getAnalyzeStatement()	Returns the default DAC index analyze statement (for this particular index).
getAnalyzeTableStatement()	Returns the default DAC table analyze statement (for the parent table).
getBitMapString	Resolves to the string `BITMAP` if it is a bitmap index; otherwise the string is empty.
getClusteredString	Resolves to the string `CLUTSTERED` if it is a bitmap index; otherwise the string is empty.
getCreateIndexStatement()	Returns the default DAC index creation statement.
getDBType()	Returns the physical data source connection type (Oracle OCI8, Oracle Thin, DB2, DB2-390, MSSQL, or Teradata).
getDropIndexStatement()	Returns the default DAC index drop statement.
getHashString()	Resolves to the string `HASH` if it is a hash index; otherwise the string is empty.
getImageSuffix()	Resolves to the table image suffix if one is specified; otherwise the string is empty.
getIndexColumns()	Returns a comma separated list of index columns.
getIndexName()	Returns the index name.
getIndexTableSpace()	Resolves to the index space name if one exists; otherwise the string is empty.
getNamedSource()	Returns the DAC physical connection name.
getRvrsScanString()	Resolves to the string `ALLOW REVERSE SCANS` if the index supports reverse scans; otherwise the string is empty.
getTableName()	Returns the table name.
getTableOwner()	Returns the table owner name.
getTableSpace()	Returns the table space name if one exists; otherwise the string is empty.
getTruncateTableStatement()	Returns the default DAC table truncate statement.
getUniqueColumns()	Returns a comma separated list of unique index columns. This function is a counterpart of the getAdditionalColumns() function.
getUniqueString()	Resolves to the string `UNIQUE` if the index is unique; otherwise the string is empty.

Table 7–2 Functions for Table Actions

Function	Description
getAnalyzeTableStatement()	Returns the default DAC table Analyze statement.
getDBType()	Returns the physical data source connection type (Oracle OCI8, Oracle Thin, DB2, DB2-390, MSSQL, or Teradata).

Table 7–2 (Cont.) Functions for Table Actions

Function	Description
getImageSuffix()	Returns the table image suffix if one exists; otherwise the string is empty.
getCreateIndexStatement()	Returns the default DAC index creation statement.
getNamedSource()	Returns the DAC physical connection name.
getDropIndexStatement()	Returns the default DAC index drop statement.
getTableName()	Returns the table name.
getTableOwnerName()	Returns the table owner.
getTableSpace()	Returns the table space name if one exists; otherwise the string is empty.
getTruncateTableStatement()	Returns the default DAC table truncate statement.

Table 7–3 Functions for Task Actions

Function	Description
getAnalyzeTableStatement()	Returns the default DAC analyze table statement.
getDBType()	Returns the physical data source connection type (Oracle OCI8, Oracle Thin, DB2, DB2-390, MSSQL, or Teradata).
getImageSuffix()	Returns the table image suffix if one exists; otherwise the string is empty.
getNamedSource()	Returns the physical connection name in DAC.
getTableName()	Returns the table name.
getTableOwner()	Returns the table owner.
getTableSpace()	Returns the table space name if one exists; otherwise the string is empty.
getTruncateTableStatement()	Returns the default DAC truncate table statement.

Note: Table-related task action functions should be used only for SQL blocks with a specified table type. For these blocks, DAC will loop through the tables of the type you specify and execute custom SQL for each table. Functions will be substituted with table-specific values for each iteration.

For example, if you wanted to gather statistics in a particular way after creating specific indexes, you would need to create index actions with two SQL blocks:

1. `getCreateIndexStatement()`
2. `DBMS_STATS.GATHER_TABLE_STATS(ownname => 'getTAbleOwner()', tabname => 'getTableName()', estimate_percent => 70, method_opt +> 'FOR ALL COLUMNS SIZE AUTO', cascade => false)`

7.5 Example of How to Use a DAC Source System Parameter in an Action

The following example illustrates how to use a DAC source system parameter in defining an action.

Assume there is a source system parameter called *COUNTRY* (note: source system parameter names are case sensitive). And, you use this parameter in an action using the following SQL statement:

```
DELETE FROM TABLE1 WHERE COUNRY_NAME='@DAC_COUNTRY'
```

Assume that during the ETL process, `COUNTRY` gets resolved to Canada. The resulting SQL that is executed by the DAC Server would be the following:

```
DELETE FROM TABLE1 WHERE COUNTRY_NAME='Canada'
```

8

Customizing DAC Objects and Designing Subject Areas

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

This chapter contains the following topics:

- Creating or Copying a Source System Container
- About Customizing the Data Warehouse
- Adding a New Table and Columns to the Data Warehouse
- Adding an Index to the Data Warehouse
- Importing New Data Warehouse Objects into the Informatica Repository
- Creating Informatica Mappings and Workflows
- Creating Tasks in the DAC for New or Modified Informatica Workflows
- Setting a Task Phase Dependency
- Creating a Task Group
- About Parameter Management
- Specifying Tablespaces for Indexes by Table Type
- Working with Configuration Tags
- Overview of Subject Areas
- Creating a Subject Area

8.1 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. 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 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. The Name field must be at least five characters long.

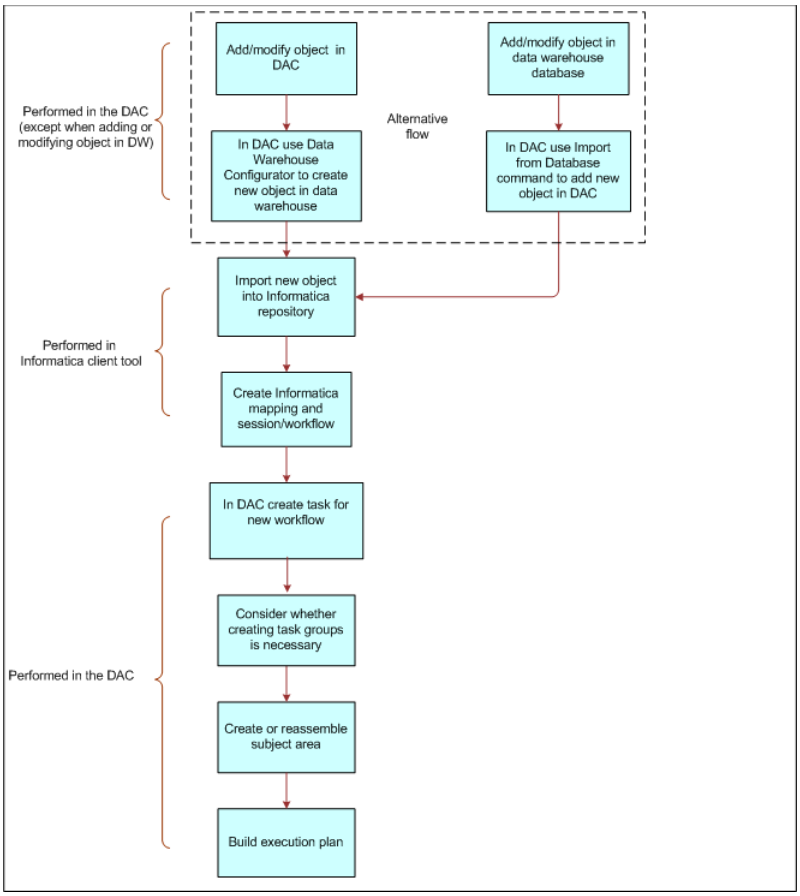
3. Select one of the following:
   - Create Empty New Source System Container
   - Create as a Copy of Source System Container
4. If you are creating an empty, new container, click OK.
5. If you are making a copy of an existing container, select the existing container from the drop-down list, and then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

8.2 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 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 Guide for Informatica PowerCenter Users*.

Figure 8–1 shows the major steps required for adding a new object to the data warehouse or modifying existing objects. As shown in Figure 8–1, you can begin the customization process by adding or modifying the new data warehouse object in 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 DAC.

Figure 8–1 Process Flow to Add New Object to Data Warehouse



8.3 Adding a New Table and Columns to the Data Warehouse

As shown in Figure 8–1, there are two alternative process flows for adding a new object to the data warehouse. You can enter the table and column definitions in DAC and then use the DAC's 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 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. From 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. From 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.

8.4 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. From 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.

8.5 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 Guide for Informatica PowerCenter Users*.

8.6 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 Guide for Informatica PowerCenter Users*.

8.7 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".

8.8 Setting a Task Phase Dependency

A task phase dependency enables you to dictate additional dependency hints for tasks that are completely unrelated to each other through source/target table relationships.

To set a task phase dependency

1. In the DAC toolbar, select the appropriate source system container from the drop-down list in the toolbar.
2. From the Menu bar, select Views, then select 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 Phase, Grain, and Scope and click OK.

   For information about these fields, see "Tasks 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.

8.9 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 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. From 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.

8.10 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
- Preconfigured Parameters
- How DAC Handles Parameters at Runtime
- Nesting Parameters within Other Parameters
- Defining a Text Type Parameter
- Defining a Database Specific Text Type Parameter
- Defining a Timestamp Type Parameter
- Defining a SQL Type Parameter

8.10.1 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.

8.10.1.1 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.

8.10.2 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.

8.10.3 How 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 Guide for Informatica PowerCenter Users*.

8.10.4 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.

8.10.5 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.

8.10.6 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.

8.10.7 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.

8.10.8 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.

8.11 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.

8.12 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. From 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. From 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. From 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. From 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.

8.13 Overview of Subject Areas

Oracle BI 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".

8.13.1 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?

8.13.1.1 Previewing and Pruning Subject Areas

You can preview a subject area to determine whether it contains the appropriate tables and tasks to suit your business needs. When you assign a fact table to a subject area, the fact table's related tables are automatically assigned to the subject area. If you determine you do not need one or more of the related tables, you can remove the tables during the subject area Assembly process. This kind of modification is referred to as *pruning* the subject area. You can also add individual tables and tasks.

To preview a subject area, follow the procedure "Creating a Subject Area" through the assembly and review stages, but do not complete the process (by clicking Accept in the Subject Area Assembly dialog box) unless the subject area is suitable for your needs.

8.13.2 How DAC Determines Tasks Required for Subject Areas

A subject area is a collection of tasks. When a subject area is defined, DAC generates a list of relevant tasks and assembles those tasks using the following logic:

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.

   Tasks that DAC automatically assigns to a subject area are indicated with the Autogenerated flag (in the Tasks subtab of the Subject Areas tab).

   You can inactivate a task from participating in a subject area by selecting the Inactive check box (in the Tasks subtab of the Subject Areas tab). When the

Inactive check box is selected, the task remains inactive even if you reassemble the subject area.

You can also remove a task from a subject area using the Add/Remove command in the Tasks subtab of the subject Areas tab, but when you remove a task it is only removed from the subject area until you reassemble the subject area.

4. Recursive selection of all tasks.

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

   Note: A task is listed only once, even if it is associated with several tables in the subject area. DAC expands or trims the total number of tasks based on the configuration rules defined as configuration tags. This process can be resource intensive as DAC loads all of the objects in the container into memory before parsing.

8.14 Creating a Subject Area

When you create a new subject area, you assign one or more fact tables to the subject area. DAC then determines which dimension and other related tables are required as well as the tasks and their order of execution.

To create a subject area

1. In DAC, select the appropriate source system container from the drop-down list in the toolbar.

   Note: 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 how to make a copy of a preconfigured source system container, see "Creating or Copying a Source System Container".

2. From the Menu bar, select Views, then select Design, and then select 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.

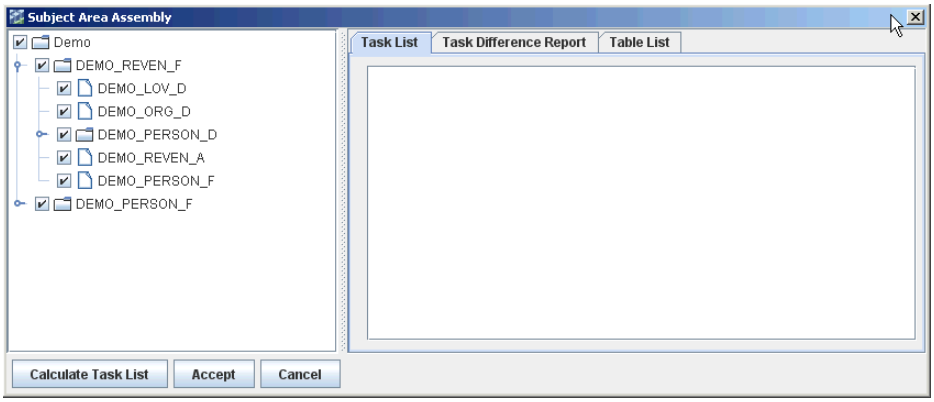
   The tables are added to the right-hand window, which represents the subject area.

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, DAC will reassemble all the subject areas listed in the top pane.

DAC assembles the selected subject area by determining what dimensions and other related tables are required.

The Subject Area Assembly dialog box appears, displaying on the left a tree view that shows the fact tables that belong to the subject area. You can expand the fact table node to view its related tables.



12. Click Calculate Task List to assemble the tasks needed to load the tables displayed in the tree view.

    If configuration tags have been assigned to any of the tasks, an information message will appear.

    A list of tasks appears in the Task List tab on the right side of the window. Also, the Table List tab displays the tables included in the subject area.

13. Click Accept to complete the subject area assembly process.

14. (Optional) Prune the subject area to better suit your business needs by removing one or more tables from the subject area.

    a. In the tree view, remove one or more tables from the subject area by deselecting the check box to the left of the table name.

    b. Click Calculate List.

    c. Review the changes to the subject area:

       The Task List tab displays the new set of tasks required to load the pruned subject area.

       The Task Difference Report tab displays the tasks that were added and the tasks that were deleted during the last assembly process.

       The Table List tab displays the tables that are included in the pruned subject area as well as the tables that were excluded during the pruning process.

    d. To accept the changes made to the subject area, click Accept.

    e. Click OK in the message box stating the subject area was successfully assembled.

15. (Optional) Click the Tasks tab to view which tasks DAC has determined are required for this subject area.

Tasks that are automatically assigned to the subject area by DAC are indicated with the Autogenerated check mark.

You can inactivate a task from participating in the subject area by selecting the Inactive check box. When the Inactive check box is selected, the task remains inactive even if you reassemble the subject area.

You can also remove a task from the subject area using the Add/Remove command, but when you remove a task it is only removed from the subject area until you reassemble the subject area.

9

Building, Running and Monitoring Execution Plans

This chapter provides information about building, running, and monitoring execution plans.

This chapter contains the following topics:

- Types of Execution Plans
- Common Extract and Load Scenarios
- Best Practices for Multi-Source Execution Plans
- How DAC Determines the Order of Task Execution within an Execution Plan
- Building and Running Single-Source and Multi-Source Execution Plans
- Unit Testing Execution Plan Tasks
- Building and Running Micro ETL Execution Plans
- Scheduling an Execution Plan
- About Refresh Dates
- Monitoring Execution Plan Processes

9.1 Types of Execution Plans

An execution plan is a unit of work that enables you to organize, schedule, and execute ETL processes. An execution plan comprises the following objects: subject areas, ordered tasks, indexes, tags, parameters, source system folders, and phases.

DAC supports single-source and multi-source execution plans, which are described in the following sections:

- About Single-Source Execution Plans
- About Multi-Source Execution Plans

9.1.1 About Single-Source Execution Plans

A single-source execution plan extracts data from a single instance of a single source system container, such as Siebel 7.8 or Oracle EBS 11. For information about data extraction and loads for single-source execution plans, see "Common Extract and Load Scenarios".

9.1.2 About Multi-Source Execution Plans

There are two types of multi-source execution plans:

- **Homogeneous**

  This type of execution plan extracts data from multiple instances of the same source system. For example, a business might have an instance of Oracle EBS 11i in one location and time zone and another instance of Oracle EBS 11i in another location and time zone. In such cases, the timing of data extraction from the different instances can be staggered to meet your business requirements.

- **Heterogeneous**

  This type of execution plan extracts data from one or more instances of *dissimilar* source systems. For example, a business might have an instance of Siebel 7.8 in one location, an instance of Oracle EBS 11i in another location, and a second instance of Oracle EBS 11i in yet a third location. You can also stagger the timing of data extraction when you use this type of execution plan.

For information about data extraction and loads for multi-source execution plans, see "Common Extract and Load Scenarios".

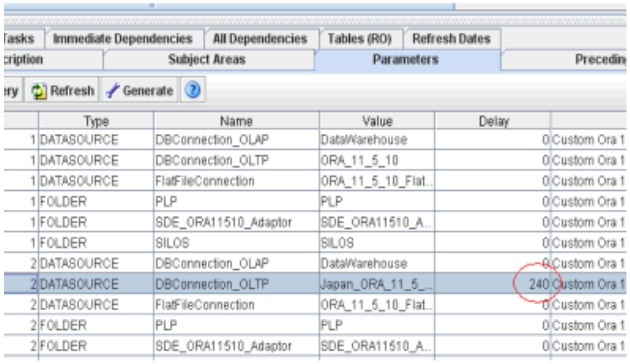
9.1.2.1 Multi-Source Order of Execution

The order in which DAC loads data from the different sources that are participating in the ETL process is determined by the priority of the physical data source connection (set in the Physical Data Sources tab of the Setup view). The Priority property ensures that tasks attempting to write to the same target table will not be in conflict.

9.1.2.2 Delay

As shown in Figure 9–1, the Delay property is located in the Parameters subtab in the Execution Plans tab of the Execute view. You set this property to specify how many minutes an extract of a data source will be delayed after the first extract of a multiple source extract process started.

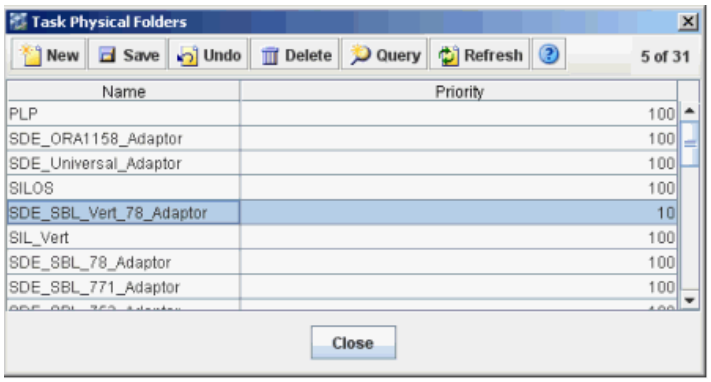
Figure 9–1 Multi-Source Delay Property



9.1.2.3 Folder Level Priorities

As shown in Figure 9–2, you need to set the folder level priority in the Task Physical Folders dialog box, which you by selecting Tools, then Seed Data, and then Task Physical Folders.

Figure 9–2 Task Physical Folders Dialog Box



9.2 Common Extract and Load Scenarios

The most common extract and load scenarios are as follows.

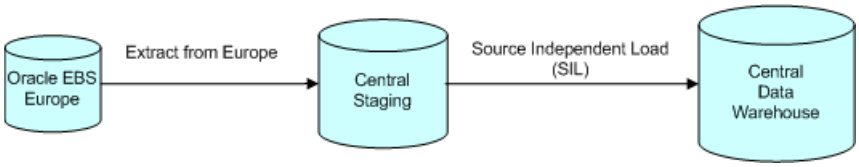
- Single Extract and Single Load Scenario
- Multiple Extract and Single Load Scenario
- Multiple Extract and Multiple Load Scenario

The multiple extract scenarios apply to both homogeneous and heterogeneous multi-source execution plan types.

9.2.1 Single Extract and Single Load Scenario

In the single extract and single load scenario, as shown in Figure 9–3, data is extracted from a single source, loaded into staging tables, and then loaded into the data warehouse.

Figure 9–3 Single Extract and Single Load Option



9.2.1.1 Truncate Table Behavior in a Single Extract Scenario

When DAC truncates a table, as part of the truncation process, it also drops and recreates indexes (if the table has indexes) and analyzes the table after the truncation.

In a single extract scenario, the truncation process for a table without indexes is as follows:

1. Truncate table.
2. Run Informatica mapping
3. Analyze table Table.

For a table with indexes, the truncation process is as follows:

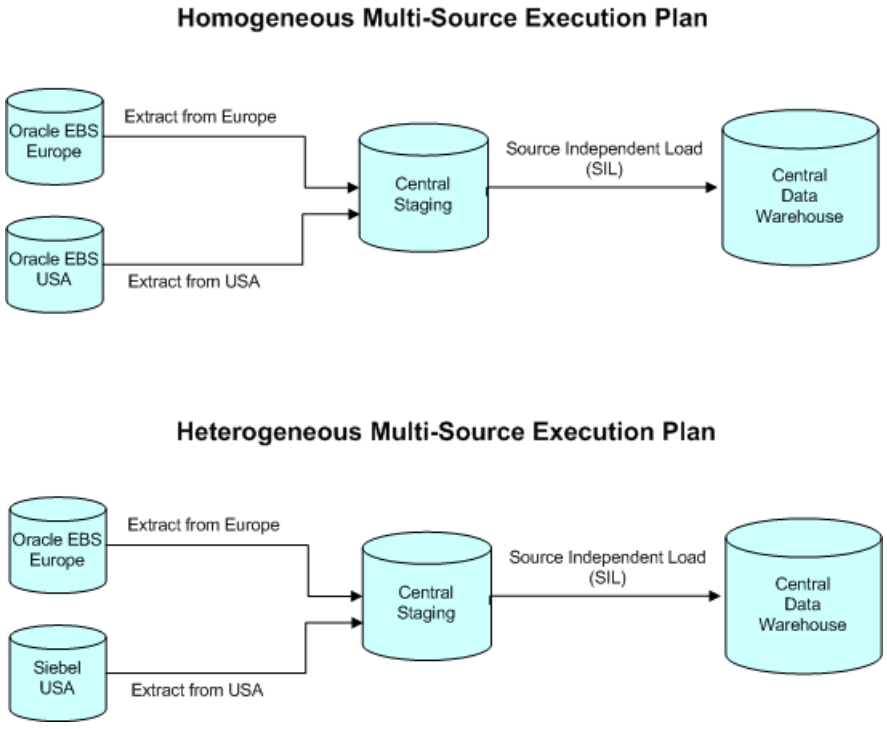
1. Truncate table.
2. Drop indexes.
3. Run Informatica mapping.
4. Recreate indexes.
5. Analyze Table.

9.2.2 Multiple Extract and Single Load Scenario

In the multiple extract and single load scenario, as shown in Figure 9–4, data is extracted from multiple sources and loaded into central staging tables. You can stagger the extracts to accommodate different time zones by setting the Delay property found in the Parameters subtab of the Execution Plans tab in the Execute view.

After all of the extract tasks have completed, the data is loaded into the data warehouse in a single load process.

Figure 9–4 Multiple Extract and Single Load Option



9.2.2.1 Truncate Table Behavior in a Multiple Extract Scenario

When DAC truncates a table, as part of the truncation process, it also drops and recreates indexes (if the table has indexes) and analyzes the table after the truncation.

If a target table is shared across different sources, it will be truncated only once. The priority of the data source determines which of the extracts truncates the tables. The task reading from the data source with the highest priority truncates the tables and drops the indices. The last task writing to the table from the data source with the highest priority creates the indexes.

In a multiple extract scenario, the truncation process for a table without indexes is as follows:

1. Truncate table (first extract task).
2. Run Informatica mapping (first extract task).
3. Run Informatica mapping (second extract task).
4. Analyze table.

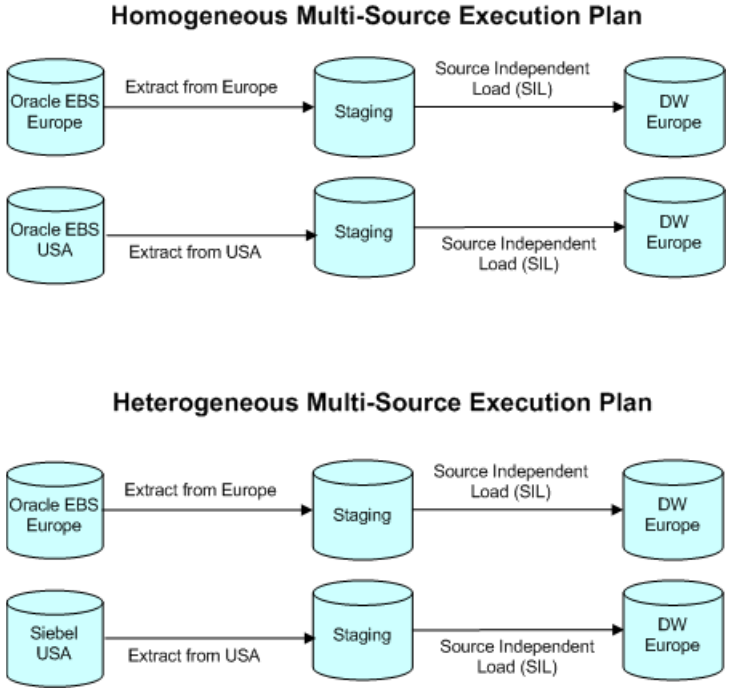
For a table with indexes, the truncation process is as follows:

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

9.2.3 Multiple Extract and Multiple Load Scenario

Figure 9–5 shows a multiple extract and multiple load scenario.

Figure 9–5 Multiple Extract and Multiple Load Scenario



9.3 Best Practices for Multi-Source Execution Plans

The following rules apply to homogeneous and heterogeneous multi-source execution plans.

Phase Dependency Blocking Behavior

DAC's Task Phase Dependency feature enables you to change the order in which tasks are executed. Task phase dependencies have a Scope property, which specifies how the Block action behaves in relation to multi-source execution plans. For more information about phase dependencies, see "Tasks Tab: Phase Dependency Subtab".

You can set the Scope property to one of the following values:

- **Both**

  Indicates the blocking action is active for tasks that have the same source and target physical data source connections.

- **Source**

  Indicates the blocking action is active for tasks that have the same source physical data source connection.

- **Target**

  Indicates the blocking action is active for tasks that have the same target physical data source connection.

- **None**

  Indicates the blocking action is active for all tasks regardless of the source and target physical data source connections.

Delay Property

The Delay property is located in the Parameters subtab in the Execution Plans tab of the Execute view. You set this property to specify how many minutes an extract of a data source will be delayed after the first extract of a multiple source extract process started.

Source Priority

For multi-source execution plans, you must assign each physical data source a priority (in the Physical Data Sources tab of the Setup view). The priority ranks the data sources and specifies the order in which DAC will load the data from the different sources. This property ensures that tasks attempting to write to the same target table will not be in conflict.

Truncate Table Behavior

If a target table is shared across different sources, it will be truncated only once. The priority of the data source determines which of the extracts truncates the tables. The task reading from the data source with the highest priority truncates the tables and drops the indexes. The last task writing to the table from the data source with the highest priority creates the indexes.

Truncate Options

Truncate options should be the same across the source system containers.

Task Groups

Tasks should belong to the same task groups across the source system containers.

Customized Tasks for Vertical Applications

In the case of Vertical Applications in which a task could have been customized in a Vertical workflow folder, the task will pick up the Vertical task folder when the source and target connections are the same based on the task folder priority.

DATASOURCE_NUM_ID

- All tables should contain the column DATASOURCE_NUM_ID.
- Unique indexes should always include the column DATASOURCE_NUM_ID.
- All the extract mappings populate the DATASOURCE_NUM_ID column from the parameter file produced by DAC.
- All the load mappings extract the value of the DATASOURCE_NUM_ID column from the staging tables.
- Level 1 aggregate tables should always contain the column DATASOURCE_NUM_ID.

Load Strategies

You should use the same load strategy for loading a table across the source system containers; that is, you should always use full loads or always use incremental loads. Avoid using a combination of load types.

Failure Restart Strategies

The Upon Failure Restart option of the Task Actions feature can be useful to restart tasks upon failure. For information about using task actions, see "About Index, Table and Task Actions".

9.4 How 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. DAC determines the order of tasks based on the following considerations:

- A task's source and target table

  The dependency algorithm 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 algorithm would determine task T2 should be executed before T1.

  The dependency algorithm 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

- DAC randomly organizes tasks that have the same property values. If some tasks need to be executed in a particular order, you can create a task group that allows you to specify an execution order.

9.5 Building and Running Single-Source and Multi-Source Execution Plans

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 PowerCenter Services and Integration Service (in the Informatica Servers tab).

Before you attempt to run a multi-source execution plan, you must first define the priority for each source. The priority specifies the order in which DAC will load the data from the different sources. For more information about the source priority property, see "Physical Data Sources Tab".

To define a source priority

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

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

To build and run a single-source or multi-source 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 top pane window or in the Edit subtab, 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 from the drop-down list.
   d. Query for the subject area you want to associate with the execution plan.
   e. Select the subject area and click Add.

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

f. Click OK to close the window.

4. Generate the runtime execution plan parameters.

a. Click the Parameters subtab, and then click Generate in the bottom pane toolbar.

b. In the Generating Parameters dialog box, enter the number of copies for each source system container, and then click OK.

c. Click OK in the informational message.

DAC automatically generates the parameters required for each copy. Not all copies require all of the possible parameters.

d. On the Parameters subtab, edit the parameters for each copy of the source system container as follows:

For each data source type, select the appropriate value from the Value drop-down list.

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

For each Informatica SIL and SDE folder, select the appropriate value in the Value drop-down list.

e. For each data source type, enter the appropriate name in the Value field.

f. (Optional) If you are extracting data from more than one source system container and want to stagger the data extracts, in the Delay field for the appropriate data source, enter a value for the number of minutes you want to delay the extract.

g. (Optional) Set the Prune Days property. This setting subtracts the Prune Days value from the LAST_REFRESH_DATE and supplies this value as the value for the $$LAST_EXTRACT_DATE parameter. See "Prune Days" for more information.

5. Click the Ordered Tasks child tab and verify the following:

a. Click Details in the toolbar, and review each task's predecessor and successor tasks to confirm tasks common to multiple sources are ordered in a manner consistent with the priority of the source connection.

b. Confirm that load tasks appear only once even if there are multiple extracts for tables common to multiple sources.

c. For tasks common to multiple sources, click Preview Run Details in the toolbar, and confirm the following:

The first common task truncates the common target table and the following tasks do not.

The first common task truncates the common target table and the following tasks do not.

For instructions on unit testing a task, see "Unit Testing Execution Plan Tasks".

6. In the top pane of the Execution Plans tab, make sure the new execution plan is highlighted, and click Build.
7. In the Building... dialog box, select the option Selected Record Only, to build only the selected execution plan.
8. 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".

9.6 Unit Testing Execution Plan Tasks

You can test how DAC will execute an individual task (and its details) without running a fully functional execution plan. And you can test tasks that belong to single-source or multi-source execution plans. All DAC tasks in an execution plan can be individually tested, regardless of their position in the dependency graph. DAC generates a parameter file named exactly as in the Informatica sessions.

Note: You cannot unit test the workflow directly from Informatica. DAC generates parameter files and log files with variable names, and, therefore, you cannot re-run workflows directly from Informatica.

To unit test an execution plan task

1. Navigate to the Execute view, then select the Execution Plans tab.
2. Select the execution plan to which the task you want to test belongs.
3. Click the Ordered Tasks child tab.
4. Select a task from the list.
5. On the toolbar, select Preview Run Details.
6. In the Preview Run Details dialog, click Execute.

   DAC executes the task and displays the results in the Preview Run Details dialog.
7. Click OK to close the Preview Run Details dialog.

9.7 Building and Running Micro ETL Execution Plans

Micro ETL execution plans are ETL processes that you schedule at very frequent intervals, such as hourly or half-hourly. They usually handle small subject areas or subsets of larger subject areas. DAC tracks refresh dates for tables in micro ETL execution plans separately from other execution plans and uses these refresh dates in the change capture process.

After a micro ETL execution plan runs, DAC populates refresh date values in the 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, 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, 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 a build and run 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. From 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. The subject areas can belong to one or more source systems.
   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 execution plan parameters.
   a. Click the Parameters subtab, and then click Generate in the bottom pane toolbar.
   b. In the Generating Parameters dialog box, enter the number of copies for each source system container, and then click OK.
   c. Click OK in the informational message.

      DAC automatically generates the parameters required for each copy. Not all copies require all of the possible parameters.
   d. On the Parameters subtab, edit the parameters for each copy of the source system container as follows:

      For each data source type, select the appropriate value from the Value drop-down list.

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

      For each Informatica SIL and SDE folder, select the appropriate value in the Value drop-down list.
   e. For each data source type, enter the appropriate name in the Value field.
   f. (Optional) If you are extracting data from more than one source system container and want to stagger the data extracts, in the Delay field for the appropriate data source, enter a value for the number of minutes you want to delay the extract.
8. In the top pane of the Execution Plans tab, make sure the new execution plan is highlighted, and click Build.

   DAC builds the execution plan.
9. Click the Ordered Tasks child tab and verify the following:
   a. Click Details in the toolbar, and review each task's predecessor and successor tasks to confirm tasks common to multiple sources are ordered in a manner consistent with the priority of the source connection.

b. Confirm that load tasks appear only once even if there are multiple extracts for tables common to multiple sources.

c. For tasks common to multiple sources, click Preview Run Details in the toolbar, and confirm the following:

   The first common task truncates the common target table and the following tasks do not.

   The first common task truncates the common target table and the following tasks do not.

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".

9.8 Scheduling an Execution Plan

Follow this procedure to schedule an execution plan.

To schedule an execution plan

1. In 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.

9.9 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). 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. 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 DAC will run the full load command.

Table 9–1 shows the possible scenarios regarding refresh dates.

Table 9–1 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.

9.10 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 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

10

Upgrading, Comparing and Merging DAC Repositories

This chapter provides instructions for using the DAC Upgrade/Merge Wizard to upgrade and merge the content of DAC repositories.

This chapter contains the following topics:

- Overview of Upgrade/Merge Wizard
- Overview of Upgrade and Merge Options
- About the Repository Upgrade (DAC 784) Option
- About the Refresh Base Option
- About the Simplified Refresh From Base Option
- About the Replace Base Option
- About the Peer to Peer Merge Option
- Resolving Object Differences in the View Difference Report

10.1 Overview of Upgrade/Merge Wizard

This section includes the following topics:

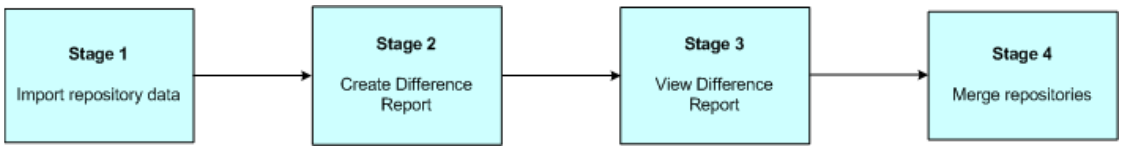
- Major Stages of the Upgrade/Merge Wizard
- Resetting the Upgrade or Merge Process

10.1.1 Major Stages of the Upgrade/Merge Wizard

The Upgrade/Merge Wizard enables you to upgrade and merge content of DAC repositories. It contains dialog boxes that guide you through the stages of an upgrade or merge, as shown in Figure 10–1.

Note: Figure 10–1 does not apply to the Simplified Refresh From Base upgrade option. For more information about this option, see "About the Simplified Refresh From Base Option".

Figure 10–1 Major Stages of the Upgrade/Merge Wizard



The Upgrade/Merge Wizard enables you to suspend an upgrade or merge while in process and return to it at a later time. You can also repeat a stage that you just completed or reset the process to start from the beginning.

Table 10–1 summarizes the possible actions you can take at each stage of the upgrade or merge process.

Table 10–1 Possible Upgrade and Merge Actions

Stage	Possible Actions
After you have completed Stage 1: Import repository data	You can perform one of the following actions: ■ Repeat Stage 1: Import repository data ■ Perform Stage 2: Create Difference Report ■ Reset the process
After you have completed Stage 2: Create Difference Report	You can perform one of the following actions: ■ Repeat Stage 2: Create Difference Report ■ Perform Stage 3: View Difference Report ■ Reset the process
After you have completed Stage 3: View Difference Report	You can perform one of the following actions: ■ Repeat Stage 3: View Difference Report ■ Perform Stage 4: Merge repositories ■ Reset the process

10.1.2 Resetting the Upgrade or Merge Process

The Upgrade/Merge Wizard enables you to reset an upgrade or merge process at any stage you are in until the final stage in which you merge the repositories. For a description of the stages of the Upgrade/Merge Wizard, see "Major Stages of the Upgrade/Merge Wizard".

When you reset an upgrade or merge process, the Upgrade/Merge Wizard reverts all the actions you have already taken in the upgrade or merge process.

To reset an upgrade or merge process

1. Open the Upgrade/Merge Wizard by selecting from the DAC toolbar Tools, then DAC Repository Management, and then Upgrade/Merge Wizard.

   The Upgrade/Merge Wizard dialog box displays the type of upgrade or merge process you previously started.

2. From the Perform drop-down list, select Reset and click OK.
3. In the Reset Upgrade Process dialog box, re-type the text in the text box to confirm you want to proceed, and click Yes.

   An informational message tells you the reset process was successful.

4. Click OK.

10.2 Overview of Upgrade and Merge Options

The Upgrade/Merge Wizard includes the following options for upgrading and merging DAC repositories:

- **Repository Upgrade (DAC 784)**

  Use this option to upgrade a DAC Repository in the release 7.8.4 format, which has a *non-partitioned* DAC Repository structure, to the new release, which has a *partitioned* DAC Repository structure. For more information, see "About the Repository Upgrade (DAC 784) Option".

- **Refresh Base**

  Use this option to upgrade the DAC Repository when you are upgrading from an older release of Oracle BI Applications to a new release. For more information, see "About the Refresh Base Option".

- **Simplified Refresh From Base**

  This option is similar to the Refresh Base option. It allows you to upgrade the DAC Repository from an older release of Oracle BI Applications to a new release without comparing repositories and creating a Difference Report. For more information, see "About the Simplified Refresh From Base Option".

- **Replace Base**

  Use this option to upgrade the DAC Repository when you are phasing out an older release of a transactional application and moving to a newer release, for example, phasing out Siebel 7.5.3 and moving to Siebel 7.8. For more information, see "About the Replace Base Option".

- **Peer to Peer Merge**

  Use this option to merge DAC repositories of different instances of the same release. For example, in a development environment you may have two instances of a DAC Repository used with Oracle BI Applications release 7.9.5 that you want to merge. For more information see, "About the Peer to Peer Merge Option".

10.3 About the Repository Upgrade (DAC 784) Option

This section includes the following topics:

- Repository Upgrade (784): High-Level Process Flow
- Repository Upgrade (784): Procedure for Upgrading

DAC versions before Oracle BI Applications 7.9 had a non-partitioned DAC Repository structure and held metadata for a single application. DAC versions released with Oracle BI Applications 7.9 and higher and with the DAC 10.1.3.4 release have partitioned DAC Repository structures. The partitioned structure, also known as a container, can hold metadata for multiple applications.

The Repository Upgrade (DAC 784) option enables you to upgrade from the DAC Repository release 7.8.4 (non-partitioned) to the new partitioned structure.

Note: If you want to upgrade a pre-7.8.4 release of a DAC Repository, you must first upgrade the repository to the 7.8.4 release before you can use the Repository Upgrade (DAC 784) option to upgrade to the new release.

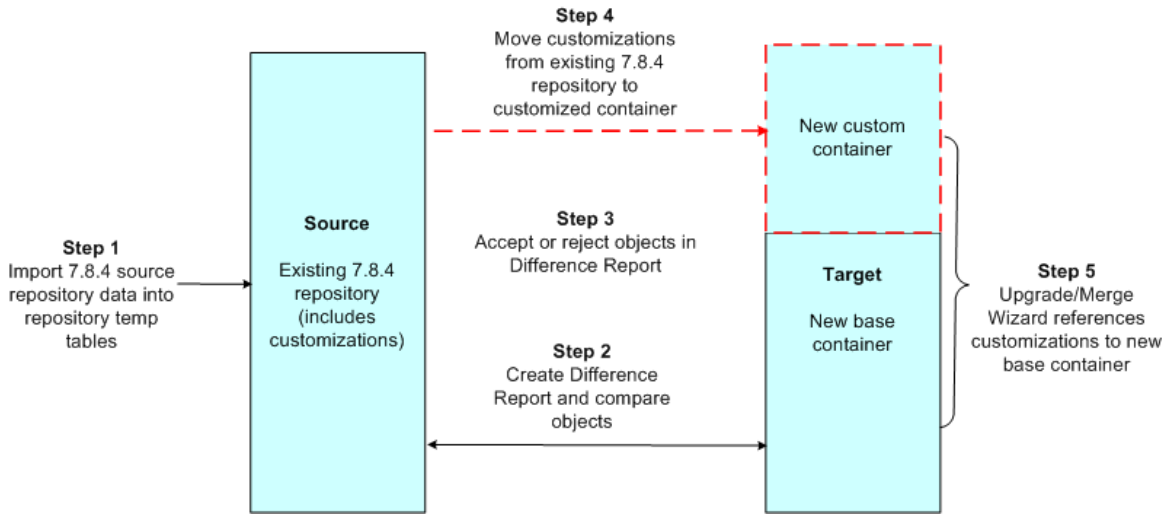
Note: In the View Difference Report, "Accept Source" may appear as the default merge action. However, you should select "Accept Target" as the merge action when the phase is shown as being different between the source and target.

10.3.1 Repository Upgrade (784): High-Level Process Flow

Figure 10–2 shows a high-level process flow for using the Repository Upgrade (DAC 784) option to upgrade an existing release 7.8.4 DAC Repository to the new release.

The term *base* is used to refer to a source or target container that you have not changed or customized in any way. The base container is also referred

Figure 10–2 Upgrade Process for Repository Upgrade (DAC 784) Option



In Step 1 of the high-level process flow, you import the existing 7.8.4 DAC Repository into the repository temporary tables. The existing 7.8.4 repository is referred to as the *source* in the Upgrade/Merge Wizard.

In Step 2, you create a Difference Report that compares the existing repository with the new base container, which is referred to as the *target*. (The new base container is the version to which you are upgrading and is the version you imported from the file system.)

In Step 3, you accept or reject the objects that the Difference Report shows as being either present or changed in the source but not the target. See Section 10.8, "Resolving Object Differences in the View Difference Report" for a description of how objects are merged based on the action you take.

In Step 4, after you have resolved the differences, you then execute the merge. In Step 5, the Upgrade/Merge Wizard references the customizations in the newly merged custom container with the new base container.

10.3.2 Repository Upgrade (784): Procedure for Upgrading

Follow this procedure to upgrade a DAC Repository in the release 7.8.4 format to the new release.

> **Note:** You cannot use the Repository Upgrade (784) option with pre-7.8.4 releases of the DAC. To confirm your existing repository is in the release 7.8.4 format, open the 7.8.4 DAC client and in the toolbar, select Help, and then select About DAC. The About DAC dialog box displays the version of DAC you are running.

Before you begin this procedure, you need to have already installed the new release of Oracle BI Applications and imported the metadata that is the same version as the metadata in the existing 7.8.4 DAC Repository. For example, if the existing 7.8.4 repository contains Siebel 6.3 metadata, you need to have imported the Siebel 6.3 source system container metadata from the new Oracle BI Applications release. If you are using Oracle BI Applications 7.9.x with the DAC 7.9.x, then you should have the correct Siebel source system container in the 7.9.x DAC Repository.

You should also review the section "Resolving Object Differences in the View Difference Report" to gain an understanding of your options for resolving object differences.

To upgrade a DAC Repository in the release 7.8.4 format to the new release

1. Edit the datamapping.xml file to reflect the upgrade environment.
   a. Navigate to the folder DAC\conf\upgrade
   b. Open the datamapping.xml file for editing.
   c. Edit the entries for folder, database connection, and subject area to reflect the upgrade environment. See the comments in the datamapping.xml file for instructions.
2. Configure the connection between the release 7.8.4 DAC Repository and the new DAC client.
   a. In the Setup view, click Physical Data Sources.
   b. Click New to create a new record for the 7.8.4 DAC Repository.
   c. Enter the appropriate information to connect to the release 7.8.4 DAC Repository database. For information about the required fields, see the section "Physical Data Sources Tab".

      > **Note:** The value for the Type field must be set to DAC Repository.

   d. Click Test Connection to confirm the connection works.
   e. Click Save.
3. Navigate to the Upgrade/Merge Wizard by selecting Tools, then DAC Repository Management, and then Upgrade/Merge Wizard.
4. From the drop-down list, select Repository Upgrade (DAC 784), and then click OK.

   The Import 7.8.4 Repository dialog box appears.

5. From the 784 Repository drop-down list, select the repository you configured in Step 1.

   Note: In order for a repository to appear in the drop-down list, it must be configured in the Physical Data Sources tab of the Setup view.

6. From the Source System Container drop-down list, select the new base container to which you are upgrading. This is the container you will compare against the existing release 7.8.4 DAC Repository.

   Caution: Make sure you select the appropriate container. If you select a container different from that contained in the existing release 7.8.4 DAC Repository, the Difference Report will compare dissimilar source system containers and will be inaccurate.

7. Select the categories of metadata from the existing release 7.8.4 DAC Repository you want to import into the repository temporary tables for comparison with the new base container. The categories are as follows:

Categories Options	Description
Logical	Imports all information contained in the DAC Design view and the execution plan information for the DAC Execute view.
Run Time	Imports ETL Run History and the last refresh date information.
System	Imports all information contained in the DAC Setup view, except passwords for servers and database connections.

   The release 7.8.4 DAC Repository tables are imported into the temporary tables.

8. Click OK in the Importing Tables dialog box when the process is finished.

   The Create Difference dialog box appears.

9. Create the Difference Report to compare the differences between the source and target DAC repositories.

   a. Enter a name for the Difference Report, or leave the default name.

   b. From the Source System Container drop-down list, select the new base container.

   c. In the New Custom Container ID/Name fields, enter an ID and a name for the custom container that will be created during the upgrade process.

      The ID and Name fields are alphanumeric. The Name field can contain spaces and must be at least five characters long. The ID field cannot contain spaces.

   d. (Optional) Enter a description for the Difference Report.

   e. Click OK.

      When the Difference Report is complete, the Creating Difference Report dialog box tells you how long the process took.

   f. Click OK.

      The View Difference Report dialog box displays the differences between the existing and new DAC repositories.

10. In the View Difference Report dialog box, resolve the differences between the existing (source) and new (target) containers, that is, between the release 7.8.4 DAC Repository and the repository that contains the new base container. For detailed information about the View Difference Report, see "Resolving Object Differences in the View Difference Report".

    To resolve the differences, you either accept or reject the objects that appear in the release 7.8.4 DAC Repository but do not appear in the new base container to which you are upgrading.

    a. In the navigation tree, select the repository object for which you want to view the differences between the existing and new repositories.

       If the object selected in the navigation tree is a hierarchical object, the subtabs for the child objects appear in the bottom pane of the Object Difference window.

    b. (Optional) Filter the objects that appear in the Object Difference window by selecting one of the options from the drop-down list in the toolbar.

    c. For parent objects in the top pane and any child objects in the bottom pane, accept or reject the object in the difference list by selecting or deselecting the Accept Source check box.

       For detailed information about these options and the merge outcome, see "Possible Repository Merge Outcomes Based on Your Decisions".

       Note: If a child object has been changed but not the parent object, the parent object will still appear in the Object Difference window even though it has not been changed.

    d. (Optional) Once you have made a decision about whether to accept or reject the difference, select the Resolved check box to indicate you have resolved the object.

    e. Repeat Steps a, b, and c, until you have resolved all object differences.

    f. Click Merge.

       The Merge dialog box appears and lists the details of the merge.

11. Click Merge to begin the merge process.

12. Click OK in the Merging Repositories dialog box when the merge process is complete.

10.4 About the Refresh Base Option

This section includes the following topics:

- Refresh Base: High-Level Process Flow
- Refresh Base: Procedure for Upgrading

The Refresh Base option enables you to upgrade an existing customized DAC Repository. You should use this option if you are upgrading from a DAC release higher than 7.8.4. If you are upgrading a DAC Repository in the release 7.8.4 format or lower-numbered releases, see "About the Repository Upgrade (DAC 784) Option".

The Refresh Base option enables you to compare a new base container with the existing customized repository and to create a Difference Report. If you want to

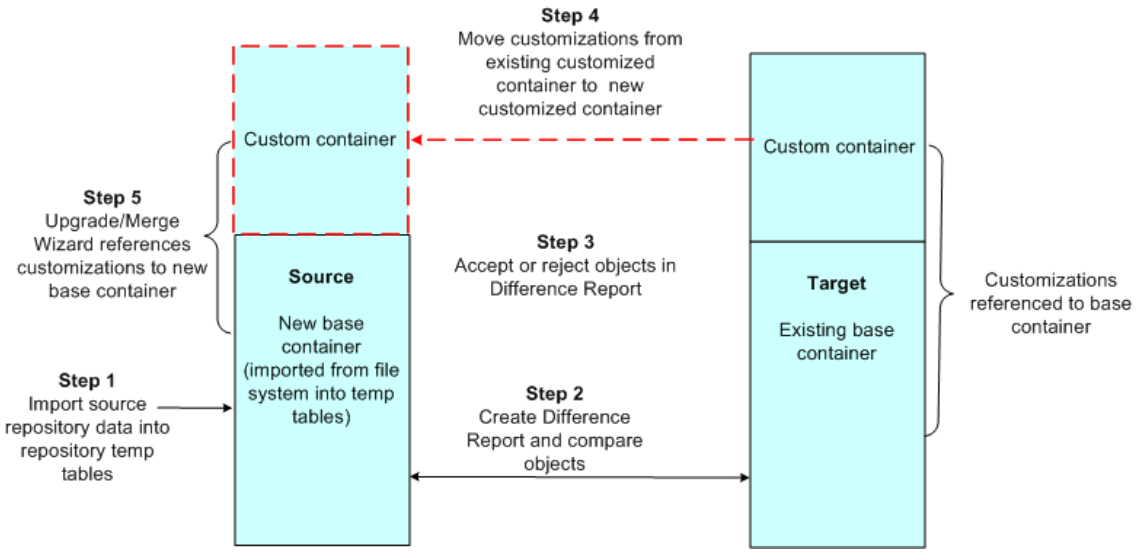
upgrade an existing customized DAC Repository without comparing repositories and creating a Difference Report, you can use the Simplified Refresh From Base option. See "About the Simplified Refresh From Base Option" for more information.

10.4.1 Refresh Base: High-Level Process Flow

Figure 10–3 shows a high-level process flow for using the Refresh Base option to upgrade existing customized repositories from DAC releases higher than 7.8.4, such as Oracle BI Applications 7.9 and higher.

The term *base* is used to refer to a source or target container that you have not changed or customized in any way.

Figure 10–3 Upgrade Process for Refresh Base Option



In Step 1 of the high-level process flow, you import the repository data for the new base container from the file system into the repository temporary tables. This repository is referred to as the *source* in the Upgrade/Merge Wizard.

In Step 2, you create a Difference Report that compares the new base container with the existing repository (including customizations). The existing customized repository is referred to as the *target*.

In Step 3, you accept or reject the objects that the Difference Report shows as being present or changed in the source but not the target. See Section 10.8.3, "Possible Repository Merge Outcomes Based on Your Decisions" for a description of how objects are merged based on the action you take.

In Step 4, after you have resolved the differences, you then execute the merge. In Step 5, the DAC references the customizations in the newly merged repository with the new base container.

Note: The repository data is imported from a file system and should be in the DAC 10.1.3.4 format. If it is in the 7.9.x format, you should do the following:

1. Restore the repository data into a database using the 7.9.x DAC (using the regular DAC import process).
2. Install and configure DAC 10.1.3.4 to the 7.9.x repository.
3. Export the relevant source system container to a file folder.

   The metadata in the folder in Step 3 above will become the source for this upgrade.

10.4.2 Refresh Base: Procedure for Upgrading

Follow this procedure to use the Refresh Base option to upgrade an existing customized repository from DAC releases higher than 7.8.4.

Before you begin this procedure you should review the section "Resolving Object Differences in the View Difference Report" to gain an understanding of your options for resolving object differences.

To upgrade an existing repository using the Refresh Base option

1. Navigate to the Upgrade/Merge Wizard by selecting Tools, then DAC Repository Management, and then Upgrade/Merge Wizard.
2. From the drop-down list, select Refresh Base, and then click OK.

   The Import Source System Container dialog box appears.
3. Click Change import/export folder to navigate to the directory that holds the metadata files for the new base container to which you are upgrading.
4. Select the container to which you are upgrading from the Source System Container drop-down list, and click OK.
5. In the Importing Tables dialog box, re-type the text in the text box to confirm you want to proceed, and click Yes.

   When the import process is complete, the Importing Tables dialog box tells you how long the process took.
6. Click OK.

   The Create Difference Report dialog box appears.
7. Create the Difference Report to view the differences between the new and existing DAC repositories.
   a. Enter a name for the Difference Report, or leave the default name.
   b. Select the appropriate existing container.
   c. (Optional) Enter a description for the Difference Report.
   d. Click OK.

      When the Difference Report is complete, the Creating Difference Report dialog box tells you how long the process took.
   e. Click OK.

      The View Difference Report dialog box displays the differences between the new and existing DAC repositories.

8. In the View Difference Report dialog box, resolve the differences between the new repository (source) and existing repository (target). For detailed information about the View Difference Report, see "Resolving Object Differences in the View Difference Report".

   To resolve the differences, you either accept or reject the objects that are listed as new or changed in the new repository (the version to which you are upgrading).

   a. In the navigation tree, select the repository object for which you want to view the differences between the new and existing repositories.

      If the object selected in the navigation tree is a hierarchical object, the subtabs for the child objects appear in the bottom pane of the Object Difference window.

   b. (Optional) Filter the objects that appear in the top, right window by selecting one of the options from the drop-down list in the toolbar.

   c. For parent objects in the top pane and any child objects in the bottom pane, accept or reject the object in the difference list by selecting or deselecting the Accept Source check box.

      For detailed information about these options and the merge outcome, see "Possible Repository Merge Outcomes Based on Your Decisions".

      Note: If a child object has been changed but not the parent object, the parent object will still appear in the Object Difference window even though it has not been changed.

   d. (Optional) Once you have made a decision about whether to accept or reject the difference, select the Resolved check box to indicate you have resolved the object.

   e. Repeat Steps a, b, and c, until you have resolved all object differences.

   f. Click Merge.

      The Merge dialog box appears and lists the details of the merge.

9. Click Merge to begin the merge process.

10. Click OK in the Merging Repositories dialog box when the merge process is complete.

10.5 About the Simplified Refresh From Base Option

The Simplified Refresh From Base option is similar to the Refresh Base option. It allows you to upgrade from a DAC Repository from an older release of Oracle BI Applications to a new release, but you cannot compare repositories and create a Difference Report.

If you want to upgrade a DAC Repository from an older release of Oracle BI Applications to a new release and you want to compare repositories and create a Difference Report before merging the repositories, you must use the Refresh Base option. See "About the Refresh Base Option" for more information.

Before you begin this procedure, do the following:

- Determine what customizations were made to your existing DAC Repository.

- Make sure you have renamed and backed up your existing DAC Repository into a different database. When you backup the DAC Repository, you export the DAC metadata, in XML format (using the DAC's Export tool), into a folder other than the standard DAC export folder where backups are stored (DAC\export). For instructions on exporting DAC metadata, see "Exporting DAC Metadata".

To upgrade the DAC metadata repository

1. Upgrade your existing DAC Repository tables to be compatible with the new DAC Repository release.
   a. Configure the new DAC client version to read the DAC metadata from your existing DAC Repository.
   b. Log in to the DAC and select Yes if prompted to upgrade the repository tables.
2. Export the custom applications from your existing custom repository to a folder other than the standard DAC export folder (DAC\export) or the folder into which you backed up metadata in the previous format.
3. Import the new DAC metadata for your application from the standard DAC export folder (DAC\export). Select the Truncate Repository Tables check box.

   For instructions on importing DAC metadata, see "Importing DAC Metadata".
4. Import the customized DAC metadata that you exported in Step 2, and deselect the Truncate Repository Tables check box.

   This will append all the custom data to the repository, thus bringing in the customizations, which include the following:
   - All modified data.
   - All newly created custom data.
   - All deleted data.
5. Refresh the source system container to locate any missing objects in your customized application. The missing objects are any new objects that the preconfigured applications may have that are not referenced in the custom applications.
   a. Navigate to the Upgrade/Merge Wizard by selecting Tools, then DAC Repository Management, and then Upgrade/Merge Wizard.
   b. From the drop-down list, select Simplified Refresh From Base, and then click OK.
   c. Select the appropriate container from the Source System Container drop-down list, and click OK.
   d. Confirm that you want to refresh the source system container.
6. Rebuild all the subject areas and execution plans in your customized application to include any new changes in the object dependency. For information about building subject areas and execution plans, see "Customizing DAC Objects and Designing Subject Areas" and "Building, Running and Monitoring Execution Plans".

10.6 About the Replace Base Option

This section includes the following topics:

- Replace Base: High-Level Process Flow

- Replace Base: Procedure for Upgrading

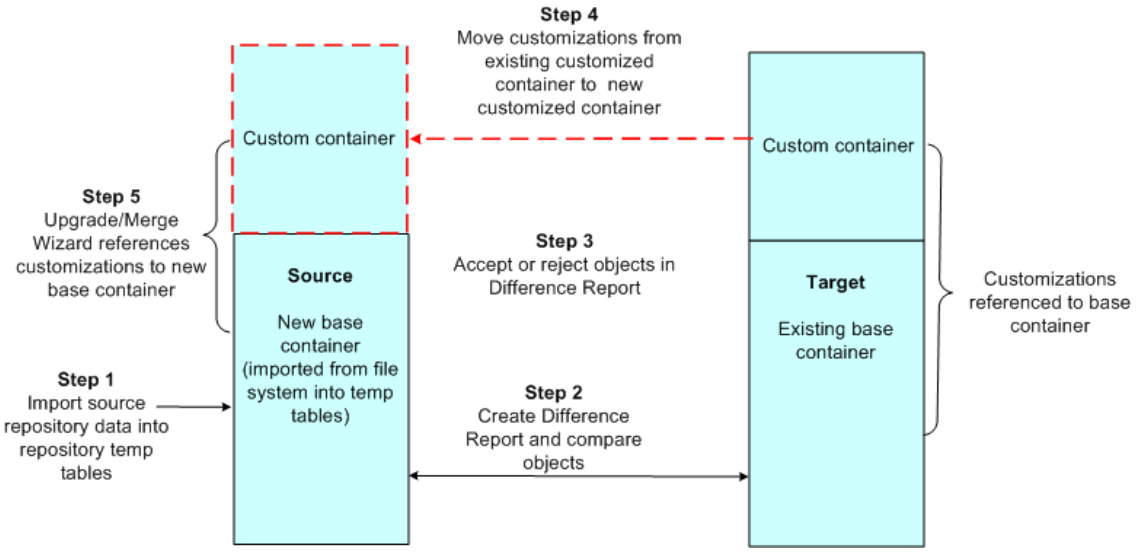
The Replace Base option enables you to upgrade the DAC Repository when you are phasing out an older release of a transactional application and moving to a newer release, for example, phasing out Siebel 7.5.3 and moving to Siebel 7.8.

10.6.1 Replace Base: High-Level Process Flow

Figure 10–4 shows a high-level process flow for using the Replace Base option to upgrade the DAC Repository when you are phasing out an older release of a transactional application and moving to a newer release.

The term *base* is used to refer to a source or target container that you have not changed or customized in any way.

Figure 10–4 Upgrade Process for Replace Base Option



In Step 1 of the high-level process flow, you import the repository data for the new base container into the repository temporary tables. This repository is referred to as the *source* in the Upgrade/Merge Wizard

In Step 2, you create a Difference Report that compares the new base container (source repository) with the existing base container (including customizations). The existing base container is referred to as the *target*

In Step 3, you accept or reject the objects that the Difference Report shows as being present in the source but not the target or changed in the source but not the target. See Section 10.8.3, "Possible Repository Merge Outcomes Based on Your Decisions" for a description of how objects are merged based on the action you take.

In Step 4, after you have resolved the differences, you then execute the merge. In Step 5, the DAC references the customizations in the newly merged repository with the new base container.

Note: The repository data is imported from a file system and should be in the DAC 10.1.3.4 format. If it is in the 7.9.x format, you should do the following:

1. Restore the repository data into a database using the 7.9.x DAC (using the regular DAC import process).
2. Install and configure DAC 10.1.3.4 to the 7.9.x repository.
3. Export the relevant source system container to a file folder.

   The metadata in the folder in Step 3 above will become the source for this upgrade.

10.6.2 Replace Base: Procedure for Upgrading

Follow this procedure to use the Replace Base option to upgrade the DAC Repository when you are phasing out an older release of a transactional application and moving to a newer release.

Before you begin this procedure you should review the section "Resolving Object Differences in the View Difference Report" to gain an understanding of your options for resolving object differences.

To upgrade an existing DAC Repository

1. Navigate to the Upgrade/Merge Wizard by selecting Tools, then DAC Repository Management, and then Upgrade/Merge Wizard.
2. From the drop-down list, select Replace Base, and then click OK.

   The Import Source System Container dialog box appears.
3. Click Change import/export folder to navigate to the directory that holds the metadata files for the new base container to which you are upgrading.
4. Select the appropriate container from the Source System Container drop-down list, and click OK.
5. In the Importing Tables dialog box, re-type the text in the text box to confirm you want to proceed, and click Yes.

   When the import process is complete, the Importing Tables dialog box tells you how long the process took.
6. Click OK.

   The Create Difference Report dialog box appears.
7. Create the Difference Report to view the differences between the new and existing DAC repositories.
   a. Enter a name for the Difference Report, or leave the default name.
   b. Select the appropriate existing container.
   c. (Optional) Enter a description for the Difference Report.
   d. Click OK.

      When the Difference Report is complete, the Creating Difference Report dialog box tells you how long the process took.
   e. Click OK.

The View Difference Report dialog box displays the differences between the new and existing DAC repositories.

8. In the View Difference Report dialog box, resolve the differences between the new repository (source) and existing repository (target). For detailed information about the View Difference Report, see "Resolving Object Differences in the View Difference Report".

   To resolve the differences, you either accept or reject the objects that are listed as new or changed in the new repository (the version to which you are upgrading).

   a. In the navigation tree, select the repository object for which you want to view the differences between the new and existing repositories.

      If the object selected in the navigation tree is a hierarchical object, the subtabs for the child objects appear in the bottom pane of the Object Difference window.

   b. (Optional) Filter the objects that appear in the top, right window by selecting one of the options from the drop-down list in the toolbar.

   c. For parent objects in the top pane and any child objects in the bottom pane, accept or reject the object in the difference list by selecting or deselecting the Accept Source check box.

      For detailed information about these options and the merge outcome, see "Possible Repository Merge Outcomes Based on Your Decisions".

      Note: If a child object has been changed but not the parent object, the parent object will still appear in the Object Difference window even though it has not been changed.

   d. (Optional) Once you have made a decision about whether to accept or reject the difference, select the Resolved check box to indicate you have resolved the object.

   e. Repeat Steps a, b, and c, until you have resolved all object differences.

   f. Click Merge.

      The Merge dialog box appears and lists the details of the merge.

9. Click Merge to begin the merge process.

10. Click OK in the Merging Repositories dialog box when the merge process is complete.

10.7 About the Peer to Peer Merge Option

This section includes the following topics:

- Peer to Peer Merge: High-Level Process Flow
- Peer to Peer Merge: Procedure for Merging

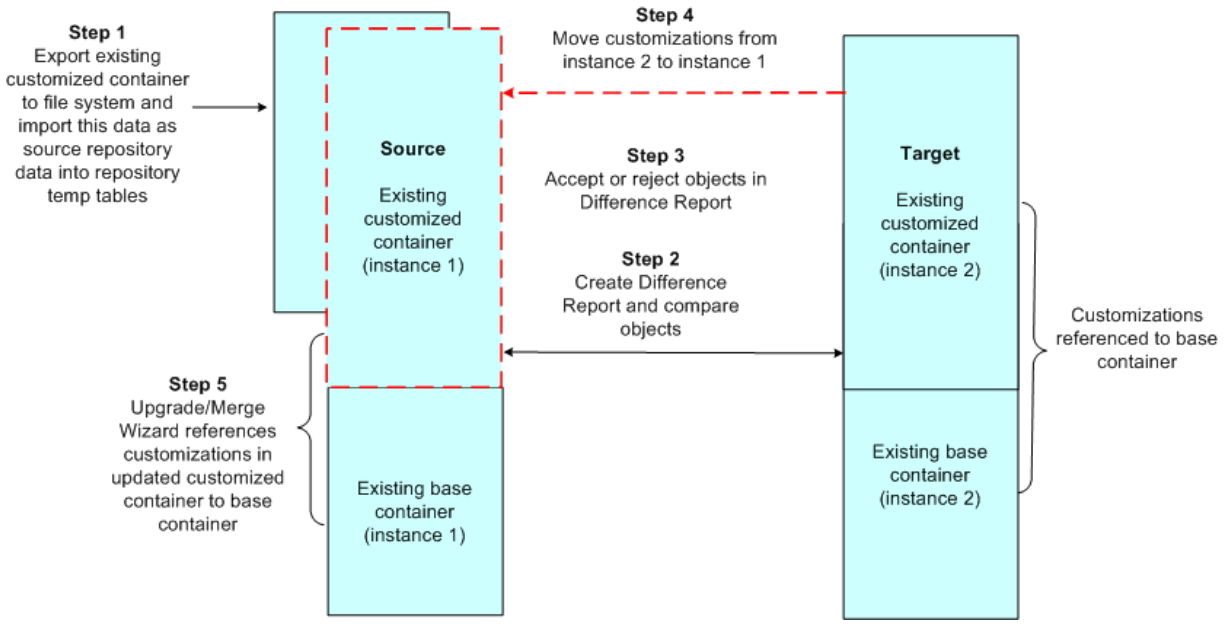
The Peer to Peer Merge option enables you to merge DAC repositories of different instances of the same release. For example, in a development environment you may have two instances of a DAC Repository used with Oracle BI Applications release 7.9.5 that you want to merge.

10.7.1 Peer to Peer Merge: High-Level Process Flow

Figure 10–5 shows a high-level process flow for using the Peer to Peer Merge option to merge DAC repositories of different instances of the same release.

The term *base* is used to refer to a source or target container that you have not changed or customized in any way.

Figure 10–5 Merge Process for Peer to Peer Merge Option



In Step 1 of the high-level process flow, you export one instance of the existing customized source system container to the file system and then import this container into the repository temporary tables. This repository is referred to as the *source* in the Upgrade/Merge Wizard.

In Step 2, you create a Difference Report that compares the instance 1 container (including customizations) with the instance 2 container (including customizations). The instance 2 container is referred to as the *target*.

In Step 3, you accept or reject the objects that the Difference Report shows as being present or changed in the source but not the target. See Section 10.8.3, "Possible Repository Merge Outcomes Based on Your Decisions" for a description of how objects are merged based on the action you take.

In Step 4, after you have resolved the differences, you then execute the merge. In Step 5, the Upgrade/Merge Wizard references the customizations in the newly merged container with the instance 1 base container.

10.7.2 Peer to Peer Merge: Procedure for Merging

Follow this procedure to use the Peer to Peer Merge option to merge DAC repositories of different instances of the same release.

Before you begin this procedure you should review the section "Resolving Object Differences in the View Difference Report" to gain an understanding of your options for resolving object differences.

To merge two DAC repositories of different instances of the same release

1. Navigate to the Upgrade/Merge Wizard by selecting Tools, then DAC Repository Management, and then Upgrade/Merge Wizard.
2. From the drop-down list, select Replace Base, and then click OK.

   The Import Source System Container dialog box appears.
3. Click Change import/export folder to navigate to the directory that holds the metadata files for instance 1 of the source system container you want to merge.
4. Select the appropriate container from the Source System Container drop-down list, and click OK.
5. In the Importing Tables dialog box, re-type the text in the text box to confirm you want to proceed, and click Yes.

   When the import process is complete, the Importing Tables dialog box tells you how long the process took.
6. Click OK.

   The Create Difference Report dialog box appears.
7. Create the Difference Report to view the differences between the instance 1 container and the instance 2 container.
   a. Enter a name for the Difference Report, or leave the default name.
   b. In the Existing Container drop-down list, select the instance 2 container.
   c. (Optional) Enter a description for the Difference Report.
   d. Click OK.

      When the Difference Report is complete, the Creating Difference Report dialog box tells you how long the process took.
   e. Click OK.

      The View Difference Report dialog box displays the differences between the instance 1 and instance 2 containers.
8. In the View Difference Report dialog box, resolve the differences between the instance 1 and instance 2 DAC repositories. The instance 1 repository is referred to as the source or existing container, and instance 2 as the target or new container. For detailed information about the View Difference Report, see "Resolving Object Differences in the View Difference Report".

   To resolve the differences, you either accept or reject the objects that appear as new or changed in the instance 1 container but do not appear in the instance 2 container.
   a. In the navigation tree, select the repository object for which you want to view the differences between the instance 1 and instance 2 containers.

      If the object selected in the navigation tree is a hierarchical object, the subtabs for the child objects appear in the bottom pane of the Object Difference window.
   b. (Optional) Filter the objects that appear in the top, right window by selecting one of the options from the drop-down list in the toolbar.
   c. For parent objects in the top pane and any child objects in the bottom pane, accept or reject the object in the difference list by selecting or deselecting the Accept Source check box.

For detailed information about these options and the merge outcome, see "Possible Repository Merge Outcomes Based on Your Decisions".

Note: If a child object has been changed but not the parent object, the parent object will still appear in the Object Difference window even though it has not been changed.

d. (Optional) Once you have made a decision about whether to accept or reject the difference, select the Resolved check box to indicate you have resolved the object.

e. Repeat Steps a, b, and c, until you have resolved all object differences.

f. Click Merge.

The Merge dialog box appears and lists the details of the merge.

9. Click Merge to begin the merge process.

10. Click OK in the Merging Repositories dialog box when the merge process is complete.

10.8 Resolving Object Differences in the View Difference Report

This section includes the following topics:

- Overview of View Difference Report
- View Difference Report Interface
- Possible Repository Merge Outcomes Based on Your Decisions

The Upgrade/Merge Wizard generates a View Difference Report for the following upgrade and merge options:

- Repository Upgrade (DAC 784)
- Refresh Base
- Replace Base
- Peer to Peer Merge

A View Difference Report is not available if you select to upgrade using the Simplified Refresh From Base option. For more information about the upgrade and merge options, see "Overview of Upgrade and Merge Options".

10.8.1 Overview of View Difference Report

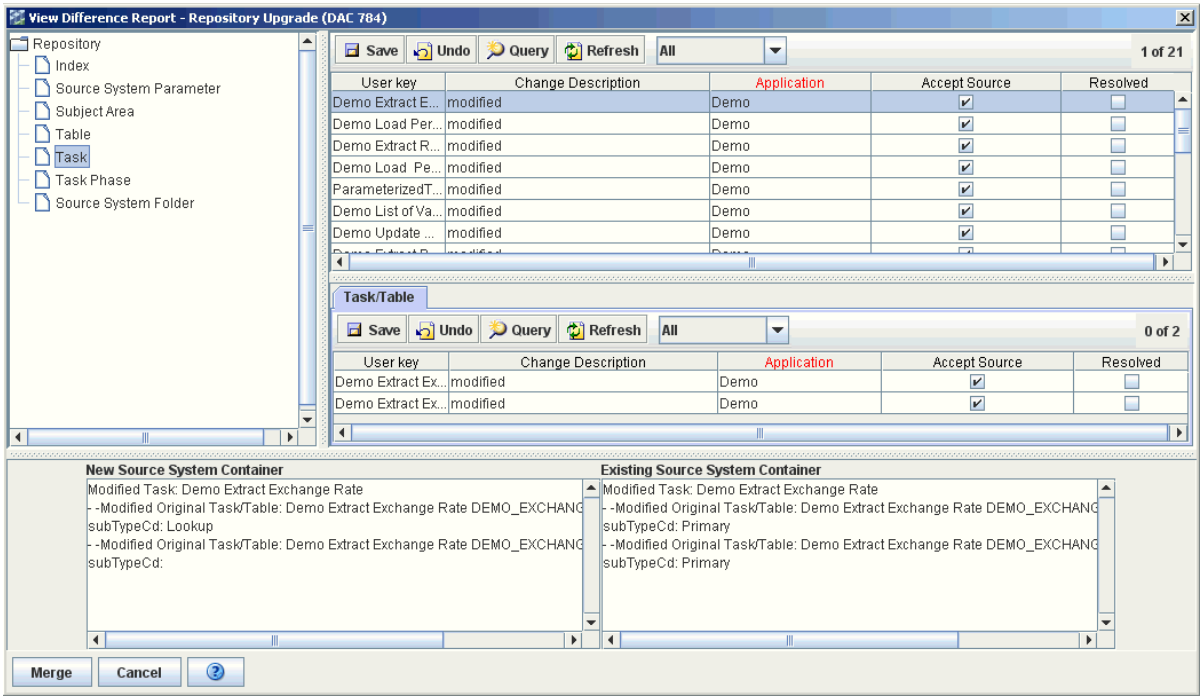
The View Difference Report dialog box enables you to resolve the differences between the existing and new DAC repositories. To resolve the differences, you either accept or reject the objects that appear in the existing repository but do not appear in the new repository.

Objects in a source system container have an ownership property. The categories of ownership are original, reference, and clone. The ownership property is maintained when an object from the existing repository is merged with the new repository. For example, if an object in the existing source system container is a reference and you want to merge it into the new repository, it will be moved into the new container as a reference. For more information about object ownership properties, see "About Object Ownership in DAC".

Table 10–2 shows the merge outcomes based on object type.

10.8.2 View Difference Report Interface

This section describes the main features of the View Difference Report.



- **Navigation Tree**

  The navigation tree displays the repository object types that are compared in the existing and new repositories. When you select a repository object in the navigation tree, the differences are listed in the object difference window.

- **Object Difference Window**

  The Object Difference window lists the differences found between the existing and new containers for the object type selected in the navigation tree. If the object selected in the navigation tree is a hierarchical object, the subtabs for the child objects appear below the Object Difference window. The following columns appear in the Object Difference window and in the subtabs:

  – **User Key**

    Unique identifier for the object.

  – **Change Description**

    Indicates the type of difference that exists between the containers.

  – **Container**

    The new or target container.

  – **Accept Source**

Selecting the Accept Source check box indicates you want to accept the change that appears in the source container. For detailed information about the possible merge outcomes, see "Possible Repository Merge Outcomes Based on Your Decisions".

  - **Resolved**

    Optional field that indicates the object difference has been resolved. Note: This field is for user reference only. It does not have an affect on the merge process.

- **Subtabs for Child Objects**

  If the object selected in the navigation tree is a hierarchical object, the subtabs for the related child objects appear in the bottom pane of the Object Difference window.

- **Text Fields**

  The text fields in at the bottom of the dialog, labeled "New Source System Container "and "Existing Source System Container," display a textual and hierarchical representation of the object difference selected in the Object Difference window.

- **Query Functionality**

  You can query for objects in the Object Difference window and in the subtabs. For more information about the DAC query functionality, see "Using the DAC Query Functionality".

- **Difference Type Filter**

  The drop-down list in the Object Difference window toolbar enables you to filter the list of differences based on the Change Description. The following filter options are available:

  - **All**. Displays all changed objects.
  - **Added-Source**. Displays objects that were added in the source container.
  - **Added-Target**. Displays objects that were added in the target container.
  - **Cloned-Source**. Displays objects that were cloned in the source container.
  - **Cloned-Target**. Displays objects that were cloned in the target container.
  - **Deleted-Source**. Displays objects that were deleted from the source container.
  - **Deleted-Target**. Displays objects that were deleted from the target container.
  - **Modified**. Displays objects that were modified differently in the source and target containers.

10.8.3 Possible Repository Merge Outcomes Based on Your Decisions

The View Difference Report shows the changes between two containers based on object type.

For more information about object types and their ownership properties, see "About Object Ownership in DAC".

> **Tip:** Use the right-click command Record Info to view the lineage of an object.

The following tables list the merge outcomes based on the decision you make about the object differences listed in the View Difference Report.

- Table 10–2, " Merge Outcomes for Repository Upgrade (DAC 784) Option"
- Table 10–3, " Merge Outcomes for Replace Base and Refresh Base Options"
- Table 10–4, " Merge Outcomes for Peer to Peer Option"

Table 10–2 Merge Outcomes for Repository Upgrade (DAC 784) Option

Change	Object Ownership Type	Description	Merge Action with User Choice: Accept Source	Merge Action with User Choice: Reject Source
Added-Source	Original	An original object has been added in the source container.	Object will be merged into the custom container as an original object.	Object will be deleted from the custom container.
Added-Source	Clone	A cloned object has been added in the source container.	Not applicable	Not applicable
Added-Source	Reference	A referenced object has been added in the source container.	Not applicable	Not applicable
Added-Target	Original	An original object has been added in the target container.	Object will be deleted from the custom container.	Object will be merged into the custom container as a referenced object to the base container.
Added-Target	Clone	A cloned object has been added in the target container.	Object will be deleted from the custom container.	Object will be merged into the custom container as a referenced object to the base container.
Added-Target	Reference	A referenced object has been added in the target container.	Object will be deleted from the custom container.	Object will be merged into the custom container as a referenced object to the base container.
Deleted-Source	Original	An original object has been deleted from the source container.	Object will be deleted from the custom container.	Object will be retained in the custom container as a reference to the base container.
Deleted-Source	Clone	A cloned object has been deleted from the source container.	Object will be deleted from the custom container.	Object will be retained in the custom container as a reference to the base container.
Deleted-Source	Reference	A referenced object has been deleted from the source container.	Object will be deleted from the custom container.	Object will be retained in the custom container as a reference to the base container.
Deleted-Target	Original	An original object has been deleted from the target container.	Object will be merged into the custom container as an original object.	Object will be deleted from the custom container.
Deleted-Target	Clone	A cloned object has been deleted from the target container.	Object will be merged into the custom container as an original object.	Object will be deleted from the custom container.

Table 10–2 (Cont.) Merge Outcomes for Repository Upgrade (DAC 784) Option

Change	Object Ownership Type	Description	Merge Action with User Choice: Accept Source	Merge Action with User Choice: Reject Source
Deleted-Target	Reference	A referenced object has been deleted from the target container.	Object will be merged into the custom container as an original object.	Object will be deleted from the custom container.
Modified	Not applicable	An object has been modified differently in the source and target containers.	Object will be merged into the custom container as a cloned object.	Object will be merged into the custom container as a referenced object to the base container.
Modified-Cloned	Clone	The parent of a cloned object has been modified in the source container.	Not applicable	Not applicable
Modified-Referenced	Reference	The parent of a referenced object has been modified in the source container.	Not applicable	Not applicable
Cloned-Source	Not applicable	A referenced object has been cloned in the source container.	Not applicable	Not applicable
Cloned-Target	Not applicable	A referenced object has been cloned in the target container.	Not applicable	Not applicable

Table 10–3 Merge Outcomes for Replace Base and Refresh Base Options

Change	Object Ownership Type	Description	Merge Action with User Choice: Accept Source	Merge Action with User Choice: Reject Source
Added-Source	Original	An original object has been added in the source container.	Object will be added to the target custom container as an original object.	Not applicable
Added-Source	Clone	A cloned object has been added in the source container.	Object will be added to the target custom container as a cloned object. Automatically adjusts the lineage for objects in child containers if the object was referenced in them.	Not applicable
Added-Source	Reference	A referenced object has been added in the source container.	Object will be added to the target container as a referenced object.	Not applicable
Added-Target	Original	An original object has been added in the target container.	Object will be deleted from the target custom container. Automatically adjusts the lineage for objects in child containers.	Object will be retained in the target custom container.

Table 10–3 (Cont.) Merge Outcomes for Replace Base and Refresh Base Options

Change	Object Ownership Type	Description	Merge Action with User Choice: Accept Source	Merge Action with User Choice: Reject Source
Added-Target	Clone	A cloned object has been added in the target container.	Object will be deleted from the target custom container. Automatically adjusts the lineage for objects in child containers.	Object will be retained in the target custom container.
Added-Target	Reference	A referenced object has been added in the target container.	Object will be deleted from the target custom container. Automatically adjusts the lineage for objects in child containers.	Object will be retained in the target custom container.
Deleted-Source	Original	An original object has been deleted from the source container.	Object will be deleted from the target custom container. Automatically adjusts the lineage for objects in child containers if the object was referenced in them.	Object will be retained in the target custom container.
Deleted-Source	Clone	A cloned object has been deleted from the source container.	Object will be deleted from the target custom container. Automatically adjusts the lineage for objects in child containers if the object was referenced in them.	Object will be retained in the target custom container.
Deleted-Source	Reference	A referenced object has been deleted from the source container.	Object will be deleted from the target custom container. Automatically adjusts the lineage for objects in child containers if the object was referenced in them.	Object will be retained in the target custom container.
Deleted-Target	Original	An original object has been deleted from the target container.	Object will be added as an original object to the target custom container .	Not applicable
Deleted-Target	Clone	A cloned object has been deleted from the target container.	Object will be added as a cloned object to the target custom container .	Not applicable
Deleted-Target	Reference	A referenced object has been deleted from the target container.	Object will be added as a referenced object to the target custom container .	Not applicable
Modified	Not applicable	An object has been modified differently in the source and target containers. The right-click command Record Info shows the object's lineage.	Not applicable	Not applicable

Table 10–3 (Cont.) Merge Outcomes for Replace Base and Refresh Base Options

Change	Object Ownership Type	Description	Merge Action with User Choice: Accept Source	Merge Action with User Choice: Reject Source
Modified-Cloned	Clone	The parent of a cloned object has been modified in the source container.	De-clone the object and re-reference it. Automatically adjusts the lineage for objects in child containers if the object was referenced in them.	Not applicable
Modified-Referenced	Reference	The parent of a referenced object has been modified in the source container.	Re-reference the object. Automatically adjusts the lineage for objects in child containers if the object was referenced in them.	Not applicable
Cloned-Source	Not applicable	A referenced object has been cloned in the source container.	Not applicable	Not applicable
Cloned-Target	Not applicable	A referenced object has been cloned in the target container.	Not applicable	Not applicable

Table 10–4 Merge Outcomes for Peer to Peer Option

Change	Object Ownership Type	Description	Merge Action with User Choice: Accept Source	Merge Action with User Choice: Reject Source
Added-Source	Original	An original object has been added in the source container.	Object will be added to the target custom container as an original object.	Not applicable
Added-Source	Clone	A cloned object has been added in the source container.	Object will be added to the target custom container as a cloned object. Automatically adjusts the lineage for objects in child containers.	Not applicable
Added-Source	Reference	A referenced object has been added in the source container.	Object will be added to the target custom container as a referenced object.	Not applicable
Added-Target	Original	An original object has been added in the target container.	Object will be deleted from the target custom container. Automatically adjusts the lineage for objects in child containers.	Object will be retained in the target custom container.

Table 10–4 (Cont.) Merge Outcomes for Peer to Peer Option

Change	Object Ownership Type	Description	Merge Action with User Choice: Accept Source	Merge Action with User Choice: Reject Source
Added-Target	Clone	A cloned object has been added in the target container.	Object will be deleted from the target custom container. Automatically adjusts the lineage for objects in child containers.	Object will be retained in the target custom container.
Added-Target	Reference	A referenced object has been added in the target container.	Object will be deleted from the target custom container. Automatically adjusts the lineage for objects in child containers.	Object will be retained in the target custom container.
Deleted-Source	Original	An original object has been deleted from the source container.	Object will be deleted from the target custom container. Automatically adjusts the lineage for objects in child containers if the objects were referenced in them.	Object will be retained in the target custom container.
Deleted-Source	Clone	A cloned object has been deleted from the source container.	Object will be deleted from the target custom container. Automatically adjusts the lineage for objects in child containers if the objects were referenced in them.	Object will be retained in the target custom container.
Deleted-Source	Reference	A referenced object has been deleted from the source container.	Object will be deleted from the target custom container. Automatically adjusts the lineage for objects in child containers if the objects were referenced in them.	Object will be retained in the target custom container.
Deleted-Target	Original	An original object has been deleted from the target container.	Object will be added to the target custom container as an original object.	Not applicable
Deleted-Target	Clone	A cloned object has been deleted from the target container.	Object will be added to the target custom container as a cloned object.	Not applicable
Deleted-Target	Reference	A referenced object has been deleted from the target container.	Object will be added to the target custom container as a referenced object.	Not applicable

Table 10–4 (Cont.) Merge Outcomes for Peer to Peer Option

Change	Object Ownership Type	Description	Merge Action with User Choice: Accept Source	Merge Action with User Choice: Reject Source
Modified	Not applicable	An object has been modified differently in the source and target containers. The right-click command Record Info shows the object's lineage.	Not applicable	Not applicable
Modified-Cloned	Clone	The parent of a cloned object has been modified in the source container.	De-clone the object and re-reference it. Automatically adjusts the lineage for objects in child containers if the objects were referenced in them.	Not applicable
Modified-Referenced	Reference	The parent of a referenced object has been modified in the source container.	Re-reference the object. Automatically adjusts the lineage for objects in child containers if the objects were referenced in them.	Not applicable
Cloned-Source	Not applicable	A referenced object has been cloned in the source container.	Clone object in target custom container. Automatically adjusts the lineage for objects in child containers.	Not applicable
Cloned-Target	Not applicable	A referenced object has been cloned in the target container.	De-clone object in target custom container. Automatically adjusts the lineage for objects in child containers.	Not applicable

11

Common Tasks Performed in the DAC

This chapter provides instructions for performing common DAC tasks.

This chapter contains the following topics:

- Importing DAC Metadata
- Exporting DAC Metadata
- Distributing DAC Metadata
- Running the DAC Server Automatically
- Command Line Access to the DAC Server
- DAC Repository Command Line Options
- Replacing an Informatica Workflow with a Custom SQL File
- Determining the Informatica Server Maximum Sessions Parameter Setting
- Determining the Number of Transactional and Data Warehouse Database Connections
- Running Two DAC Servers on the Same Machine
- Customizing Index and Analyze Table Syntaxes
- Overview of Change Capture Process (Siebel Sources Only)
- Using the Change Capture Filter
- Tracking Deleted Records
- Pointing Multiple Informatica PowerCenter Services to a Single Informatica Repository
- Handling ETL Failures with the DAC
- Forcing Password Encryption When DAC Interacts With Informatica
- Upgrading the Data Warehouse Schema on Teradata
- Using JDBC Connection URL Override to Handle Database Connections
- Using Parameters to Specify Full or Incremental Loads
- Mapping Multiple Database-Specific Informatica Workflows to the Same DAC Task
- Connecting to the DAC Repository When Using Oracle RAC

11.1 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.

11.2 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.

11.3 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".

11.4 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.

11.5 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.

Refer to the file daccommandline.bat/.sh for usage.

11.5.1 Command Line Operations

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

11.5.2 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 value in the DAC System Properties tab is not a positive number.
- There are no active Informatica PowerCenter Services or Integration Service registered in the Informatica Servers tab.
- One or more Informatica PowerCenter Services or Integration Service do not have the passwords defined in the Informatica Servers tab.
- One or more Informatica PowerCenter Services or Integration Service do not have a Maximum Sessions number properly defined in the Informatica Servers tab.
- One or more data sources do not have the Table Owner or Table Owner Password values properly defined in the Physical Data Sources tab.

- One or more data sources do not have a maximum number of connections (Max Num Connections) value properly defined in the Physical Data Sources tab.
- One or more data sources do not have a Data Source Number defined in the Physical Data Sources tab.

11.5.2.1 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.

11.5.2.2 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.

11.5.3 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.

   a. 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
```

11.5.4 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. -wait option lets you start the execution plan in synchronous mode.
StopETL	Stops the operation of an execution plan. You must specify an execution plan name.
ETLStatus	If you do not specify an execution plan name, the status of the execution plan that last ran is returned. If you specify an execution plan name, the status of the specified execution plan is returned.
DatabaseStatus	Verifies whether the DAC server can connect to all active database connections. You do not need to specify an execution plan name.
InformaticaStatus	Verifies whether the DAC server is able to ping all active Informatica PowerCenter Services machines.

Note: The method names are case insensitive. Execution plan names are case sensitive. Also, if the execution plan name contains spaces, place beginning and ending double quotes around the name.

For example:

Command Line	Description
`dacCmdLine EtlStatus`	Returns the status of the execution plan that last ran.
`dacCmdLine EtlStatus Forecast`	Returns the status of the last instance of the Forecast execution plan.
`dacCmdLine StopEtl Forecast`	If the execution plan currently running is Forecast, the operation will be terminated. Otherwise, the request is ignored.
`dacCmdLine databasestatus`	Returns the health status of all the database connections as defined in the DAC repository from the DAC server.
`dacCmdLine InformaticaStatus`	Returns the health status of all the Informatica PowerCenter Services connections as defined in the DAC client on the Informatica Services tab.

11.6 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.

11.6.1 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.

11.6.2 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.

11.6.3 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).

11.6.4 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).

11.6.5 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.

11.6.6 Drop Schema

The DROPSCHEMA option drops the schema of the DAC repository.

Syntax:

```
DROPSCHEMA
```

11.6.7 Analyze

The ANALYZE option analyzes the DAC repository tables.

Syntax:

```
ANALYZE
```

11.6.8 Upgrade

The UPGRADE option upgrades the DAC repository.

Syntax:

```
UPGRADE
```

11.6.9 Set Password

The SETPASSWORD option sets the passwords for the PowerCenter Services, Integration Service, 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.

11.7 Replacing an Informatica Workflow with a Custom SQL File

You can improve the performance of loads sometimes by replacing Informatica workflows with the Actions feature. For more information about actions, see "About Index, Table and Task Actions".

To replace an Informatica workflow with a custom SQL file

1. Create a SQL action to be used to load the table, and unit test it.

   You can create one action for a full load and one for an incremental load, or you can use the same file for both full and incremental loads.

2. In the Tasks tab of the DAC Design view, query for the task for which you want to replace the Informatica workflow.
3. Change the Execution Type to SQL.
4. Replace the workflow name in the Command for Incremental Load or Command for Full Load fields with the SQL action.

11.8 Determining the Informatica Server Maximum Sessions Parameter Setting

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

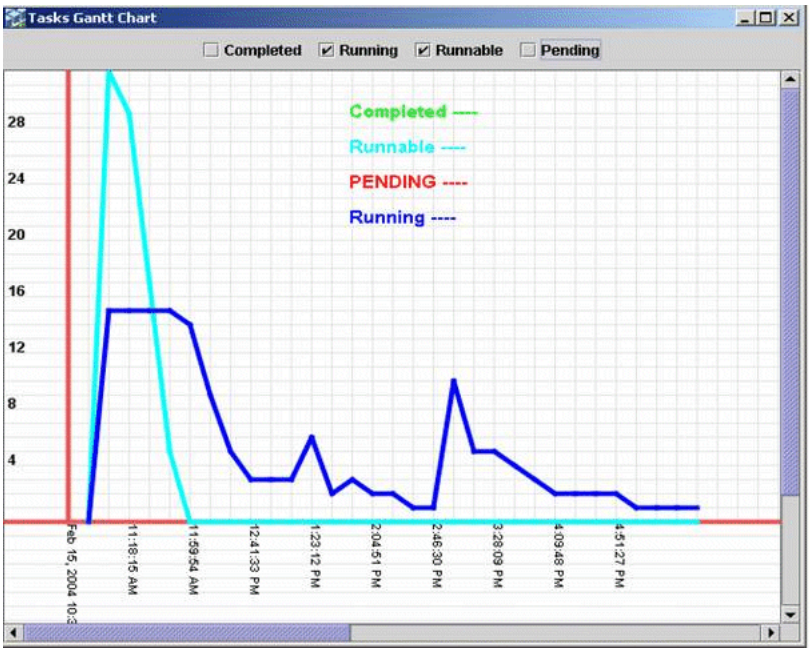
You should consider the following factors when determining the Maximum Sessions parameter value:

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

  For an optimal run, the runnable queue should be at zero or should reach zero in a short time. For example, Figure 11–1 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 11–1 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 11–1 Sample Performance Run



11.9 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 in the PowerCenter Services) because these values have a one to one relationship.

11.10 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.

11.11 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.

11.12 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:

11.12.1 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".

11.12.2 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).

11.12.2.1 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.

11.12.2.1.1 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.

11.12.2.1.2 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'
```

11.12.2.2 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
```

11.13 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.

11.14 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.

11.15 Pointing Multiple Informatica PowerCenter Services to a Single Informatica Repository

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

11.16 Handling ETL Failures with the DAC

This section includes the following topics:

- When the Execution of an Execution Plan Fails
- Discarding the Current Run Execution Plan
- Failure of Aggregator Transformation Tasks with Sorted Input

11.16.1 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.

11.16.2 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.

11.16.3 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.

11.16.4 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 PowerCenter Designer, navigate to Mapping Designer, open the corresponding mapping, and in the Aggregator transformation, remove the check from the Sorted Input check box.

11.17 Forcing Password Encryption When DAC Interacts With Informatica

DAC sends the Informatica repository and server passwords un-encrypted when communicating with Informatica through pmcmd and pmrep commands.

You can force password encryption by following the procedure below:

Note: In the examples included in the following procedure, the Informatica server and Informatica Repository server use the password Administrator.

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

   ```
   pmpasswd Administrator -e CRYPT_SYSTEM
   ```

   This step will produce something similar to the following text:

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


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

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

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

   Value– `dMGpMvpsuQwXD5UvRmq0OZxhppTWK0Y7fzBtxHL04Gg=`

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

3. Modify the file DAC\conf\infa_command.xml by replacing all occurrences of <-p> with <-pv> and <-x> with <-X>.
4. Stop the DAC server.
5. Log into the DAC client and navigate to the Setup menu and choose the Informatica Servers tab.
6. Highlight the Informatica Server record and enter the name of the environment variable that you created in Step 2 of this procedure as the value of Password. Save the record.
7. Repeat the previous step for the Informatica Repository Server.
8. Close and re-open the client to test the connections.
9. If the DAC server is located on the same machine, start the DAC server and run ETL.
10. Verify that DAC issues the following `pmcmd` command.

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

11. If the DAC server is on another Windows machine, do the following:
    a. Set the environmental variable on the server machine and modify the DAC\conf\infa_command.xml.
    b. Shut down the DAC server when testing Informatica connectivity or unit testing a DAC task via the DAC client.
12. If the DAC server is on a non-Windows machine, do the following:
    a. Set the environmental variable on the server machine and modify the DAC\conf\infa_command.xml.
    b. Shut down the DAC server when testing Informatica connectivity or unit testing a DAC task via the DAC client.

11.18 Upgrading the Data Warehouse Schema on Teradata

When the data warehouse schema is on a Teradata database and you use the Data Warehouse Configuration Wizard to upgrade the data warehouse schema, DAC generates four files: two SQL files and two log files. DAC stores these files in DAC\conf\sqlgen\sql\teradata. A description of the files follows.

- **upgrade-regular.sql.** This file contains scripts that DAC has verified are necessary to upgrade the data warehouse schema. The scripts handle new or altered tables and columns. For example, they may increase the size of simple data types or change a column from null to not null.
- **upgrade-questionable.sql.** This file contains scripts that DAC suggests may be required to upgrade the data warehouse schema. These scripts are intended to be an aid in determining upgrade requirements and must be reviewed by a Teradata DBA and corrected if necessary.
- **upgradedwtables_sql.log.** This file is the log file that corresponds to the upgrade-regular.sql file. It is for information purposes only.
- **upgrade-issues.log.** This file is the log file that corresponds to the file upgrade-questionable.sql. It is for information purposes only.

11.19 Using JDBC Connection URL Override to Handle Database Connections

DAC provides a mechanism for overriding the JDBC connection strings for some database types. This is done by modifying the connection_template.xml document. The location of this document is DAC\conf\connection_template.xml. Note that these changes to the connection_template.xml document apply to all instances of the database type.

For example, if the URL to connect to MSSQL is modified in the file connection_ template.xml, the modifications apply to all MSSQL connections that appear in DAC (for example, source connections, target connections, and connection to the DAC Server). Note that in the case where there are multiple source connections of type MSSQL and users need to connect to each of them using a different URL, this process will not work.

You can override connection strings with different JDBC URLs for each connection. You can configure the connections using the following methods:

- You can configure connections to different databases by using the fields JDBC URL Override and JDBC Driver Override to add connection specific JDBC URLs. To access these fields, go to the Setup menu and choose Physical Data Sources tab. When entering text into these fields, you must be sure to conform to the database specifications.
- You can configure the DAC Client connection to create or connect to the DAC repository with user specified URLs in the DAC client login screen.
- The DAC Server can be configured using specific JDBC URLs.
  - If the DAC Server is installed on Windows, you can configure specific JDBC URLs by accessing the client and using the Server Set Up menu.
  - If the DAC Server is installed on Unix or Linux, you can configure the JDBC URLs by executing ServerSetupPrompt.sh and following the on screen instructions.

11.20 Using Parameters to Specify Full or Incremental Loads

Parameters can be registered in DAC as either Source System Parameters or Task Level Parameters.

Source System Parameters – A list of parameter names and the values applicable to each source system can be registered by accessing DAC, then Design View, and using the Source System Parameter tab.

Task Level Parameters – Parameters that apply to all tasks under a source system may be registered under the Source System Parameters. If there are parameters that are specific to particular tasks, developers can create such task specific parameters by accessing the Task tab and using the Parameters subtab. Task level parameters have priority over source system parameters. If the same parameter is registered as a source system parameter and as a task level parameter, DAC will evaluate the parameter with the task level value.

You can specify different values for both Source System Parameters and Task Level Parameters for full load ETL runs and/or for incremental load ETL runs. The "Load Type" is for specifying a value as it applies to the parameter.

For example, if you have a parameter called "SourceSystemParameter" at the source system level, you can define this parameter as shown in Table 11–1.

Table 11–1 Example of Source System Parameter

SourceSystemParameter	Full	Source_System_Parameter_ Full_Value
SourceSystemParameter	Incremental	Source_System_Parameter_ Incremental_Value
SourceSystemParameter	Both	Source_System_Parameter_ Both_Value

During ETL runtime, DAC automatically evaluates this parameter to "Source_System_ Parameter_Full_Value" if the ETL is in Full mode and to "Source_System_Parameter_ Incremental_Value" if the ETL is in an incremental run.

In this example, load type "Both" is redundant, as there is a value defined for full and incremental values. If a value is undefined for Full or Incremental, then DAC picks the value defined for both.

Note that the behavior is the same for parameters at the task level.

11.21 Mapping Multiple Database-Specific Informatica Workflows to the Same DAC Task

This section includes instructions for mapping multiple, database-specific Informatica workflows to the same DAC task. This is accomplished by parameterizing the Informatica workflow command. At runtime, DAC determines which workflow to run based on the parameterization.

Follow the instructions below to map multiple, database-specific workflows to the same DAC task. These instructions use SIL_PersonDimension_Full as an example of a full command and SIL_PersonDimension as an example of an incremental command on an Oracle database and SIL_PersonDimension_Full_TD and SIL_ PersonDimension_TD as full and incremental commands, respectively, on a Teradata database.

To map multiple database-specific Informatica workflows to the same DAC task:

1. In the DAC Design view, go to the Tasks tab.
2. Query for the task to which you want add multiple workflows.
3. Select the task, and then click the Parameters subtab.
4. Create a new parameter for a full load command:
   a. Click New in the subtab toolbar.
   b. In the Name field, enter $$workflow_CMD_PARAMETER.
   c. In the Data Type field, select DB Specific Text.
   d. In the Load Type field, select Full.
   e. Click in the Value field to open the Enter Parameter Value dialog box.
   f. In the Connection Type field, select @DAC_TARGET_DBTYPE.
   g. In the appropriate database fields, enter the full command name for both database types.

      For example, enter SIL_PersonDimension_Full in the Oracle field and SIL_ PersonDimension_Full_TD in the Teradata field.

5. Create a new parameter for an incremental load command:
   a. Click New in the subtab toolbar.
   b. In the Name field, enter $$workflow_CMD_PARAMETER.
   c. In the Data Type field, select DB Specific Text.
   d. In the Load Type field, select Incremental.
   e. Click in the Value field to open the Enter Parameter Value dialog box.
   f. In the Connection Type field, select @DAC_TARGET_DBTYPE.
   g. In the appropriate database fields, enter the incremental command name for both database types.

      For example, enter SIL_PersonDimension in the Oracle field and SIL_PersonDimension_TD in the Teradata field.
6. With the same task selected, click the Edit subtab.
7. In the Command for Incremental Load field, enter @DAC_$$workflow_CMD_PARAMETER.
8. In the Command for Full Load field, enter @DAC_$$workflow_CMD_PARAMETER.
9. Click Save.

11.22 Connecting to the DAC Repository When Using Oracle RAC

When the DAC repository is on an Oracle database that uses Oracle Real Application Cluster (RAC), the standard URL for configuring the connections between the DAC Client and the DAC repository and the DAC Server and the DAC repository does not work. (The standard URL is jdbc:oracle:thin:@<HOST>:1521:<SID>.)

To configure the connection between the DAC client and the DAC repository and between the DAC Server and DAC repository use the following URL:

jdbc:oracle:thin:@(DESCRIPTION=(LOAD_BALANCE=on)
(ADDRESS=(PROTOCOL=TCP)(HOST=<host1>) (PORT=<port1>))
(ADDRESS=(PROTOCOL=TCP)(HOST=<host2>) (PORT=<port2>))
(CONNECT_DATA=(SERVICE_NAME=<service name>)))
...

1. Enter this URL in the "DB URL" field in the Configuring… dialog. This dialog is used to configure the connection between the DAC Client and the DAC repository.
2. Repository Connection Information tab in the Server Configuration dialog. This dialog is used to configure the connection between the DAC Server and the DAC Client.

12

DAC Functional Reference

This chapter describes the functionality available in the Data Warehouse Administration Console (DAC) tabs.

It contains the following topics:

- Common Elements of Interface Tabs
- Design View Tabs
- Setup View Tabs
- Execute View Tabs

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.

- Configuration Tags Tab
- 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.

For information using the Index Action feature to override default index behavior, see "About Index, Table and Task Actions".

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.

Databases

Lists the databases associated with the selected 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.

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.

Always Drop & Create Bitmap

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

Indices Tab: Actions Subtab

The Actions subtab lists the actions that have been set up for the selected index. For a description of index actions, see the following:

- About Index, Table and Task Actions
- Defining a SQL Script for an Action
- Assigning an Action to a Repository Object

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.

Source System Folders Tab

The Source System Folders tab lists the Informatica folders associated with the selected source system container.

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

A subject area is a logical grouping of tables related to a particular subject or application context. It also includes the tasks that are associated with the tables, as well as the tasks required to load the tables.

Subject areas are assigned to execution plans, which can be scheduled for full or incremental loads.

The Subject Areas tab lists all the subject areas associated with the selected source system container. It enables you to view and edit existing subjects areas and to create new ones.

For more information, see "Customizing DAC Objects and Designing Subject Areas" and "Creating a Subject Area".

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".

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: Extended Tables (RO) Subtab

The Extended Tables (RO) subtab is a read-only tab that lists the extended tables associated with the selected subject area.

Included

Indicates whether the extended table is included in the selected subject area, indicating your choice about inclusion or exclusion of the tables in the assembly process.

For more information about subject areas, see "Customizing DAC Objects and Designing Subject Areas".

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.

For information about adding tables to a subject area, see"Customizing DAC Objects and Designing Subject Areas" and "Creating a Subject Area".

Name

The name of the table associated with the selected subject area.

Subject Areas Tab: Tasks Subtab

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

When you inactivate a task, it remains inactive even if you reassemble the subject area. When you remove a task from a subject area, it will be added back to the subject area upon reassembly.

For more information about subject areas, see "Customizing DAC Objects and Designing Subject Areas".

Parent Group

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

Phase

The 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 the physical database tables defined in the database schema that are associated with the selected source system container. It enables you to view and edit existing tables and to create new ones.

For information about adding new tables, see "Adding a New Table and Columns to the Data Warehouse".

Table Type

The table type.

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: Actions Subtab

The Actions subtab lists the actions that have been set up for the selected table. For a description of task actions, see the following:

- About Index, Table and Task Actions
- Defining a SQL Script for an Action
- Assigning an Action to a Repository Object

The table action types enable you to trigger SQL scripts to analyze or truncate tables. Table actions for analyzing or truncating tables override all other table properties.

Action Type

The default actions on a table during ETL execution are truncating a table and analyzing a table. If you want to override any of these syntaxes, you can define actions per mode (full or incremental). Once an action is defined, it overrides the default behavior.

Note: You can associate an action to multiple objects at the same time. First, identify the objects that you want to associate an action with by using the query functionality. Then, right-click on the results displayed, and select Associate Actions. Next, select an action and the mode you want to associate it with.

The table action types are the following:

- **Analyze Table**

Use this type to analyze tables.

- **Truncate Table**

  Use this type to truncate tables.

Load Type

The load type specifies whether the SQL script is to be called for incremental runs, full runs, or both.

Action

Actions are the SQL statement or stored procedure that you define. You can define one or more SQL statements or stored procedures for each action.

Tables Tab: Conditional for Tasks (RO) Subtab

The Conditional for Tasks (RO) subtab displays a read-only list of tasks that have defined this table as one of the conditional tables.

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.

Tables Tab: Columns Subtab

The Columns subtab displays the columns that belong to the selected table.

Tables Tab: Indices (RO) Subtab

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.

Tables Tab: Related Tables Subtab

The Related Tables subtab lists tables that are related to the selected table. Related tables participate in the ETL process in addition to the tables that are associated with this table.

Tables Tab: Source for Tasks (RO) Subtab

The Source for Tasks (RO) subtab displays a read-only list of tasks that use the selected table as a source.

Task Phase

Task phase of the ETL process. This information is primarily used for dependency generation. Certain phases, such as Change Capture and Query Index Creation, are not available for you to assign to a task. The DAC server uses the task phase to prioritize tasks and to generate a summary of the time taken for each of the phases.

Build Image

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

Type

Table type.

Tables Tab: Target for Tasks (RO) Subtab

The Target for Tasks (RO) subtab displays a read-only list of tasks that use the selected table as a target.

Task Phase

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.

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. It also enables you to add child tasks to a task group selected in the top window.

Primary Source
Logical database connection for the primary source database.

Primary Target
Logical database connection for the primary target database.

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 extracting 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.

Group Order

The order in which the task is defined to execute in a certain group.

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 manually on the other sources also.

- **IMG_SYNC**

  Used for internal change capture. If you are using multiple Siebel transactional sources, you can create this task for the additional tasks for doing similar change capture sync processes. You cannot change the behavior of the change capture sync process. This task requires change capture tables to be created on the other sources also. This task should be used with discretion for Siebel sources only.

Execution Priority

Indicates the order in which the task is executed. If two or more similar tasks (tasks having same phase, similar truncate properties, same number of successors, same number of source tables) have the same priority, the order occurs randomly.

Build Image

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

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: Actions Subtab

The Actions subtab lists the actions that have been set up for the selected task. For a description of task actions, see the following:

- About Index, Table and Task Actions
- Defining a SQL Script for an Action
- Assigning an Action to a Repository Object

Action Type

Action types are predefined categories of task behaviors that trigger the execution of a SQL script. The following types are available:

- **Preceding Action**

  Use this type to execute a SQL script before a task runs.

- **Success Action**

  Use this type to execute a SQL script after a task runs successfully.

- **Failure Action**

  Use this type to execute a SQL script if a task fails during its execution.

- **Restart Action**

  Use this type to execute a SQL script when a task that previously failed is restarted.

- **Upon Failure Restart Action**

  Use this type to execute a SQL script to restart a task that fails.

Load Type

The load type specifies whether the SQL script is to be called for incremental runs, full runs, or both.

Action

You define the task action in the form of a SQL statement or stored procedure. You can define one or more SQL statements for each action. Double-click in the Action field to open the Choose Action dialog box, where you can select the appropriate action.

You define the SQL statement or stored procedure for an action in the Task Actions dialog box, which you access by selecting Tools, then Seed Data, then Actions, and then Task Actions. For instructions, see "Defining a SQL Script for an Action".

Tasks Tab: Conditional Tables Subtab

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.

Tasks 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.

Scope

For multi-source execution plans only. Specifies how the Block action of the phase dependency behaves in relation to multi-source execution plans. Possible values are the following:

- Both

  Indicates the blocking action is active for tasks that have the same source and target physical data source connections.

- Source

  Indicates the blocking action is active for tasks that have the same source physical data source connection.

- Target

  Indicates the blocking action is active for tasks that have the same target physical data source connection.

- None

  Indicates the blocking action is active for all tasks regardless of the source and target physical data source connections.

Phase

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

Tasks Tab: Refresh Date Tables Subtab

When a task gets executed, DAC determines the read and write mode based on the refresh dates that the DAC stores for the database connection and the source/target table name combination of that task. However, if you want to determine the mode of both read and write operations based on a table which is not a source/target table, you can define it in the Refresh Date Tables subtab. For example, suppose have an aggregate table based on W_REVENUE_F and its dimensions, there are two ways of populating the aggregate table. In Full mode, the table gets truncated, and all the aggregate values get recomputed. In the Incremental mode, the delta aggregate values are computed based on the new/updated records in the base fact table. The incremental strategy is efficient for a small subset of rows you would expect in an incremental run. The primary transactional table on the OLTP side for this table is S_REVN. Suppose the data is being incrementally extracted from one source, and at a subsequent time a new data source is added to the execution plan. The big number of incoming rows make the incremental load inefficient, but recomputing of the aggregates more efficient. Hence, the decision to load the table in Full mode should depend on the ultimate source table, which is qualified as OLTP.S_REVN, rather than based on the immediate source table Datawarehouse.W_REVENUE_F.

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.

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.

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.

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.

No Run

Generates tasks in the Task Details subtab of the Current Run tab but does not execute them.

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.

Worker Pool Size

The worker pool size is the number of worker threads that perform operations such as drop/create indexes, truncate/analyze tables, and ETL jobs like SQL and Informatica workflows.

The property's value corresponds to the number of task details that are anticipated to run in parallel. The default size is 50, which assumes that 10 tasks run in parallel and each task has five detail sub-tasks (for example, truncate, drop index, create index, analyze table) running in parallel.

It is prudent to increase this number when running more ETL tasks and/or task details in parallel. For example the Worker Pool Size can be set to 100 if the number of parallel indexes is set to two per table.

Running multiple threads can be resource intensive; therefore, you should set this value as low as possible.

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.

Service

For Informatica 7.x installations, indicates the host machine name where the Informatica Server is installed. For Informatica 8.x installations, indicates the Integration Service name.

Server Port

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

Domain

For Informatica 8.x installations only. Indicates the domain file location.

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.

Dependency 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.

Num Parallel Indexes
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.

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

Physical Data Sources Tab: Index Spaces Subtab

The Index Spaces subtab enables 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: Parallel Indexes Subtab

Enables you to specify how many indexes can be created in parallel for a given table.

Note: Use this property to specify the number of parallel indexes DAC will create for a specific table. Use the Num Parallel Indexes property in the Physical Data Sources tab to specify the number of parallel indexes DAC will create for all tables associated with a specified physical data source connection.

Name Column

Name of the table on which the indexes will be created.

Number of Parallel Indexes Column

Number of indexes that can be created in parallel for the specified table.

Physical Data Sources Tab: Analyze Frequencies Subtab

Analyze frequencies in days by table type.

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.

Analyze Date

Indicates when the table was analyzed.

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: Phase 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 phase of the run.

End Time
End time of the selected phase of the run.

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

Current Run Tab: Run Type Summary (RO)

The Run Type Summary (RO) subtab is a read-only tab that indicates the number of task details by the execution type.

Current 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. The prioritization is also based on the number of dependent tasks, the number of source tables, and the average time taken by a task.

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.

For more information, see "Building, Running and Monitoring Execution Plans".

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.

The Prune Days property can be set at the execution plan level for the entire plan or in the execution plan parameters for specific source connections. Prune Days value set at the source level override the Prune Days value set at the execution plan level.

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

The Build button does the following:

- Collects all the tasks from all the subject areas.
- Substitutes the logical values with the physical values as defined in the parameters subtab (primary source/target connection and folder information).
- Removes any redundancies based on the task name and primary source/target connection information.
- Creates multiple instances of the same task to accommodate the number of copies of the parameters.
- Comes up with the ordered list of tasks with the dependencies computed among them.

For more information about building an execution plan, see "Building and Running Single-Source and Multi-Source Execution Plans".

Execution Plans Tab: All Dependencies Subtab

The All Dependencies subtab opens in Query mode. It enables you to query for tasks that have a dependent relationship. The columns in this tab are the same as those in the Immediate Dependencies subtab.

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

Note: The All Dependencies information is not subject to export and, therefore, not importable elsewhere. However, if you want to compute all dependencies, you can do so by right-clicking and selecting Compute All Dependencies.

Execution Plans Tab: Following Tasks Subtab

The Following Tasks subtab lists the tasks with the phase Post-ETL.

Execution Priority

Indicates the order among the following tasks in which this task is executed.

Command for Incremental Load

Command associated with the task.

Source System

Source system container from which the task extracts data.

Execution Plans Tab: Immediate Dependencies Subtab

The Immediate Dependencies subtab opens in Query mode. It enables you to query for tasks that have an immediate dependent relationship between tasks that are generated during the automatic task generation process.

Depth
The level of the task's dependency. Tasks that have no dependencies are depth 0. Tasks that depend on other tasks of depth 0 are depth 1, and so on.

Task Name
Name of immediate dependent task.

Source
Source table from which the task extracts data.

Target
Target table into which the task loads data.

Execution Plans Tab: Ordered Tasks Subtab
The Ordered Tasks subtab lists tasks associated with the selected execution plan and the order in which they can be executed.

Depth
Depth of the task

Group
Indicates the name of the group, if the task belongs to one.

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.

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

Preview Run Details Button
The Preview Run Details Gives a summary of the details of what gets executed when the particular task runs part of the execution plan.

It includes the following:

- Pre-ETL actions
- Upon Failure Restart actions
- Truncate table information
- List of indexes to be dropped and created
- Full or incremental command based on the current situation

- Tables to be analyzed
- Upon Success action
- Upon Failure action

Execution Plans Tab: Parameters Subtab

The Parameters subtab lists the parameters of the selected execution plan for database connections and Informatica folders.

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

Copy Number

Number of the data source.

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.

Delay

Indicates how many minutes an extract of a data source will be delayed after the start of a multiple source ETL.

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.

Execution 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 for Incremental Load

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.

Execution Plans Tab: Tables (RO) Subtab

All the tables that are touched by the execution plan by database connection, type (source/target), and subtype (primary/auxiliary/lookup.)

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".

Run History Tab: Audit Trail (RO) Subtab

See "Current Run Tab: Audit Trail (RO) Subtab".

Run History Tab: Phase Summary (RO) Subtab

See "Current Run Tab: Phase Summary (RO) Subtab".

Run History Tab: Run Type Summary (RO) Subtab

See "Current Run Tab: Run Type Summary (RO)".

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.

A

Troubleshooting DAC

This appendix contains troubleshooting topics for the Data Warehouse Administration Console (DAC).

It contains the following topics:

- Handling Parameter Files with Multi-Line Parameters
- Restoring the DAC Repository on Unicode Oracle Databases

A.1 Handling Parameter Files with Multi-Line Parameters

Informatica workflows initiated by DAC fail with error code 17 and the error message "Parameter file does not exist" when the parameter file has multi-line parameters. See the below text for an example. Note that in the following example, an error occurs because DAC issues `pmcmd` with -lpf in the syntax.

```
$$SYND_DS_PARTITION_TRUNCATE_SQL_TEXT=SELECT
LTRIM(MAX(SYS_CONNECT_BY_PATH('execute immediate ''ALTER TABLE
getTableName() TRUNCATE PARTITION ' || '"' ||COUNTRY_REGION_NAME|| '"' ||''''

,';')) KEEP (DENSE_RANK LAST ORDER BY curr),';') AS SQL_TXTFROM
(SELECT SOURCE,COUNTRY_REGION_NAME,ROW_NUMBER() OVER (PARTITION BY SOURCE
 ORDER BY COUNTRY_REGION_NAME) AS curr, ROW_NUMBER() OVER (PARTITION BY SOURCE

ORDER BY COUNTRY_REGION_NAME) -1 AS prev FROM W_SYND_PARTITION_TMP

WHERE SOURCE='W_SYNDD_DS_FS')
CONNECT BY prev = PRIOR curr START WITH curr = 1
```

To prevent this issue, edit <DAC_Config_Location>\conf-shared\infa_command.xml and replace all instances of <-lpf> with <-paramfile>.

This workaround will ensure that DAC uses -paramfile in the `pmcmd` syntax and Informatica can recognize the multi-line parameters.

A.2 Restoring the DAC Repository on Unicode Oracle Databases

When the DAC repository resides on a non-Unicode Oracle database, the Informatica workflow names may not fit into the corresponding DAC fields, which causes tasks to fail during ETL processes.

To work properly in multi-byte character environments, the DAC repository should be created with the Unicode option selected.

Perform the following procedure to fix an existing DAC repository with this problem. You can use the DAC Client or DAC command line parameters to perform the procedure.

1. Connect to the existing DAC repository.
2. Export the entire repository (logical, system, and run time categories).
3. Stop the DAC Server and all DAC Clients except for the one you are using.
4. Drop the current repository.
5. If you are using a DAC Client, restart it.
6. Create a new repository with the Unicode option selected.
7. Import the DAC metadata that you exported in step 2.
8. Re-enter the passwords for all Informatica services and all physical data source connections.

Index

A

actions
- about, 7-1
- assigning to repository object, 7-5
- defining SQL script for, 7-2
- example, 7-7
- functions, 7-5
- overview of, 7-1
- types, 7-1

assigning actions
- to repository object, 7-5

authentication file
- DAC, 4-3

B

best practices
- columns, 5-4
- configuration tags, 5-5
- execution plans, 5-5
- indexes, 5-3
- repository objects, 5-1
- source system containers, 5-1
- subject areas, 5-5
- tables, 5-3
- task group, 5-3
- tasks, 5-2

C

change capture
- about, 11-13
- filter, 11-17

columns
- best practices, 5-4

configuration tags
- best practices, 5-5
- working with, 8-13

D

DAC
- authentication file, 4-3
- basic concepts, 3-1
- first time logging in, 4-3
- important features, 3-1
- introduction, 3-1
- object ownership, 3-5
- process life cycle, 3-3
- quick start, 4-1

DAC repository
- command line options, 11-7

DAC server
- command line access, 11-4
- handling failures, 11-19
- running automatically, 11-3
- running two on same machine, 11-11

data flow
- Online Analytical Processing (OLAP) database, about and diagram, 2-2

data warehouse
- architecture, 2-1
- customizing, 8-2
- overview, 2-1

Data Warehouse Administration Console (DAC)
- DAC features, 3-1
- DAC window, 6-1
- editable lists, 6-11
- exporting metadata, 11-2
- importing metadata, 11-2
- menu bar, 6-2
- navigation tree, 6-10
- object ownership, 3-5
- top pane toolbar, 6-5
- user interface, 6-1

deleted records
- tracking, 11-17

E

ETL
- extract and load processes, 4-5
- full load, 4-5
- incremental load, 4-5

execution plan
- micro ETL, 9-10
- monitoring processes, 9-14
- scheduling, 9-13

execution plans
- about multi-source, 9-2
- about single-source, 9-1
- best practices, 5-5

building and running, 9-8
heterogeneous, about, 9-2
homogeneous, about, 9-2
micro ETL, 9-10
monitoring, 9-14
scheduling, 9-13
types of, 9-1
unit testing, 9-10
extract and load processes
common scenarios, 9-3

F

flat views
querying, 6-13
full load, 4-5

I

indexes
adding to data warehouse, 8-5
best practices, 5-3
Informatica
mappings, creating, 8-6
replacing workflow with SQL file, 11-9
server sessions, 11-10
Informatica repository
importing objects, 8-5
Informatica server
pointing multiple servers to single repository, 11-19
PowerCenter Services, 2-3

L

log in
DAC, 4-3

M

multi-source execution plans
best practices
multi-source execution plans, 9-5

O

object
ownership, 3-5
Online Analytical Processing database
Data Warehouse, data flow into, 2-2
Oracle Business Analytics Data Warehouse
overview, 2-1
Oracle Business Analytics Warehouse
adding columns, 8-3
adding indices, 8-5
adding new table, 8-3
architecture, 2-1
architecture components, 2-2
customizing, 8-2
overview, 2-1

P

Parameter Management
about, 8-8
parameters
at runtime, 8-9
data types, 8-8
defining database specific text type parameters, 8-11
defining SQL type parameters, 8-12
defining text type parameter, 8-10
defining timestamp type parameters, 8-11
nesting, 8-10
overview, 8-8
preconfigured, 8-9

Q

query functionality
flat views querying, 6-13
query commands, 6-12
query operators, 6-12
query procedures, 6-13

R

refresh dates
about, 9-13
repository objects
best practices, 5-1
right-click menu
common commands, 6-6
Design view commands, 6-7
Execute view commands, 6-9
Setup view commands, 6-9

S

source system container
about, 3-4
best practices, 5-1
copying, 8-1
creating, 8-1
DAC Repository objects, 3-4
SQL script
defining, 7-2
subject area
customizing, 8-1
designing, 8-17
subject areas
best practices, 5-5

T

tables
adding new tables, 8-3
best practices, 5-3
truncate behavior, multi-source ETL, 9-4
truncate behavior, single-source ETL, 9-3
tablespaces
specifying for indexes by table type, 8-13

task group
best practices, 5-3
creating, 8-7
task phase dependency
setting, 8-7
Tasks
creating tasks for workflows, 8-6
tasks
best practices, 5-2

U

upgrade
Upgrade/Merge Wizard, 10-1
Upgrade/Merge Wizard
about, 10-1
Difference Report, 10-17
major stages of upgrade/merge, 10-1
Peer to Peer Merge option, 10-14
Refresh Base option, 10-10
Refresh Base upgrade option, 10-7
Replace Base option, 10-11
Repository Upgrade (DAC 784) option, 10-3
resetting, 10-2
resolving object differences, 10-17
upgrade/merge options, 10-3
user account
about, 4-1
creating, deleting, inactivation, 4-2
management, 4-1