Oracle® Health Sciences Data Management Workbench

User's Guide

Release 2.3.1

E35217-02

September 2013

ORACLE®

Oracle Health Sciences Data Management Workbench User's Guide, Release 2.3.1

E35217-02

Contents

3 Creating Clinical Data Models

4 Using Libraries

5 Transforming Data to Standard Structures

6 Creating Validation Checks

7 Configuring Your Discrepancy Workflow

8 Administration

9 Data Processing

Part II Managing Study Data

10 Using the Home Page

11 Reviewing Data

12 Cleaning Data

Preface

The *Oracle Health Sciences Data Management Workbench User's Guide* describes how to set up clinical studies in the Data Management Workbench (Oracle DMW) application and how to use the application to view, clean, and analyze clinical patient data.

This preface includes the following topics:

Audience

This guide is for everyone who uses the Oracle Health Sciences Data Management Workbench application through the user interface. The audience includes any site user, sponsor user, medical reviewer, or data manager who is responsible for reviewing clinical data from various sources, and then cleaning the clinical data by creating, routing, and resolving discrepancies.

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.

Finding Information and Patches on My Oracle Support

Your source for the latest information about Oracle Health Sciences Data Management Workbench is Oracle Support's self-service website My Oracle Support.

Before you install and use Oracle DMW, always visit the My Oracle Support website for the latest information, including alerts, white papers, and bulletins.

Creating a My Oracle Support Account

You must register at My Oracle Support to obtain a user name and password account before you can enter the website.

To register for My Oracle Support:

1. Open a web browser to https://support.oracle.com.
2. Click the **Register** link to create a My Oracle Support account. The registration page opens.
3. Follow the instructions on the registration page.

Signing In to My Oracle Support

To sign in to My Oracle Support:

1. Open a web browser to https://support.oracle.com.
2. Click **Sign In.**
3. Enter your user name and password.
4. Click **Go** to open the My Oracle Support home page.

Finding Information on My Oracle Support

There are many ways to find information on My Oracle Support.

Searching by Article ID

The fastest way to search for information, including alerts, white papers, and bulletins is by the article ID number, if you know it.

To search by article ID:

1. Sign in to My Oracle Support at https://support.oracle.com.
2. Enter the arcticle ID in the text box in the upper right corner of the My Oracle Support page.
3. Click the magnifying glass icon to the right of the search box (or press the Enter key) to execute your search.

   The Knowledge Browser displays the results of your search. If the article is found, click the link to view the text and attachments, if any.

Searching by Product and Topic

You can use the following My Oracle Support tools to browse and search the knowledge base:

- Product Focus — On the Knowledge page under Select Product, type part of the product name and the system immediately filters the product list by the letters you have typed. (You do not need to type "Oracle.") Select the product you want from the filtered list and then use other search or browse tools to find the information you need.
- Advanced Search — You can specify one or more search criteria, such as source, exact phrase, and related product, to find information. This option is available from the **Advanced** link on almost all pages.

Finding Patches on My Oracle Support

Be sure to check My Oracle Support for the latest patches, if any, for your product. You can search for patches by patch ID or number, or by product or family.

To locate and download a patch:

1. Sign in to My Oracle Support at https://support.oracle.com.
2. Click the **Patches & Updates** tab. The Patches & Updates page opens and displays the Patch Search region. You have the following options:
   - In the **Patch ID or Number** field, enter the number of the patch you want. (This number is the same as the primary bug number fixed by the patch.) This option is useful if you already know the patch number.
   - To find a patch by product name, release, and platform, click the **Product or Family** link to enter one or more search criteria.
3. Click **Search** to execute your query. The Patch Search Results page opens.
4. Click the patch ID number. The system displays details about the patch. In addition, you can view the Read Me file before downloading the patch.
5. Click **Download.** Follow the instructions on the screen to download, save, and install the patch files.

Finding Oracle Documentation

The Oracle website contains links to all Oracle user and reference documentation. You can view or download a single document or an entire product library.

Finding Oracle Health Sciences Documentation

To get user documentation for Oracle Health Sciences applications, go to the Oracle Health Sciences documentation page on oracle.com at:

http://www.oracle.com/technetwork/documentation/hsgbu-154445.html

or, for the documentation for this product, to:

http://www.oracle.com/technetwork/documentation/hsgbu-clinical-407519.html

---

Note: Always check oracle.com to ensure you have the latest updates to the documentation.

Finding Other Oracle Documentation

To get user documentation for other Oracle products:

1. Go to the following web page:

   http://www.oracle.com/technology/documentation/index.html

   Alternatively, you can go to http://www.oracle.com, point to the Support tab, and then click **Product Documentation.**
2. Scroll to Health Sciences and click the link.
3. Click the **Clinical Documentation** the link.
4. Click the link for the documentation you need.

Related Documents

This section lists the documents in the documentation set, followed by their part number. The most recent version of each guide is posted on the Oracle website; see "Finding Oracle Health Sciences Documentation" on page xvii.

Oracle DMW Documentation Set Available on Oracle Website

- *Oracle Health Sciences Data Management Workbench Installation Guide* (Part E35223)
- *Oracle Health Sciences Data Management Workbench User's Guide* (Part E35217)
- *Oracle Health Sciences Data Management Workbench and Oracle Life Sciences Data Hub Security Guide* (Part E38924)

Oracle LSH Documentation Set Available on Oracle Website

The following Oracle LSH manuals contain information about Oracle DMW as well as Oracle LSH:

- *Oracle Life Sciences Data Hub Installation Guide* (Part E35295)
- *Oracle Life Sciences Data Hub System Administrator's Guide* (Part E35297)
- *Oracle Life Sciences Data Hub Application Programming Interface Guide* (Part E35306)

The following Oracle LSH manuals contain information that may be helpful to Oracle DMW developers:

- *Oracle Life Sciences Data Hub Application Developer's Guide* (Part E35298)
- *Oracle Life Sciences Data Hub Adapter Toolkit Guide* (Part E35307)
- *Oracle Life Sciences Data Hub Implementation Guide* (Part E35296)
- *Oracle Life Sciences Data Hub User's Guide* (Part E35305)

Product Release Notes

The release notes for the Oracle DMW and Oracle LSH products are available on the My Oracle Support website. Their article ID is listed in the installation 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.

Part I

Setting Up Studies

This section contains the following topics:

1

Introduction

Oracle Health Sciences Data Management Workbench (Oracle DMW) is designed for two basic functions:

- "Data Aggregation and Standardization" on page 1-1
- "Data Review and Cleaning" on page 1-2

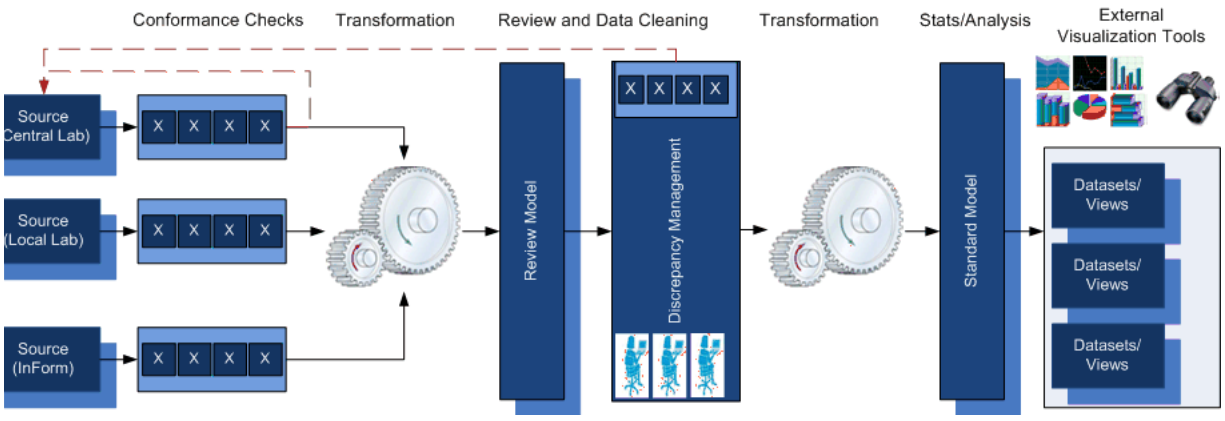
It is designed for integration with other systems:

- "Integration with InForm" on page 1-5
- "Integration with Labs" on page 1-5
- "Integration with SAS" on page 1-5
- "Integration with Informatica" on page 1-6
- "Integration with Data Visualization Tools" on page 1-6
- "Integration with Oracle Life Sciences Data Hub" on page 1-6

Data Aggregation and Standardization

Oracle Health Sciences Data Management Workbench (Oracle DMW) lets you load and merge clinical data and metadata from various sources, including InForm and labs, in one location where you can then transform raw clinical data to successive standard structures suitable for review and analysis. These structures are sets of tables called *clinical data models* that can be reused from a library or from study templates.

Figure 1–1 Clinical Data Transformation in a Study



The diagram shows a simple study setup in Oracle DMW. All studies have a clinical data model for each data source: here, a local lab, a central lab, and an electronic data capture system—InForm. The tables in these clinical data models are generated with the same structure as in the source system. As Oracle SQL Loader loads data from each source it checks data conformance. Then a transformation program reads the raw source data, derives values, merges and rearranges data as required, and writes to a target model whose purpose is data review. Validation (edit) checks run on data in the review model and identify data discrepancies. Then another transformation program reads data from the review model and writes to a statistics analysis model in SDTM format.

You can create any number of clinical data models and transformations from one to another. You can have parallel models; a single linear chain is not required. Creating a transformation from one model to another is a partially automated process and if you make changes to the source or target model of an existing transformation, you can use the Upgrade feature to upgrade the transformation to reflect these changes.

If you build up a set of validated clinical data models and study templates, you can reuse them in many studies, focusing validation effort on any changes you make in a particular study.

If you reuse standard source and target models without change, you can reuse the transformation between them without change.

Data Lineage The system tracks each data point from tables in the source models through each transformation to the final model. The system recognizes the unique key of each source record, enabling the exchange of comments and data updates between Oracle DMW and source data systems.

Data Blinding You can hide sensitive data by blinding entire tables or providing masking values for specific columns, rows, or cells. Once blinded, sensitive data cannot be viewed, even through multiple transformations, without explicit authorization by a user with a combination of special privileges.

Data Export You can create an Oracle LSH data mart to export data in files. Supported formats include SAS Transport (XPort and CPort), SAS datasets, fixed length or delimited text, or Oracle Export, optionally compressed into zip files.

Data Review and Cleaning

All Data—One Location Oracle DMW provides a full set of features that allow a data manager to easily review clinical data from various sources at one central location.

Seamless Hand-off Data managers, medical monitors, and biostatisticians can all work in a single environment, handing off work to one another as required.

Identifying Discrepant Data

A **discrepancy**—called a *query* in InForm—is associated with a single data point that has been identified as faulty or possibly faulty. Identifying and resolving discrepancies and correcting the underlying data helps ensure that clinical data is complete, accurate, and compliant with the study protocol.

Discrepancies are identified and created in several ways:

- **InForm queries** are imported as discrepancies with their source context intact. Oracle DMW users can add a comment or question to an InForm query

discrepancy and send it back to InForm, where a CRA or other InForm user can respond and/or correct the data point and send it back. InForm queries are visible in Oracle DMW but must be resolved in InForm.

- You can write **validation check programs** to examine data, including checking lab ranges, checking CRF data against lab data, and comparing multiple data points across CRFs, and create discrepancies against data points that are, or may be, faulty. You can write a validation check so that it creates either Candidate- or Open-state discrepancies. You can set up validation checks to automatically close discrepancies they created when the underlying data point is corrected, or require manual review before closing.
- Users can examine data in the Listings page, write ad hoc queries to identify faulty data, and create discrepancies **manually**.

The following sections describe Oracle DMW features used for data review and cleaning:

- "Listings Pages" on page 1-3
- "Discrepancies Page" on page 1-4
- "Custom Flags" on page 1-4
- "Customizing the Workflow" on page 1-5

Listings Pages

The Listings pages display records in a selected clinical data model:

- The **Default Listings** page displays all data in the model, table by table, consistent with the privileges of the user.
- The **VC (Validation Check) Listings** page displays records containing data identified as discrepant with related data as specified by a validation check.
- The **Custom Listings** page displays records retrieved by queries developed by data managers using the Query Builder. These queries can be saved and made available to all users with the required access.

You set up security so that, for example, data managers have access to the data in the input and Review models and Biostatisticians have access to the data in the Analysis model. Access can be restricted by table within the model, and special privileges are required to see blinded data. These restrictions apply to all three Listings pages.

Data Manager In any of the Listings pages, the data manager can:

- Write custom queries to identify faulty data.
- Review data identified as faulty by validation checks.
- Filter data by flag, category, or value such as subject or subject visit.
- Open discrepancies against or apply flags to a single data point or to many at once—for example, to all data identified by a custom query or validation check.
- View InForm queries against data points and InForm status flags.
- View the data lineage of any data point back to its source system and forward to subsequent clinical data models.
- With the required privileges, perform an audited blind break to see real, sensitive data that is otherwise hidden.

Biostatistician A biostatistician can, for example:

- Filter data by flags applied by special validation checks created for the purpose of tracking the completeness and cleanliness of data in individual CRFs, labeling those that are ready for analysis.
- Write custom queries to check for a given condition and mark the resulting data with a flag.
- View and filter data at any time, without waiting for an official hand-off. Data loads can trigger transformation to the Review model, which can trigger transformation to the Analysis model so that statisticians always have the most current data possible.
- With the required privileges, perform an audited blind break to see real, sensitive data that is otherwise masked.
- View data graphically and interactively in a visualization tool; see "Integration with Data Visualization Tools" on page 1-6.

Discrepancies Page

The Discrepancies page displays only data that has been identified as discrepant within a selected clinical data model, including:

- Discrepancies raised by validation checks
- Discrepancies raised manually
- InForm queries imported as discrepancies. These must be resolved in InForm, but Oracle DMW users can send messages to InForm users about InForm queries.

Data Manager and Medical Monitor can both act on discrepancies:

- Filter discrepancies by subject, visit, table, item, state, tag, data source, or categories you define.
- View the full record that contains the discrepant data point.
- View the history of the discrepancy.
- Apply an action to change the status of a discrepancy. For example, a data manager might change the status from Candidate to Open and requiring medical review.

  The medical reviewer can filter for discrepancies requiring medical review, view the data in context, and change the status to Canceled or Closed, or send a message to InForm or a lab requesting action there.
- Send discrepancies on InForm data to InForm.
- Export discrepancies on lab data to a spreadsheet to be sent to the lab.

Custom Flags

You can create any number of flags, each with any number of states, for use in tracking data review. These flags can be applied by data managers or other users, or by validation checks.

In addition, InForm Subject and Subject Visit status flags are imported and viewable in Oracle DMW.

Customizing the Workflow

Resolving a discrepancy can involve multiple users and groups. Oracle DMW comes with a set of discrepancy states, allowed transitions from one state to another, and predefined actions available for users to move a discrepancy from one state to another. These actions constitute a workflow among users that you can use out of the box. However, if you need a more complex, fine grained workflow to satisfy your standard operating procedures, you can define custom actions that apply custom tags that serve as substates, effectively creating a custom workflow.

Integration with InForm

Oracle DMW provides a bidirectional exchange of study data, including InForm queries and Oracle DMW discrepancies, between InForm and Oracle DMW. Loading data from InForm can trigger all downstream data processing, including validation checks and transformations of new and updated data from one data model to the next.

- In each study, the system automatically creates the InForm source clinical data model based on metadata from InForm
- The system loads data from InForm according to a configurable schedule
- Data loads can trigger running transformation programs to propagate data updates to downstream clinical data models
- The system stores the unique InForm context of each data point and recognizes data changes made in InForm.

InForm queries raised in InForm must be resolved in InForm, but can be viewed in Oracle DMW. Oracle DMW users can comment on InForm queries and InForm users can reply. Oracle DMW users and validation checks can raise discrepancies against InForm data, and users can send these discrepancies to InForm as queries.

Oracle DMW can maintain multiple InForm studies in a single database.

Integration with Labs

Oracle DMW supports partially automated exchange of files with any number of labs. The File Watcher feature checks for files from labs in a specified location and automatically processes them upon detection, loading data into the clinical data model created for this purpose.

Loading data from a lab can trigger all downstream data processing, including validation checks and transformations of the new and updated data from one clinical data model to the next.

Discrepancies can be exported to a file and sent to a contact person at the lab.

The system stores the unique external context of each data point and recognizes data changes made at the lab.

The same features could support integration with any system providing data in SAS or text files.

Integration with SAS

You can integrate SAS (purchased separately) in order to write custom programs in SAS and execute them, and to create SAS CPort, XPort, and dataset data marts to export data from Oracle DMW.

Integration with Informatica

You can integrate Informatica (purchased separately) in order to write custom transformation programs in Informatica and execute them.

Integration with Data Visualization Tools

You can use external data visualization tools to allow users to view clinical data graphically and interactively with the protection of Oracle DMW security and blinding access privileges. Oracle Business Intelligence Enterprise Edition is included for this purpose, and third party tools that already have an adapter to Oracle Life Sciences Data Hub (Oracle LSH) are available. You must create a Business Area object in Oracle LSH to take advantage of this functionality.

You can integrate other third-party tools using the Oracle LSH generic visualization adapter.

Integration with Oracle Life Sciences Data Hub

Oracle DMW is installed on top of Oracle Life Sciences Data Hub (Oracle LSH), which provides its underlying internal data model, execution engine, validation lifecycle, and security system. The Oracle DMW database is an Oracle LSH database. Oracle DMW studies are Oracle LSH domains, and clinical data models are a new Oracle LSH object type, as are codelists, transformation mappings, and more.

You can use Oracle LSH features including:

- Data marts to **export SAS datasets, CPort or XPort files, text files or Oracle Export files** based on the tables in a clinical data model.
- Business areas to view and report on data in a **data visualization** tool.
- **Snapshot labels**; see "Applying Snapshot Labels" on page 8-18

See the first chapter of the *Oracle Life Sciences Data Hub Implementation Guide* for an overview. See "Object Ownership" on page A-6 for more information on the relationship between Oracle LSH and Oracle DMW objects.

2

Creating Studies and Study Templates

This section contains the following topics:

When you log in, select a study and a lifecycle mode to work in:

- "Selecting a Study" on page 10-1
- "Selecting a Lifecycle Mode" on page 10-1

About Studies, Study Templates, and Libraries

An Oracle Health Sciences Data Management Workbench (Oracle DMW) study consists of sets of tables, programs, and other components that allow you to review, clean, and analyze clinical data. When you start using the system you must create new studies, explicitly defining each component in a partially automated process.

When appropriate, you can mark a study as a template. The template includes all study components, including transformations and validation checks as well as clinical data models. You can then apply the template to new studies. Applying a template creates a copy of each study template component's metadata in the new study. You can modify or delete template components and add components as required. You can build up a set of study templates; for example, for different therapeutic areas or study types.

Libraries contain clinical data models, which are groups of tables. Library models are available across studies and are intended to be used as standards. When you create a study model from a library model, the system maintains a relationship between the two. If you update the library model, you can propagate the changes to any study model created from it.

By contrast, applying a study template is a simple Copy operation. The system does not maintain a relationship between the original template and the copy, and you cannot update study objects automatically if their corresponding template object is updated.

You also create code lists in a library for use as lists of allowed column values. Code lists remain in the library for reference; they are not copied into studies.

> **Note:** You create other objects used in the data review process, including tags, flags, and categories, in the Administration page. They are not study components and are not part of study templates, but are available for use in any study; see Chapter 8, "Administration."

Study templates and libraries contain only metadata, or object definitions. Studies also contain metadata, but to use the objects (data models, tables, programs and so on) you must *install* them. Installation creates object instances and the actual database tables and packages required to store data and run programs. For more information on object definitions and instances, see the *Oracle Life Sciences Data Hub Application Developer's Guide*.

Creating and Using Studies and Study Templates

A DMW study or study template includes:

- **Clinical data models**, each containing a set of tables. Each data model is intended for a particular purpose; for example, reviewing data or transforming data into standard CDISC-formats for analysis; see Chapter 3, "Creating Clinical Data Models" for more information.

  > **Note:** Input clinical data models—models that receive data directly from InForm or a lab—are not included in study templates. They must reflect the exact structure of the source data and are created from the source data for each study. Only one InForm model can be in a study.

- **Mappings and transformation programs** to read data from one clinical data model and write to the next ; see "Transforming Data to Standard Structures" on page 5-1.
- **Validation check programs** to validate data in a clinical data model and raise discrepancies; also called **edit checks**; see "Creating Validation Checks" on page 6-1.
- **Saved custom listings** created by data managers, medical reviewers, or other end users to analyze and clean data; see Chapter 11, "Reviewing Data".
- **Business areas** to support data visualization tools; see "Setting Up a Data Visualization Tool" on page 8-19.

> **Tip:** You can apply only one study template to a study. Oracle recommends creating study templates that are as complete as possible for each study type, and then deleting any tables, validation checks, or other components not needed in a particular study after applying the template.

This section contains the following topics:

Creating a Study

To create a study:

1. On the Home page, click the Add icon (+) in the Studies pane.
2. Enter a name and description for the study.
3. Select the following attributes if appropriate:
   - **Template**: If selected, this study is available as a template for other studies. You can select this setting later, after testing the study; see "Making a Study Available as a Template" on page 2-4.
   - **Active**: Select this check box to indicate that the study is being set up or is ongoing. Deselect the box to indicate that the study is closed. The system does not use this field in Release 2.3.1.
4. **Therapeutic Area (or other category)**: Select the category to which the study belongs. The label for this field and the choices available are customizable by your company. The libraries you develop of clinical data models will be divided into these categories; see "Setting Up Library and Study Categories" on page 8-15.
5. **Study Size**: Select **Small**, **Medium**, or **Large** to indicate the amount of patient data to be collected in this study relative to other studies. The system uses this setting to help determine which partition to use for this study's data in certain cross-study internal tables; see "Configuring Partitioning" on page 8-16 for more information.

   You cannot change this value after saving.
6. Save.

Applying a Template to a Study

If a study template already exists that suits your needs, apply it to your study. You can apply only one template to a study.

To see a template in order to decide if you want to use it, query for it in the Studies pane of the Home page, then navigate to its clinical data models, transformations, and validation checks. You can also look at its Listings pages.

To apply a template to your study:

1. In the Home page, create or open your study.
2. Go to Study Configuration, then click the Study Templates node. The system displays a list of study templates with their descriptions.
3. If necessary, query by example to find the template you want:
   a. If empty fields are not already displayed above each column, click the Filter icon.
   b. In the empty field above the Study Name column, enter all or part of the template's name.
   c. In the empty field above the Description column, enter a key word that should be in the description, such as the therapeutic area or other study type for which the template is intended.
   d. Press Enter.
4. Select the template and click **Apply Template to Study**.

Creating Study Components

You can add components manually to complete the study:

1. Create an input clinical data model—set of tables—for each external study data source, including InForm and each lab. The system automatically creates tables with the same structure as in the source system and loads data into them; see "Creating a Study Clinical Data Model" on page 3-4 for more information.
2. Create target clinical data models for reviewing and analyzing data. You can develop standard library clinical data models and reuse them in multiple studies.
3. Create code lists and associate them with table columns to define lists of allowed values. Code lists are available for use in all studies; see "Creating Code Lists" on page 4-3.
4. Create a transformation for each target data model: Map one or more source data models to each target model and map tables in the source models to tables in the target model. Define table mappings as joins, unions, pivots, unpivots, or directo one-to-one mappings, and the system generates the supporting program code. Map columns in source tables to columns in target tables, using expressions and functions as required. See Chapter 5, "Transforming Data to Standard Structures" for more information.
5. Write validation (edit) checks to detect discrepant data; see Chapter 6, "Creating Validation Checks" for more information.

In addition, create tags and categories for use in filtering discrepancies and managing your discrepancy workflow. Create flags for use in reviewing data and tracking subject/visit status. These are available for use in all studies; see Chapter 8, "Administration".

Making a Study Available as a Template

If you want to make your study available for use as a template for other studies, mark it as a template:

1. On the Home page Studies pane, select the study and click the Edit icon.
2. Edit the description to make it easy for others to decide whether to select this study for use as a template. The maximum length is 2000.

   Tip: Oracle recommends developing company standards for this purpose.

3. Select the **Template** check box.
4. Click **OK**.

Using the Validation Lifecycle

When you create a study, the system automatically creates three lifecycle areas for the study. Within each lifecycle area, each clinical data model has its own database schema. Each lifecycle area has a different purpose:

- **Development** to create clinical data models, transformation programs, and validation checks either manually or from a study template or library as required for the study. You can load data into the development lifecycle schema and do initial testing and fine-tuning and begin formal testing there.
- **Quality Control** for formal testing of all study components.

- **Production** to load, review, and clean production study data. The system prevents destructive changes to tables and models in a production environment.

You select the lifecycle area you want to work in on the Home page in the bottom left corner. To change from one lifecycle area to another you must return to the Home page, select the lifecycle area and reselect the study if necessary, and then go back to the page you were working in.

As you test study components in the development or quality control lifecycle areas, you promote them to the next lifecycle stage: from development to quality control or from quality control to production. You can associate documents supporting the change in validation status, such as log files or signoff documents, with the study component.

As soon as you promote them they become visible with the new validation status in the next lifecycle area. However, you must install the current version of the object in the higher lifecycle area for the changes to take effect.

Similarly, if you need to make a change to a study component with a validation status of either quality control or production, the system changes its status to development as soon as you check it out. You make your changes, install the study component, test it and promote it when appropriate.

You can set an object's validation status to **Retired**. If the object is executable, like transformations or validation check batches, it can no longer be executed. This has no effect on objects currently in use, but prevents creating other objects based on the retired object as part of a study template or library.

3

Creating Clinical Data Models

This section contains the following topics:

To find a clinical data model:

1. For study models, select the study and lifecycle area context in the Home page and then go to the **Study Configuration** page.

   For library models, go to the **Library** page and select a **Study Type**.

2. In the Clinical Data Model field, type part or all of the model's name and press Enter. The system lists all data models in the study or library whose name contains the string you typed.

3. Select the clinical data model you want by clicking its name or pressing the Down arrow and press Enter.

About Clinical Data Models

An Oracle Health Sciences Data Management Workbench (Oracle DMW) clinical data model is a logical group of tables.

You must create a clinical data model for each external data source—InForm or lab—in each study. When you upload SAS datasets or properly formatted text files, or connect to an InForm study reporting database, the system automatically creates tables in the corresponding model with the same structure.

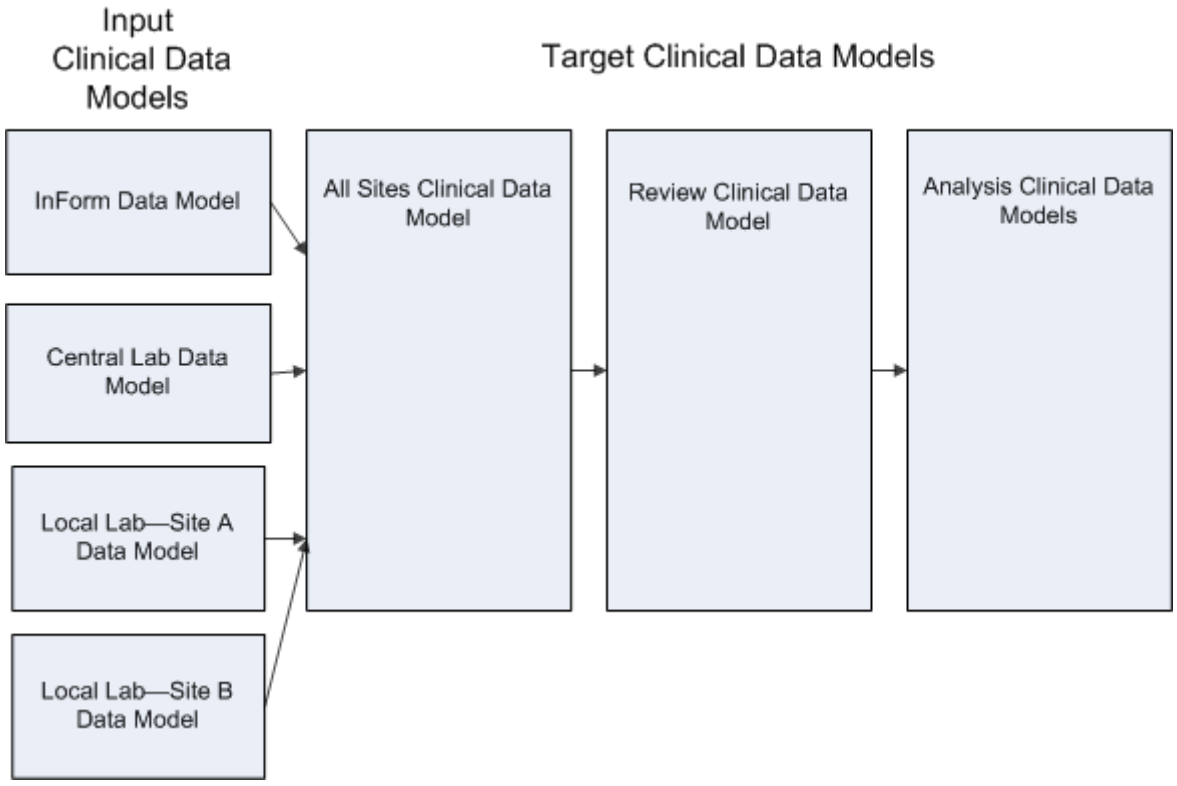
You can create other clinical data models with tables that have structures better suited to reviewing, analyzing, or reporting data—for example, your company's internal

standard or CDISC standards SDTM or ODM—and transform raw clinical data to these structures in an appropriate sequence; see Chapter 5, "Transforming Data to Standard Structures."

Library Models You can develop a library of clinical data models that are available for reuse in any study. If you update a library model, you can synchronize study models that were created based on the library model. However, if you have made changes in the study data model, you will lose them if you synchronize with the library model.

Study Models You can create a clinical data model for use in a particular study by:

- Copying an existing library model into the study
- Applying a study template that contains the model you want
- Uploading SAS or text files from which the system can read the data structure and automatically create tables to match; see "Required Syntax for Table Metadata Text Files" on page A-3 for more information.
- Manually defining each table and column



Installation

Before you can load data into a clinical data model you must **install** the model. This process creates a database schema for the model and creates actual database tables from the table metdata. You cannot run any transformation programs or validation checks until you install the model and the program.

When you install a clinical data model, the system automatically creates or updates a query that selects all columns and rows from all tables in the data model, including

any masking of blinded data. The system uses this view to display data on the Listings page. This view also serves as the starting point for custom listings.

> **Note:** When you install a transformation, the system also installs its target clinical data model.

Data Blinding

The system supports blinding sensitive data at several levels: whole tables, whole columns, whole rows meeting specified criteria, and cells meeting specified criteria. You can specify masking values for use with blinded columns, rows, and cells.

See the *Oracle Life Sciences Data Hub Implementation Guide* for information about blinding-related security user privileges.

> **Note:** A cell is the intersection of a table column and row; a single data point for a record.

Blinding Data in Input Models

Blinding requirements vary by source type:

- **Text files**: Tables created by uploading text metadata files can be created with some blinding-related attributes set; see "Required Syntax for Table Metadata Text Files" on page A-3.
- **SAS**: tables created by uploading SAS datasets are created as nonblinded by default. If data should be blinded, you must define blinding attributes manually for tables in the input data model.

  > **Note:** You can create a table initially from a text metadata file and subsequently load data into it from a SAS file.

- **InForm**: *Hidden* data in InForm has different behavior and rules than blinded data in Oracle DMW, so **you must manually define blinding within Oracle DMW for InForm data** during initial study setup. By default, all tables are created as not blinded.

Blinding Data in Target Models

As you define downstream clinical data models for review and analysis, you must define the blinding attributes of tables and columns, rows, and cells that are targets of blinded source tables, including masking values.

The system does not allow you to map a blinded source table to a nonblinded target table unless you have special privileges and explicitly authorize the creation of the target table as nonblinded. This authorization is intended to be used only when the mapping is done in such a way that the target table contains no blinded data: only columns and rows containing nonblinded data are mapped to the target table. The authorization action is audited.

Blinding Data in Custom Listings and Validation Check Listings

When you create a custom listing or a validation check, the system generates the target table. If any of the source tables are blinded, the system makes the whole target table blinded by default. To see any data on the Custom or VC Listings page, you must

explicitly authorize the creation of the target table as nonblinded. You must then use the Expression Builder to mask specific columns, rows, or cells to ensure that no sensitive data that would break the blind is displayed. Special privileges are required for the authorization.

Note: If the target table is blinded, users with the required privileges can view the real data (not the masking values) if they request to do so. Each such access is audited.

Unblinding Data

A person with the required privileges can unblind the data in a table so that users with normal privileges can see the sensitive data, normally at the end of a study. Unblinding undoes all types of blinding: whole table, whole column, or row or cell values meeting specified criteria.

Special privileges are required to unblind data and to view unblinded data.

To unblind data a blinded table:

1. Navigate to the table in its clinical data model in the Study Configuration tab.
2. Select the table and click the Edit icon.
3. Deselect the **Blinded** check box.
4. Click **OK**.

Requirements

- You must define a **primary key** for every table you create. This is required to trace the data lineage of each record from one clinical data model to the next and to support sending discrepancies back to their source system and recognizing them when the source system sends them back to Oracle DMW. You can define a primary key in the Constraints user interface tab or by uploading a text file; see "Required Syntax for Table Metadata Text Files" on page A-3.
- All the data you plan to compare in a **validation check** must be included in the same clinical data model.
- You must check out a clinical data model to modify it. Each checkout creates a new **version**.
- Each clinical data model begins with a **validation status** of Development and must be promoted to Quality Control and then Production before being used in a production environment; see "Using the Validation Lifecycle" on page 2-4.

Creating a Study Clinical Data Model

Clinical data models are required in every study.

To create a clinical data model:

1. Select the appropriate study in the Home page and click **Study Configuration**.
2. Click the Add icon (+) for Clinical Data Models. The Create Clinical Data Model window appears.
3. Enter a name and description for the model; see "Naming Objects" on page A-1 for restrictions.

4. Select the model type:
   - **Input** if its data will be loaded from an external system such as InForm or a lab.
   - **Target** if its data will be derived from another model in the system.
5. If you selected Input, select an Input Type:
   - **File** if you plan to load data from files, as from a lab.
   - **InForm** if you plan to load data from InForm. In this case, skip the next step. Click **OK** and go to "Configuring the InForm Connector" on page 3-16.
6. **File Type** (Required if Input Type is File): Select the type of file to be loaded into this model: **SAS** or **Text.** If the lab that is sending you data for this model may send both types of files, you must define two models, one for each file type.
7. Under **Select from source** select the source, if any:
   - **None** to define tables and columns manually.
   - **Load from file** to create tables by uploading files. The system can create tables from SAS Transport (CPort or XPort) files, or a .zip file that contains one or more SAS datasets or text metadata (.mdd) files. Metadata files are the only way to automatically create tables with all blinding attributes set; see "Required Syntax for Table Metadata Text Files" on page A-3.

     Note: You can create a table initially from a metadata file and subsequently load data into it from a SAS file.

     Browse to the file.
   - **Import from Library** to copy a library model. Additional fields appear.

     You can type all or part of the name of the model you want to copy and/or the library that contains it to filter the models displayed. The library, or namespace, is the therapeutic area or whatever other category your company uses.

     Select the model and click **OK**.
8. Click **OK**.

   Notes: If you have not already created a primary key for every table in the model, you must do so and then install or reinstall the model.

9. If the model is an input model, configure its method of importing data:
   - For InForm models, go to InForm Configuration; see "Configuring the InForm Connector" on page 3-16.
   - For File models, go to Watcher Configuration; see "Configuring File Watcher" on page 3-20.
10. Click **Install Model**. You must install the model before you can load data. The model must be installable.

    A model is not installable if it does not have any tables or if any of its tables are not installable. A table is not installable if it has no columns.

In addition, the table must have a primary key constraint and if the table has a Blinding Flag set to Yes, blinding must be completely defined. However, these requirements do not prevent the installation of the model.

Adding Tables to a Clinical Data Model

This section contains the following topics:

If you create a study model from a library model or a study template, all the tables and columns are included. If you are creating an InForm input model, the system creates tables and columns automatically; you can skip this section and go to "Configuring the InForm Connector" on page 3-16.

Adding Tables from a File

You can upload certain types of files (see below) into a clinical data model to automatically create a table with columns in the same structure as in the file. You can do this either when you first create a model or afterward by modifying the model:

1. Navigate to the clinical data model to which you want to add a table.
2. Select the model and click **Check Out**. (If someone else has checked it out, you cannot work on it.)
3. Click the Edit icon (a pencil) in the upper left corner. The Edit Clinical Data Model Table window opens.
4. Under **Select from source**, select **Load from file**. A Browse field appears.
5. Browse to the file you want to load and click **Browse**. The file must be one of the following and must be compatible with the File Input Type setting for the model:
   - SAS dataset—creates one table
   - SAS CPort file—creates multiple tables
   - SAS XPort file—creates multiple tables
   - MDD file—creates one table
   - Zipped file containing one or more .mdd files (one .mdd file per table).

     You can include data files in the zipped file; they must have an extension of .txt or .csv and contain data in the required format; see "Required Syntax for Table Metadata Text Files" on page A-3. Metadata and data files must use the table name as the file name, plus the appropriate extension.

     Notes: All tables that contain clinical data must have a **primary key**; see "Requirements" on page 3-4.
6. Click **OK**. See "Setting Additional Attributes" below.

Adding and Modifying Tables Manually

To add tables to the clinical data model manually or to specify blinding attributes:

1. Navigate to the model to which you want to add a table. Check out the model if it is not already checked out.

   Note: Do not make structural changes to tables in InForm models; see "Modifying an InForm Input Model" on page 3-29 for information on the changes that are and are not allowed.

2. In the Tables tab, click the Add icon (+). The Create Clinical Data Model Table window opens.
3. Enter values in the following fields:
   - **Name**: Enter a name for the table; see "Naming Objects" on page A-1 for restrictions.
   - **Description**: (Optional) Enter a description of the table.
   - **Oracle Name**: By default, the system populates this with the value you entered for the name. However, it should not be longer than 30 characters; see "Automatic Name Truncation" on page A-2.
   - **SAS Name**: By default, the system populates this with the value you entered for the name. However, it should not be longer than 32 characters; see "Automatic Name Truncation" on page A-2.
   - **SAS Label**: (Optional) By default, the system populates this with the value you entered for the name. It can be up to 256 characters.
   - **Alias:** Enter an alternate name for the table to be used consistently across studies for tables that store the same type of information but may have different names; for example, if the table's name is DEMO or DEMOG, enter DEMOGRAPHY. The system uses aliases to generate proposed mappings between tables in different models.

     You can use aliases to provide an easily identifiable, user-friendly name for the table. However, shorter names work better in the user interface.

     You can enter multiple aliases separated by commas, with no space after the comma.
   - **UOW (Unit of Work) Processing Type**: See "Setting Data Processing-Related Attributes" on page 3-8
   - **SDTM Identifier**: If this table corresponds to an SDTM standard Subject or Subject Visit table, select its identifier from the list; see "Adding Subject Visit and Subject Tables" on page 3-13 for information about requirements.
4. Set blinding attributes; see "Setting Blinding-Related Attributes" on page 3-8.
5. Click **OK**.

Setting Additional Attributes

Oracle DMW tables have attributes that SAS datasets do not have. After uploading the file and creating the table(s) you must set these attributes in the clinical data model page.

Defining a Primary Key

A primary key is required. You can add a primary key to each dataset before uploading the dataset or upload the file to create the table(s), then click the Edit icon for each table in turn and follow instructions in "Adding Constraints to a Table" on page 3-12.

Setting Blinding-Related Attributes

By default, a table's Blinding Flag is set to No. If the table will ever contain blinded data, you must change this setting. If it will contain blinded data but you want to display the table's nonsensitive data, you must specify **column-, row-, or cell-level blinding**. You can set the Blinding Flag and Blinding Status in an .mdd file; see "Required Syntax for Table Metadata Text Files" on page A-3, but you must define the blinding type and criteria in the user interface; see "Data Blinding" on page 3-3 for further information.

After you create a table, set its blinding attributes:

- **Blinded**: Select if the table may ever contain any sensitive data that should be hidden. The default value is unchecked. This setting corresponds to the Blinding Flag table attribute.
- **Blinding Type**: (Available only if Blinded is selected.) Select one:
  - **Table**: Select to hide all data in the table. The table appears in the Default Listings page, but none of its data is displayed unless a user with special privileges requests to view it. Such viewing is audited.
  - **Column**: Select to mask all values in one or more columns, or in certain rows. Then click **OK** and select the columns.

    Note: If you select Column, you must specify a masking value for the blinded column by editing the column definition in the Columns tab. If you want to mask the values only in some rows—cell-level blinding—you can enter masking criteria during column definition too; see "Adding Columns to a Table" on page 3-9 and "Specifying Masking Attributes for a Column" on page 3-11.

  - **Row**: Select to hide certain rows in their entirety, or mask the values in certain cells.
- **Blinding Criteria**: (Available only for Row-level blinding.) Click **Expression Builder** to specify which rows should be hidden from users with insufficient privileges. The Expression Builder window opens.

  Build an expression. For example, to hide all rows whose Test column's value is `LiverCount`, select column **Test**, operator **is**, and value **LiverCount**.

  Available operations include: `less than, less than or equal to, greater than, greater than or equal to, is, is not, is blank, is not blank`.

  You can add as many criteria as you need, combining criteria with `and, or, not`, or `()`. See "Using the Expression Builder" on page 5-12 for more information.

Setting Data Processing-Related Attributes

After you create a table and define its primary key, set its UOW (Unit of Work) Process Type attribute; see "Data Processing Types and Modes" on page 9-1.

Note: You must define a primary key before you set the UOW Process Type attribute. When you define a primary key, the Process Type attribute value changes to **Reload**. Click the Edit icon (a pencil) to specify the UOW Process Type value.

Select one of the following:

- **Non UOW**: Jobs writing to the table will use Reload processing by default.
- **Subject**: Jobs writing to the table will use UOW processing with Subject as the unit of work. The table must have a column designated with the Subject SDTM Identifier, and must have a primary key that includes the Subject column.
- **Subject Visit**: Jobs writing to the table will use UOW processing with Subject Visit as the unit of work. The table must have one column designated with the Subject SDTM Identifier and another with the Visit SDTM Identifier, and both columns must be included in the primary key.

Tip: **Always** define tables as UOW if the Subject column is part of the primary key and as Subject Visit UOW if both columns are part of the primary key, because:

- Many tables serve as both sources and targets, and defining tables with UOW processing facilitates UOW processing when the table serves as a source, even if it is not possible to use UOW processing when the table is a target—for example, for tables whose data is loaded from an external system.
- If a table is defined as UOW the system will use UOW processing if possible; if not it will use Reload processing.

See "Data Processing Types and Modes" on page 9-1 and "Loading Data" on page 9-5for more information.

Adding Columns to a Table

To add columns to a table manually:

1. In the Column tab, click the Add icon (+). The Create Clinical Data Model Column window opens. The system automatically checks out the table if it is not already checked out.
2. Enter values in the following fields:
   - **Name**: Enter a name for the column; see "Naming Objects" on page A-1 for restrictions.
   - **Description**: (Optional) Enter a description of the column.
   - **Oracle Data Type**: Select the appropriate data type: Varchar2, Number, or Date. All standard rules for Oracle data types apply.
     - **DATE**. For each Date value, Oracle stores the following information: century, year, month, date, hour, minute, and second. Although date and time information can be represented in both character and number datatypes, the Date datatype has special associated properties.
     - **NUMBER**. Stores zero, positive, and negative fixed and floating-point numbers. A Number column can contain a number with or without a decimal marker and/or a sign (-).

  - **VARCHAR2**. Specifies a variable-length character string. For each row, the system stores each value in the column as a variable-length field unless a value exceeds the column's maximum length, in which case the system returns an error.

- **Length**: The requirements vary according to the data type:
  - **DATE**: No length required.
  - **VARCHAR2**: (Required) The default value is 50. The value must be between 1 and 4000.
  - **NUMBER**: (Required) The default value is 10. The maximum value is 38.

  If the data type is Number you must also enter a value for **Precision**; the total number of digits to the right of the decimal point allowed. For example, if Precision is set to 2 and a data value of 34.333 is entered in this column, the system stores the data value as 34.33. Oracle guarantees the portability of numbers with precision ranging from 1 to 38.
- **SDTM Identifier**: If the column corresponds to one of the standard identifiers supported by DMW and has a compatible data type, it is good practice to select it from the list because the system uses this information in several ways; see "Using SDTM Identifiers for Columns" on page 3-11.
- **Oracle Name**: By default, the system populates this with the value you entered for the name. However, it should not be longer than 30 characters; see "Automatic Name Truncation" on page A-2.
- **SAS Name**: By default, the system populates this with the value you entered for the name. However, it should not be longer than 32 characters; see "Automatic Name Truncation" on page A-2.
- **SAS Label**: (Optional) By default, the system populates this with the value you entered for the name. It can be up to 256 characters.
- **SAS Format**: By default, the system enters a dollar sign ($) followed by the value you entered in the Length field.
- **Default Value**: (Optional). You can enter a value to serve as the default for this column.
- **Aliases**: Enter one or more aliases, or alternate names for the column. If you want more than one alias, enter a comma-separated list with no spaces; for example: `dm,demo,demog,demography`

  The system uses these in automapping transformations.
- **Nullable**: If selected, having a value in this column is not required. This is the default value. If not selected, all rows must have a value in this column.
- **Code List**: (Optional) If the column should be populated with a limited set of values that are defined in a code list, select the appropriate therapeutic area (or other category) and then the code list. You can apply a code list only to columns with a data type of varchar2.

Note: If the table may need to be pivoted from a horizontal (short fat) structure to a vertical (tall skinny) structure during a transformation, the pivot column must be associated with a code list; see "Pivot" on page 5-10.

---

3. Enter masking attribute values; available only if the table has a masking type of Column; see "Specifying Masking Attributes for a Column" on page 3-11.
4. Click **OK**.

Using SDTM Identifiers for Columns

If a column corresponds to one of the standard identifiers supported by DMW and has a compatible data type, it is good practice select it from the list. The system uses these identifiers:

- In the Automap feature for transformations.
- USUBJID and VISITNUM SDTM identifiers are **required in any table to support filtering** records in the Listings pages.
- USUBJID and VISITNUM SDTM identifiers are **required in tables supporting Subject Visit Unit of Work processing**.
- USUBJID is **required in tables supporting Subject Unit of Work processing**.

Oracle DMW supports these identifiers:

- **Actual Visit Date** (Not an SDTM variable; date)
- **Country of Investigator Site** (**COUNTRY**; varchar2)
- **Visit Cycle** (**EPOCH**; varchar2)
- **Investigator ID** (**INVID**; varchar2)
- **Investigator Name** (**INVNAM**; varchar2)
- **Site ID** (**SITEID**; varchar2)
- **Site Name** (Not an SDTM variable; varchar2)
- **Study ID** (**STUDYID**; varchar2)
- **Subject ID within study** (**SUBJID**; varchar2)
- **Unique Subject ID across studies** (**USUBJID**; varchar2)
- **Visit Name** (**VISIT**; varchar2)
- **Visit Day** (**VISITDY**; varchar2)
- **Visit Number** (**VISITNUM**; number)
- **Visit Start Date** (**SVSTDTC**; varchar2)

Specifying Masking Attributes for a Column

If you selected a blinding type of Column for a table, at least one of the columns in the table must have a masking level specified as either column or cell. The default value is None, so you do not need to change nonblinded columns.

1. Select the column and click the Edit icon.
2. Select a **Masking Level**:
   - **Cell**: Masks the real data only in certain rows in this column. You must enter masking criteria to specify which row values are masked.
   - **Column**: Masks the real data in this column in every row.
   - **None**: No rows are masked in this column.

3. **Masking Value**: Do one of the following to specify what value(s) to display instead of the real values:
   - Enter a constant value to be displayed in every row.
   - Enter an expression that generates multiple values for the system to display.
   - Use the expression builder to create an expression to generate multiple values for the system to display.
4. **Masking Criteria** (Required only for cell-level masking): Click the Expression Builder icon to specify the criteria for blinding cells in the column; see "Using the Expression Builder" on page 5-12.

Adding Constraints to a Table

All clinical data model tables must have a primary key. This is required to support data lineage tracing; see "How the System Tracks Data Lineage" on page 5-24. If you create tables by uploading text files you can define constraints at the same time; see "Required Syntax for Table Metadata Text Files" on page A-3. InForm tables' constraints are imported as part of a metadata load and cannot be modified in the input model.

If you upload SAS files you must create the primary key and any other constraints manually:

1. Navigate to the model that contains the table and select the table in the Tables tab.
2. In the Constraints tab, click the Add icon (+).
3. Enter values:
   - **Constraint** Enter a name for the constraint. It must be unique among constraints for the table and must not contain special characters or Oracle or SQL reserved words.
   - **Description** (Optional)
   - **Constraint Type** Select one:
     - **Bitmap Index**: A bitmap index stores rowids (row IDs) associated with a key value as a bitmap. Each bit in the bitmap corresponds to a possible rowid. If a particular bit is set, the row with the corresponding rowid contains the key value.
     - **Check**: The check constraint allows you to specify allowable values for a particular column. Enter one allowed value in the **Add Value** field and click the arrow icon to move the value into the right-hand column; repeat for each value.

       If any row contains a different value for the column, the system does not insert the record but generates an error to the program writing to the table instance. If the program does not handle the error, the job fails.

       Note: You can also use code lists to specify allowed values for a column, but the system behavior is different: it inserts the record with the nonconforming value and reports a violation.

     - **Non-Unique Index**: A non-unique index keeps rows sorted on the specified column or columns to speed up queries.

– **Primary Key**: (Required) A primary key is a column or set of columns whose value(s) identify a row in a table as unique. The system uses the primary key to trace data lineage; see "How the System Tracks Data Lineage" on page 5-24.

  Primary key columns cannot have a null value in any row.

  The system automatically creates an index based on the primary key, which it uses to enforce a unique constraint and to speed up queries on the table.

  Check **Supports Duplicate** if you want to support inserting records with the same primary key value within a single data load, which is normally not desirable but may be required in a few cases. Selecting this option ensures that all records are loaded and not deleted but requires careful checking of the data. See "Supporting Duplicate Primary Keys in a Load" on page 9-7 for more information.

– **Unique Key** A unique key is similar to a primary key in that it can include one or more columns whose value(s) identify a row as unique. The difference is that the system allows null values in the columns that are part of a unique key.

  Any number of rows can include null (empty) values. A null in a column (or even all nullable columns in a composite unique key) satisfies the unique key constraint. However, you cannot have identical non-null values in the columns of a partially null composite unique key constraint.

Notes: The Not Null constraint is handled as an attribute called Nullable for each column.

For more information on Oracle constraints, see *Oracle® Database SQL Language Reference* 11g Release 2 (11.2) at `http://docs.oracle.com/cd/E11882_01/server.112/e26088.pdf`

- **Columns**: To include a column in the constraint, select it from the list on the left and use the arrow icon (>>>) to move it to the right.

Adding Subject Visit and Subject Tables

Oracle DMW supports filtering data and discrepancies by subject and visit. You may wish to track subject visit completeness using flags and custom programs. To support this functionality, you must include one Subject Visit table in at least one clinical data model in each study.

Oracle DMW ships with SDTM-compatible tables that you can copy. Alternatively, you can designate existing tables as your Subject and Subject Visit tables as long as they comply with the requirements in the following sections:

- "Subject Visit Table" on page 3-14
- "Subject Table" on page 3-15

Although Oracle DMW ships with both Subject and Subject Visit tables, the system uses only the Subject Visit table to support filters.

Copying Subject Visit and Subject Tables

To copy Subject or Subject Visit tables into a clinical data model:

1. In the Study Configuration page, select the clinical data model, check it out, and click **Copy Subject/Visit from Library**. The Copy Subject/Visit from Library window opens.
2. Select one:
   - **Copy from Library**: Allows you to copy the Subject, Subject Visit, or both tables created in a library clinical data model by your company; see "Subject Visit Table Requirements" on page 3-14.
   - **Default Structure**: Allows you to copy the Subject, Subject Visit, or both tables that are shipped with Oracle DMW. The table and columns are already associated with the appropriate SDTM identifiers; see "Shipped Subject Table Metadata" on page 3-15 and "Shipped Subject Visit Table Metadata" on page 3-14.
3. If you selected Copy from Library, select the study type—therapeutic area or other category—that contains the library model whose table(s) you want to copy, and then select the library model.

   If you selected Default Structure, proceed to the next step.
4. Select one:
   - **Subject** to copy only the **Subject** table
   - **Subject Visit** to copy only the **Subject Visit** table
   - **Both** to copy both
5. Click **OK**.

Subject Visit Table

If you designate an existing table as the Subject Visit table, it must satisy these Subject Visit Table Requirements. Alternatively, you can copy the shipped Subject table; see Shipped Subject Visit Table Metadata.

Subject Visit Table Requirements

If you designate another table as the SDTM Subject table:

- Its primary key must include the Unique Subject ID and Visit Number columns in that order, and no other columns.
- These columns must be linked to USUBJID and VISITNUM SDTM identifiers.

Filter Requirements See "Defining Tables to Support Filtering in the Listings Pages" on page 3-16 for information on additional requirements.

Shipped Subject Visit Table Metadata

The shipped Subject Visit table follows SDTM specifications.

Primary Key The shipped Subject Visit table's primary key is the USUBJID and VISITNUM columns.

Columns The shipped Subject Visit table columns are:

- STUDYID (Study Identifier) is the unique ID of the study.
- DOMAIN (Domain Abbreviation) is an abbreviation for the CDISC domain; for example AE, CM. In the Subject Visit table the value must be SV.

- USUBJID (Unique Subject Identifier) is the unique ID for each subject across all studies for all applications or submissions involving the product.
- VISITNUM (Visit Number) is the clinical encounter number; a numeric version of VISIT, used for sorting. Decimal numbering may be useful for inserting unplanned visits.
- VISIT (Visit Name) is the protocol-defined description of the clinical encounter of description of unplanned visit. May be used in addition to VISITNUM and/or VISITDY as a text description of the clinical encounter.
- VISITDY (Planned Study Day of Visit) is the planned study day of the start of the visit based on RFSTDTC in Demographics.
- SVSTDTC (Start Date/Time of Visit) is the start date/time of a subject's visit, represented in ISO 8601 character format.
- SVENDTC (Study Day of End of Visit) is the end date/time of a subject's visit, represented in ISO 8601 character format.
- SVSTDY (Study Day of Start of Visit) is the study day of start of visit relative to the sponsor-defined RFSTDTC.
- SVENDY (Study Day of End of Visit) is the study day of end of visit relative to the sponsor-defined RFSTDTC.
- SVUPDES (Description of Unplanned Visit) is a description of what happened to the subject during an unplanned visit. Null for protocol-defined visits.

Subject Table

Note: The system does not use the Subject table internally.

Shipped Subject Table Metadata

The shipped Subject table is based on the *CDISC SDTM Implementation Guide (Version 3.1.2)*. It includes all SDTM columns and also COUNTRY and SITE_ID columns.

Primary Key The shipped Subject table's primary key is the Subject ID, called USUBJID.

Columns The shipped Subject table columns are:

- STUDYID is the unique ID of the study.
- DOMAIN is an abbreviation for the SDTM Domain.
- USUBJID is the unique ID for each subject across studies. This is the primary key of the table.
- SESEQ is the sequence number.
- ELEMENT is a basic building block for time within a study.
- ETCD is the element code.
- SESTDTC is the start timestamp of the element, including date and time.
- SEENDTC is the end timestamp of the element, including date and time.
- TAETORD in the planned order of elements within a study arm.
- EPOCH is a set of elements.

- SEUPDES is a description of an unplanned element.

Defining Tables to Support Filtering in the Listings Pages

Users can create and reuse filters in the Listings pages to display only the data they need to work on at a particular time; see "Creating and Using Filters" on page 11-6. To support this functionality, you must define tables as follows:

- "Subject Visit Table" on page 3-16
- "Subject and Visit SDTM Variable Columns in All Tables" on page 3-16

How Filters Work

The Filter logic looks first at the Subject Visit table and finds each distinct Subject/Visit combination that meets the criteria of the filter. It then displays those Subject/Visits in the table that the user is viewing in the Listings page.

The user must specify the clinical data model that contains the Subject Visit table to use. The table can be in any clinical data model, but the user must have view privileges on the table. Therefore, if you have one group of users with security privileges to only one clinical data model and another group of users with access only to a different model, you need to have a Subject Visit table in both models or grant all users access to the Subject Visit table in one model. You can grant access to a single table in a model; see "Applying Security" on page 3-27.

Subject Visit Table

You must create at least one Subject Visit table in one clinical data model that has the SDTM ID for Study Visit table and meets the Subject Visit Table Requirements.

In addition, the user can filter on values in certain Subject Visit table columns only if the Subject Visit table contains the columns and the columns are linked to the following SDTM identifiers: COUNTRY, SITENAME, SITEID, INVNAM, INVID, EPOCH, VISIT, VISITNUM, VISITDY, SVSTDTC, SUBJID, USUBJID. See "Using SDTM Identifiers for Columns" on page 3-11.

Subject and Visit SDTM Variable Columns in All Tables

Filter logic requires a join of the Subject Visit table with the table being viewed in the Listings page on the USUBJID and VISITNUM columns. Therefore, all tables whose records users may need to view and act on must have columns for unique subject ID and visit number, and those columns must be linked to the corresponding SDTM identifiers USUBJID and VISITNUM.

Security Privileges for Making Filters Public

A special privilege is required for making filters public. There is a separate privilege for each lifecycle area. These privileges, MRKFILPUB_DEV, MRKFILPUB_QC, and MRKFILPUB_PROD, have been added to the predefined roles DME_STUDY_DEVELOPER, DME_STUDY_QC, and DME_STUDY_PROD, respectively. The predefined role DME_STUDY_ADMIN has all three.

Configuring the InForm Connector

The system uses a database connection to import data and metadata from InForm and a web service to send discrepancies and query comments back to InForm. You configure both for each study's InForm input clinical data model in the Development

lifecycle area. The system applies these settings to the QC and Production lifecycle areas as well.

> **Note:** There can be only one InForm clinical data model in a study, and it must be configured from a metadata load from the InForm reporting database. Do not make any manual structural changes to tables or columns in the InForm model. It must reflect the structure in InForm. If you make changes to the structure in InForm, reload the metadata. See "Modifying an InForm Input Model" on page 3-29 for details.
>
> InForm models are not included in study templates.

1. Navigate to the study and model and then the InForm Configuration tab.
2. **Remote Location**: Select or add the InForm reporting database; see "Adding or Editing a Remote Location" on page 3-18. You can also correct existing remote locations as necessary by selecting them and clicking the Edit icon.
3. **Remote Study Account Name**: Enter the name of the database account that owns the study tables and views in the InForm reporting database to be loaded into Oracle DMW.
4. **InForm LifeCycle**: Select the InForm lifecycle stage—Development, Quality Control, or Production—from which you want to load data into the current DMW lifecycle area. You can load data from any of these lifecycle stages into an Oracle DMW Development or QC lifecycle area. You can load only Production data into a Production lifecycle area.

   You select the Oracle DMW lifecycle area context when you select a study in the Home page.
5. **Webservice Location**: Select the appropriate web service location or add it; see "Adding or Editing a Web Service Location" on page 3-18. This is required to send Oracle DMW-originated discrepancies to InForm as InForm queries.

   You can correct an existing one if needed by selecting it and clicking the Edit icon.
6. **Schedule Production Data Load**: Select check box to schedule data loads at regular intervals in a production environment. This option is available only when the current lifecycle context is Production and you must have data load privileges to set the schedule; see "Scheduling Data Loads in the Production Lifecycle Area" on page 3-20.

   In all lifecycle areas you can load data manually as needed by clicking **Load Data** if you have the required privileges; see "Loading Data" on page 3-19.
7. **InForm URL**: Enter the URL for the study's InForm website; for example:

   http://*your_InForm.your_company*.com/*your_trial*/pfts.dll

   After you enter a value, an arrow icon appears. Click the icon to test the URL you entered. If the InForm login screen opens, you have succeeded. (The new window may open behind the current one.)
8. Save.
9. Click **Load Metadata**. The system reads table structures for the study in InForm and recreates them in Oracle DMW. This button is displayed only if you have the privileges required to use it. To check the progress of the job, click **Refresh**.

In the Development lifecycle area, loading metadata automatically installs the area and no additional installation is required. In the QC and Production lifecycle areas you must click **Install Model** to create the actual database tables based on the InForm metadata, which is required before loading data; see "Loading and Synchronizing InForm Metadata" on page 3-20. You can then load data; see "Loading Data" on page 3-19.

Adding or Editing a Remote Location

You can add or edit a remote location only if data loading is not enabled.

Provide information about the InForm reporting database for the study to create the database link:

1. Enter a value for:
   - **Name**: Enter a name for the InForm database. Do not use spaces, slashes, or special characters other than underscore (_).
   - **ConnectString**: Enter the text for the Using clause of the Create Database Link SQL statement. This is normally the same as the Description clause of a TNSNAMES definition; for example:

     ```
     (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=host_name.company_
     domain.com)(PORT=1521))(CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=trial_
     name.world)))
     ```

     Note: No spaces are allowed.

   - **Username**: Enter the name of the Oracle DMW read-only access account that has been set up in the InForm database for this remote location. See the section "Create Oracle Accounts for the DMW InForm Connector" in the *Oracle Health Sciences Data Management Workbench Installation Guide* for further details. The system will use this account to connect.
   - **Password**: Enter the password for the same user account. The system changes and encrypts the password after creating the database link.

     In the unlikely event that you need to change the password, your InForm database administrator must change the password for the Oracle DMW read-only access account. You can then select **Enable Password Entry?** in the Edit Remote Location window and enter the new password. The system then recreates the database link and changes and encrypts the password.
2. Click **Test Connection** to make sure it is set up correctly. The system returns a success or failure message above the Test Connection button.
3. Click **OK**.

Adding or Editing a Web Service Location

You can add or edit a web service location only if data loading is not enabled.

1. Enter a value for:
   - **Name**: Enter a name for the web service location.
   - **WSDL URL**: Enter the web service URL for the InForm Adapter's Enhanced Discrepancy Interface; for example:

```
http://your_InForm_Adapter_Server_Name/InFormAdapter/Discrepancy/
DiscrepancyService.svc?wsdl
```

- **Trial Name**: Enter the study name as defined when the trial was registered in the InForm Adapter; see the *Oracle Health Sciences InForm Adapter Installation Guide* for details.
- **Username**: Enter the name of the account set up for authenticating Oracle DMW web service transactions. The default name is DMW_AUTH. See the section "Create Users in InForm" in the *Oracle Health Sciences Data Management Workbench Installation Guide*.

  Enter the name of a valid user account on the web service location. The system will use this user name to connect.

  Note: The Username and Password fields are not displayed if your company chose to install Oracle DMW supporting HTTP, not HTTPS.

- **Password**: Enter the password required for the same user account. The system encrypts the password.

2. Click **Test Connection** to make sure it is set up correctly. The system returns a success or failure message above the Test Connection button.
3. Click **OK**.

Loading Data

In an InForm model's InForm Configuration tab, click **Load Data** to immediately load the latest data from InForm.

The system loads data into the lifecycle area currently in context—the one you selected on the Home page, now displayed at the top of the page.

The InForm lifecycle stage of the data being loaded depends on the setting of the Remote Study Account field, whose lifecycle stage is indicated in the InForm Lifecycle field.

Data Loading Rules You can load data at any InForm lifecycle stage into a model's Development or QC lifecycle area. You can load only InForm Production data into a model's Production lifecycle area.

To load data into either the QC or Production lifecycle area:

- The validation status of the model must be equal to or greater than the model's lifecycle stage; that is, a QC lifecycle model must have a validation status of at least QC, and a Production lifecycle model must have a validation status of Production.
- The most current version of the model must be installed.
- Data loading must be enabled, not suspended.

Suspending and Resuming Data Loading

If you need to stop data loading temporarily, click **Suspend**.

The system replaces the **Suspend** button with the **Resume** button. When the issue is resolved, click **Resume**.

Scheduling Data Loads in the Production Lifecycle Area

If the lifecycle context is Production and you have Production data loading privileges, you can schedule data loads:

1. Install the model if it has not already been installed.
2. If the status currently allows data loading—if the **Suspend** button is active— click **Suspend**.
3. Select **Schedule Production Data Load**.
4. Specify a **Fetch Frequency**: the number of minutes between automatic data loads.

> **Note:** A data load will not start until the previous one completes.

5. Save.
6. Click **Resume**. The first scheduled data load starts immediately.

Loading and Synchronizing InForm Metadata

If you have the required privileges, you can click the **Load Metadata** button to create or update tables in the InForm clinical data model. In the Development lifecycle area, loading metadata automatically installs the area and no additional installation is required. In the QC and Production lifecycle areas you must click **Install Model** to create or update the actual database tables based on the InForm metadata, which is required before loading data.

If InForm data structures are changed during the course of the study—for example, columns added or increased in length—the system detects these changes during the next data load and, if the user associated with the data load has the appropriate privileges, automatically reloads the metadata and synchronizes the changes.

If the user doesn't have privileges to load metadata:

- The data load fails.
- The log file states that metadata must be reloaded.
- The message "Metadata needs to be reloaded" is displayed in the InForm Configuration tab for the clinical data model.

A user with the required privileges must then click the **Load Metadata** button to reload and synchronize metadata.

The required privileges for InForm configuration, including metadata loading, are part of the roles DMW_STUDY_INFORM_CONFIG and DMW_STUDY_ADMIN .

Data loading, suspending, and resuming privileges are included in the roles DMW_STUDY_DEVELOPER, DMW_STUDY_QC, and DMW_STUDY_PROD for each lifecycle area, respectively, as well as DMW_STUDY_ADMIN.

Configuring File Watcher

This section contains the following topics:

- "Suspending and Resuming File Loading" on page 3-24
- "Viewing Detected Files and Forcing Actions" on page 3-25

For input clinical data models that load data from files, such as lab models, configure the File Watcher feature to look for data files that match your specifications in a location created for this purpose at intervals you specify. When the File Watcher service finds a file that matches your specifications in the relevant location it puts it in the queue for the Distributed Processing (DP) Server to load data from the file into the database tables of the input model.

You can load data in SAS CPORT or XPORT files or .zip files with multiple SAS datasets or text files; one dataset or text file per clinical data model table. Text data files can be .txt or .csv format, and .txt files can be delimited or use fixed column lengths.

The data in the file and the tables and columns in the clinical data model must have the same structure or the data load fails. You can create tables and columns initially by uploading the same data files you want to load, but if structural modifications are required afterward you must make them manually in the user interface and reinstall the model.

Each data load can automatically trigger downstream transformations and validation checks so that the data updates are automatically processed and propagated to your downstream models. To set this up, check the Can Trigger attribute for this model when you add it as a source to a transformation; see "Creating Model Mappings" on page 5-2.

By default, each study has six File Watchers, one for each combination of the two file types—SAS and text— and three lifecycle stages—Development, Quality Control, and Production. Each of the six study File Watchers watches a single folder to detect files to load. You specify one or more file name patterns to watch for for each file-type input clinical data model.

You can configure File Watcher for a clinical data model either before or after installing the model.

You can see information about data files that have been detected and submitted for loading on the Detected Files tab; see "Viewing Detected Files and Forcing Actions" on page 3-25.

See "Setting Up File Watcher for the Instance" on page 8-4.

Configuring File Watcher for a Model

To configure the study File Watcher for the clinical data model:

1. In the Study Configuration page, navigate to the clinical data model's Watcher Configuration tab. The page displays the type of file being loaded (specified during clinical data model definition) and the three study File Watchers for that file type.
2. Enter data load parameter values. These differ for each file type:
   - "Setting SAS Data Load Parameters" on page 3-22
   - "Setting Text Data Load Parameters" on page 3-22
3. In the **Data Source** field, select the name of the original source of the data being loaded, such as Lab XYZ.

   Your administrator creates a list of data sources; see "Viewing and Setting Up Lab Data Sources" on page 8-1.

4. Save. You must also create one or more file specifications; see "Creating File Specifications" on page 3-22

Setting SAS Data Load Parameters

For SAS files there are two data load parameters:

- **SAS File Type**: The allowed values are CPORT, XPORT, and SAS Dataset.
- **Max Errors**: The number of errors allowed per dataset before the load fails.

Setting Text Data Load Parameters

For Text files you must specify either:

- **Fixed**: If you specify **Fixed** format, the system uses the target table column length to determine the length of each data value. The file must contain the correct number of characters for each value in each record, in column order.
- **Delimited**: The system uses a delimiter character you specify to determine when one column value ends and the next one begins. You must also specify:
  - **Delimiter Character**: For delimited files, enter the character to be inserted between column values in each row; for example, a comma (,) or a pipe (|).
  - **Enclosing Character**: If any data value may contain the delimiter character, you need to surround each data point with another character. Specify the character used to enclose each value. The default character is double quotation marks (").

Note: With either Fixed or Delimited format the files can have an extension of either .txt or .csv, and you can upload multiple text files at the same time by including them in a .zip file. The system extracts and processes the files.

For either type, specify:

- **Skip Records**: If you want to prevent loading records at the beginning of the file, enter the number of records you want the system to skip. The default value is zero (0).
- **Max Errors**: Tolerance factor; the maximum number of invalid rows you are willing to tolerate before the load process with an error. The default value is 0. See "Format Checks on Loaded Files" on page 9-6 for more information.
- **Rows Before Commit**: Enter the number of rows you want the system to process before committing processed rows to the database.
- **Date Format**: Enter the exact date format used in the data file, if any. Do not enter a value here if the data file does not contain a date field. When you start to type a value, the system displays a list of values from which you can select.

Creating File Specifications

File specifications determine which of the files that are detected by the Watcher will actually be loaded into the clinical data model lifecycle area. All of the study's files for a particular lifecycle stage are in the same watched folder, so you must use different file naming conventions for each clinical data model.

File specifications contain a file name pattern to watch for and a schedule for loading detected files that match that file name pattern.

When you create or update a File Specification, log in as a user with the security privileges required to load data into the corresponding lifecycle tables. The system uses the account that defined or most recently updated the File Specification to run the data load job.

Under Watcher Policies, click the Add icon (+) and enter values:

1. Enter a name and description for the File Specification.
2. **File Name**: Enter a file name or regular expression for files to be loaded into the selected lifecycle stage for the clinical data model. All of the study's files for a particular lifecycle stage are in the same watched folder, so you must be sure that file names are unique across all models in the study. For example, you can use a naming convention includes the model name, but using the model name is not required.

   For example, Study A with file input models DMX, DMY, and DMZ, where labs are expected to include a date just before the extension:

   - StudyA_DMX_.*.zip
   - StudyA_specialLabs_.*.cport
   - StudyA_DMZ_.*.zip

   Note: File name pattern regular expressions use the POSIX standard Extended Regular Expression syntax found here: http://docs.oracle.com/cd/B28359_01/server.111/b28286/ap_posix001.htm.

   An asterisk (*) in POSIX Extended Regular Expression syntax looks for any preceding character. The dot (.) before the asterisk (*) in the examples above means "any character or no characters."

   Note: Any File Watcher Policy created in Oracle DMW 2.3 will continue to work in 2.3.1, now called a File Specification. The 2.3.1 File Name field contains the value defined as the Child Path in 2.3, which could include folders. In 2.3.1 the system continues to watch for the folder and file pattern specified as the Child Path, even though the new File Name field cannot include a folder. If you try to change a File Name value that includes a folder, you get an error.

   ---

3. **Execution Priority**: Enter the priority for loading data corresponding to this File Specification relative to others: Low, Normal, or High. The system uses this value to determine when to run these load jobs.
4. **Submission Mode**: Select one:

   - **Incremental**: The system loads all data in the file, inserting new records, updating records with changes, and refreshing the timestamps of records that are reloaded without change. It does not delete any records.
   - **Full**: The system inserts, updates, and refreshes reloaded records as in incremental processing and in addition, compares the unique keys of records

in the file to existing records and **deletes any records that are not included in the file**.

- **UOW Load**: For each UOW (Unit of Work—either subject or subject visit) that has any new or changed records, the system processes all records, inserting new records, updating records with changes, and refreshing the timestamps of records that are reloaded without change. The system does not process records for units of work—subjects or subject visits—that have no new or changed records. The system **deletes any existing records that are not reloaded within processed units of work**. It does not delete records that are not present in the source if no other records from the same unit of work are reloaded.

  Tip: You may want to create two File Specifications, one for a frequent Incremental or UOW Load and another for a less frequent Full load. Incremental loads are faster and can be run on a subset of data. Full loads are more time-consuming but they detect when to delete data. Be careful to always load the complete set of current data when you use Full processing; see Chapter 9, "Data Processing".

  Note: All deletions are "soft" deletions: records have an end timestamp equal to the load's date and time and *are no longer available in the system*. However, they still exist in the database and have an audit trail.

5. **Dataload Type**: Select Immediate or Scheduled.
   - **Immediate**: The Watcher searches continuously and queues the job as soon as it finds a file. This option loads data as soon as possible, continuously.
   - **Scheduled**: The Watcher searches at the interval you specify and queues the job as soon as it finds a file.
6. **Frequency**: If you selected Scheduled, select a frequency for the Watcher Policy to look for a new data file in the specified location in days, hours, or minutes.
7. **Start Datetime**: If you selected Scheduled, enter the date and time when you want scheduled file watching to begin.
8. **End Datetime**: Enter the date and time when you want file watching to end. You can enter a date far in the future and change it at any time.
9. Click **OK** to save.

Suspending and Resuming File Loading

You can stop the system from watching for a particular filename pattern by selecting the file specification and clicking the appropriate icon:

- If the file specification is currenting running, select it and click the Suspend icon (| |).
- If the file specification is currenting suspended, select it and click the Resume icon (>).

Migrating a Release 2.3 File Watcher to 2.3.1

If you configured File Watcher for a model in Oracle DMW 2.3, you can upgrade it to the 2.3.1 structure, which is simpler and more secure, by clicking the **Migrate** button.

Migrated Watchers look for files in the folders defined for the Study File Watchers; see "Setting Up Study File Watchers" on page 8-6. Existing File Specifications will continue to work but you may wish to delete these File Specifications and create new ones to coordinate with the 2.3.1 Study File Watchers.

Viewing Detected Files and Forcing Actions

In the Detected Files tab you can see a list of all detected files and their status.

You can also load or reload and delete files from here, overriding any scheduled load or deletion, by selecting the file and clicking **Data Load** or **Delete**. This may be useful if a load failed and needs to be retried, or if a file was faulty.

You can see information about files that have been detected and submitted. Once a data load job has been started, you can see the job status in the Data Loads tab on the Home page; see "Viewing Data Load Information" on page 10-2. To see the data, go to the Listings page.

For each detected file the system displays:

- **File Name**
- **File Spec Name**: The name of the File Specification
- **Status**: The possible statuses are:
  - **DETECTED**: The file has been detected in the watched folder but has not yet been submitted. (The scheduled submission time is in the Data Load Date column.)
  - **SUBMITTED**: The file has been submitted. This status does not change when the load is completed. However, you can go to the Data Load tab on the Home page to see if the load was successful.
  - **MISSING**: The file was detected but deleted before or after it was submitted, but before the scheduled deletion date.
  - **DELETED**: The file was deleted by File Watcher as scheduled.
- **File Modified**
- **Data Load Date**
- **Deletion Date**
- **Detection Date**
- **Error**
- **Date Missing**

If you have the required privileges, you can:

Load Data You can select a file and click **Load Data** to immediately load the file; for example, if the original load failed. The system will not load the file on the scheduled date, if any, after a manual load.

Delete You can also delete any of the files listed at any time.

See "Viewing Data Load Information" on page 10-2 for job statuses. To see the data, go to the Listings page.

Updating Validation Status

Each study has three lifecycle areas: Development, Quality Control, and Production. Each clinical data model has a validation status of either Development, Quality Control, Production, or Retired. When you install a model, the system creates a database schema for it in the lifecycle area corresponding to its validation status. The schema also contains the installed packages for the transformation programs that write to tables in the model and the validation checks and saved custom listings that run on data in the model.

When you first create a clinical data model, the system gives the new model a validation status of Development.

When you have finished working on the model, upgrade its validation status to QA. The model then appears in the list of models in the QC lifecycle. A user with QC privileges can install the model in the QC area, creating a schema for the model and database tables in it for all the metadata tables. The QC user then upgrades the model's validation status to Production. A user with Production privileges can then install the model in the Production lifecycle area, creating a schema for it there.

When you check a model out to modify it, the system creates a new version of it with a validation status of Development. The installed version of the object in the QC and Production areas continue to function as before.

The system installs models in Upgrade mode, which can handle nondestructive changes to table structure like adding a column. However, if table metadata is changed in a destructive way, such as removing a column, the system displays a warning that installing the model would result in a loss of data. If you choose to continue anyway, the system drops and replaces the table and its data is deleted.

You must handle transformations and validation checks the same way—upgrade their validation status and then install them in the next higher lifecycle area.

You select the lifecycle area you want to work in at the bottom of the Home page in the Lifecycle Mode section. You may have privileges to work in only one.

To change the validation status of a clinical data model, table, transformation, validation check, or saved custom listing:

1. Select the object and click **Update Validation Status**. The object must be checked in. The validation status applies to the current checked in version.

   The Validation Status Update window opens, displaying the current validation status.

2. From the Validation Status drop-down list, select the new status:

   - **Development**: This is the default status for all new objects and is intended for objects being worked on or tested.
   - **Quality Control**: This is intended as the status for objects that have passed initial testing and are undergoing formal testing. Objects must have a status of Installable to be promoted to QC from Development. Objects with this status are eligible to be in the Development or Quality Control Lifecycle area.
   - **Production**: This is intended as the status for objects that have passed formal testing and are suitable for use in an active or frozen study. Objects with this status are eligible to be in any lifecycle area, including Production.
   - **Retired**: Objects with this status are no longer available for use. If you set a library model to Retired, users can no longer create study models based on the library model. You cannot set an installed study model to Retired.

3. Click **OK** or, if your company's standards require supporting documentation such as test results or a requirements document to justify the change in validation status, click the **+** icon in the Supporting Documents pane. The Add Supporting Document window opens.
4. Enter a name and description for the document and browse for the document on your computer, then click **OK**. The system uploads the document.

Applying Security

To view data, a user must belong to a user group with access to the table containing the data. To create and modify studies, clinical data models, tables, validation checks, and transformations—all of which are metadata *objects*—a user must belong to a user group with access to the object. The user must also have a role in the user group with the privileges required to view data or create or modify the object.

Objects are contained within other objects: therapeutic areas (or another organizational category) contain studies, which contain clinical data models, each with a Development, Quality Control, and Production lifecycle area, which contain tables, transformations, and validation checks installed as database tables and packages. Studies also contain the metadata for models, transformations and validation checks; see "Object Ownership" on page A-6.

When you assign a user group to an object the user group assignment is automatically inherited by all the objects contained in it. For example, when a user group is assigned to a study, the user group has access to all objects in a study. Or, when a user group is assigned to a clinical data model, it has access to all objects in the model, including the Development, Quality Control, and Production lifecycle areas. If you want a user group to have access to only one lifecycle area, you must either assign it explicitly to the one it should have access to, or assign it at a higher level and revoke its access to the areas it should not have access to.

For information about setting up user groups, roles, and user accounts, see the *Oracle Life Sciences Data Hub System Administrator's Guide*.

Assigning User Groups to Objects

To change user group assignments to an object:

1. Select the object and click **Apply Security**. The Apply Security window opens with the name of the object displayed and a list of all user groups currently associated with the object in any way. The Assignment Status column shows the current state of each user group's access to the object:
   - **Inherited**: The user group has access to the object because it was explicitly assigned to an object that directly or indirectly contains the current object.
   - **Assigned**: The user group has access to the object because it was explicitly assigned to the object.
   - **Revoked**: The user group does not have access to the object. Its inherited access has been revoked from the object.

   User groups that do not have access to the object are not displayed.
2. In the **Assign To** field, select one of the following:
   - **Metadata**: This choice assigns the user group to the definitional metadata for the object. Access to object metadata is required to modify studies, clinical data models, tables, validation checks, and transformations.

Note: To create models, transformations, and validation checks, the user needs to belong to a user group assigned to the study metadata, and have Create privileges on the object type to be created: model, transformation, or validation check. See the *Oracle Life Sciences Data Hub Implementation Guide* for more information.

- **Development**: This choice assigns the user group to the installed object in the Development lifecycle area database schema. This access is required for creating, modifying, or executing the object in the Development lifecycle area and loading or viewing data in the Development lifecycle area.
- **QC**: This choice assigns the user group to the installed object in the Quality Control (QC) lifecycle area database schema. This access is required for testing or executing the object in the QC lifecycle area and loading or viewing data in the QC lifecycle area.
- **Production**: This choice assigns the user group to the installed object in the Production lifecycle area database schema. This access is required for executing the object in the Production lifecycle area and loading or viewing data in the Production lifecycle area.

3. Make a user group assignment change:
   - To assign a user group that is not currently associated with the object, click **Assign**. The Assign User Group window appears. Query for the user group if necessary, then select it and click **Apply**.
   - To revoke the access of a user group that has an Inherited status, select the group and click **Revoke**.
   - To reinstate the access of a group whose status is Revoked, select the group and click **Unrevoke**.
   - To remove the access of a group whose status is Assigned, select the group and click **Unassign**.

Modifying Clinical Data Models

This section contains the following topics:

See also "Adding and Modifying Tables Manually" on page 3-7

Modifying a Non-InForm Model Manually

In target models and file-type input models you can create tables and columns initially by uploading SAS files, but if any modifications are required afterward you must make them manually in the user interface and reinstall the model.

Note: DO NOT add or remove tables or columns or make any other structural changes to an InForm clinical data model. The InForm model must have exactly the same table and column structures as in InForm. If you change these structures in InForm, use the Load Metadata job in the InForm Configuration tab to synchronize the changes here. See "Modifying an InForm Input Model" on page 3-29.

However, nonstructural changes are allowed; see "Manual Changes Allowed in InForm Models" on page 3-30.

To view or modify a clinical data model:

1. Navigate to the model in the Study Configuration or Library page.
2. To modify the model, click **Check Out**. The system creates a new version of the model for you to modify. This button is visible only if you have the privileges required to modify the model.
3. Make changes. If you have the required privileges, you can:
   - **Add, modify, or delete tables** by clicking on the appropriate icon and making the changes; see "Adding Tables to a Clinical Data Model" on page 3-6.
   - **Add, modify, or delete columns** in a table by selecting the table in the upper pane and clicking the appropriate icon in the Columns tab and making the changes; see "Adding Columns to a Table" on page 3-9.
   - **Add, modify, or delete constraints** in a table by selecting the table in the upper pane and clicking the appropriate icon in the Constraints tab and making the changes; see "Adding Constraints to a Table" on page 3-12.
   - **Update to Current Library Version** If the study model was created from a library model and the library model has been updated, the **Upgrade to Latest Version** button appears. Click it to update your study model to the new library version.

     Note: Any changes that have been made to the study model will be lost.

   - **Update Validation Status** Click this button to change the validation status (possible values are Development, Quality Control, or Production) or to upload a supporting document for the validation status change; see "Updating Validation Status" on page 3-26. The system displays this button only if you have the privileges required.
4. Click **Install Model**. You must install the model after you make changes; otherwise the system continues to use the old version. The model must have a status of Installable.

   A model is not installable if it does not have any tables or if any of its tables are not installable. A table is not installable if it has no columns.

Modifying an InForm Input Model

The InForm model must have exactly the same table structures as in InForm. If you change these structures in InForm, use the Load Metadata job in the InForm Configuration tab to synchronize the changes automatically.

Manual Changes Not Allowed in InForm Models

DO NOT make any manual structural changes in an InForm clinical data model, including:

- Adding or dropping tables or columns
- Modifying table or column names
- Modifiying column data type, length, precision or nullable status
- Adding or removing constraints or unique indexes

Automatic Synchronization If data structures are changed *in InForm* during the course of the study, the system detects these changes during the next data load and, if the user associated with the data load has the required privileges, automatically reloads the metadata and synchronizes the changes.

If the user doesn't have privileges, the dataload fails with a message in the log file that metadata needs to be reloaded. A user with the required privileges can then click the **Load Metadata** button to reload and synchronize metadata.

Manual Changes Allowed in InForm Models

You can modify table and column attributes, including:

- Blinding-related attributes
- SAS-related attributes
- Column and table aliases

To check out an InForm model:

1. Go to the InForm Configuration tab.
2. Click the **Suspend** button to suspend data loading. The **Check Out** button appears.
3. Check out the model.
4. Make your changes; see:
   - "Adding Constraints to a Table" on page 3-12
   - "Defining a Primary Key" on page 3-8
   - "Setting Blinding-Related Attributes" on page 3-8
   - "Setting Data Processing-Related Attributes" on page 3-8
   - "Adding and Modifying Tables Manually" on page 3-7 for information on adding tables aliases and SDTM identifiers—but do not add or remove tables or modify other things
   - "Adding Columns to a Table" on page 3-9 for information on adding column aliases, code lists, and SDTM identifiers—but do not add or remove columns or do other things
   - "Updating Validation Status" on page 3-26
5. Click **Install Model**. You must install the model after you make changes; otherwise the system continues to use the old version. The model must have a status of Installable.

   A model is not installable if it does not have any tables or if any of its tables are not installable. A table is not installable if it has no columns.

Removing a Clinical Data Model

To delete a clinical data model, select it in the Clinical Data Model pane and click the red **X** Remove icon. You must have the Remove Model privilege to do this.

Upgrading a Clinical Data Model to the Latest Library Version

If a clinical data model was created from a library model, and a new version of the library model exists, the Upgrade to Latest Version button appears. Click the button to synchronize the study model with the library model.

Note: Any custom changes you have made to the study model are lost.

Viewing Data

After you have installed a study clinical data model and loaded data into its tables, you can view the data in the Listings page; see "Viewing All Study Data Using Default Listings" on page 11-2.

4

Using Libraries

This section contains the following topics:

See also "Setting Up Library and Study Categories" on page 8-15.

About Library Clinical Data Models

Youcreate a library clinical data model directly in the library.

Libraries correspond to the categories—for example, therapeutic areas—that your company creates for the purpose of grouping studies and library data models. If you promote a study data model to the library, it is promoted to the library for the study's category. If you want to create standard data models that are appropriate for many study categories, you may want to create a category and study especially for that purpose and call them something like Generic or Standard. When you create a study data model from a library data model, you can select a data model from any library; see "Setting Up Library and Study Categories" on page 8-15.

The system tracks all study clinical data models based on a library model. If the library model is modified, the system detects the study models based on previous library model versions and allows users to upgrade the study models or choose not to upgrade them.

When you create a study model based on a library model, you can modify the study copy as necessary, adding and deleting tables and columns. However, if you update your study copy to the latest library version you lose your changes.

Creating and Modifying Library Clinical Data Models

Clinical data models in a library are available for reuse in studies. When you modify a library model, any study models created from it can be updated to the latest version.

Creating a Library Clinical Data Model

To create a library clinical data model:

1. Click **Library**.
2. Click the Add icon (+) for Clinical Data Models. The Create Clinical Data Model window appears.

3. Enter a name and description for the model; see "Naming Objects" on page A-1 for restrictions.
4. Select the library in which the data model belongs. The label of this field as well as the options are configurable by your company; see "Setting Up Library and Study Categories" on page 8-15.
5. Under **Select from source** select the source, if any:
   - **None** to define tables and column manually.
   - **Load from file** to create tables by uploading files. The system can create tables from zipped SAS datasets, SAS Transport (CPort or XPort) files, or a .zip file that contains one or more text metadata (.mdd) files. Metadata files are the only way to automatically create tables with constraints or with all blinding attributes set; see "Required Syntax for Table Metadata Text Files" on page A-3.

     Browse to the file.
6. Click **OK**.

> **Notes:** If you have not already created a primary key for every table in the model, you must do so; see "Adding Constraints to a Table" on page 3-12.

7. Check the status. Make sure it has a status of Installable before you check it in. A model is not installable if it does not have any tables or if any of its tables are not installable. A table is not installable if it has no columns.

Modifying Library Clinical Data Models

To modify a library model:

1. Go to the Library page and select a study category.
2. In the search field in the upper left, enter part or all of the model name and press Enter. The system lists all models whose name contains the string you typed.
3. Select the model you want, either by clicking on it or by using the down arrow and then pressing Enter.
4. To modify the model, click **Check Out**. The system creates a new version of the model for you to modify. This button is visible only if you have the privileges required to modify the model.
5. Make changes. If you have the required privileges, you can:
   - **Add, modify, or delete tables** by clicking on the appropriate icon and making the changes; see "Adding Tables to a Clinical Data Model" on page 3-6.
   - **Add, modify, or delete columns** in a table by selecting the table in the upper pane and clicking the appropriate icon in the Columns tab and making the changes; see "Adding Columns to a Table" on page 3-9.
   - **Add, modify, or delete constraints** in a table by selecting the table in the upper pane and clicking the appropriate icon in the Constraints tab and making the changes; see "Adding Constraints to a Table" on page 3-12.
   - **Update Validation Status** The system displays this button only if you have the privileges required. Click this button to change the validation status (possible values are Development, Quality Control, or Production) or to upload a

supporting document for the validation status change; see "Updating Validation Status" on page 3-26.

6. Check the status. Make sure it is installable before you check it in. A model is not installable if any of its tables are not installable. A table is not installable if it has no columns.

Removing a Clinical Data Model

To delete a clinical data model, select it in the Clinical Data Model pane and click the red **X** Remove icon. You must have the Remove Model privilege to do this.

Creating Code Lists

Code lists specify a list of valid values that you can associate with a table column.

1. In the Library page, select the Code Lists pane, then click the **+** icon.
2. **Codelist Name**: Enter a name for the code list that will help users know what it is.
3. **Description**: (Optional)
4. Select a therapeutic area or other library category. The code list will be available for use in study and library clinical data models in the same library.
5. Click **OK**.

Add Code Values Select the code list. In the Code List Values tab, click the **+** icon and enter:

1. **Code**: The code must be unique within the code list. The code is the allowed clinical data value; the string that the system tests for when columns are associated with the code list.
2. **Code Value**: You can use this as a more descriptive equivalent of the code. The system does not evaluate this value.
3. Click **OK**.
4. Repeat for each value.
5. Click **Check In** to make the code list available for use.

Code List Examples

Common code lists include:

- Code list YESNO with codes Y and N and code values Yes and No.
- Code list SEX with codes M and F and code values Male and Female.

Removing a Code List

To remove a code list, navigate to the Library, then Code Lists. Select the code list and click the Remove (red X) icon.

Note: You cannot remove a code list if it is associated with a table column.

5

Transforming Data to Standard Structures

This section contains the following topics:

About Transformations

To move data from one clinical data model to another—for example, merging data in InForm and lab input models into a single Review model, or transforming a Review model to an SDTM Analysis model—you create a transformation.

The system provides a user interface that supports complex data transformations, but for some types of transformations you need to write a custom program.

- At the table level, the user interface supports:
  - joins, self-joins, unions, pivots, unpivots, or direct 1-to-1 mappings
  - creating staging tables to hold data at an intermediate stage in the transformation; for example, you can create a staging layer to normalize source data before transforming it to your standard review model
  - defining criteria, or source filters, for the data to be included in the Where clause and written to the target table
- At the column level, mappings are constrained by parent table mappings and can include:

- Structural mappings that implement the table-level relationship; for example, in a Many-to-1 table relationship, mapping one column from each source table to a single column in the target table.
- Functional mappings to support data derivations, including writing expressions on one or more columns for data manipulations such as unit conversions, string manipulations, or inserting a hardcoded value. You can also call SQL functions and lookup tables from a library you develop in Oracle Life Sciences Data Hub (Oracle LSH). These functions can take column values as input parameter values.

Upon request, the system automatically maps as many tables and columns as possible within mapped source and target models. You can review the suggested mappings and accept as many as appropriate, then manually map the rest. The system uses these mappings to generate one PL/SQL program for the transformation to each target table.

You can set up transformations to run on a schedule and to automatically trigger all validation checks in the target model and any downstream transformations and validation checks for other models in sequence, so that all models and users always have the most current data possible.

If a table mapping or any column mapping within a table mapping requires code that you cannot define in the user interface; for example, data aggregation or conditional logic that selects data from a source table to populate a target field, you can write a custom program in Oracle LSH for the table transformation and attach it to the table mapping in the user interface; see "Defining a Custom Program" on page 5-17.

Transformation programs are included in study templates, as are all clinical data models except input models of type InForm.

If you are using both source and target models as is, you can also use the mapping as is. If you make changes to the models, you can run upgrade mapping; see "Upgrading a Mapping to Reflect Model Metadata Changes" on page 5-17. You can run Automap to help map input models to template models.

You must create the source and target data models and tables before you can create the transformation between them.

Creating Mappings

To create transformation mappings, do the following tasks in order:

1. Creating Model Mappings
2. Creating Table Mappings
3. Creating Column Mappings

Creating Model Mappings

You must first map at the highest level: identifying the target clinical data model and one or more source models. For example, map all your input models from InForm and from each lab to a standard review model for studies of a particular type such as cardiology or immunology.

1. Go to **Study Configuration**, then **Transformations** at the bottom of the right-hand pane. The system displays a list of all the study's clinical data models defined as targets—that is, data models that are populated by data from within the system by a transformation, not from an external data source such as InForm or a lab.

2. Select the clinical data model to be the target of this transformation.

   The system displays its tables in the Target Tables pane and in the Source Tables pane in case you want to populate some target tables with data already transferred or transformed into other target tables, or if you will use staging tables to populate the target table.

3. Click **Source Model** in the Source Tables pane. The Add Model window opens., listing all other models in the study and indicating if each one is an Input or Target-type model.

   a. Select one or more models to feed data into the transformation's target model.

   b. For each, select the **Can Trigger** check box if you want the completion of a job updating data in the source model to trigger the execution of this transformation.

   c. Click **OK**.

   The system lists the models in the Model column and all tables in the selected model(s) in the Table column.

Creating Table Mappings

After you have mapped one or more source models to the target model, you must specify how to handle every table in the target model: either mark it as Not Used or map it to one or more source tables and specify the mapping details.

This section contains the following topics:

Marking Target Tables as Not Used

If you do not need a particular table in a target model that you have copied from a library or study template, you can mark it as Not Used. The system then does not consider the table when computing the status of the transformation mapping; the status can be Complete without mapping the table.

To mark a target table as Not Used, select it and click **Mark as Not Used**.

Using Automap

When you click **Automap**, the system attempts to map all tables in the target data model and all the columns in the mapped tables. You must then check the generated mappings and select those you want to accept.

If you define mappings manually before running Automap, your mappings remain unchanged.

The Automap logic uses the following to generate mappings:

1. Matching metadata: table and column names, column data type and length
2. Table and column alias matches
3. Historical mappings in the same installation

Accepting and Rejecting Automatic Mappings After Automap runs, the Transformation Automap window opens, listing the target tables that have been automapped.

To see the column mappings for a table, click the table's node (**+**). The system displays the target table's columns and the source columns Automap has found to map them to. The Type column shows the type of logic used to find the source column: Name Match, Alias Match, Datatype Match, Historical Map, or Partial Name/Alias Match.

Review all table and column mappings, deselect any you do not want or are not sure about, and then click **Accept Selected Mappings**.

Mapping Tables Manually

To map a target table manually:

1. Select one or more source tables to map to a single target table, select the target table, and click the **Map** button in the target table's row.

   You can use Ctrl+Click or Shift+Click to select and map multiple source tables to the same target at the same time. You can also map a single source table to a target table and add another source table later.

2. **Program**: If the table mapping requires a custom program—for example, because one or more of its target column mappings requires a more complex expression than can be created in the user interface—select the program to use. See "Defining a Custom Program" on page 5-17.

3. Select a **Transformation Type**: Direct, Join, Union, Pivot, Unpivot, or Custom.

   > **Note:** In most cases, even if you are using a custom program, select the actual type of table transformation your program is performing: Direct, Join, Union, Pivot, or Unpivot. Use Custom only if your program cannot support data lineage tracing; see "Defining a Custom Program" on page 5-17.

4. If you specify a transformation type that requires further definition—Join, Pivot, or Unpivot—the system displays an icon after you select that type. Click the icon to supply details; see "Table Transformation Types" on page 5-7.

5. **Authorize**: This setting is modifiable only if one or more source tables is defined as blinded (its Blinding flag is set to Yes) at any level (whole table, column, row or cell) *and* you have the required blinding-related privileges. If one or more source tables contains blinded data, by default the target table is completely blinded and only users with special privileges can view the data. However, if you are certain the target table contains only nonblinded data and you have the required privileges, you can authorize that the target table's data not be blinded. This authorization is audited.

   > **Note:** If you have the required privileges, you can change the blinding attributes of the target table so that only the appropriate columns, rows, or cells are blinded. Do this in the Clinical Data Model page; see "Setting Blinding-Related Attributes" on page 3-8.

6. Click **Save**. You can save your work at any point but must save in order to map columns.

Adding Expressions on Mapped Sources

Select the target table. The system displays the source table or tables mapped to the target in the Mapped Sources tab, with their:

- **InForm Ref Path Name** if the source table originated in InForm; this fully identifies a field on a CRF.
- **Logical Names** are aliases defined in the clinical data model definition for the table; these values are used in automapping.

You can:

- Add another **alias** for the table to be used in an expression, if needed; for example, if you need to self-join the table to itself, to differentiate the two usages of the table.

  To create a self-join, select the table in the Source Tables pane and the Target Tables pane and click Map. The system adds the source table to the Mapped Sources tab again. Enter an alias for one or both occurrences and save. Then select either occurrence and write an expression.

- Write an **expression** to operate on the source table(s)' data during the data transformation to the target table by clicking the Source Filters icon; this expression becomes part of the Where clause, restricting the set of source data that participates in the transformation; see "Using the Expression Builder" on page 5-12.

> **Note:** You can write expressions to define derivations at the column level in the Column Mapping page.

Creating Staging Tables

Complex transformation may require multiple steps with a layer of staging tables to hold data at intermediate steps. For example, to make transformations as reusable as possible, you may want to start by performing source-dependent conversions and derivations into a normalized structure in the staging layer. You then perform reusable source-independent conversions and derivations on the normalized structure to write to the final target table.

Since your source data is likely to be in different structures in different studies, you will need to do some manual remapping from the source to the staging tables when you reuse the transformation and its target model. However, the mappings from the normalized staging structure to the target tables are reusable if they are standard.

You select source tables and columns, and create expressions, joins, and criteria as required and the system generates a staging table and adds it to the target data model. You can use all the same table- and column-level mapping functionality to map to or from a staging table as from a source or target table.

You can map one or more staging tables to another, if two intermediate stages are required. You cannot self-join a staging table. The system maintains data lineage tracing through the staging layer.

To create a staging table:

1. Click **Create Staging**.
2. Enter a name and description for the table.
3. **Supports Duplicate**: If the table should support duplicate primary key values in a single job, select the check box. This should be used only rarely, but if a data

source such as a small lab with limited IT resources cannot ensure that there are no duplicate primary key values in a single data load, this setting allows all the data to be loaded. You may want to also use the setting for tables the original input model table writes to; see "Supporting Duplicate Primary Keys in a Load" on page 9-7.

4. **Create Target Mapping?**: If checked, the system creates the mappings from the source table(s) to the staging table when it creates the staging table.
5. **Model Name**: In the Source pane, select the source model from the drop-down list. The system displays the model's tables below.
6. **Table and Columns**: To add a whole table with all its columns as a source, select it and click the Add (+) icon. To add individual columns in the table, expand the table's node, select the column or columns and click the Add (+) icon.

   In addition to being used as sources, these columns become the columns of the staging table. Add comma-separated aliases if necessary.
7. For each column that should be part of the primary key of the staging table, select the check box in the Primary Key column.
8. If the source-to-staging transformation requires an expression at the column level, click the Edit icon (pencil) to use the Expression Builder. To write an expression using two or more column values as inputs, select one of them in the Selected Columns tab, then select the other(s) in the Sources pane and click the Arrow icon. The system adds them to the same row in the Selected Columns tab.
9. Define joins and Where clause criteria as in the Query Builder; see "Building the Validation Check Query" on page 6-3.
10. Save.

Creating Column Mappings

After you have mapped one or more source tables to a target table, you must specify how to handle every column in the target table.

If you do not need a column in the target model, select it and click **Mark as Not Used**. The system then does not consider the column when computing the status of the transformation mapping; the status can be Complete without mapping the column.

Column mappings are constrained by table mappings; columns can be mapped only if their tables are mapped, and they can be mapped only in ways that are consistent with the Transformation Type of their table mapping.

To map a target table's columns:

1. In the Target Tables pane, select the target table whose columns you want to map and click the icon in the **Map Column** column. The Map Columns window opens.
2. Click **Automap** (optional). The system automatically maps as many columns as it can. You can then add and change the mappings if necessary, using the following steps.
3. Select one or more source columns to map to a single target column, select the target column, and click the **Map** button in the target column's row.

   You can use Ctrl+Click or Shift+Click to select multiple source columns. You can also map a single source column to a target table and add another source column later.

4. **Expression** (Optional): If you need to write an expression operating on the source column data to populate the target column, select the target column in the Target Columns pane so that the source columns mapped to it appear in the Mapped Sources tab. Click the icon in the Source Filters column in the Mapped Sources tab. The Expression Builder opens in the Define Source Filters window; see "Using the Expression Builder" on page 5-12. This expression becomes part of the Where clause.
5. **Set Preferred Path**: If you create a Many-to-1 column mapping, you must select one of the source columns as the Preferred Path in the Mapped Sources tab. If a user creates a discrepancy against a data point in the target column, the system displays the discrepancy against the source value in the Preferred Path column and not the other source column(s). When the discrepancy is sent back to the data point's source system, it is associated with this data point.

   In a 1-to-1 mapping, the system automatically sets the Preferred flag of the single source column to Yes when you save the mapping.
6. Continue until all source and target tables and columns complete. The system displays the check icon for the target table when its mapping is complete, meaning that all columns are either mapped or marked as Not Used.
7. **Save**. You can save your work at any point but must Save when you have finished and all mappings are complete.
8. Install using the **Install Map** button. The system generates a PL/SQL transformation program using the mappings, checks it in, and generates the package in the database. It also tries to install the target model if it is not already installed.

   When you run the transformation program it copies data from the source tables to the target tables. To run the program, go to the Transformations tab in the Home page.

Undoing Mappings

To undo a table or column mapping:

1. Select the target table or column in the Target Table or Target Columns pane. The system displays the source tables or columns mapped to it in the Mapped Sources tab below.
2. In the Mapped Sources tab, select the source table or column whose mapping you want to undo.
3. Click the Remove icon (X) in the Mapped Sources tab.

Table Transformation Types

The system autogenerates relations between source and target tables based on the following operations (except Custom, which requires you to write your own transformation program):

- Direct
- Join
- Union
- Pivot
- Unpivot

- Custom

For more information on joins, unions, pivot, and unpivot, see *Oracle® Database SQL Language Reference 11g Release 2 (11.2)* at:
http://st-doc.us.oracle.com/11/112/server.112/e26088.pdf.

Direct

One or more source tables feed data to a single target table. This is the default value. Column maps in a direct table relation may have a 1-to-1 or Many-to-1 relation.

To create a Direct transformation:

1. Select the source and target tables, then select a Transformation Type of **Direct**.
2. Click the icon in the **Map Column** column. The system displays the source and target columns.
3. Select each target column in turn and do one of the following:
   - Select the source column(s) that will feed data to it and click **Map**.
   - Click **Mark as Not Used**.
4. Save.

Join

Two or more source tables feed data to a single target table in a join relationship with a join condition. Column maps in a join relation may have a 1-to-1 or Many-to-1 relation.

To define a Join transformation:

1. Select the source table(s) and target table and click **Map**, then select a Transformation Type of **Join**.
2. Click the Join icon that appears in the Transformation Type column. The Define Joins window opens.
3. Click the Add (+) icon in the Define Joins window. The system populates the Table drop-down lists with the mapped tables.
4. Select Table 1 (left) and Table 2 (right, also known as the foreign key table) to be joined.
5. Specify if it is to be an outer join on either the left or right side. Leave the checkbox unselected to create an inner join.

   An *inner join* (sometimes called a *simple join*) returns only those rows that satisfy the join condition.

   An *outer join* extends the result of a simple join. An outer join returns all rows that satisfy the join condition and also returns some or all of those rows from one table for which no rows from the other satisfy the join condition:

   - A *left outer join* returns returns all rows from Table 1 and only rows meeting the join condition from Table 2. Rows in Table 1 that do not have a corresponding row in Table 2 have null values in the columns from Table 2.
   - A *right outer join* returns returns all rows from Table 2 and only rows meeting the join condition from Table 1. Rows in Table 2 that do not have a corresponding row in Table 1 have null values in the columns from Table 1.

- A *full outer join* returns all rows from both or all tables. Rows in either table that do not have a corresponding row in the other table have null values in the columns from the other table.

6. If you are joining more than two tables, click the join Add (+) icon again and define the next join. Also follow the next steps for each pair of tables.
7. Click the Add (+) icon in the Join Details pane. The system populates the Table lists with the columns in each joined table.
8. Create the join condition for the Where clause by selecting a column from each table and the operator required. For example, where both tables have a column called USUBJID, select those columns and select an operator of *equals* (=).

   To specify additional Join conditions on other columns, click the Add (+) icon in the Join Details pane again as many times as required.
9. Click **OK**. The system displays the join in the Joins tab below.
10. Click the icon in the **Map Column** column. The system displays the source and target columns.
11. Select each target column in turn and do one of the following: a
    - Select one or more source columns and click **Map**.
    - Click **Mark as Not Used**.
12. Save.

Union

Two or more entire tables feed data to a single target table that is a superset of all columns in the sources. The assumption is that the source tables are logically related with at least some overlapping content, such as two versions of a Vital Signs form or two local lab vendors providing results.

Data Lineage Tracing In a union, the Preferred Path drop-down in the Mapped Sources pane is disabled. The system automatically defines the preferred path:

- If the column mapping is 1-to-1; where each row in the target is populated by a single row in one of the source tables, the system automatically selects the corresponding row in the appropriate source as the preferred path.
- If the mapping is Many-to-1, the system randomly selects one of the source columns as the preferred path. If you want to select the preferred path yourself, use a staging table to perform the union (creating a 1-to-1 column mapping) and then write the expression on the columns in the staging table to the end-target table.

To create a Union transformation:

1. Select the source and target tables and click **Map**, then select a Transformation Type of **Union**.
2. Click the icon in the **Map Column** column. The system displays the source and target columns.
3. Select each target column in turn and do one of the following: a
    - Select one source column and click **Map**. You can select multiple sources but this has implications for Data Lineage Tracing.
    - Click **Mark as Not Used**.

4. Save.

Pivot

A pivot converts a source table with a vertical (tall, skinny) structure, such as lab data or ODM, into a target table that represents the same data in a more horizontal (short, fat) structure—with more columns and fewer rows; see Pivot Example.

To create a Pivot transformation:

1. Select the source and target tables and click **Map**, then select a Transformation Type of **Pivot**.
2. Click the Pivot icon that appears in the Transformation Type column. The Pivot/Unpivot window appears.
3. Query for and select the pivot column from the source table columns. The column must be associated with a code list that specifies the allowed values in the column; for example, the column Test may have a code list of the lab test names collected in a particular visit.
4. Click **OK**.
5. Click the icon in the **Map Column** column. The system displays the source and target columns.
6. In the **View** drop-down, select **Columns** and then select **Filter Value** if it is not already checked.
7. Scroll over to the **Filter Value** column. For each pivoted column in the target table, select the pivot column value to identify the row in the source table from which to get the value for the target column. The list is populated by the code list you specified.

   For the nonpivoted columns you normally create a direct 1-to-1 map but Many-to-1 maps with expressions are also allowed.
8. Save.

Pivot Example Lab results are shipped one per row, but the review data model requires one row containing all lab results for each patient at the same visit.

Table 5–1 Source Table in Pivot Example (Tall, Skinny Table)

SubjID	Date	Visit	Test	Unit	Value
972	03262112	5	IG	mh/dl	853
972	03262112	5	Lith	null	neg
972	03262112	5	PTH	pg/mL	285
989	03312112	3	IG	mh/dl	824
989	03312112	3	Lith	null	pos
989	03312112	3	PTH	pg/mL	290

The Test column, which contains the lab test name in the source table, is the pivot column. It is associated with a code list whose values are IG, Lith, and PTH. The source columns Unit and Value are also pivoted. The columns SubjID, Date, and Visit are not pivoted.

Table 5–2 Target Table in Pivot Example (Short, Fat Table)

SubjID	Date	Visit	IG	IG_Unit	Lith	Lith_Unit	PTH	PTH_Unit
972	03262112	5	853	mh/dl	neg	null	285	pg/mL
989	03312112	3	824	mh/dl	pos	null	290	pg/mL

Unpivot

An unpivot converts a source table with a horizontal (short, fat) structure, into a target table that represents the same data in a more vertical (tall, skinny) structure—with more rows and fewer columns; for use with tables where multiple columns collect the same data, such as the same assessment repeated in each section of a CRF; see Unpivot Example.

To create an Unpivot transformation:

1. Select the source and target tables and click **Map**, then select a Transformation Type of **Unpivot**.
2. Click the Unpivot icon that appears in the Transformation Type column. The Pivot/Unpivot window appears.
3. Query for and select the pivot column from the target table columns. The column must be associated with a code list that specifies the allowed values in the column; for example, the column Section may have a code list with values 0hr, 1hr, and 2hrs.
4. Click **OK**.
5. Click the icon in the **Map Column** column. The system displays the source and target columns.
6. For each pivoted column, select the target column and all the source columns that will feed data to it, and click **Map**.

   For the nonpivoted columns you normally create a direct 1-to-1 map but Many-to-1 maps with expressions are also allowed.
7. In the Mapped Sources tab, **Filter Value** field, for each pivoted column in the source table, select the pivot column value to identify the row in the target table into which to put the source column value. The list is populated by the code list you specified.
8. Save.

Unpivot Example Multiple observations are collected in an InForm CRF using sections rather than itemsets. A flat form is created with three sections, one section for each time point in that visit for blood draws. In InForm this is one CRF instance and one record but the standard review data model in use requires that these be three separate records.

Certain metadata values—the section name in this case—should be inserted as data in the corresponding row for that section.

Certain values in the flat section—the subject ID, visit, and date—should repeat on each row. These are the nonpivoted columns and the source column must be mapped to the target column.

The Section (Sect) column in the source table is the pivot column. It is associated with a code list containingthe values 0hr, 1hr, and 2hr.

Multiple columns in the source table—for example Sect1_Test, Sect2_Test, and Sect3_Test, map to a single column in the target table: Test.

Table 5–3 Source Table Columns in Unpivot Example (Short, Fat Table)

Subj ID	Date	Visit	Sect	Sect1 _Test	Sect1 _Unit	Sect1_ Value	Sect	Sect2 _Test	Sect2 _Unit	Sect2_ Value	Sect	Sect3 _Test	Sect3 _Unit	Sect3_ Value
509	01082112	1	0hr	Hb	gl	8	1hr	Hb	gl	8	2hr	Hb	gl	9
598	02092112	1	0hr	Hb	gl	8	1hr	Hb	gl	9	2hr	Hb	gl	10
613	02112112	1	0hr	Hb	gl	9	1hr	Hb	gl	10	2hr	Hb	gl	10

The system automatically generates a row in the target table for each value in the code list per set of nonpivoted values (SubjID, Date, and Visit) and populates the Section column in the target table with the code list values as shown in Table 5–4, " Target Table Columns in Unpivot Example (Tall, Skinny Table)".

Table 5–4 Target Table Columns in Unpivot Example (Tall, Skinny Table)

SubjID	Date	Visit	Section	Test	Unit	Value
509	01082112	1	0hr	Hb	gl	8
509	01082112	1	1hr	Hb	gl	8
509	01082112	1	2hr	Hb	gl	9
598	02092112	1	0hr	Hb	gl	8
598	02092112	1	1hr	Hb	gl	9
598	02092112	1	2hr	Hb	gl	9
613	02112112	1	0hr	Hb	gl	9
613	02112112	1	1hr	Hb	gl	10
613	02112112	1	2hr	Hb	gl	10

Custom

Select a Transformation Type of Custom only if you need a custom program AND it performs operations on data in such a way that it is not possible to track all data points in the source model that contributed to each data point in the target model; see "Defining a Custom Program" on page 5-17 for more information.

Otherwise, even if you use a custom program, select the type of transformation it actually performs: join, union, pivot, or unpivot. The system then generates the code required for data lineage tracing.

Using the Expression Builder

For more information about expressions, see:

- "Passing Data as Input Parameter Values" on page 5-14
- "Passing Metadata as Input Parameter Values" on page 5-14
- "Passing Constant Values" on page 5-14
- "Developing a Library of SQL Functions" on page 5-15.

Also see the *Oracle® Database SQL Language Reference 11g Release 2 (11.2)* at http://st-doc.us.oracle.com/11/112/server.112/e26088.pdf.

If you need to perform a more complex operation on the source data, for example using a lookup table or populating staging tables, see "Defining a Custom Program" on page 5-17.

You build an expression gradually, clicking **Add** after defining each part. The system then generates corresponding code and displays it in the Expression Text pane. You can validate the code at any time.

1. In the Mapped Sources tab, click the Source Filters icon. The Define Source Filters window opens.
2. In Expression Criteria, select one of the following to build the expression from left to right.
   - **Add Group** to add the parentheses () that surround a string in the expression.
   - **Add Item** to add a column, a constant value, or call a function.

   Note: Functions must already be defined in Oracle LSH. You select the one you need; see "Developing a Library of SQL Functions" on page 5-15.

3. When you add an item, in the Expression Item pane select either **Column** or **Function**.

   To create an expression for a column:

   a. For Item Type, select **Column**.

   b. For **Column**, click **Select**. The Select Column window appears, displaying the source columns that are currently mapped to the target column. You can query (see "Querying By Example" on page 10-9) above any of the attribute columns to find the table column you want. Select a column, then click **OK**.

   c. If needed, select an operator from the list.

   d. If needed, enter a value.

   e. If needed, select a conjunction from the list.

   f. Click **Add**. The system generates and displays the SQL expression in the Expression Text pane.

      You can click **Validate** to check the generated code.

   g. Define additional items to complete the expression as necessary.

   To reference a function:

   a. For Item Type, select **Function**. The Select Function window appears, displaying a list of packages created for this purpose; see "Developing a Library of SQL Functions" on page 5-15.

   b. Select the package that contains the function you need.

   c. Select the function and click **OK**.

   d. If needed, select an operator: ALL, IN, IS NOT NULL, NOT LIKE, LIKE, NOT IN, IS NULL, || (concatenate).

   e. If needed, enter a value.

f. If needed, select a conjunction from the list.

g. Click **Add**. The system generates and displays the SQL expression in the Expression Text pane.

   You can click **Validate** to check the generated code.

h. Define additional groups and items to complete the expression as necessary.

To make a correction:

a. Select the faulty item in the Expression Criteria pane. An Update button appears in the Expression Item pane.

b. Make your changes in the Expression Item pane and click **Update**.

Passing Data as Input Parameter Values

Use the `&` delimiter and fully qualified format (*`data_model.table.column`*) to indicate input parameter values to SQL functions or custom functions in the expression. The default input is the column value if no metadata is specified after the column name.

For example, to calculate a subject's age from his date of birth:

```
round((sysdate - {Review.LAB_SRC.dob})/365)
```

where `Review` is the data model name, `LAB_SRC` is the table name, and `dob` is the column name. No metadata follows the column name, so by default the system passes the Date of Birth (dob) data value to the expression.

Passing Metadata as Input Parameter Values

To pass metadata, add the metadata type after the fully qualified column name. To pass code list-related data or metadata, enter a dollar sign ($) after the column name and then the code list name. The default input is the code list value if no metadata is specified.

```
util_pkg.SIConversion4Height ({InForm.RD_VITALS.Height},
{InForm.RD_VITALS.Height.DataType}, {InForm.RD_VITALS.Height$U},
{InForm.RD_VITALS.Height$U.CodeListID})
```

where, on the second line, `InForm` is the data model name, `RD_VITALS` is the table name, `Height` is the column name and `DataType` is the metadata.

This expression passes four parameter values into the SIConversion4Height function so that it can convert the subject's height to a different unit if the unit used is the wrong one for the target data model: the data value for a subject's height, the data type of the Height column, the unit in which the height was collected, and the code list ID associated with the column that provides the allowed units for Height; for example, inches and centimeters.

Passing Constant Values

You can hard-code a value for a target column using an expression that contains only a constant value or by calling a SQL function based on constants; for example:

```
round(3.14 * power(10, 2))
```

Developing a Library of SQL Functions

You can develop a library of standard PL/SQL functions, including conversions and derivations, and call them in an expression. Custom functions are always static references and can reference lookup tables.

You must define each function in Oracle Life Science Data Hub as an Oracle LSH program; see "Defining a Custom Program" on page 5-17.

Each static reference program must:

- Be created in the Oracle LSH DMW_UTILS domain.
- Have a validation status of Production.
- Have its source code marked as **Shareable** in Oracle LSH.

Validating Mappings

The system does not display error messages as you create each mapping, allowing you to work more quickly. You can run the Validate Map job at any time for either the whole target model or for an individual target table.

To see the mapping messages for a target table and its columns, click its mapping icon, then hover over or click the table or column to see its error message.

Validation Error Messages

The error messages you may see are, in alphabetical order:

- DME_XFMVAL_AUTH_YES_ERR: One or more source tables is blinded and the target table is not blinded. You must either set the target table's Authorize flag to Yes if you know that it will not contain any of the sensitive, blinded data from the source table(s), or go back to the clinical data model and set the target table's blinding attributes appropriately. Special privileges are required for either action.
- DME_XFMVAL_COL_DATATYPE_ERR: The data type of the source and target columns must be the same.
- DME_XFMVAL_COL_LEN_TRUNC_ERR: To prevent data loss, the target column must have a length equal to or greater than the source columns.
- DME_XFMVAL_COL_NOSRC_ERR: Mapping required. Add one or more source columns or specify a constant value or mark as Not Used.
- DME_XFMVAL_COL_NOTRGT_ERR: Each column mapping must have one target column.
- DME_XFMVAL_COL_SRC_TRGT_SAME: A column cannot be mapped to itself.
- DME_XFMVAL_ERROR_NOT_FOUND: All mappings are valid.
- DME_XFMVAL_MOD_CYLIC_TAB_ERR: This model has circular table mappings.
- DME_XFMVAL_MOD_SRC_NOTFOUND: A model mapping must have at least one source model.
- DME_XFMVAL_MOD_SRC_TRGT_SAME : A model cannot be mapped to itself.
- DME_XFMVAL_MOD_TAB_INVALID: A model mapping must have valid table mappings.
- DME_XFMVAL_MOD_TAB_NOTFOUND: A model mapping must have at least one table mapping.

- DME_XFMVAL_MOD_TRGT_NOT_ONE: A model mapping must have one target model.
- DME_XFMVAL_NOTNULLCOL_HARDCOD: Not null columns in the target table must be mapped to either a source column or a constant value.
- DME_XFMVAL_NOTNULLCOL_NOMAPERR: Mapping required. Not null columns in the target table cannot be left unmapped.
- DME_XFMVAL_PRMCOLS_HARDCOD: All primary key columns in the target table cannot be mapped to a constant value.
- DME_XFMVAL_PRMCOLS_UNMAPPED: Mapping Required. All primary key columns in the target table are unmapped.
- DME_XFMVAL_PRMCOL_HARDCOD_ERR: Primary key column in the target table cannot be mapped both to source column or to a constant value simultaneously.
- DME_XFMVAL_PRMCOL_MAP_ERR: Primary key column in the target table must be either mapped to source column or to a constant value.
- DME_XFMVAL_PRMCOL_NOMAPERR: Mapping Required. Primary key column in the target table cannot be left unmapped.
- DME_XFMVAL_TAB_DIR_MULTSRC_ER: Table mappings of type Direct can have only one source table.
- DME_XFMVAL_TAB_JOIN_ONESRC_ERR: Table mappings of type Join must have at least two source tables.
- DME_XFMVAL_TAB_NOCOLMAP_ERR: Table mappings must have at least one column mapping.
- DME_XFMVAL_TAB_NOSRC_ERR: Each table mapping must have at least one source table.
- DME_XFMVAL_TAB_NOTRGT_ERR: Each table mapping must have one target table.
- DME_XFMVAL_TAB_SRC_TRGT_SAME: A table cannot be mapped to itself.
- DME_XFMVAL_TAB_UNION_ONESRC_ER: Table mappings of type Union must have at least two source tables.
- DME_XFMVAL_XFORM_TYPE_ERR: Incorrect Table Map Type. It can be Direct, Join, Union, Pivot or Unpivot.

Installing Transformations

After you have completed mapping, install the transformation using the **Install Map** button. The system:

- Uses the mappings, joins and other transformation types, and expressions to generate a PL/SQL program and its packages in the database.
- Links the transformation program to the target model and installs the target model.
- Generates the auxiliary target table columns and logic required to maintain the source system context and data lineage tracing for each record; see "How the System Tracks Data Lineage" on page 5-24.

The transformation must be Installable to be installed. It is not Installable if:

- Its status is Incomplete; this indicates that at least one table or column in the target model is neither mapped nor marked as Not Used.
- One or more source or target tables are not Installable.
- If it has a custom program, the program is not Installable, which may be due to not having source code or table descriptors.
- One or more expressions has invalid code.

To see the log file:

1. Navigate to the Home page, Transformations tab.
2. Find the job by querying for the target model name in the Target column, and the date and time in the Submit Date/Time column.
3. Click the icon in the Install Job Log column. The system displays the log file.

Running a Transformation

After you have completed all mappings and any required derivations, saved, and installed the transformation, you can run the generated program in the Transformations tab of the Home page to actually read data from one data model and write to the next.; see "Viewing and Running Transformations" on page 10-3.

Modifying a Transformation

To modify a transformation you must first check it out at the model level. While you have it checked out, noone else can modify any part of it. When you have finished your work, you must check it in and install it for the changes to take effect.

Each time you check out a mapping, the system creates a new version of it at a Lifecycle stage of Development.

After checking out a transformation you can choose to Uncheck it. The system then deletes the new version with your changes.

Upgrading a Mapping to Reflect Model Metadata Changes

When either a source or target model referenced by a transformation is versioned and the later version installed, you can automatically upgrade the mapping to synchronize it with the model changes. The system sets **Upgrade Required** to **Yes** and activates the **Upgrade Map** button. Click it to perform the upgrade.

Defining a Custom Program

If the transformation to any target column or table is too complex for the user interface, you can create a custom program that must take care of all column mappings for the target table; for example:

- To refer to a lookup table outside the data model with shared code lists, lab reference ranges, or SI conversion factors
- To perform data aggregation, case statements, or complex calculations
- To call an API to set a flag on records that meet specified criteria; see the *Oracle Life Sciences Data Hub Application Programming Interface Guide* for information on APIs for Oracle LSH and Oracle DMW, and see "Sample Program that Calls the API to Set a Flag" on page 5-22.

- You need to use 1-to-Many or Many-to-Many column mappings

You can write a PL/SQL, SAS, or Informatica program in an interactive editor and upload it and define a program object in Oracle Life Sciences Data Hub (Oracle LSH). You can reuse custom programs.

> **Note:** To create and use a SAS or Informatica program, you must purchase SAS or Informatica separately and integrate it with Oracle LSH. See the *Oracle Life Sciences Data Hub Installation Guide* and the *Oracle Life Sciences Data Hub System Administrator's Guide* for instructions.

Creating a custom program requires:

1. "Enabling Data Lineage Tracing in a Custom Program" on page 5-18
2. "Creating a Custom Program in Oracle Life Sciences Data Hub" on page 5-19
3. "Completing the Transformation" on page 5-20

This section also contains:

- "Sample Programs that Populate Auxiliary Columns with the Source Surrogate Key" on page 5-21
- "Sample Program that Calls the API to Set a Flag" on page 5-22

Enabling Data Lineage Tracing in a Custom Program

If possible, enable data lineage tracing so that the system can display discrepancies in models before and after the one where they were created, and display information about contributing data points upstream and resulting data points downstream from each data point.

> **Note:** If your custom program performs operations on data in such a way that it will not be possible to track all data points in the source model that contributed to each data point in the target model—for example, aggregations—there is no need to follow these instructions. Data lineage tracing will not work. Users will be able to create discrepancies in both the source and target model, but the system will only be able to display these discrepancies in the model in which they were created.
>
> If you are performing a pivot or unpivot in a custom program, the source columns' surrogate key values must be modified before writing to the auxiliary column in the target table or data lineage tracing will not work. However, you can create an intermediate model containing a table and create a pivot or unpivot transformation to it in the user interface, then create a custom program from the intermediate table to the target table.

Add Auxiliary Columns to the Target Table When you create a custom program for a transformation you must add one auxiliary column to the target table to store the surrogate key for each source table record. The system uses these columns to support data lineage tracing; see "How the System Tracks Data Lineage" on page 5-24.

To add auxiliary columns:

1. In the Study Configuration page, navigate to your target data model and check it out.
2. Select the target table and click the + icon in the Columns pane. Add one column for each source table that will feed data into it in the transformation. These columns must be of data type varchar2 and length 4000. There is no required naming convention for these columns, but see "Naming Objects" on page A-1.
3. Save and check in the data model.

Note: If you start to map to the target table in the user interface before creating the auxiliary columns, you will not see the columns in the user interface. If you then add the columns and go back to the transformation, you must check in the transformation and check it out again in order to have a new version that can "see" the columns.

However, you can write the custom program in Oracle LSH before or after creating the tranformation.

Populate the Auxiliary Columns Your program code must populate one auxiliary column with the value of the CDR$SKEY column in each source table. See "Sample Programs that Populate Auxiliary Columns with the Source Surrogate Key" on page 5-21 for example SAS and PL/SQL code required to do this.

Creating a Custom Program in Oracle Life Sciences Data Hub

To create a custom program:

1. Log in to Oracle LSH, select its Applications tab if it is not already selected, and search for the DMW_UTILS domain under the DMW_DOMAIN.
2. In the DMW_UTILS domain, navigate down to the subdomain your company has set up for storing custom programs of this type; see "Setting Up Custom Program and Function Categories" on page 8-16. Click the Manage Definitions icon on that row.
3. In the Maintain Library Domain window Create field select **Program** and click **Go**. Enter a name and description and select the Program Type: PLSQL, SAS, or Informatica. Click **Apply**.
4. In the Table Descriptors subtab, click Add Target From **Library**, then select **Create a Table Descriptor from an existing Table definition** and click the Search icon for the Definition Source field.
5. In the pop-up window:
   a. In the **Domain** list, select DMW_DOMAIN, the appropriate category for your study, and your study.
   b. Select **Display Table Definitions Under DataModel**. The system displays the Data Models field. Select the clinical data model and enter the table name if you know it, then click **Go**.
   c. When the system returns the table you are searching for, click its Quick Select icon. The system returns you to the Create Table Descriptor page with the selected table displayed in the Definition Source field. Click **Apply**.
6. If you are adding a source table, click **Update** in the Table Descriptor page and change **Is Target** to **No** and click **Apply**, then click **Return** to return to the program page.

Repeat Steps 5-7 until you have added each source table. For transformations, you must also add the target table.

7. Write the program. If you have integrated SAS with Oracle LSH you can click the Launch IDE button from the program page.

   Your program must handle populating all target columns, including populating the new auxiliary column(s) with the value of the internal CDR$SKEY (surrogate key) column in each source table.

8. In the Source Code subtab, click **Add**, then select **Create a new Source Code definition and instance**. Enter all required values. For a SAS program, the File Type should be **Program**, not Macro.

   Note: Source Code names:

   - **must not** include Oracle or PL/SQL reserved words or special characters; see "Avoid Special Characters and Reserved Words" on page A-1.
   - **must** include a file extension; for example, `.sas` for SAS or `.sql` for PL/SQL.
   - The Oracle name must not be the same as the Oracle name of either a table descriptor or another source code in the same PL/SQL program.

9. Upload the file containing your program and click **Apply**.

10. If your code uses parameters, formally define them as Oracle LSH parameter objects in the Parameters subtab. However, using parameters makes sense only for functions called from an expression where you can provide input values. DMW transformation submission does not allow setting parameter values.

    Note: If you are writing an Informatica program, you must populate the WF Name parameter with the name of the Informatica workflow that you want execute. Execution fails without this value.

11. If required, you can call Oracle LSH or Oracle DMW public APIs from your code; see the *Oracle Life Sciences Data Hub Application Programming Interface Guide* for a list and descriptions of the public APIs and information about how to call them.

12. Check in the program in Oracle LSH.

13. Return to Oracle DMW. If you are defining a validation check, see "Creating a Custom Validation Check" on page 6-7. If you are defining a transformation, follow instructions in the next section.

Completing the Transformation

If you have not already created auxiliary key column(s) in the target table, do so now; see "Enabling Data Lineage Tracing in a Custom Program" on page 5-18.

1. In Oracle DMW, navigate to the transformation mapping for the target data model and table.

   a. Select the source table(s) and target table.

   b. Click the icon in the Program column and select the Oracle LSH program.

c. Specify the actual Transformation Type.

Note: In most cases DO NOT select a Transformation Type of Custom. Select the actual transformation type that your code performs. The system uses the Transformation Type to ensure that data lineage tracing works correctly.

Select a Transformation Type of Custom if you are performing operations that make it impossible to trace data lineage, such as aggregations. In that case, users can create discrepancies on the source or target table but the system can display these discrepancies only in the model in which they were created.

---

d. Map the source and target columns, including mapping the CDR$SKEY column in each source table to the corresponding surrogate key column you created in the target table. (The CDR$SKEY columns are normally not visible in the Column Mapping pane, but this changes when the table Transformation Type is Custom.)

2. Install the whole transformation.

   The system creates an instance of the custom program in the study's Development lifecycle area and the corresponding database schema and maps its table descriptors to the source and target table instances. You can now run the transformation in the Activities page.

Sample Programs that Populate Auxiliary Columns with the Source Surrogate Key

The following example SAS and PL/SLQL programs are straightforward ones that you could create in the user interface using a Transformation Type of Direct and Join, respectively. However, they show how to populate auxiliary keys in the target table with the surrogate key value from the source table(s).

Example SAS Program

```
PROC SQL;
INSERT
INTO Target.vitals_tgt
  (
    STUDYID,
    SITEID,
    SUBJID,
    VISITNUM,
    INITIALS,
    BIRTHDT,
    HEIGHT,
    HEIGHTU,
    WEIGHT,
    WEIGHTU,
    SOURCE_KEY
  )
SELECT STUDYID,
  SITEID,
  SUBJID,
  VISITNUM,
  INITIALS,
  BIRTHDT,
  HEIGHT,
```

```
  HEIGHTU,
  WEIGHT,
  WEIGHTU,
  CDR_SKEY
FROM Source.vitals;
QUIT;
```

Example PL/SQL Program

```
CREATE OR REPLACE PACKAGE VITALS_PKG AS
  PROCEDURE loadVitals;
END VITALS_PKG ;
/

CREATE OR REPLACE PACKAGE BODY VITALS_PKG AS
  PROCEDURE loadVitals IS
   BEGIN
     insert into vitals_tgt (
      STUDYID,
      SITEID,
      SUBJID,
      VISITNUM,
      INITIALS,
      BIRTHDT,
      HEIGHT,
      HEIGHTU,
      WEIGHT,
      WEIGHTU,
      SEX,
      RACE,
      VITALS_SKEY,
      DEMOG_SKEY
     ) select
      a.STUDYID,
      a.SITEID,
      a.SUBJID,
      a.VISITNUM,
      a.INITIALS,
      a.BIRTHDT,
      a.HEIGHT,
      a.HEIGHTU,
      a.WEIGHT,
      a.WEIGHTU,
      b.SEX,
      b.RACE,
      a.CDR$SKEY,
      b.CDR$SKEY
     from vitals a, demog b
     where a.studyid = b.studyid
     and a.subjid = b.subjid;
   END loadVitals;
END VITALS_PKG;
/
```

Sample Program that Calls the API to Set a Flag

This sample program reads flags on source table data, and if the flag is 'Complete' for a row, it inserts that row into the target table, and on the target table it assigns the 'Complete' flag to all rows.

```
CREATE OR REPLACE PACKAGE VITALS_PKG AS
  PROCEDURE loadVitals;
END VITALS_PKG ;
/

CREATE OR REPLACE PACKAGE BODY VITALS_PKG AS
  PROCEDURE loadVitals IS
  x_return_status VARCHAR2(1);
  x_msg_count     NUMBER;
  x_msg_data      VARCHAR2(1000);
  oFlagName dme_flag_name_type;
  vFlagState varchar2(100);
  BEGIN
    -- get the flag
    dme_pub_flag_name.getFlagName(
              p_api_version => 1.0
            , p_init_msg_list => CDR_PUB_DEF_CONSTANTS.G_FALSE
            , p_commit => CDR_PUB_DEF_CONSTANTS.G_TRUE
            , p_validation_level => CDR_PUB_DEF_CONSTANTS.G_VALID_LEVEL_FULL
            , x_return_status => x_return_status
            , x_msg_count => x_msg_count
            , x_msg_data => x_msg_data
            , pi_company_id => cdr_pub_def_constants.current_company_id
            , pi_flag_namestr => 'Completeness'
            , pio_dme_flag_name => oFlagName
    );
    for row in (select
      STUDYID,
      SITEID,
      SUBJID,
      VISITNUM,
      INITIALS,
      BIRTHDT,
      HEIGHT,
      HEIGHTU,
      WEIGHT,
      WEIGHTU,
      CDR$SKEY
     from vitals
    ) loop
      dme_pub_flag_data.getFlag(
          p_api_version => 1.0
        , p_init_msg_list => CDR_PUB_DEF_CONSTANTS.G_FALSE
        , p_commit => CDR_PUB_DEF_CONSTANTS.G_TRUE
        , p_validation_level => CDR_PUB_DEF_CONSTANTS.G_VALID_LEVEL_FULL
        , x_return_status => x_return_status
        , x_msg_count => x_msg_count
        , x_msg_data => x_msg_data
        , pi_company_id => cdr_pub_def_constants.current_company_id
        , pi_tab_obj_id => cdr_pub_df_mapping.GET_TAB_INST_ID('VITALS')
        , pi_skey_value => row.cdr$skey
        , pi_flag_id => oFlagName.flag_id
        , po_flag_state => vFlagState
      );

      if vFlagState = 'Complete' then
        insert into vitals_tgt (
          STUDYID,
          SITEID,
          SUBJID,
```

```
          VISITNUM,
          INITIALS,
          BIRTHDT,
          HEIGHT,
          HEIGHTU,
          WEIGHT,
          WEIGHTU,
          SOURCE_KEY
         )
         values (row.studyid, row.siteid, row.subjid, row.visitnum, row.initials,
row.birthdt, row.height, row.heightu, row.weight, row.weightu, row.cdr$skey);
      end if;
   end loop;
   for row in (select * from vitals_tgt) loop
      dme_pub_flag_data.setFlag(
          p_api_version => 1.0
        , p_init_msg_list => CDR_PUB_DEF_CONSTANTS.G_FALSE
        , p_commit => CDR_PUB_DEF_CONSTANTS.G_TRUE
        , p_validation_level => CDR_PUB_DEF_CONSTANTS.G_VALID_LEVEL_FULL
        , x_return_status => x_return_status
        , x_msg_count => x_msg_count
        , x_msg_data => x_msg_data
        , pi_company_id => cdr_pub_def_constants.current_company_id
        , pi_tab_obj_id => cdr_pub_df_mapping.GET_TAB_INST_ID('VITALS_TGT')
        , pi_skey_value => row.cdr$skey
        , pi_flag_id => oFlagName.flag_id
        , pi_flag_state => 'Complete'
      );
   end loop;
  END loadVitals;
END VITALS_PKG;
/
```

How the System Tracks Data Lineage

The system tracks each data point in each target clinical data model back to the raw source data—one or more data points in one or more input models that contributed to it, and all models in between—and to all data points it contributes to in downstream models.

Maintaining this context, or *data lineage*, is required in order to pass discrepancies back and forth between the system and its source data systems, InForm and labs, and to recognize a discrepancy as the same discrepancy in all sequential models.

You can see a data point's source and target data lineage in the Listings page.

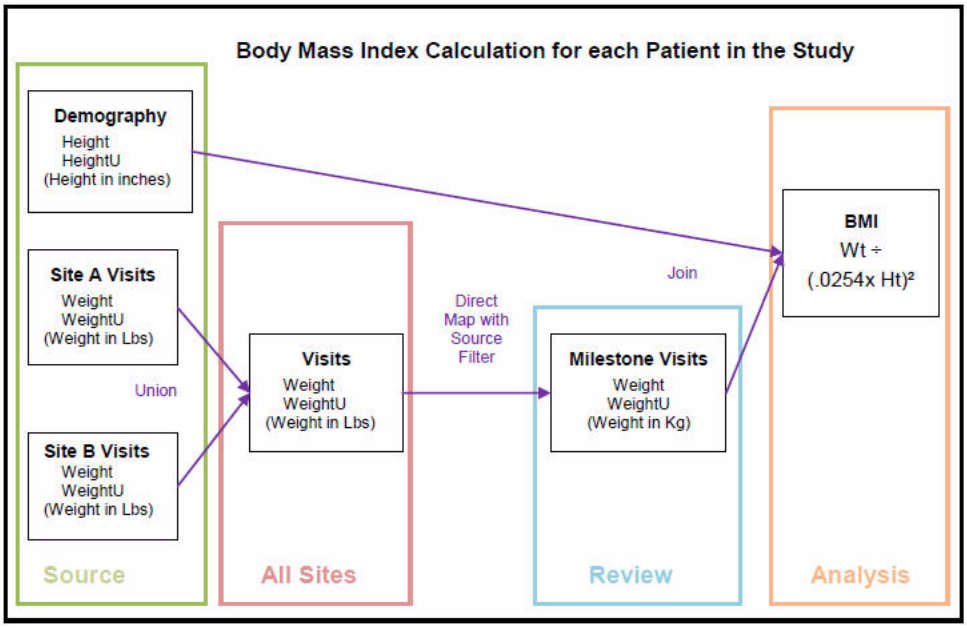
The system uses several mechanisms to maintain context:

- **Mappings**: The system stores the table and column mappings you define as part of a transformation. The system also generates record-level mappings during transformation execution.
- **Surrogate Keys**: The surrogate key value for each record is a concatenation of the table instance identifier followed by the values in the primary key columns in the order specified in the primary key constraint, separated by tildes (~); for example, *table ID~subject~ visit~ crf~test*.
- **Generated Columns to Store Surrogate Key Values**: When a transformation program is installed it adds one auxiliary column to each target table for each source table that writes to the target, to store the surrogate key of source records.

These mechanisms provide enough metadata to trace back a transformed data point to its contributing source data points, even through multiple transformations.

Context Example

In this example, multiple data points in three data models contribute to the calculation of each subject's Body Mass Index (BMI).



The clinical data model called Source is an input model with raw data from InForm including the following tables:

- Table Demography includes the columns Height and Height Unit and has data for all subjects.
- Table Site A Visits includes the columns Weight and Weight Units and has data for subjects at Site A.
- Table Site B Visits includes the columns Weight and Weight Units and has data for subjects at Site B.

Model All Sites has a table called Visits that is a union of Source tables Site A Visits and Site B Visits. It stores the Weight and Weight Unit for all subjects at both sites.

Model Review has a table called Milestone Visits created as a direct map with a source filter that copies only data for the milestone visits 1, 4, 7, and 10. The table has Weight and Weight Unit for all subjects at both sites, now all converted to kilograms.

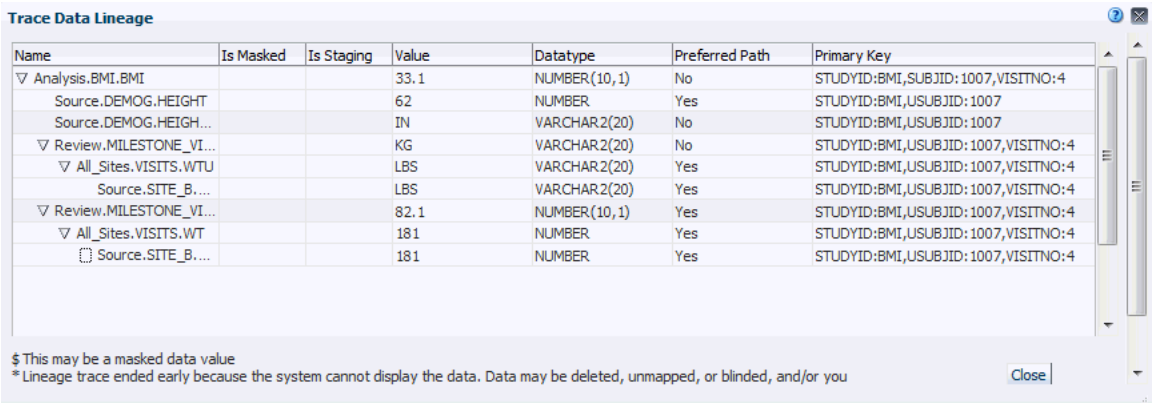
The model Analysis has a table called BMI whose BMI column value is the computation for Body Mass Index using the height from Demography and weight from Milestone Visits.

Table 5–5 Data in Table BMI

Study	Subject	Visit	Site	Gender	Age	BMI	BMIU
BMI	1001	4	MGH	FEMALE	34	22.7	KG/M**2
BMI	1002	4	MGH	FEMALE	24	24.5	KG/M**2
BMI	1003	4	MGH	MALE	42	21.9	KG/M**2
BMI	1004	4	MGH	FEMALE	28	26.3	KG/M**2
BMI	1005	4	McLean	MALE	29	24.6	KG/M**2
BMI	1006	4	McLean	MALE	54	28.8	KG/M**2
BMI	1007	4	McLean	FEMALE	42	33.1	KG/M**2

A BMI value over 30 is outside the normal range. The reviewer would like to investigate the source data for Subject 1007, a female aged 42 with a BMI of 33.1. In the Default Listings page, the reviewer selects the BMI data value 33.1 for Subject 1007 and selects **View Source Data**, then **Trace Data Lineage**.

Figure 5–1 Source Data Lineage Tracing Example



The Trace Data Lineage window appears, displaying the selected data point in the top row. In the following rows, at the first level of indentation, are the four data points from two tables that feed data directly into the BMI calculation. Source data points that are not from input models—in this case, both a Weight value of 82.1 and a WeightU value of KG —have a node you can expand to see the upstream data points feeding data into them. Source data points from an input lab or InForm model—in this case, a Height value of 62 and a HeightU value of IN—have no node to expand because they are the ultimate source within Oracle DMW.

Columns are displayed in the format *data_model.table.column*.

The user can also choose to see downstream data.

If data has been masked to prevent divulging which subjects are receiving the study drug, a dollar sign ($) is displayed in the unlabeled column immediately after the column name in the relevant row or rows.

Note: The system cannot always display the entire lineage. There may be a problem with the data: it may be deleted, unmapped, or blinded; or you may not have the privileges required to view the data. The system displays an asterisk (*) when this is the case.

6

Creating Validation Checks

This section contains the following topics:

See "Viewing and Running Validation Check Batches" on page 10-5.

About Validation Checks

Validation checks—also called edit checks—are programs designed to identify flawed data: *discrepancies* (called *queries* in InForm). Each one must check for a single problem and apply the same text, state, and action to each discrepancy created. Validation checks can test any combination of data—including lab and InForm data and data from different CRFs—contained in a single data model.

The same flawed data point can have several discrepancies created against it raised by different validation checks; for example, if the value is both out of range and does not make sense compared to a related data point.

The first time a validation check is executed, it automatically creates a target table with columns corresponding to source table columns you select and a row for each discrepancy identified. Each time it runs it updates the table with new or changed data. The data in this table is displayed in the VC (Validation Check) Listings page.

Validation checks are part of a study or study template. When you apply a study template to a study, the validation checks from the study template are copied to the target study.

Automated Discrepancy Closure You can set up a validation check so that it closes any discrepancies it previously created if the underlying data point has been updated in such a way that it is no longer discrepant.

Manual Discrepancy Closure Alternatively, you can set up a validation check so that, when the underlying data point is no longer discrepant, the validation check marks the discrepancy as "Answered" but a data manager must review the discrepancy and data change and manually close it.

Custom Programs You can define most validation checks, including comparing related data points across tests, visits, or sources, directly in the Validation Check window. However, if you need a more complex program, you can write a program in SAS or PL/SQL and upload it to Oracle Life Sciences Data Hub (Oracle LSH) and reference the program from a validation check; see "Creating a Custom Validation Check" on page 6-7 for more information.

Creating a Validation Check Batch

Validation checks are executed in batches of one or more. Before you create a validation check you must create the batch in which to execute it. You can use batches to group validation checks in logical ways, for example:

- Checks that have dependencies on each other so they must be executed in a particular order.
- Standard checks kept in a set for ease of reuse in many studies.
- Checks that should all be executed manually, or triggered by the same event, or scheduled for the same frequency.

To create a validation check batch:

1. Select your study in the Home page and click **Study Configuration**.
2. Select **Validation Checks** at the bottom of the left pane.
3. Select the clinical data model whose data you want to check.
4. Click **Create a New Batch**.
5. Enter a name for the batch that is unique within the model. If it is not unique within the study, the system automatically appends "_1" to its name.
6. Enter a description (optional). The description is displayed in the places where you can run the batch.
7. If the validation checks in the batch should be run in a particular order select the **Ordered Execution?** check box. You specify the order when you create the validation checks.

   Validation checks in unordered batches run in parallel.
8. Check **Can be Triggered** to allow the successful completion of a transformation or data load writing to the clinical data model that this batch runs against to trigger the execution of this batch.
9. Click **OK**.
10. Create the validation checks to run in the batch; see "Creating a Validation Check" on page 6-2.

Creating a Validation Check

1. Before you create a validation check, you must create a batch in which to execute it; see "Creating a Validation Check Batch" on page 6-2.
2. Select the batch. If it is not already checked out, click **Check Out**.
3. Click **Create a New VC**. Then see:
   - "Building the Validation Check Query" on page 6-3
   - "Defining Validation Check Details" on page 6-5

Building the Validation Check Query

To create the query for the validation check:

1. Enter a name and description for the query. The name is displayed on the VC Listings page; the description is not. The Listings page can display about 25 characters of the name without requiring scrolling to see the whole name. You may want to use a naming convention to make it easier to find queries later.
2. Select **Authorize access to this listing for users without Blind Break rights** if you know that only nonblinded data will be displayed in the listing. This option is available only if at least one source table contains blinded data and if you have the required blinding-related privileges.

   If any source table is blinded at any level and this attribute is not selected, the system automatically blinds the entire target table, so that by default no data is displayed on the VC Listings page for this validation check. Only a user with the required Blind Break privileges can view the data.
3. Select **Create VC Using a Custom Program** if you need more complex logic than you can create here and you have already created the program in Oracle LSH; see "Creating a Custom Program in Oracle Life Sciences Data Hub" on page 5-19. Then click the Edit icon to select the custom program.

   If you select **Create VC Using a Custom Program**, select the source table(s) and skip the next steps. Your program must take care of selecting columns for display in the target table on the VC Listings page, creating joins, and specifying criteria if required.
4. Specify the columns to display; see "Selecting Columns to Display in VC Listings" on page 6-3.
5. If needed, create joins between tables; see "Creating Joins" on page 6-4.
6. Specify query criteria; see "Specifying Criteria" on page 6-5.

Selecting Columns to Display in VC Listings

In the Select Columns tab, identify the columns you want to display for each record retrieved by the query. You can give columns a different header for display and combine or write expressions on one or more columns as required. The system creates a SELECT clause for the query based on your specifications.

1. In the Source pane, expand the node for the table or tables whose data you want to display in the VC Listings page for each record retrieved.
2. Select the columns you want to display. You can use Ctrl+click and Shift+click to select multiple columns at a time. You can also select a table to add all its columns and then remove the ones you do not want to display.
3. Click the Add (+) icon. The system displays the columns you selected under **Selected Columns**.
4. If you want to display a heading for the column different from the column name, enter it in the Alias field.
5. **Expression**: If you want to write an expression on a column, click its Edit (pencil) icon; see "Using the Expression Builder" on page 5-12.

   To write an expression that operates on multiple columns you must add each column to the same row:

   a. After moving one column into Selected Columns, highlight it there.

b. Select the second column in the Source pan and click the arrow icon in the Source pane. The system adds the second column to the same row. If you need a third column for your expression, add it the same way.

c. Enter an alias for the column that will display the results of the expression.

For example, you may want to display the red blood cell count collected three times in a visit and then display the average of the three results. To do this, select the column for red blood cell count in the Source pane four times and add it to Selected Columns. Then select the last one under Selected Columns, select it again under Source and click the arrow icon, and then repeat. All three are now in the same row. Enter an alias such as "RBC Avg" in the row with all three. Then write an expression for each row to populate the first three columns with data from each sample and the last to calculate and display the average. See "Using the Expression Builder" on page 5-12.

Creating Joins

To compare columns in different tables, you must define a join between the tables.

To create a join, select and add the tables to be joined in the Source pane and then:

1. Go to the Joins tab and click the Add (+) icon in the Joins pane. The system adds a row.
2. Click in the row. The system displays two drop-down lists containing the tables in the join.
3. Select the tables to be joined.
4. Specify if it is to be an outer join on either the left or right side. Leave the checkbox unselected to create an inner join.

   An *inner join* (sometimes called a *simple join*) returns only those rows that satisfy the join condition.

   An *outer join* extends the result of a simple join. An outer join returns all rows that satisfy the join condition and also returns some or all of those rows from one table for which no rows from the other satisfy the join condition:

   - A *left outer join* returns returns all rows from Table 1 and only rows meeting the join condition from Table 2. Rows in Table 1 that do not have a corresponding row in Table 2 have null values in the columns from Table 2.
   - A *right outer join* returns returns all rows from Table 2 and only rows meeting the join condition from Table 1. Rows in Table 2 that do not have a corresponding row in Table 1 have null values in the columns from Table 1.
   - A *full outer join* returns all rows from both or all tables. Rows in either table that do not have a corresponding row in the other table have null values in the columns from the other table.
5. Click the Add (+) icon in the Join Details pane. The system adds a row.
6. Click in the row. The system displays two drop-down lists containing the columns for each table and another for operators.
7. Create the Join condition for the Where clause by selecting a column from each table and the operator required. For example, where both tables have a column called USUBJID, select those columns and select an operator of *equals* (=).

   To specify additional Join conditions, click the Add (+) icon in the Join Details pane again as many times as required.

8. Click **OK**.

Specifying Criteria

In the Criteria tab, identify the table(s) whose columns you want to operate on to determine which rows to retrieve. The system creates a WHERE clause based on your specifications.

1. Select the table or tables you need in the Source pane and click the Add icon. If you need to operate on columns from two or more tables in the same expression, add them to the same row:
   a. After moving one table into the Criteria tab, highlight it there.
   b. Select the second table in the Source pan and click the arrow icon in the Source pane. The system adds the second table to the same row in the Criteria tab. If you need a third table for your expression, add it the same way.
2. Click the edit icon to open the Expression Builder. See "Using the Expression Builder" on page 5-12.
3. Save. The system generates a system PL/SQL program.

Defining Validation Check Details

Validation checks must be associated with a *batch* of validation checks that are executed together.

1. Select the batch. If it is not already checked out, click **Check Out**.
2. Click **Create a New VC**.
3. Enter a name for the validation check that is unique within the batch. If it is not unique within the batch, the system automatically appends "_1" to its name. Avoid using special characters except underscore (_) and reserved words; see "Naming Objects" on page A-1.
4. All the other fields affect the discrepancies identified by the validation check:
   - **Allow Auto Close**: If selected, rerunning the validation check automatically closes any discrepancy it created if the underlying data point is modified so that it no longer satisfies the criteria of the validation check.

     If not selected, rerunning the validation check changes the state of discrepancies not satisfying the criteria to Answered, so that they can be manually closed after review.
   - **Category**: Select a category for the validation check. The validation check applies this category to each discrepancy it creates. Users can filter and count discrepancies by category.

     See "Viewing and Creating Categories" on page 8-9.
   - **Discrepancy Initial State**: Select a state to be applied to discrepancies created by this validation check: **Open** or **Candidate**. You can require review in Oracle DMW or send discrepancies to InForm immediately in either state.
   - **Discrepancy Text**: Enter the text to be displayed for the discrepancy. This is the message that will be used for communication with InForm, lab, or Oracle DMW users. Describe the problem with the data and/or the action required.
   - **Initial Discrepancy Action**: Select the action you want the validation check to immediately perform on the new discrepancies, if any; see "Actions" on page 7-3.

- **Primary Source Table**: You must designate one data point as the primary one against which discrepancies are created. Select the table that contains the primary data point. If the validation check logic processes only one data point, select its table. If the logic processes two or more data points, you must designate one of them as primary.

  Note: You can ensure that all data points are displayed with the discrepancy in the VC Listings page by adding their columns in the Selected Columns tab.

- **Primary Source Column**: Select the column that contains the primary data point.
- **Execution Order**: Enter a number to indicate the order in which this validation check should be run, in relation to other validation checks in the same batch. The system runs the lowest numbered check first, then the next highest, and so on. The numbers do not need to be consecutive, and you may want to use numbers divisible by 10, for example, so that you can add a new validation check at any point. This setting is available only in batches with ordered execution. The default value is 100. Up to 6 digits are allowed.
- **Continue on Error**: If selected, the batch continues to execute subsequent validation checks even after one fails. This setting is available only in batches with ordered execution. Unordered batches always run all validation checks.

Installing Validation Checks

Before you can run a validation check, you must select its batch and select **Install**. This creates database packages for the generated program. If any source tables are not installed, the Installation job tries to install the whole clinical data model.

The batch must be Installable to be installed. It is not Installable if:

- It does not have any validation checks.
- One or more source tables are not installable.
- If it has a custom program, the program is not installable, which may be due to not having source code or table descriptors.

If the batch does not install successfully, the system displays an error message with a Job ID. You can view the log file on the Home page's Validation Checks tab.

Running a Validation Check Batch

After you have installed the validation check batch, you can run it in the **Home** page, Validation Check tab.

You can run the job at any time on demand or schedule it to run on a regular basis or link it to other jobs as part of a scheduled or triggered forward-chained execution; see "Viewing and Running Validation Check Batches" on page 10-5.

The resulting data is displayed in the VC (Validation Checks) Listings page.

Modifying a Validation Check

To make any change to a validation check other than enabling or disabling it, you must check out the batch. To make most changes, click **Update** and follow instructions for "Creating a Validation Check" on page 6-2.

The system also supports the following types of changes:

- "Reordering Validation Checks" on page 6-7
- "Disabling a Validation Check" on page 6-7
- "Upgrading a Validation Check Batch" on page 6-7

Reordering Validation Checks

To change the execution order of an ordered batch, click **Re-order**. The system displays all validation checks in the batch with their Execution Order number. Edit the numbers as required. They do not need to be consecutive. The system runs the validation check with the lowest number first, then the next higher number, and so on. For example, it would run checks with these numbers in the following order: 20, 30, 35, 36, 40, 100, 200.

Note: If a validation check has Enabled set to No, the system ignores its execution order number and does not run it.

Disabling a Validation Check

You can prevent a validation check from being executed when the batch is executed by selecting it and then selecting **Disable**. To include in the batch execution after disabling it, select it and then select **Enable**.

You can change this setting when the validation check batch is checked in or out.

Upgrading a Validation Check Batch

If there have been metadata changes in the source clinical data model—for example, change of column length—that affect any validation check in a batch, the batch becomes upgradable.

1. Go to the Validation Check Batch list in either the Home or the Study Configuration page.
2. Select the batch and click Upgrade. The system updates the table and column mappings behind the scenes.

Creating a Custom Validation Check

If you want to write a program that is too complex for the user interface, you can write a PL/SQL or SAS program in an editor and upload it to an Oracle Life Sciences Data Hub (Oracle LSH) program, then associate it with a validation check. For example, if you need to refer to a table outside the data model such as shared codelist tables, lab reference ranges, or SI conversion factors, you must create a custom program and use a static reference within it.

You also need a custom program to call APIs for assigning flags to records or applying actions to records.

If you have already written an appropriate SAS or PL/SQL program outside of Oracle LSH, you can upload it to an Oracle LSH program. However, to support data lineage tracing, you will need to add columns to the target table and populate them as described below.

> **Note:** To create and use a SAS program, you must purchase SAS separately and integrate it with Oracle LSH. See the *Oracle Life Sciences Data Hub Installation Guide* for instructions.

1. Create the program the same way you do for a transformation; see "Creating a Custom Program in Oracle Life Sciences Data Hub" on page 5-19 except for this additional task:

   Each validation check should have a single target table. To support data lineage tracing, this target table must have one column per source table to store the surrogate key for the source table, which is a concatenation of the primary key values for each record; see "How the System Tracks Data Lineage" on page 5-24.

   a. In the Oracle LSH Program page, create a new Table Descriptor and Table instance for the target table.

   b. In addition to the columns you need to display the results of the validation check, add one column for each source source table with a name like *source_table*_SKEY.

   c. In the source code, populate each of these columns with the value from the CDR$SKEY column of each source table.

2. Define the validation check in a batch and enter values for all required fields.

3. Select **Create VC using a Custom Program**. The system displays a window where you can select the source table(s) for the custom validation check program.

4. When the Select a Program icon appears, click it and select the program. You can use the Query By Example fields above any column to search for all or part of a value in that column; for example, enter `%cardio%` to search for a program in the Cardiology category your company has set up; see "Setting Up Custom Program and Function Categories" on page 8-16.

7

Configuring Your Discrepancy Workflow

This section contains the following topics:

See also:

- "Viewing and Creating Categories" on page 8-9
- "Viewing and Creating Discrepancy Tags" on page 8-10
- "Creating and Using Flags" on page 8-11

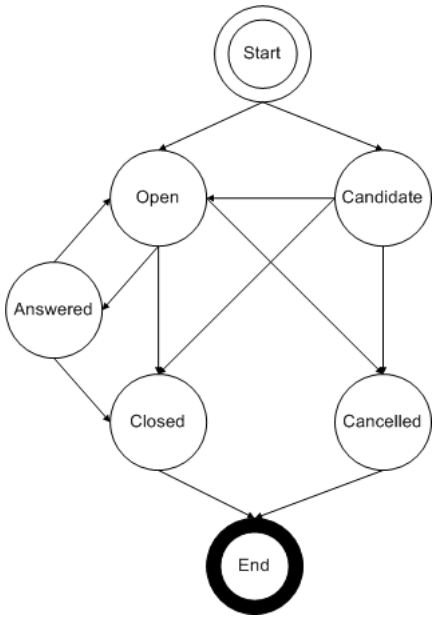
Discrepancy States and Allowed Transitions

Oracle Health Sciences Data Management Workbench (Oracle DMW) comes with a set of discrepancy states, allowed transitions between them, and **action**s that change the state of the discrepancy and/or apply a **tag**. The allowed transitions differ depending on the source of the discrepant data. You can define custom actions, following the same rules.

Discrepancies on Lab Data

When a discrepancy is created on lab data in Oracle DMW, either by a user or by a validation check, the system allows the maximum number of states and transitions, as shown in Figure 7–1, "Discrepancy States and Allowed Transitions for Lab Data Discrepancies":

Figure 7–1 Discrepancy States and Allowed Transitions for Lab Data Discrepancies



As shown in the diagram, for lab data discrepancies:

- Candidate and Open are the two valid beginning states for a discrepancy.
- Cancelled and Closed are the two valid end states for a discrepancy.
- Candidate discrepancies can move to Open, Cancelled, or Closed.
- Open discrepancies can move to Cancelled, Answered, or Closed.
- Answered discrepancies can move back to Open or to Closed. They cannot be reopened as Candidate.
- Discrepancies can be closed from the Candidate, Open, or Answered state.

Discrepancies on InForm Data

Discrepancies can be raised against data from InForm in either system:

- Queries raised in InForm are loaded into Oracle DMW as discrepancies with the same state as in InForm.
- Either a validation check or a user can raise a discrepancy in Oracle DMW against data that originated in InForm.

When a discrepancy is sent to InForm, the system limits the actions an Oracle DMW user can take on the discrepancy until it is sent back to Oracle DMW. The same limited actions are allowed whether the discrepancy originated in InForm or in Oracle DMW. While the discrepancy is in InForm, Oracle DMW users can apply comments and categories, which are never sent to InForm.

In addition, discrepancies on InForm data, whether raised in InForm or in DMW:

- Can be answered only in InForm. An Oracle DMW user cannot change its state to Answered. An InForm user can answer the query either by changing the underlying data point or by providing explanatory text. InForm then changes the state to Answered and exports the update to Oracle DMW.

- Can transition to a state of Cancelled only if there has been no data change in InForm.
- Can be closed in either system. The change is then propagated to the other system.

Communication with InForm

State changes made in Oracle DMW are sent to InForm almost immediately using a web service.

Changes made in InForm are updated in Oracle DMW in the next data load from InForm for the study.

Actions

Users apply *actions* to discrepancies to move them from one state to another. The system comes with a set of predefined actions, and you can create your own custom actions. An action includes:

- **Start and Result States**: An action is available to a user only for discrepancies that are in the action's defined start state. When the user applies an action to a discrepancy, the discrepancy's state changes to the defined result state. If you create custom actions, the start and result states must constitute an allowed transition (see "Discrepancy States and Allowed Transitions" on page 7-1) or be the same. If the start and result states are the same, they normally have different start tags, result tags, or both.

  Note: No defined *action* is required to create a discrepancy. Users and validation checks can create discrepancies in either of the two valid start states, Candidate or Open.

- **Start and Result Tags** (Optional): Tags can be used to allow a dialog between users without changing the state or to effectively create a substate. Users can filter records in the Discrepancies page based on tag values.

  A tag is a label applied to a discrepancy when a user performs an *action* that specifies it. An action with a start tag specified is available to a user only for discrepancies that have that tag.

  When the user applies an action to a discrepancy, the system applies the action's defined result tag (if there is one) to the discrepancy.

- **Enabled**: This attribute determines whether an action is available for use. You can use this attribute, for example, to ensure that users have access only to the custom actions you define, and not to the predefined actions.

  Each time a user applies an action, the system maintains an unmodifiable audit trail of the event.

  Note: Users with access to the Discrepancies page can add comments to a discrepancy at any time. It is not necessary to apply an action in order to add a comment.

Out-of-the-Box Workflow

The system comes with a set of actions that users can apply to move a discrepancy from one state to another and, in most cases, apply a tag to the discrepancy.

Predefined Actions

Table 7–1, " Predefined Actions Available at each Discrepancy State" lists all predefined actions. The system displays only those that are valid given the current state of the selected discrepancy. For example, although there are three predefined actions named Close, only one of them is displayed to the user at a time, depending on the current state of the discrepancy the user has selected; either Candidate, Open, or Answered.

For each action named Close the system applies a different tag, depending on the start state:

- Discrepancies that started in the Candidate state get the tag ClosedAsIs.
- Discrepancies that started in the Answered state get the tag ClosedByAnswer.
- Discrepancies that started in the Open state get the tag ClosedByDataChange.

  Only the appropriate action is available to the user.

> **Note:** None of the predefined actions specifies a starting tag. They can be applied to a discrepancy in the appropriate starting state with any tag or with no tag.

Table 7–1 Predefined Actions Available at each Discrepancy State

Start State	Action Name	Routing Operation	Result State	Result Tag
Candidate	**Open**	None	Open	None
Candidate	**Cancel**	None	Cancelled	None
Candidate	**Open in InForm**	SendToInForm	Open	SentToInForm (This action creates a new Open InForm query.)
Candidate	**Send to InForm**	SendToInForm	Candidate	SentToInForm (This action creates a new Candidate InForm query.)
Candidate	**Close Discrepancy**	None	Closed	ClosedAsIs
Candidate	**Needs DM Review**	None	Candidate	NeedsDMReview
Candidate	**Send to Spreadsheet**	Send to Spreadsheet	Open	None
Open	**Cancel**	None	Cancelled	None
Open	**Send to InForm**	Send To InForm	Open	Sent To InForm (This action creates a new Open InForm query.)
Open	**Send to Spreadsheet**	Send To Spreadsheet	Open	Sent to Spreadsheet (This action generates a .csv file in the location you specify that includes all selected discrepancies. The file can then be sent to the lab.)
Open	**Needs DM Review**	None	Open	NeedsDMReview

Table 7–1 (Cont.) Predefined Actions Available at each Discrepancy State

Start State	Action Name	Routing Operation	Result State	Result Tag
Open	**Answer**	None	Answered	Answered By User Response (if a user uses the action Answer) Answered By Data Change (if a data change occurred in the source system)
Open	**Close**	None	Closed	Closed By Data Change (if a data change occurred in the source system)
Answered	**Reopen**	None	Open	None
Answered	**Close**	None	Closed	Closed With Answer

Although Table 7–1 lists all valid actions based on the applicable state for a discrepancy, not all the actions are valid for every discrepancy in that state. The system automatically filters available actions based on the following factors:

- Existence of the specified **Start Tag** value, if any.
- The data source; for example, the **Send to InForm** action is not available for discrepancies on lab data.
- When a discrepancy has been sent to an external system, no actions are available in Oracle DMW.

Out-of-the-Box Workflows

Using only predefined actions, you still have many options:

- You can choose to create discrepancies in either the Candidate or Open state.

> **Note:** InForm queries are imported in their InForm state; you cannot change that.

- You can send either Candidate or Open discrepancies to their source—lab or InForm—for review, or require manual review in Oracle DMW first.
- You can require a formal medical review of each discrepancy or not.
- You can require a formal data management review of each discrepancy or not.
- You can set up validation checks so that when they run, they automatically close discrepancies they created when the underlying data is corrected in such a way that the validation check logic no longer sees the data as discrepant.
- Alternatively, you can require manual review of validation check-created discrepancies whose underlying data has been changed.

You can disable any actions you do not want to use.

Lab Workflow Example

1. A validation check with Autoclose set to Yes detects several lab test results that are out of range, and creates a discrepancy for each one. These discrepancies are displayed in the VC Listings page under the name of the validation check and in the Discrepancies page.

2. In the Discrepancies page, the data manager clicks **Send to Spreadsheet** to export the records to a file on her personal computer and sends the file to the lab.
3. Lab personnel open the file in Microsoft Excel and check the test result values against the original source data. In Excel, they fix the data for all but one of the discrepancies, ready for the next data load back into Oracle DMW.

   The lab personnel also send information separately about how the remaining discrepancy should be handled, and the data manager updates the discrepancy manually in Oracle DMW.
4. The next time the validation check runs, it automatically closes the discrepancies on the corrected data and makes no change to the other discrepancy.

InForm Workflow Example

1. Using Custom Listings, a data manager detects several BMI values that are out of range, creates a discrepancy for each one, and sends the discrepancies to InForm as Open queries.
2. The CRA checks all the data in InForm and is able to resolve all queries by correcting one data point for each one. The CRA then sets all queries in InForm to Answered.
3. The InForm Connector detects the data changes and loads the updated data and queries (discrepancies) into the correct study InForm data model. In DMW the discrepancies will be updated to have the same state as in InForm and the discrepancy history will include an "Updated from InForm" entry.
4. In the Discrepancy page, the data manager looks for discrepancies in the Answered state and closes them. The system sends the update to Closed state to InForm to close the queries there.

Creating a Custom Workflow by Creating Custom Actions

Your company's standard operating procedures may require additional or different workflow steps. You can create a set of actions to suit your company's procedures—within the predefined transition rules—and disable any shipped actions you do not need. If you need more states, create tags to serve as substates and create actions that specify a starting and resulting tag.

Custom InForm Workflow Example

In the case of the out-of-range BMI values:

1. The CRA responds that the weight, height, and units for one of the subjects are all correct and the patient is extraordinarily heavy. The discrepancy when imported back into the system has a state of Answered with a tag Answered by User Response.
2. The data manager wants to send the discrepancy for medical review to determine if the subject qualifies for participation in the study. This requires a custom action:

Table 7–2 Example of Custom Actions

Start State	Custom Action	Routing	Start Tag	Result State	Result Tag
Answered	**Answered but Requires Medical Review**	None	AnsweredByUser Response	Answered	Needs Medical Review
Answered	**Send to DM**	None	Needs Medical Review	Answered	Remove Subject from Study
Answered	**Remove Subject**	Send to InForm	Remove Subject from Study	Answered	Remove Subject from Study

3. The medical reviewer uses another custom action, Send to DM (with a Start State of Answered) to respond to the data manager that the subject is not eligible for the study and should be removed, leaving the discrepancy state set to Answered.
4. The data manager uses a custom action, Remove Subject, to send the discrepancy back to InForm instructing the CRA to remove the subject from the study.
5. The CRA removes the subject from the study and sends a comment back informing the data manager.
6. The data manager uses a predefined action to close the discrepancy.

Custom Lab Workflow Example

Table 7–3 lists custom actions that allow a custom workflow between data management and medical review teams. In this case, company policy is for all lab test discrepancies to go through medical review first.

When a validation check creates a discrepancy on lab data, it gives the discrepancy a state of Candidate. The data manager reviews the discrepancy and applies a custom action that applies the tag Needs Medical Review while leaving the discrepancy state as Candidate. The medical reviewer then either applies the Needs DM Review tag, with a result state of Open, or the Send to Spreadsheet action and then sends the discrepancy to the lab.

Table 7–3 Examples of Custom Actions for Workflow between Data Management and Review Teams

Start State	Custom Action	Routing Operation	Start Tag	Result State	Result Tag
Candidate	**Answer Data Management**	None	NeedsMedReview	Answered	MedAnswered
Candidate	**Assign to Data Management**	None	NeedsMedReview	Open	MedResponded

Creating and Editing Actions

To create new actions or edit the predefined ones, you must use public APIs. Information is included in the *Oracle Life Sciences Data Hub Application Programming Interface Guide*You can define or modify the following attributes:

- **Name**: The action's name does not appear in the user interface. It is required and the combination of Name and Start State must be unique in the database.

  For example, you can have two actions with the name XYZ, one with a Start State of Open and the other with a Start State of Candidate, but you cannot have two actions named XYZ with a Start State of Open.

- **Menu Label**: Enter text to appear in the user interface. This is what the user will see and apply to a discrepancy. The label is required but does not need to be unique.
- **Start State**: Select the state that discrepancies must have for this action to be available to a user. This action will be available to users only for discrepancies that are in this state.
- **Result State**: Select the state that the system should apply to discrepancies when the user applies this action. The result state must be an allowed transition state from the start state; see "Discrepancy States and Allowed Transitions" on page 7-1.
- **Start Tag** (Optional): Select the tag that discrepancies must have for this action to be available to a user. You can create new tags in the Administration page; see "Viewing and Creating Discrepancy Tags" on page 8-10. Tags can be used as substates, for communication within a state, and as filters in the Discrepancies page.
- **Result Tag** (Optional): Select the tag that the system should apply to discrepancies when the user applies this action, if any.
- **Routing Operation** (Optional): If the discrepancy needs to be checked or resolved in the source data system, select the appropriate operation:
  - **SendToInform**: The system sends the discrepancy to InForm immediately and applies a transition restriction of Limited to it.
  - **ExportToCSV**: The system creates a .csv file suitable for export to a lab as is or as an Excel spreadsheet.
- **Enabled**: Select this attribute to make this action available for use. Deselect it to prevent the action's use.

8

Administration

The following administration tasks are done in the Oracle Health Sciences Data Management Workbench (Oracle DMW) user interface:

The following administration tasks are done outside the Oracle DMW user interface:

See also the *Oracle Life Sciences Data Hub System Administrator's Guide* for additional administrative tasks in the Oracle Life Sciences Data Hub, including the chapters on services and profiles.

Viewing and Setting Up Lab Data Sources

In the Administration page, you can create a list of all the labs or other entities that send files to be loaded into the system. This list populates the list of values for File Watcher data sources. You can view the existing file data sources in the Lab tab.

Adding a Lab Data Source

To add a data source:

1. Go to the Administration page, click **Data Sources**, click **Lab**, and then click the Add (+) icon.
2. The system uses only the value of the data source name. All other fields are for your own records.
   - **Data Source Name**: Enter the name of the data source; for example, `Central Labs`. This value appears in the list of data sources in the File Watcher configuration tab for clinical data models during study configuration.
   - **Description**: Enter a description of the data source.
   - **Contact**: Enter the name of the person who should receive discprepancies.
   - **Address**: Enter the address of the lab or other data source.
   - **Contact Number**: Enter the telephone number of the contact.
   - **Email Address**: Enter the email address of the contact.
   - **Send All Open Discrepancies?**: This field has no effect in Release 2.3.1.

Modifying a Lab Data Source

To change any part of a lab data source definition, click the Edit (pencil) icon, make the changes you need, and click OK. See "Adding a Lab Data Source" on page 8-1 for details about each field.

Removing a Lab Data Source

To remove a lab data source, select it and click the Remove (X) icon.

Viewing and Setting Up InForm Data Sources

In the InForm tab of the Administration page you can define remote locations and web service locations for InForm studies. These definitions are available for use in the InForm Configuration tab of the Study Configuration page. They can be created, edited, and deleted there as well as here.

Adding a Remote Location

You must information about the InForm reporting database for the study:

1. Enter a value for:
   - **Name**: Enter a name for the InForm database. Do not use spaces, slashes, or special characters other than underscore (_).
   - **ConnectString**: Enter the text for the Using clause of the Create Database Link SQL statement. This is normally the same as the Description Clause of a TNSNAMES definition; for example:

     ```
     (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=host_name.company_
     domain.com)(PORT=1521))(CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=trial_
     name.world)))
     ```

     Note: No spaces are allowed.

- **Username**: Enter the name of the Oracle DMW read-only access account that has been set up in the InForm database for this remote location. See the section "Create Oracle Accounts for the DMW InForm Connector" in the *Oracle Health Sciences Data Management Workbench Installation Guide* for further details. The system will use this account to connect.
- **Password**: Enter the password required for the same user account. The system changes and encrypts the password after creating the database link.

  In the unlikely event that you need to change the password, your InForm database administrator must change the password for the Oracle DMW read-only access account. You can then select **Enable Password Entry?** in the Edit Remote Location window and enter the new password. The system then recreates the database link and changes and encrypts the password.

2. Click **Test Connection** to make sure it is set up correctly. The system returns a success or failure message above the Test Connection button.
3. Click **OK**.

Adding a Web Service Location

1. Enter a value for:
   - **Name**: Enter a name for the web service location.
   - **WSDL URL**: Enter the web service URL for the InForm Adapter's Enhanced Discrepancy Interface; for example:

     `http://your_InForm_Adapter_Server_Name/InFormAdapter/Discrepancy/DiscrepancyService.svc?wsdl`

   - **Trial Name**: Enter the study name as defined when the trial was registered in the InForm Adapter; see the *Oracle Health Sciences InForm Adapter Installation Guide* for details.
   - **Username**: Enter the name of the account set up for authenticating Oracle DMW web service transactions. The default name is DMW_AUTH. See the section "Create Users in InForm" in the *Oracle Health Sciences Data Management Workbench Installation Guide*.

     Enter the name of a valid user account on the web service location. The system will use this user name to connect.
   - **Password**: Enter the password required for the same user account. The system encrypts the password.
2. Click **Test Connection** to make sure it is set up correctly. The system returns a success or failure message above the Test Connection button.
3. Click **OK**.

Removing a Remote Location or Web Service Location

To remove a remote location or web service location, select it and click the Delete icon (the red X).

Modifying a Remote Location or Web Service Location

To modify a remote location or web service location, select it and click the Edit icon (the pencil). Modify values as required; see "Adding a Remote Location" on page 8-2 or "Adding a Web Service Location" on page 8-3.

Setting Up File Watcher for the Instance

This section contains the following topics:

Setting Up Root Folders for File Watchers

You must create six folders, one for each combination of file type—SAS or text—and lifecycle—Development, Quality Control, and Production—to be used by all Study File Watchers. Oracle recommends using a naming convention that includes the file type and lifecycle mode so people can be sure they are putting data files in the right place.

These six folders do not need to be in the same folder or even on the same machine.

Both the File Watcher service(s) and the DP Server(s) must have access to these folders; for example, you can set up an NSF mount of the file system to each machine where one of these DP Servers is installed.

In addition, you must register their locations in the Oracle LSH profiles listed below; see the *Oracle Life Sciences Data Hub System Administrator's Guide* for instructions.:

- `DME:FWR_ROOT_FOLDER_TEXT_DEV` for text files in the Development life cycle area
- `DME:FWR_ROOT_FOLDER_TEXT_QC` for text files in the Quality Control life cycle area
- `DME:FWR_ROOT_FOLDER_TEXT_PROD` for text files in the Production life cycle area
- `DME:FWR_ROOT_FOLDER_SAS_DEV` for SAS files in the Development life cycle area
- `DME:FWR_ROOT_FOLDER_SAS_QC` for SAS files in the Quality Control life cycle area
- `DME:FWR_ROOT_FOLDER_SAS_PROD` for SAS files in the Production life cycle area

Note: You must define all six folders for File Watcher to work. However, if you do not need one or more of them for a particular study or even for any studies, you do not have to use them.

Changing the Root Folder Location

Changing the root folder location is disruptive to the flow of work in DMW and should only be done with caution. However, you can modify the location if necessary:

1. Create one or more new root folders in a new location.
2. Change the value of the LSH profile to the new location.

The system then:

- Migrates all existing Study File Watchers to the new root folder(s).
- Does **not** move existing files to the new location.

You must determine which files have not been loaded, if any, and move them to the new location.

Any files already submitted for data load at the time of the change will be processed.

Setting Up Study Watched Folders

For each Study, define a Study Folder Name; see "Creating and Editing Study File Watchers" on page 8-6 and create a subfolder of that name under each of the six file type/lifecycle area combination root folders; see "Setting Up Root Folders for File Watchers" on page 8-4.

The six root folders should contain one subfolder for each study in the DMW instance. Study folder names must be unique across the DMW instance.

Setting Up the File Watcher Service

The File Watcher service detects files that have been placed in watched folders so the system can then load them into DMW. Instructions for setting up the File Watcher service are included in the *Oracle Health Sciences Data Management Workbench Installation Guide*.

The File Watcher service must have access to the root folder; the root folder must be located either on the same machine or NFS mounted to the machine.

You can have one or two File Watcher services:

- If you are using the same machine for processing both SAS and text files, you need only one service.
- If your SAS and text data loading installations are on separate machines that do not share file systems, you need a File Watcher service on each machine.

Setting Up the Distributed Processing Server for File Watcher

The operating system account that runs the Distributed Processing (DP) Server, which runs the File Watcher Service, must have read, write, and delete privileges on the files to be processed by File Watcher, including the root folder and any study folders that contain data files.

The Distributed Processing (DP) Server communicates with the database and processing engine to run data load jobs. You can use DP Servers on multiple machines, not necessarily just the one or two computers where the File Watcher service is installed. The DP Server also requires access to the root folder for the data files to be loaded.

Define Service Locations and Services

For each machine where you have installed SAS or SQL*Loader to run SAS and text data loads, respectively, you must define the following in Oracle LSH:

- One Service Location for the computer.
- One or more Services for the Service Location of the following types:
  - **SAS** to load SAS files
  - **Text for SQL*Loader** to load text files

Each Service has a priority assigned. If your data loads will have different priorities, you should define one Service for each priority (Low, Normal, or High) you may need. For each Service you must also specify the number of Service Instances to create. You should specify a number equal to the number of data loads of the same priority that "Setting Up Root Folders for File Watchers" on page 8-4may run at the same time.

You can define DP Service Locations and Services on other computers with access to the machine where the root folders are located and use those services to run data load jobs as well; see "Selecting the Distributed Processing Server" on page 8-7.

Start the DP Server Service

In addition, you must start the DP Server service on the machine, entering values for the following parameters related to File Watcher:

- *FW_ENABLED* Set to **Yes** to start the File Watcher Service or **No** if you are not using Oracle DMW.
- *FW_FREQ* Refresh frequency in seconds. This value specifies the minimum interval between requests to the database to check if there is a new set of Watcher Configurations. This value cannot be set lower than 60 seconds. A high setting will result in a delay between the user's addition or adjustment of a Watcher Configuration in Oracle DMW and the changes' taking effect in file detection behavior.
- *FW_POLL* Polling frequency in seconds. The polling frequency represents the minimum interval at which a File Watcher Service may run to detect if there are any files in the watched location that should be loaded into Oracle DMW. The minimum value permitted is 60 seconds.

You define DP Service Locations and Services in the Oracle LSH user interface and start the service in UNIX. See the *Oracle Life Sciences Data Hub System Administrator's Guide* for more information.

Setting Up Study File Watchers

This section contains the following topics:

After the administrator creates the Study File Watchers, the study designer or data manager must then configure File Watchers and File Specifications for each clinical data model in the study that loads data from files; see "Configuring File Watcher" on page 3-20 for more information.

You can suspend and resume File Watchers; see "Restarting Watchers that Are Not Working" on page 8-8 and "Suspending and Resuming Automatic Data Loading" on page 8-9.

Creating and Editing Study File Watchers

For each study you must specify a single folder name to be used for loading data from files. The system uses this folder name to create six study File Watchers, one for each combination of the two possible input file types—SAS and text—and each of the three lifecycle stages—Development, Quality Control, and Production. Each Watcher watches a different directory path ending in a folder of the name you specify here.

You must create the actual directories with this folder name in the six locations you specify in the LSH profile values; see "Setting Up Root Folders for File Watchers" on page 8-4.

Edit You cannot change the study or folder name. You can change the file deletion period.

Create To create study File Watchers:

1. In the Administration tab, click File Watcher in the left pane. The Watcher Listing window opens.
2. Click the Add (+) icon. The Create Study File Watcher window opens.

> **Note:** You will receive an error if you try to create Study File Watchers before the root folders are defined in the LSH profiles.

3. **Study**: Select the name of the study from the drop-down list. The list includes studies with no existing Study File Watcher.
4. **Study Folder Name**: Enter a valid UNIX case-sensitive directory name with **no** underscores (_) to be used as the final directory of the Watched Folder path after each file type/lifecycle area root folder. The Study Folder Name must be **unique within the DMW instance** and you will receive an error if the name you enter is already in use. You can see existing folder names on the Watcher Listing pane of the Administration page.

   You must create six directories matching this name; see "Setting Up Study Watched Folders" on page 8-5.
5. **Delete Files After**: Enter the number of days after the load date when the file should be deleted. If the file cannot be loaded because it does not meet the file specifications for the folder, the system deletes the file this number of days after the file detection date.

   Files are permanently deleted. If you do not want your files deleted, enter a very large number. However, Oracle recommends archiving your files in a different secure location and keeping them in the watched folders for a minimal amount of time after they have been loaded.
6. Click **OK** to save a new set of study Watchers or **Regenerate** to save changes to existing ones.

Selecting the Distributed Processing Server

Specify the Distributed Processing (DP) Service Location(s) where the the File Watcher service is installed. These settings apply to all the Study File Watchers for the Oracle DMW instance.

1. In the Watcher Listing pane of the Administration page, click **DP Server**. The DP Servers for File Watcher window opens.
2. **Text DP Server**: From the drop-down list, select the DP Service Location you want to use to detect text data load jobs. Both the DP Server and the SQL*Loader processing engine must have access to the root folders for the text file type.
3. **SAS DP Server**: From the drop-down list, select the DP Service Location you want to use to detect SAS data load jobs. The DP Server and the SAS processing engine must have access to the root folders for the SAS file type.
4. Click **OK** to save.

Monitoring File Watchers

Click File Watchers at the bottom of the right-hand pane in the Administration page to see the Watcher Listing page.

This section contains the following topics:

Viewing File Watchers

The Watcher Listings page displays all File Watchers in the DMW instance with their current status, including six for each study, one for each file type—SAS or text—and lifecycle—Development, Quality Control, and Production—combination.

- **To see all Watchers for a single study**, enter all or part of the study name in the blank field above Study Name and press Enter.
- **To see if a Watcher is for text or SAS files**, display the File Type column. In the **View** drop-down list, select **Columns**, then **File Type**. Columns displaying the **data load parameter values** are also available.
- **To see a Watcher's File Specifications**, select the Watcher in the Watcher Listing pane. The system displays its File Specifications in the File Specifications pane.

> **Note:** This displays the File Specifications from all Data Models for the study that match the file type and lifecycle for the selected Watcher.

For information on each field, see "Configuring File Watcher" on page 3-20. To see the list of data load jobs for a study, their end status and a link to their log file, go to the Home page, Data Loads tab.

See also "Changing User Interface Display" on page 10-7 and "Querying By Example" on page 10-9.

Stopping and Starting File Watchers

In this window you can:

Restarting Watchers that Are Not Working

To detect files and load data, the Watcher's status must be Running and the Distributed Processing (DP) Server status must be Online. If not:

- **Resume the Watcher File Specification**: You can resume a Watcher or one of its File Specifications that has been suspended by selecting it and clicking **Resume**.

> **Note:** If the DP Server status is Offline, the Watcher cannot run, even if its status is Running.

- **Restarting the DP Server**: If the DP Server status is Offline you can restart it using instructions in the *Oracle Life Sciences Data Hub System Administrator's Guide* chapter on stopping and starting services and queues.

Suspending and Resuming Automatic Data Loading

If you have set up scheduled data loading and need to stop it temporarily; for example, because a lab is sending files using the wrong format, you can suspend loads for the Watcher or a single File Specification and resume them when the issue is resolved.

To conserve system resources, suspend data loading for all of a study's File Watchers at the end of a study.

To stop or start a File Watcher:

1. Go to the Administration page and click File Watcher in the left pane. The Watcher Listing window opens. If necessary, click the Filter icon to query for the Watcher by its study or status.
2. Select the File Watcher you want to stop or start.
   - If it is currently stopped, click the **Resume** button to start it.
   - If is it currently running, click the **Suspend** button to stop it.

   Or, to start or stop only one of the filename patterns a Watcher looks for, select the File Specification you want to stop or start.
   - If it is currently stopped, click the **Resume** button to start it.
   - If is it currently running, click the **Suspend** button to stop it.

Viewing and Creating Categories

A category is a label that can be applied to discrepancies—either by a validation check or by a user directly in the Discrepancies page—or to validation checks or flags. If a category is applied to a validation check, the validation check automatically applies the category to all discrepancies it creates.

A particular discrepancy, validation check, or flag can have only one category. Users can filter these objects by category.

Categories allow you to group flags by which data they are appropriate for. For example, the InForm flags imported with InForm queries are available only to data that originated in InForm. You can create other flags for lab data, for example, by linking categories to clinical data models of type Input and subtype File. In Release 2.3.1 linking categories to model types and subtypes is possible only by using a public API; see the *Oracle Life Sciences Data Hub Application Programming Interface Guide*.

You define any categories you need; for example Range, Date Alignment, CTC Grading Rules, Preprocessing.

To create a category:

1. Click **Categories** at the bottom of the right-hand pane in the Administration page, then click the Add (+) icon.
2. Enter values in the following fields:
   - **Category Name**: The label is displayed in the user interface as a filter and in all places where a user can read or apply it.
   - **Description**: (Optional) Enter a description explaining the intended use of the category. The maximum length is 500 characters. The description is displayed only on the Administration page.
   - **For Discrepancies?**: Select this option to allow users to assign the category to discrepancies.

- **For Validation Checks?**: Select this option to allow users to assign the category to validation checks. All discrepancies created by validation checks assigned to the category will also be assigned to the category.
- **For Flags?**: Select this option to allow users to assign the category to flags.

Note: Flags are applied to data records, not discrepancies. Flags appear in the user interface as available to assign to a particular record only if they are assigned to a category that has been mapped to the type of data model the record is in. For example, the predefined InForm flags belong to the category InForm Flags which has been mapped to the clinical data model type Input and subtype InForm. You can create a category for flags appropriate to data in models of type Input and subtype File, or models of type Target.

In Release 2.3.1 you must use public API createModelFlagcatMap to map a category to a clinical data model type or subtype. See the *Oracle Life Sciences Data Hub Application Programming Interface Guide* for more information.

You can map the same category to more than one model type or subtype; for example, you might have flags that you wanted to assign to input data no matter whether it was from InForm or a lab. You could create a category called Input Data Flags and call the API twice to assign it to both subtypes of Input models. A model type or subtype can have more than one category mapped to it.

---

3. Save. The system assigns an ID to the category that you can see in the Category Listing page.

Viewing and Creating Discrepancy Tags

A tag is a label that a user can apply to one or more discrepancies with an *action*. A discrepancy can have only one tag at a time. Tags have several uses:

- Users can filter the data they view in the Discrepancies page by tag.
- You can use tags to effectively create discrepancy substates. Define actions whose start and end state is the same—for example, Open—but in which one applies a tag as a result tag and the others specifies the same tag as its start tag.
- You can use tags to direct communication among teams while the discrepancy remains in the same state.

The system comes with a set of predefined system tags and actions; see "Out-of-the-Box Workflow" on page 7-4 for a list of system tags as well as the system actions that apply them. You can change the name of system tags and enable or disable them. You can also create custom tags and custom actions, and you can use custom actions to apply system or custom tags.

Tags are available in all studies.

Creating a New Tag

To create a tag:

1. Click **Tags** at the bottom of the right-hand pane in the Administration page, then click the Add (+) icon..

2. Enter values in the following fields:
   - **Tag Name**: This text will be visible in the Discrepancies page. Spaces and special characters are allowed. The maximum length is 100 characters.
   - **Description**: Enter a description to help users decide if they should apply the tag.
   - **Enabled?**: Check to allow the tag to be used.
3. Click **OK**.

Modifying Existing Tags

All existing tags are displayed in the Tag Listing page. The System Tag? column check box is selected if the tag is predefined and shipped with the system and unchecked if your company created the tag. User-defined tags are listed first. All tags have a system-generated ID in the first column.

You can edit each tag, including system tags, as follows:

- **Tag Name**: Any changes you make here are reflected in the user interface wherever the tag name is displayed.
- **Description**
- **Enabled**: Select this check box to allow the tag to be used when creating or updating actions. Deselect it to prevent its use, except in existing actions that already use the tag.

Creating and Using Flags

Flags are an important data review tool. Flags can be applied to records (not discrepancies) by validation checks and by users in the Listings page.

The system imports InForm flags with the InForm records they are assigned to in InForm. These flag assignments cannot be changed in Oracle DMW, but you can use them to analyze records and apply custom flags.

You can create any number of custom flags, each with any number of states, for use within the system. These flag assignments are not sent to InForm. A record can have multiple flags applied, each with a single state.

You can define flags for two basic purposes:

- The clinical data review process
- Tracking subject visit completeness

For a validation check to apply a flag to records it must call the Flag API from a custom program. You must write the program in either SAS or PL/SQL and create a Program definition in Oracle LSH to store and version it, then associate the program with a validation check; see "Creating a Custom Validation Check" on page 6-7 for more information.

Each flag you define has a Subject Visit attribute. If set to **Yes**, the flag should be applied only to records in the Subject Visit table. You can use Subject Visit flags to track the state of data completeness and cleanliness for each subject visit. To do this, define clinical record flags to track completeness and cleanliness (number of discrepancies) for each CRF record in other tables, write a program to analyze records and apply those flags, and write a program to aggregate these values to set the Subject Visit flag for each subject visit.

Clinical Record Flag Examples You could create the following flags and accompanying custom programs to help track the review process of individual subject data records:

- Create a flag called **Record Complete** with two states: Yes and No. Write a custom program to ascertain that all required fields in a record have a value and apply the flag with the state Yes if they do. You can use this flag to filter for missing data and to calculate Subject Visit completeness.
- Create a flag called **Validation Checks** with the states Validation Not Done, Validation Incomplete, andValidation Complete. Write a custom program that compares the timestamp of the last data update for each unlocked record to the execution timestamp for all validation checks run for that study and sets the **Validation Checks** flag for each record with the appropriate state.
- Create a flag called **Number of Discrepancies** with the states 0, 1, 2, 3 and so on. Write a custom program to count the number of discrepancies in a record and set the state to the corresponding number.
- Create a flag called **Data Readiness** with states: Not Clean, Clean for Medical Review, and Clean for Analysis. Write a custom program that uses the above flags or others to set its values.

Subject Visit Flag Example Create a subject visit flag called **Subject Visit Complete** with a single state of On. Write a custom program based on aggregating flag values for the individual clinical record flags in the examples above: loop through each subject/visit combination and set the **Subject Visit Complete** flag to On when every record for the current subject and visit in the Adverse Events, Concommitant Medications, and Vital Signs tables (for example) has the **Record Complete** flag set to On, the **Validation Checks** flag set to Validation Complete, and the **Number of Discrepancies** set to an acceptable number.

Creating a Flag

To create a flag:

1. Navigate to Administration, then Flags, then click the **+** icon.
2. Enter values in the following fields:
   - **Name**: Enter the text you want to appear in the user interface for the flag; in the examples above: Record Complete, Validation Checks, or Data Readiness. The maximum length is 80 characters.
   - **Description**: Enter a description or explanation of the purpose of the flag. The maximum length is 256 characters.
   - **Category** (Mandatory): Select a single category for the flag. You can define the list of categories if you have the required privileges; see "Viewing and Creating Categories" on page 8-9.
   - **Subject Visit Flag?** Select this option if the flag is to be applied to records in the Subject Visit table. Leave it deselected (the default value) if the flag is to be applied to clinical data records.
   - **User Settable?** If selected, users can set this flag in the Listings Pages. If not selected, the flag is imported from InForm and cannot be changed in Oracle DMW. All flags you create have this attribute selected.

- **States**: You can define any number of mutually exclusive states for each flag. You can delete this state and add others, but the flag must have at least one state or it cannot be applied to records.

3. Click OK.

Flag Rules

The following rules apply to flags:

- A record can have any number of flags applied to it.
- A record can have only one state of each flag applied to it at a time.
- The system does not enforce any rules concerning the transition of a record from one flag state to another.
- A user can apply a flag to a record once and change the state any number of times. The user cannot apply a flag to the same record more than once at a time.
- A user can clear a flag's association with a record. No history of the flag's previous association is preserved.

Comparing Flags, Tags, and Categories

The following table compares flags, tags, and categories. All are available for use in any study.

Table 8–1 Flags, Tags, and Categories

Object	Applied To	Applied By	Used as Filter?	Has Multiple States?
Flag	Records	User, if flag is user-settable; otherwise, system.	No	Yes
Tag	Discrepancies	User, via actions	Yes	No, but can be used to create discrepancy substates
Category	Discrepancies, Validation Checks, Flags	Validation Check or User	Yes	No

Setting Up Security

This section contains the following topics:

About Security

Studies, data models, transformations, and validation checks are all *objects*. Users are allowed to perform an operation on an object when they:

- belong to a user group that is assigned to the object either explicitly or by inheritance
- and are assigned to a role within that user group that allows the operation on the object

Oracle Health Sciences Data Management Workbench (Oracle DMW) is installed on top of Oracle LSH and uses its security system, which is itself based in part on the Oracle Applications (Oracle E-Business Suite) security system.

The following tasks are done in the Oracle LSH and Oracle Applications user interface and are documented in the *Oracle Life Sciences Data Hub System Administrator's Guide*.

- **Assign specialized administrative roles to a few users** to create Domains to contain and organize studies, grant blind break and unblind privileges, and add users to user groups.
- **Create user accounts** for all users, including application roles for all users.
- **Create user groups**.
- **Assign object security roles to user groups**. You can use shipped roles created for use with Oracle DMW or modify them as required; see Appendix B, "Predefined Roles.".
- **Assign users to object security roles within user groups**.
- **Assign user groups to objects in Oracle DMW**. This is the only task that is performed in the Oracle DMW user interface; see "Applying Security" on page 3-27.

Simple Security Setup

Using the predefined object security roles and predefined application roles as they are, you can set up a security system as follows:

- Create a single user group and assign all the predefined roles to it in the Oracle LSH user interface. The predefined roles all begin with "DMW" and are described in Appendix B, "Predefined Roles." Object security roles determine which actions a user can take on which objects.
- Create a user account for each user and give each user account the appropriate Oracle DMW application role. In some cases a user may need two roles; for example, developers who should be able to create library data models and code lists as well as study objects. Application roles control access to Oracle DMW user interface pages.

  Blinding-specific application roles are required to view sensitive, currently blinded data (Blind Break) and to unblind data (Unblind) so that people with lesser blinding-related privileges (Read Unblind) can view sensitive data that is no longer blinded; for example, at the completion of a study. The predefined roles all include blinding-related privileges on objects (table instances), but both this object security and the corresponding blinding-related application role are required for a user to access blinded data.
- Assign the user group to the study as a whole in the Oracle DMW Home page by selecting the study in the list of studies, then clicking the Security key icon at the top of the list of studies; see "Applying Security" on page 3-27. Assign the user group successively to Metadata, Development, QC, and Production. The object security roles control which users can perform which operations on objects in the each lifecycle area.
- Assign the user group to the InForm Family adapter domain in Oracle LSH; see the *Oracle Life Sciences Data Hub System Administrator's Guide*.

Note: In the rest of this document this adapter is referred to as the InForm Connector to distinguish it from the InForm Adapter that is a separately installed product whose purpose is to integrate InForm with other products. The adapter to which you assign user groups is shipped as part of Oracle LSH and Oracle DMW for the purpose of integrating Oracle DMW with InForm.

- Object security privileges on library models and code lists are not included in any of the predefined roles. In order to allow some users to create library models and code lists you can create a role in Oracle LSH with these privileges and assign it to the same or a different user group, and assign the user group to the study category domain that contains the study. The users will then be able to create library models and code lists available in all studies.

Custom Security Setup

You can customize your security setup in many ways:

- Modify the shipped object-security roles or create entirely new roles
- Create multiple user groups with different roles
- Assign user groups to objects within a study rather than the entire study; see "Object Ownership" on page A-6 to help you understand what the possibilities and effects are.
- Create roles with privileges on library models and code lists and use them as in the "Simple Security Setup" on page 8-14.

Setting Up Library and Study Categories

Oracle DMW uses Oracle LSH domain objects as containers to hold studies, study templates, and clinical data models and code lists intended for reuse.

When a user creates something that is stored in a library—a study, library model, or code list—or when they create a study model based on a library model, he or she must first select a library.

By default, the label in the user interface is "Therapeutic Area" so that you can categorize studies and models by therapeutic area. However, you can configure this label to be whatever you want, and you must create the corresponding values—libraries—for the user to select; therapeutic areas or other categories.

For example, if you use the default categorization of Therapeutic Area, you need to create therapeutic areas to serve as libraries, such as Cardiology and Immunology, including all that your company works on. Alternatively, you could categorize by Drug Program or Drug.

You must decide how to categorize and name your libraries. Consider:

- You can create only one flat list of values. So if you want to categorize all studies by drug as well as therapeutic area you must create categories such as "Cardiology Drug 12345" and "Cardiology Drug 678910."
- If you want to create a library of data models that are generic enough to be used in multiple therapeutic areas, you might want to create a category called "Generic."
- When a user creates a new study or library data model and selects the name of the library, the selected library becomes its namespace. This has implications for object

security; you can assign a user group to the library and by default that user group assignment is inherited by all studies, models, validation checks and transformations contained in the domain. You can revoke this inherited assignment at any level, or assign a user group at a lower level.

To configure the label and values:

Classification Label **Therapeutic Area** is the default value of the label, so if you want to classify your studies by therapeutic area, you do not need to set a new label value. If you want to use a different label, you must set profile DMW_STUDY_CLASSIFICATION_UI_LABEL as described in the *Oracle Life Sciences Data Hub System Administrator's Guide*. Your custom label then appears in the user interface in place of "Therapeutic Area."

Classification Values To create the list of values—a list of therapeutic areas or other categories—you must define a set of Oracle LSH domains in the root domain DMW_DOMAIN in Oracle LSH. These domains also serve as the namespace for studies and library clinical data models.

You can create Oracle LSH domains two ways:

- In the Oracle LSH user interface; for instructions see the *Oracle Life Sciences Data Hub Application Developer's Guide*
- By running an Oracle LSH API; for instructions see the *Oracle Life Sciences Data Hub Application Programming Interface Guide*

Setting Up Custom Program and Function Categories

To create a custom program or function for use in a transformation from one clinical data model to another or in a validation check, programmers must create a program in Oracle Life Sciences Data Hub (Oracle LSH); see "Defining a Custom Program" on page 5-17. These programs must be created in the DMW_UTILS domain in Oracle LSH which is created during Oracle LSH post-installation.

You can create domains within the DMW_UTILS domain to help users find appropriate custom programs and functions and reuse them, promoting standardization.

The process is the same as for creating domains to serve as libraries; see "Setting Up Library and Study Categories" on page 8-15.

Configuring Partitioning

When a user creates a study in Oracle DMW, he or she must specify a study size of either Small, Medium, or Large in terms of the volume of data expected, relative to other studies. The system uses this value, together with lookup values that you can modify, to determine which database partition to use for storing certain types of data in internal cross-study tables.

Partitioning is an Oracle Database feature that allows tables and indexes to be subdivided into smaller pieces and managed cost-effectively on different disk storage tiers with a finer level of granularity to improve access performance.

When a user specifies the study size and saves, the system assigns two partition IDs to the study: one for Development and Quality Control and the other for Production. Both IDs are unique across all studies in the instance.

Partitioned Tables

The tables that use partitioning do not store clinical patient data, but they do store data that is likely to be created in proportion to the volume of clinical data, especially the discrepancies table.

The system adds a column for the partition ID in all affected tables. The internal SYS_CONTEXT tracks the partition ID as well as the current study and lifecycle area. All internal queries to affected tables must include the partition ID and call an internal API to get the ID from the SYS_CONTEXT to run most efficiently.

The tables are:

- DME_CTXT_SKEYS_MAP: This table is used to trace the lineage between source and target tables used in transformations. It contains data that maps between specific source and target records. Each record in the target table of a transformation has at least one corresponding record in the DME_CTXT_SKEYS_MAP table, and for some transformations there are multiple records. For example, in a direct map there is one record per target table record; in a join of three tables, there are three record per target table record.
- DME_OPOBJ_CONTEXT_MAP: This table is used to highlight discrepancies on the Listings display. The table contains an entry for each primary discrepancy and all of its secondary discrepancies that are traced through the transformation data lineage. The secondary discrepancies are traced to data items both upstream and downstream through the connected transformations. This table is also used to obtain the primary source data item for a discrepancy.
- DME_DISCREPANCIES: This table stores discrepancy-related information such as the table, column, model, study, and lifecycle of the record on which each discrepancy is created. This table also records discrepancy state, comments, and discrepancy ID.
- DME_FLAG_DATA: This table stores flag assignments to records that that users add and modify in the Listings pages.
- DME_DISC_ACTION_HISTORY: This table stores the history of actions performed on specific discrepancies.
- DME_DISC_CSV_FILES: This stable stores sets of discrepancies exported as CSV files.

Developing Guidelines for Setting Study Size

Oracle recommends that you develop guidelines to help study designers categorize studies as small, medium, or large, and to help you in Specifying the Number of Similarly Sized Studies per Partition.

You can develop guidelines based, for example, on the number of subjects and the number and size of CRFs in the protocol.

Specifying the Number of Similarly Sized Studies per Partition

The maximum number of small, medium, and large studies using a single partition is determined by lookups called DME_PARTITION_DEVQC and the DME_PARTITION_PROD that control the Development/Quality Control partition and the Production partition respectively. The default settings for both are:

- Small studies: 20
- Medium studies: 5

- Large studies: 1

You can change these values; see the chapter on look-ups in the *Oracle Life Sciences Data Hub System Administrator's Guide* for more information.

For example, if you have a relatively small amount of data in your Development and Quality Control lifecycle areas and a huge amount of data in Production, you might change to settings like:

- For the Development/QC partition:
  - Small studies: 20
  - Medium studies: 15
  - Large studies: 11
- For the Production partition:
  - Small studies: 10
  - Medium studies: 4
  - Large studies: 2

There is a limit to the number of partitions that can be created for a database table: max=1024k-1.

Each partition can hold any number of records for each study; that is, any number of discrepancies or flags for given study.

Each lookup value must be different from the others; you cannot have the same number of small and medium studies per partition, for example.

Assignment Algorithm

Studies are assigned to partitions according to their defined size and in the order they are added.

For example, using the default value of 5 medium studies per partition, the system creates two partitions the first time a medium study is created, one for Development/QC and the other for Production, and assigns that study and the next 4 medium studies to those partitions. When the 6th medium study is created the system creates two new partitions (Development/QC and Production) and assigns the 6th through 10th medium studies to those partitions, and so on.

Applying Snapshot Labels

You can use Oracle LSH to apply snapshot labels to data in a clinical data model lifecycle area.

1. Log in to Oracle LSH.
2. In the Applications tab, **Select Domain** field, search for and select `DMW_DOMAIN`. The system displays the DMW_UTILS domain and the Study and Library category domains (see "Setting Up Library and Study Categories" on page 8-15).
3. Expand the node for the appropriate category. The system displays all studies in that category.
4. Expand the node for the appropriate study. The system displays the Development, Quality Control, and Production life cycle application areas for the study.

5. Click the link for the appropriate lifecycle area. The system takes you to the properties page for the lifecycle area, with a list of work areas, one for each installed clinical data model. The model name is included in the work area description.
6. Click the link for the appropriate work area. The system takes you to the properties page for the work area, with a list of installed tables and other objects in the clinical data model.

You can then follow instructions in the section "Managing Table Instance Snapshot Labels in a Work Area" in the Work Areas chapter of the *Oracle Life Sciences Data Hub Application Developer's Guide*.

Loading Reference Tables

If you have a library of lookup, or conversion, tables, you can upload them to Oracle LSH, where your custom programs and functions can reference them. Use Oracle LSH load sets to upload them.

- If your lookups are contained in SAS files, you can upload many at once in SAS CPort or XPort format using an Oracle LSH SAS load set.
- If your lookups are Oracle database tables, you can upload many at once by using an Oracle LSH Oracle Tables and Views load set.
- If your lookups are in any other format, you can create metadata files in a required format and use a Oracle LSH Text load set to upload many at once; see "Required Syntax for Table Metadata Text Files" on page A-3.

The basic steps required are:

1. Log in to Oracle LSH.
2. In the Applications tab, **Select Domain** field, search for and select `DMW_DOMAIN`. The system displays domains, including the DMW_UTILS domain. You may want to load your reference tables into the DMW_UTILS domain, which will also contain your custom programs, but this is not required.
3. Click the domain you want to use. The system opens its properties page.
4. Create an application area and then a work area inside it in the DMW_UTILS domain.
5. Create the load set in the work area you created.
6. Install the target tables in the same work area.
7. Run the load set to load the data.
8. When referencing these lookups from custom programs, use static reference type source code.

For instructions, see the *Oracle Life Sciences Data Hub Application Developer's Guide* chapter on load sets.

Setting Up a Data Visualization Tool

You can integrate the system with an external data visualization tool so that users can view data graphically and interactively, one model at a time. Oracle Business Intelligence Enterprise Edition (OBIEE) is included for this purpose, and there are third-party tools such as JReview and Spotfire available for use with Oracle LSH that

will work with Oracle DMW. You can also create your own adapter to another tool; see the *Oracle Life Sciences Data Hub Adapter Toolkit Guide*.

Install OBIEE or the third-party tool with access to the database and integrate it with Oracle LSH. Instructions for OBIEE are included in the *Oracle Life Sciences Data Hub Installation Guide* and the *Oracle Life Sciences Data Hub System Administrator's Guide*. For third-party tools, follow instructions provided by the maker.

You must define an Oracle LSH Business Area in the Work Area of a clinical data model and map it to each table instance whose data you want to make available in the visualization tool, ; see the chapter on Business Areas in the *Oracle Life Sciences Data Hub Application Developer's Guide*.

9

Data Processing

This section contains the following topics:

To run transformations and validation checks, go to the Activities page. To load data, go to the Study Configuration/Clinical Data Models page in the InForm or File Watcher tab.

Data Processing Types and Modes

Oracle Health Sciences Data Management Workbench (Oracle DMW) supports two types of data processing: Reload Processing and Unit of Work Processing (UOW). Both types can be run in either Full or Incremental mode. UOW processing has an additional mode—UOW Load—that is available only for loading data.

The definition of the target table determines the default data processing *type*—Reload or UOW—that the system uses for a job. You select the *mode*—Full, Incremental, or Load—when you schedule or submit a job.

Both types of processing require a primary key on the target Table instance. It is not necessary to have a primary or unique key defined in an external source; for example, external SAS files that do not have a primary key defined can be loaded as long as the target table has a primary key.

The target tables of validation checks, which are created automatically by the system, use Reload processing.

Reload Processing

In Reload processing, the system processes all records in the source(s) and compares the primary keys of the source records with the primary keys of the records already in the target table(s), if any, to determine which records to insert, update, and (in Full mode only) delete. If a reloaded record does not include any data changes, the system simply changes its refresh timestamp to the timestamp of the current job.

Full

Full Reload processing performs insertions, updates, refreshes, and deletions: if a record is not included as input, the system soft-deletes it—the record still exists but is **no longer available for use**. The audit trail for the record is available.

Note: Do not use Full mode if the source contains only new data or any other subset of data because **any data not reloaded will be deleted**.

Use Full mode **only** if you are confident that the data being loaded or read is the complete set of data.

Incremental

Incremental Reload processing performs insertions, updates, and refreshes but no deletions. If a record is not reloaded it remains available but its timestamp is not updated.

Incremental processing is faster than full, so you may want to use incremental processing frequently and full processing less frequently but regularly, to ensure that data is appropriately deleted.

Unit of Work Processing

A Unit of Work (UOW) is all records associated with either a particular subject or a particular subject visit. Tables with a UOW processing type must have either the subject or subject and visit columns marked as SDTM identifiers and the Unit of Work must also be defined: either Subject or Subject/Visit.

UOW processing reads all source records and notes the *UOW key*—the value of the subject or subject and visit columns—for each record, and adds the UOW key to an *execution set*—the set of subjects or subject visits to be processed. The system then processes only records for the units of work represented in the execution set and no records for other units.

See Table 9–1, " Deletion Behavior in Unit of Work Processing Modes" to compare UOW modes.

Target tables defined for Unit of Work processing also accept Reload processing.

Full UOW

Full UOW processing examines timestamps in the source UOW table(s) to determine which records have changed since the last Full UOW or Full Reload processing job and creates the execution set for those records' UOW key (subject or subject visit) and no others. If the program has never been run in any mode, all units of work are included in the execution set. It inserts, updates, and refreshes all records belonging to subjects or subject visits in the execution set and no others. If records previously existed in the target tables it also deletes records:

- Records in processed units of work (subject or subject visit) that are not reloaded are deleted.
- **Entire units of work are deleted** if they exist in the target table but are not reloaded.

  For example, if a transformation reads from an Adverse Event table and writes to a Severe Adverse Events table and has previously inserted records with a Serious flag set to Y for a particular subject, a change to the Serious flag to N for all of a subject's records results in no records being inserted and, since the subject is in the UOW execution set, the deletion of all records for that subject from the target.

Incremental UOW

Incremental UOW processing examines records' timestamps in the source UOW table(s) to determine which records have changed since the last processing job in any mode (UOW or Reload, Full or Incremental), and creates the execution set for those records' UOW key (subject or subject visit) and no others. If the program has never been run in any mode, all units of work are included in the execution set. It performs insertions, updates, and refreshes but no deletions.

As with Reload processing, UOW's Incremental mode is faster than Full mode, so you may want to use incremental UOW processing frequently and full UOW processing less frequently but regularly, to ensure that data is appropriately deleted.

Note: The end result of Incremental UOW processing is the same as Incremental Reload processing. Both process all new and changed records and delete no records. The end result of the two Full modes is also the same.

Which process will be faster depends on the volume of changed data being processed and whether changes are concentrated in specific units of work or spread fairly evenly across all units. Compared to Reload, UOW processing has overhead costs in detecting affected subjects or subject/visits, but it is more efficient in that it processes records only in units with changes, not all records.

In general, UOW will probably be faster than Reload when the number of incremental changes is small or concentrated in relatively few units of work.

In addition, if you use custom programs in transformations, using UOW processing takes care of finding incremental data changes; your code does not need to handle that.

UOW Load

After loading data the system identifies the distinct set of UOW keys for the records that were inserted, modified or refreshed in the load, creates an execution set consisting of these units of work, and processes all records within these units of work and no others. Any records in a unit of work included in the execution set that is not included in the file is deleted.

This is the only type of UOW processing available during data loading. If you want different deletion behavior you can use Reload processing ; see Table 9–2, " Deletion Behavior in Data Loading Processing Modes".

Note: Use UOW Load mode **only** if the file being loaded contains the **full current set of data** (new, modified and unchanged) for subjects or subject visits with any new or modified data because any data not reloaded in processed units of work **will be deleted**.

Instead, **use Incremental Reload** processing to load data containing just new or modified records.

Table 9–1 Deletion Behavior in Unit of Work Processing Modes

Unit of Work Mode	Delete within Reloaded Unit?	Delete All Records for Nonreloaded Unit?
UOW Load	Yes	No
Full UOW	Yes	Yes
Incremental UOW	No	No

Data Processing in Transformation Programs

A transformation reads from one or more source data models and writes to a single target data model. The transformation consists of a separate program for each target table. By default, each program uses the processing type of the target table—Reload or UOW. The user who submits or schedules the transformation to run chooses either Full or Incremental mode.

The system uses the default processing type in the selected mode for each target table whenever possible. However, if the target table's processing type is UOW, the system may not be able to use it:

- Any source tables that are not themselves identified as UOW tables are processed using Reload because the system cannot determine which subjects or subject visits may have been impacted by the changes. The end result is the same, but processing may be slower.
- Similarly, if the target is defined with Subject Visit UOW and a source table is defined as Subject UOW:
  - In Subject Visit UOW source tables, the transformation logic processes all records for impacted subject visits in the table and it also processes the records for subjects that are impacted in the Subject UOW source tables.
  - In Subject UOW source tables, the transformation logic processes all the impacted subjects.

You can enable UOW processing by defining all tables as UOW whenever possible: when they have subject and visit columns, designate the columns with the corresponding SDTM identifier and set the processing type to UOW Subject Visit or just Subject if there is no Visit column. Do this even if you do not plan to use UOW processing to write to the table to enhance downstream processing performance.

If a source table is not a subject-oriented table, such as a lookup or reference table, and there is any change in data in it since the last execution of the program, the system uses Full Reload processing on all source tables.

Populating Surrogate Keys for Data Lineage Tracing

To enable the system to trace the lineage of every data point back to its source, all types and modes of data processing do the following the first time a transformation is run:

- Create one additional column in the target table for each source table; if two source tables feed into a target table, the transformation logic adds two columns to the target table
- Populate each SKEY column with the surrogate key value of the record in the source table. That value is the concatenated primary key values of the record in the source table, separated by pipes; for example:

SUBJID_VALUE|*VISINUM_VALUE*|*TEST_VALUE*

See "How the System Tracks Data Lineage" on page 5-24 for more information.

Data Processing in Validation Check Programs

The target tables of validation checks, which are created automatically by the system, use Reload Processing. The user can choose Full or Incremental mode when scheduling or submitting the validation check batch.

Loading Data

The system supports loading data from SAS or text files or from InForm:

- "Processing Data Loads from Files" on page 9-5
- "Processing InForm Data Loads" on page 9-6

See also "Format Checks on Loaded Files" on page 9-6.

You set up data loading when you define input clinical data models.

Processing Data Loads from Files

You can load text or SAS files (data sets, XPORT, or CPORT files) into an input data model. You set up a File Watcher for each input data model. The File Watcher then detects when a data file appears in a specified location and proceeds to load the data automatically. See "Configuring File Watcher" on page 3-20 for more information.

The system supports three processing modes for loading data from files:

- Full (Reload)
- Incremental (Reload)
- UOW Load

Note: Do not use Full mode if the file being loaded contains only new data or a subset of data because any data not reloaded will be deleted.

Do not use UOW Load mode if the file being loaded contains only new data or a subset of data for subjects or subject visits because any data not reloaded in processed units of work will be deleted. Instead, use Incremental (Reload) processing.

Note: All deletions are "soft" deletions: records have an end timestamp equal to the load's date and time and are **no longer available in the system**. However, they still exist in the database and have an audit trail.

Table 9–2 Deletion Behavior in Data Loading Processing Modes

Processing Mode	Uses UOW Logic?	Deletion Behavior
UOW Load	Yes	Deletes nonreloaded records within units of work —subjects or subject visits—with new, changed, or refreshed records.

Table 9–2 (Cont.) Deletion Behavior in Data Loading Processing Modes

Processing Mode	Uses UOW Logic?	Deletion Behavior
Incremental Reload	No	Does not delete any data.
Full Reload	No	Deletes all records that are not reloaded.

Processing InForm Data Loads

Each InForm study has one InForm input clinical data model for each lifecycle stage. You set up a connection and schedule and it proceeds automatically. See "Configuring the InForm Connector" on page 3-16 for more information.

Format Checks on Loaded Files

The system uses the Oracle SQL Loader to load SAS and text files in both Reload and Unit of Work modes, and makes use of its ability to test that incoming data meets the format requirements of target columns, including data type, length, code list values (if the column is associated with a code list), and the nullable and check constraints.

Records that pass the format checks are inserted into the table one record at a time using the Oracle Insert statement. Records that are rejected are captured and validated for other errors. All the errors found in each record are reported in a file called ComprehensiveErrRpt.csv.

The error file contains one row for each record with an error that says "ORIGINAL ERROR" in the Column Name column and another row for the original error and additional rows for any other errors it finds on the same record. Even though the record is rejected after the first error, the system continues to check all column values for the record and lists all errors in the error file.

Table 9–3, " Error File Example" shows the error file entries for two records. The first record has two errors and the second has one. The string "CDR_W37_1D0156B9.TXT_TV486627301.Name" represents the full path of the erroring field, Name, for Record 1. The first part, "CDR_W37_1D0156B9," is the schema name and the second, "TXT_TV486627301," is the view based on the target table.

Records with errors are not inserted into the target tables.

When you configure File Watcher, you can define the maximum number of rejected records allowed for the load before the load fails using the Max Errors parameter.

Table 9–3 Error File Example

TABLE_NAME	FILE_NAME	REC_NUM	COLUMN_NAME	VALUE	ERROR_MESSAGE
DEMOG	DEMOG.txt	1	ORIGINAL_ERROR	Captured in the TextLoad.log	ORA-20100: ORA-01400: cannot insert NULL into ("CDR_W37_1D0156B9.TXT_TV486627301.NAME")
DEMOG	DEMOG.txt	1	NAME		ORA-01400: cannot insert NULL into ("CDR_W37_1D0156B9.TXT_TV486627301.NAME")

Table 9–3 (Cont.) Error File Example

TABLE_ NAME	FILE_NAME	REC_ NUM	COLUMN_ NAME	VALUE	ERROR_MESSAGE
DEMOG	DEMOG.txt	1	HT		ORA-01400: cannot insert NULL into ("CDR_W37_1D0156B9.TXT_ TV486627301.HT")
DEMOG	DEMOG.txt	32	ORIGINAL_ ERROR	Captured in the TextLoad.log	ORA-20100: ORA-02290: check constraint (("CDR_W37_1D0156B9.TXT_TV486627301 .AGE_CK) violated
DEMOG	DEMOG.txt	32	AGE	05	ORA-02290: check constraint (("CDR_W37_1D0156B9.TXT_TV486627301 .AGE_CK) violated

Supporting Duplicate Primary Keys in a Load

In rare cases you may need to allow a single load of source data to contain records with duplicate primary key values; for example, when loading data from a small lab that may not be able to guarantee uniqueness. In those cases you can define a composite key, but it may not be sufficient to ensure uniqueness.

If you need to support duplicate primary key values within a single data load, check **Supports Duplicate** when you define the primary key for the target table in the input data model. Selecting this option ensures that all records are loaded and not deleted but requires careful checking of the data.

When you select **Supports Duplicate**, the system adds the column CDR$DUP_NUM to the target table. During each data load, the system detects whether multiple incoming records have an identical primary key value and:

- The system inserts a value of `1` to the CDR$DUP_NUM column for each record with the first occurrence of a set of primary key values.
- If another record with the same primary key values is loaded in the same data load, the system inserts a value of `2` into its CDR$DUP_NUM column, and `3` for the third record with the same primary key values, and so on.
- During subsequent data loads, the system assumes that the first record with a particular set of primary key values is the same as the existing one with the CDR$DUP_NUM value `1`, the second is the same as `2`, and so on. If two records exist with values `1` and `2`, and the next load contains three records with the same values, the system gives the third record a CDR$DUP_NUM value of `3`.

> **Note:** For this system to work, **the lab must always reload all records in the same order**, adding new records to the end of the file.

The CDR$DUP_NUM value becomes part of the surrogate key as well as the primary key and is used for data lineage tracing; see "Populating Surrogate Keys for Data Lineage Tracing" on page 9-4.

Part II

Managing Study Data

This section contains the following topics:

10

Using the Home Page

This section covers the Home page and ways to change the columns and records displayed in many pages of the user interface:

Selecting a Study

You must first select a study in the Home page before you can work in it in Oracle Health Sciences Data Management Workbench (Oracle DMW).

Select the study you want to view or work on from the Studies list and then click the link to the page you want to work in.

Or, if the list is too long:

1. In field above the list, type all or part of the name of the study.
2. Press Enter.
3. Select the study you want in the list. The system displays links to all the user interface pages you can access.

> **Note:** The next time you log in, the system automatically loads the last study you were working in. To change to a different study, delete the study name in the field above the list and type all or part of the name of the study you want to work in.

Selecting a Lifecycle Mode

You must select a study lifecycle mode before you can work in a study. You may have security privileges for only one or two modes of the three:

- **Development** is for studies being set up and modified.
- **Quality Control** is for studies being tested.
- **Production** is for live studies and also for viewing completed studies.

Viewing Data Load Information

In the Home page, Data Loads tab, you can view information on past data loads for the selected study, including:

> **Note:** You can query on most column to filter the data loads displayed; see "Querying By Example" on page 10-9.

- **Data Model**: The Input clinical data model that is the target of the load.
- **Type**:
  - **FILEWATCHER** for data loaded from a file by File Watcher.
  - **INFORM** for data loaded from InForm.
- **File Name/InForm Load Process**: The system displays different information depending on the data load type:
  - For files, the full path of the file loaded.
  - For InForm, the type of data load: INFORM_DEFINITION, INFORM_METADATA, INFORM_MANUAL_DATA or INFORM_SCHEDULED_DATA.
- **Last Job**: The job ID of the most recent load for the clinical data model.
- **Last Load**: The date and time of the most recent execution.
- **Last Load Results**: **Success**, **Warnings**, or **Failure** for completed jobs. Click the icon to view or download the log file generated by Oracle DMW.

  There are other statuses for incomplete jobs; see "Statuses for Uncompleted Jobs" on page 10-2.
- **View Output** (File loads only): Click the icon to view or download the Oracle SQL Loader log file.
- **View Error Report** (File loads only): Click the icon to view or download the error file, if any.

See "Changing User Interface Display" on page 10-7 and "Querying By Example" on page 10-9.

You set up data loading when you define input clinical data models. See "Configuring the InForm Connector" on page 3-16 and "Configuring File Watcher" on page 3-20 for more information.

Statuses for Uncompleted Jobs

The system uses the following statuses in addition to **Success**, **Warnings**, and **Failure** for completed jobs:

Pending: The job has not yet started running.

Started: The job has begun pre-processing.

Executing: The Program has connected to database and is running.

Finalizing: The job has begun post-processing.

Aborted: The job has been manually stopped while underway.

On Hold: The job is waiting for the quiesce process to complete.

Expired: The system removed the job from the queue after the timeout interval passed.

Duplicate: The job is a duplicate of another job; the currency of the source data, parameter values, and executable instance version are the same. Therefore the system does not rerun the job unless the person submitting the job chooses to force reexecution. To view the job that this one would duplicate, paste the Duplicate Job ID in the search window and locate the job.

Viewing and Running Transformations

This section contains the following topics:

Viewing Transformation Job History

Transformations are displayed by the name of their target data model. The upper pane lists each transformation once and displays information about it and the most recent execution and installation of it.

View Table Transformations Click a transformation's triangular node to see all the target tables and information on the most recent execution of the transformation of the specific table.

View Run History Select a transformation in the upper pane and the system displays information on all past executions in the Run History pane.

View Log Files The following columns are for different types of jobs. Click the icon to view the log file.

- Log: The most recent manually submitted job
- Triggered Job Log: The most recent triggered job
- Install Job Log: The most recent installation of the transformation

Running a Transformation

You can run transformations manually and, if they are set up to support it, they can be triggered by a job that writes data to any of their source tables; see "Automatically Triggering Transformations and Validation Checks by Upstream Processes" on page 10-5.

To run a transformation manually, Select the Transformation and Submit the Job.

Select the Transformation

Select the transformation you want to run and click **Submit**.

The **Submit** button does not appear if:

- You have not selected a transformation.

- The selected transformation has not been installed; check the Install Status column.

Submit the Job

To execute a transformation:

1. Enter values in the following fields:
   - **Submission Mode**: Select one:
     - **Full** mode normally takes longer and includes data deletion.
     - **Incremental** normally is faster and does not include data deletion.

     If you are submitting a transformation for a single table and the table is defined with Unit of Work processing, you can also select:
     - **Full UOW** mode normally takes longer and includes data deletion.
     - **Incremental UOW** normally is faster and does not include data deletion

     See "Data Processing Types and Modes" on page 9-1 for more information.

     Note: You may want to set up regular Incremental loads at frequent intervals and regular Full loads at longer intervals. See "Data Processing Types and Modes" on page 9-1 for more information.

   - **Force Execution**: Before running a job, the system checks the source data currency, parameter values, and the version number of the programs. If none of these has changed since the last run, the results would be unchanged if the job ran again, so the system does not execute the job and returns a status of Success.

     If you want to run the job anyway, select **Force Execution**. The system then uses **Full mode**, regardless of the Submission Mode setting.
   - **Submission Type**: Select:
     - **Immediate** to run the job once, as soon as possible
     - **Scheduled** to set up a regular schedule
     - **Deferred** to run the job once, at a future time
   - **Frequency**: If you selected **Scheduled**, enter a number of minutes, hours, or days that you want to elapse between runs, and select the time unit.
   - **Start and End Date**:
     - If you selected **Scheduled**, select a date and time to begin running on schedule and a date and time to stop.
     - If you selected **Deferred**, select a date and time to run the job.
   - **Trigger Downstream Transformations and Validation Checks**: Select this check box if you want this job (or each job if you are setting up regularly scheduled runs) to trigger validation checks in the target model and transformations from the target model to all others that come after it, in sequence. This can happen only if the source models in the affected transformations are set up to trigger downstream processes; see "Creating Model Mappings" on page 5-2.
   - Click **Refresh** at any time for an update.

Automatically Triggering Transformations and Validation Checks by Upstream Processes

You can set up a chain of transformation and validation check executions beginning when you load data into the system or at any point thereafter by setting the Can Trigger attribute of a source model in a transformation to Yes. This ensures that the data in all participating models is always as up-to-date as possible.

Forward Chaining Example A study has Input models called InForm, Central Lab and Local Lab that all feed data to the Review model, which feeds data to the Analysis model, which feeds data into the Submission model.

In the Review model's transformation, for which all Input models are source models, set the Can Trigger attribute to Yes for each Input model. Whenever data is loaded into any of these Input models, the transformation to the Review model is triggered, as are all the validation check batches defined in the Review model.

In the Analysis model's transformation, which has only the Review model as a source in this example, set Can Trigger to Yes for Review as a source model. Whenever data is loaded into the Review model—as the result of a data load to any of its source models or because of a manual or scheduled execution—the transformation to the Analysis model is triggered, as well as all validation check batches defined in the Analysis model.

In the Submission model's transformation, you may prefer not to set the source model's Can Trigger attribute to Yes. It may not be important to have the latest data as soon as possible if the purpose of this model is to export final data for regulatory submission, and you may prefer not to use system resources for frequent executions.

Viewing and Running Validation Check Batches

The Validation Checks tab displays validation check batches for the selected clinical data model.

When you select a batch, the system displays all its validation checks in the Validate Checks pane and information on all its past executions in the Run History pane.

To submit a batch, select it and click **Submit**.

The **Submit** button does not appear if:

- You have not selected a validation check batch.
- The selected batch has not been installed; check the Installed column.

You must run validation checks as a batch. It is possible to *disable* a validation check in the Study Configuration page so that it is not included in the batch execution. To find out if a check was included in the batch, check the log file in the Run History pane.

Submit the Validation Check Batch

To run a validation check batch:

1. Enter values in the following fields:
   - **Submission Mode**: Select one:
     - **Full** mode takes longer and includes data deletion.
     - **Incremental** is faster and does not include data deletion.

- **Force Execution**: Before running a job, the system checks the source data currency, parameter values, and version number of the transformation or validation check programs. If Force Execution is not selected and none of these has changed since the last run, the system does not execute the job.

  If you want to run the job anyway, select Force Execution.
- **Submission Type**: Select **Immediate** to execute the job once, as soon as possible. Select **Scheduled** to set up a regular schedule.
- **Frequency**: If you selected **Scheduled**, enter a number of minutes, hours, or days that you want to elapse between runs, and select the time unit.
- **Start and End Date**: If you selected **Scheduled**, select a date and time to begin execution and a date and time to end execution.
- **Trigger Downstream Transformations and Validation Checks**: Select this check box if you want the system to detect all transformations and validation checks set up for this data model and all others that come after it, and submit them sequentially.
- Click **Refresh** at any time for an update.
- To check the log file, click the Log icon in the Run History pane.

Viewing Validation Check Batch Job History

Click a validation check batch in the upper pane to view its job history in the Run History pane. You can:

View Log File Click the icon in the Log column to view or download the log file.

Cancel Job Click to cancel a pending manually submitted job.

Cancel Triggered Job Click to cancel a pending triggered job; see "Automatically Triggering Transformations and Validation Checks by Upstream Processes" on page 10-5.

Viewing Data Files Not Processed

If the File Watcher service detects a file in a watched location that cannot be loaded because no File Specification for any clinical data model in the study matches it, it displays information about the file on the **Files Not Processed** tab of the Home page for the selected study.

Files may not match any File Specification because:

- They are misnamed, which may include case differences.
- There is a mistake in the File Specification regular expression; see "Creating File Specifications" on page 3-22.

The Files Not Processed screen displays the following information about each unloadable file:

- **Label**: Description
- **File Name**
- **Path**: Full path to the file on the file system
- **Status**: The possible statuses are:

  - **DETECTED**: The file has been detected in the watched folder but has not yet been submitted. (The scheduled submission time is in the Data Load Date column.)
  - **MISSING**: The file was detected but deleted before the scheduled deletion date.
  - **DELETED**: The file was deleted by File Watcher as scheduled.
- **Detection Date**
- **Deletion Date**
- **Data Missing**

Changing User Interface Display

Many pages in the user interface that are laid out like tables with columns and rows allow you to change the data display in several ways:

- "Sorting Rows by Column Values" on page 10-7
- "Showing and Hiding Selected Columns" on page 10-8
- "Changing Column Order" on page 10-8
- "Freezing Columns" on page 10-8
- "Detaching a Pane" on page 10-8

Sorting Rows by Column Values

You can sort on values in a single column or up to three columns.

> **Note:** The system sorts only the records that are currently displayed. For example, if there are a total of 200 rows but the page shows only 50 at a time, when you sort by column value, only the 50 currently displayed rows are sorted. To be sure you have seen all the rows, you must view and sort all four pages in this example.

Sorting on a Single Column

You can click on the up and down arrow on the right side of any column heading to sort on that column. You may need to widen the column to see the arrows.

Sorting on Multiple Columns

To sort on up to three column values:

1. Select **View**, then **Advanced Sort**.
2. In the **Sort By** field, select the primary column to sort on and then select either:
   - **Ascending** to sort in alphabetical or numeric order (a...z or 0...9).
   - **Descending** to sort in reverse alphabetical or numeric order (z...a or 9...0).
3. In the first **Then By** field, select the secondary column to sort on and either **Ascending** or **Descending**. Within the sort order you specified in Step 1, the system sorts records in the order you specify here.

4. In the second **Then By** field, select the tertiary column to sort on and either **Ascending** or **Descending**. Within the sort order you specified in Steps 1 and 2, the system sorts records in the order you specify here.
5. Click **OK**.

Showing and Hiding Selected Columns

To hide columns you don't need to see in order to make more room for columns you do need to see:

1. Select **View**, then **Columns**. The system displays all available columns, with a check next to those that are currently displayed. If you want to add or hide a single column, click the column name in the list.
2. To show or hide more columns, select **Show More Columns**.
3. Use the arrows to move columns from the **Hidden Columns** area to the **Visible Columns** area or the reverse. You can also reorder columns here.
4. Click **OK**.

Changing Column Order

To make it easier to see the columns you need most without scrolling or put the columns you need in a more logical order:

1. Select **View**, then **Reorder Columns**.
2. Use the arrows to move columns up or down in relation to each other. Columns at the top of the list are displayed leftmost on the page.
3. Click **OK**.

Alternatively, you can drag and drop columns directly on the page using the mouse.

Freezing Columns

You can "freeze" the leftmost column or columns so that they remain visible even when you scroll far to the right. If the columns you want to have always visible are not located farthest to the left, reorder columns so that they are.

Note: You cannot sort records in a frozen column. If you want to sort records in either ascending or descending order, unfreeze the column first.

To freeze or unfreeze a column:

1. Select the column heading.
2. Click the Freeze or Unfreeze icon or

   click **Freeze** or **Unfreeze** in the **View** menu.

Detaching a Pane

When you select **Detach**, the system displays the user interface pane as a separate, larger window. You can reattach it at any time by selecting **Attach**.

Querying By Example

In many pages that are laid out like tables with columns and rows you can filter the rows displayed:

1. If empty fields above each column are not already displayed, click the Query By Example, or Filter, icon:

   The system displays an empty field above each column heading.

2. Enter a value in one or more fields above a column heading. Use the wildcard `%` before and after a string to return all values containing the string in that column. Values are case-sensitive.

3. Press Enter. The system displays only rows that have the specified value or values.

To show all rows again, delete all values in the Query By Example fields and press Enter.

You can hide the empty fields by clicking the Query By Example icon again.

Using Online Help

Click the help icon (?) to see online help. Help icons that appear next to a specific field display help just for that field. Help icons that appear at the top of a tab, page, or pop-up window display help for that tab, page, or window that usually includes a description of each field.

The first time you open a help window in a session may take a long time. However, if you leave the window open, the next time you click a help icon, the display will be much faster.

11

Reviewing Data

You can view data and discrepancies in three pages:

The following actions are available on all three Listings pages:

In the Listings pages you can view clinical data, create discrepancies identifying possibly flawed data, and view existing discrepancies imported from InForm (queries) or raised by validation (edit) checks or other people. You can create your own custom listings. If necessary, and if you have the required privileges, you can amend clinical data in a controlled way.

To see data listings:

1. On the Home page, select the study and lifecycle mode if necessary and click **Listings**.
2. Under **Listings**, select the name of the clinical data model you want to view.
3. In the right-hand panel, select the type of listing you want to view:
   - **Default Listings** for access to all study data to which you have access.
   - **Custom Listings** to create your own listings.
   - **VC Listings** to see discrepancies identified by validation (edit) checks.

Note: The links for Custom and VC Listings are located at the bottom of the panel.

4. **Model Name**: Select the clinical data model you want to view. The system displays the tables in that model.

   A clinical data model is a set of tables grouped by your company for a purpose. Each study has several data models that store data in exactly the form loaded from the source—InForm or a lab. People in your company then combine and transform that raw data as required for review and analysis in sequence, creating a data model for each purpose and perhaps intermediate data models as well.

   Your role may have access to only one model.

5. Select the table whose data you want to view. The system displays the data, with table column names (items) across the top and a row for each record.

Viewing All Study Data Using Default Listings

Default listings show all data in all tables in a clinical data model with the exception of blinded, masked data. The most current data in the system is always displayed.

InForm queries are converted to discrepancies and displayed and tracked along with discrepancies created manually or by validation checks.

Records with one or more discrepancies have a yellow rectangle on the left and the word **Yes** in the DISCREPANCY_EXISTS column. Each data point in the row with a discrepancy is highlighted in yellow.

You can select multiple records using Ctrl+click or Shift+click at the beginning of each row.

You can do all of the following tasks in any Listings page:

- "Creating Discrepancies Manually" on page 11-15
- "Showing Discrepancies" on page 11-15
- "Showing Flags" on page 11-16
- "Flagging Data" on page 11-16
- "Viewing Data Lineage" on page 11-17
- "Viewing Blinded Data" on page 11-18
- "Exporting All to Excel" on page 11-18

Viewing Validation Check Listings

Validation checks (edit checks) identify faulty or questionable data; for example, missing or out-of-range values, or values that do not make sense when compared to related values. Validation checks generate a discrepancy against each data point they identify, give the discrepancy a status of either Open or Candidate, and may apply a tag identifying the discrepancy as requiring medical review first. Validation checks may be set up to automatically close discrepancies they created if the underlying data point is fixed. If not, manual resolution is required.

Each validation check tests for a single problem and writes all the open or candidate discrepancies it finds to a single table, which is displayed in VC Listings.

Select a Validation Check under VC Listings to see its results in the main pane.

Validation checks run in batches on a regular basis, set up in the Activities page; see "Viewing and Running Validation Check Batches" on page 10-5.

You can do all of the following tasks in any Listings page:

- "Creating Discrepancies Manually" on page 11-15
- "Showing Discrepancies" on page 11-15
- "Showing Flags" on page 11-16
- "Flagging Data" on page 11-16
- "Viewing Data Lineage" on page 11-17
- "Viewing Blinded Data" on page 11-18
- "Exporting All to Excel" on page 11-18

Creating and Viewing Custom Listings

You can view custom listings saved as public and create ad hoc listings.

To view a saved listing, select it in the Custom Listing pane. The system displays the data on the right. The most current data in the system is always displayed, regardless of when the listing was created.

Click the + icon in the Custom Listing pane to create your own listing; see "Using the Query Builder to Create Custom Listings" on page 11-3.

You can do all of the following tasks in any Listings page:

- "Creating Discrepancies Manually" on page 11-15
- "Showing Discrepancies" on page 11-15
- "Showing Flags" on page 11-16
- "Flagging Data" on page 11-16
- "Viewing Data Lineage" on page 11-17
- "Viewing Blinded Data" on page 11-18
- "Exporting All to Excel" on page 11-18

Using the Query Builder to Create Custom Listings

To create a new custom listing:

1. In the Source pane, select the clinical data model whose data you want to view.
2. Click the Add + icon to create a new custom listing. The Query Details pane appears.
3. Enter a name and description for the query. The name is displayed on the Listings page; the description is not. The Listings page can display about 25 characters of the name without requiring scrolling to see the whole name. You may want to use a naming convention to make it easier to find queries later.
4. Select **Mark as Public** if you want all users with the required security access to the data model and to the Listings pages to be able to view the custom listing produced by this query. The query is not public until you save it.

5. Select **Authorize access to this listing for users without Blind Break rights** if you know that only nonblinded data will be displayed in the listing. This option is available only if at least one source table contains blinded data and if you have the required blinding-related privileges.

   If any source table is blinded at any level and this attribute is not selected, the system automatically blinds the entire target table, so that by default no data is displayed on the Custom Listings page for this listing. Only a user with the required Blind Break privileges can view the data.

6. Select **Use a Custom Program** if you need more complex logic than you can create in the Expression Builder. Then click the Edit icon to select the custom program. See "Defining a Custom Program" on page 5-17.

7. Specify the columns to display; see "Selecting Columns to Display in Custom Listings" on page 11-4.

8. If you need to use data in two or more tables, create joins between the tables; see "Creating Joins" on page 11-5.

9. Specify query criteria; see "Specifying Criteria" on page 11-5.

10. See "Testing and Saving the Query" on page 11-6.

Selecting Columns to Display in Custom Listings

In the Select Columns tab, identify the columns you want to display for each record retrieved by the query. You can give columns a different header for display and combine or write expressions on one or more columns as required. The system creates a SELECT clause for the query based on your specifications.

1. In the Source pane, expand the node for the table or tables whose data you want to display. in the Custom Listings page for each record retrieved.

2. Select the columns you want to display. You can use Ctrl+click and Shift+click to select multiple columns at a time. You can also select table to add all its columns, and then remove the ones you do not want to display.

3. Click the Add (+) icon. The system displays the columns you selected under **Selected Columns**.

4. If you want to display a heading for the column different from the column name, enter it in the Alias field.

5. **Expression**: Enter the expression, if any, to operate on the column.

   To write an expression that operates on multiple columns you must add each column to the same row:

   a. After moving one column into Selected Columns, highlight it there.

   b. Select the second column in the Source pane and click the arrow icon in the Source pane. The system adds the second column to the same row. If you need a third column for your expression, add it the same way.

   c. Enter an alias for the column that will display the results of the expression.

   For example, you may want to display the red blood cell count collected three times in a visit and then display the average of the three results. To do this, select the column for red blood cell count in the Source pane four times and add it to Selected Columns. Then select the last one under Selected Columns, select it again under Source and click the arrow icon, and then repeat. All three are now in the same row. Enter an alias such as "RBC Avg" in the row with all three. Then write an expression for each row to populate the first three columns with data from each

sample and the last to calculate and display the average. See "Using the Expression Builder" on page 5-12.

Creating Joins

If you write an expression for multiple columns, the columns must be either in the same table or in joined tables. To create a join, select and add the tables to be joined in the Source pane and then:

1. Go to the Joins tab and click the Add (+) icon in the Joins pane. The system adds a row.
2. Click in the row. The system displays two drop-down lists containing the tables in the join.
3. Select the tables to be joined.
4. Specify if it is to be an outer join on either the left or right side. Leave the checkbox unselected to create an inner join.

   An *inner join* writes rows to the target table from all source tables only if the rows' data meets the join condition.

   An *outer join* extends the result of a simple join. An outer join returns all rows that satisfy the join condition and also returns some or all of those rows from one table for which no rows from the other satisfy the join condition.

   A *left outer join* returns returns all rows from the first table and only rows meeting the join condition from the second table.

   A *right outer join* returns returns all rows from the second table and only rows meeting the join condition from the first table.

   A *full outer join* returns all rows from both or all tables.
5. Click the Add (+) icon in the Join Details pane. The system adds a row.
6. Click in the row. The system displays two drop-down lists containing the columns for each table and another for operators.
7. Create the Join condition for the Where clause by selecting a column from each table and the operator required. For example, where both tables have a column called USUBJID, select those columns and select an operator of *equals* (=).

   To specify additional Join conditions, click the Add (+) icon in the Join Details pane again as many times as required.
8. Click **OK**.

Specifying Criteria

In the Criteria tab, identify the table(s) whose columns you want to operate on to determine which rows to retrieve. The system creates a WHERE clause based on your specifications.

1. Select the table or tables you need in the Source pane and click the Add icon. If you need to operate on columns from two or more tables in the same expression, add them to the same row:
   a. After moving one table into the Criteria tab, highlight it there.
   b. Select the second table in the Source pan and click the arrow icon in the Source pane. The system adds the second table to the same row in the Criteria tab. If you need a third table for your expression, add it the same way.

2. Click the edit icon to open the Expression Builder. See "Using the Expression Builder" on page 5-12.

Testing and Saving the Query

You can test the query or view the generated code at any time using these buttons:

- **Test**: The system generates PL/SQL code and displays the records retrieved in the Test Results pane. If the code is invalid, read the log file to find out why.
- **View Source**: The system generates and displays the PL/SQL code resulting from the query you define in the query builder.
- **Save**: The Save process includes validating the code. If the code is invalid, read the log file. The Save process creates a custom program behind the scenes and may take some time. Until you save, the query is a temporary view.

  Saved queries are available under Custom Listings. They are available to all users with the required privileges if you mark them as Public.

  Note: In Release 2.3.1 you can only validate a query if you save it, at which point it becomes publicly available. In addition, saved queries cannot be versioned. You must create a new query.

- **Update Status**: Select either QC (Quality Control) or Production if you want to see data in those lifecycle areas. When you first save a Custom Listing, the system puts it in the Development lifecycle area.
- **Delete Query**: You can select a query and delete it.

Modifying Custom Listings

You can edit a query to modify your custom listing. In this case the original version is no longer available for use. If you want to keep the old version as well as the new version, copy the query first; see "Copying Custom Listings" on page 11-6.

To edit an existing query, select and click the Edit (pencil) icon.

Copying Custom Listings

To use an existing query as the starting point for a new one without losing the original query, select it and click the Save As (floppy disk) icon.

Creating and Using Filters

You can use filters to limit the data you see in the Listings pages by setting criteria for the data you want to see.

To open the main Filters window, click the Filter icon in a Listings page:  . Then:

- **Select the filter(s)** you want to apply. The drop-down list for each type of filter displays all the public and private filters to which you have access.

  To include shared filters, check the **Show Filters Shared by Others** check box. Shared filters you created are always displayed in the drop-down list, even if this box is unchecked.

- **Add filters** if existing ones do not meet your needs. Click the Add (+) icon for the filter type. Filters you create are private filters unless you make them shared or public. Creating and modifying public filters requires special privileges.
- **Edit filters**: Select the filter from the drop-down list and click the Edit (pencil) icon to see the filter criteria or modify them. You can edit private and shared filters you created, and public filters anyone created if you have special privileges.
- **Delete saved filters** if you no longer need them by selecting a filter and clicking the Delete (X) icon. You can delete filters you created and, if you have special privileges, public filters.
- **Make your private filters Shared or Public** by selecting them and changing the value of the other drop-down list to **Shared** or **Public**. Special privileges are required to make filters public. Shared and public filters are available to everyone with the privileges required to view data in the study.
- Select **Keep Active** to keep the filters in effect until you change studies or lifecycle modes, or log out. If you do not select Keep Active and leave the Listings pages, it is no longer in effect when you return. Filters apply only on the Listings pages.

> **Note:** To save a filter for use after you log out, edit it, give it a name, and save.
>
> Filter names must be unique across all private, shared, and public filters that you can see in a particular study and lifecycle.

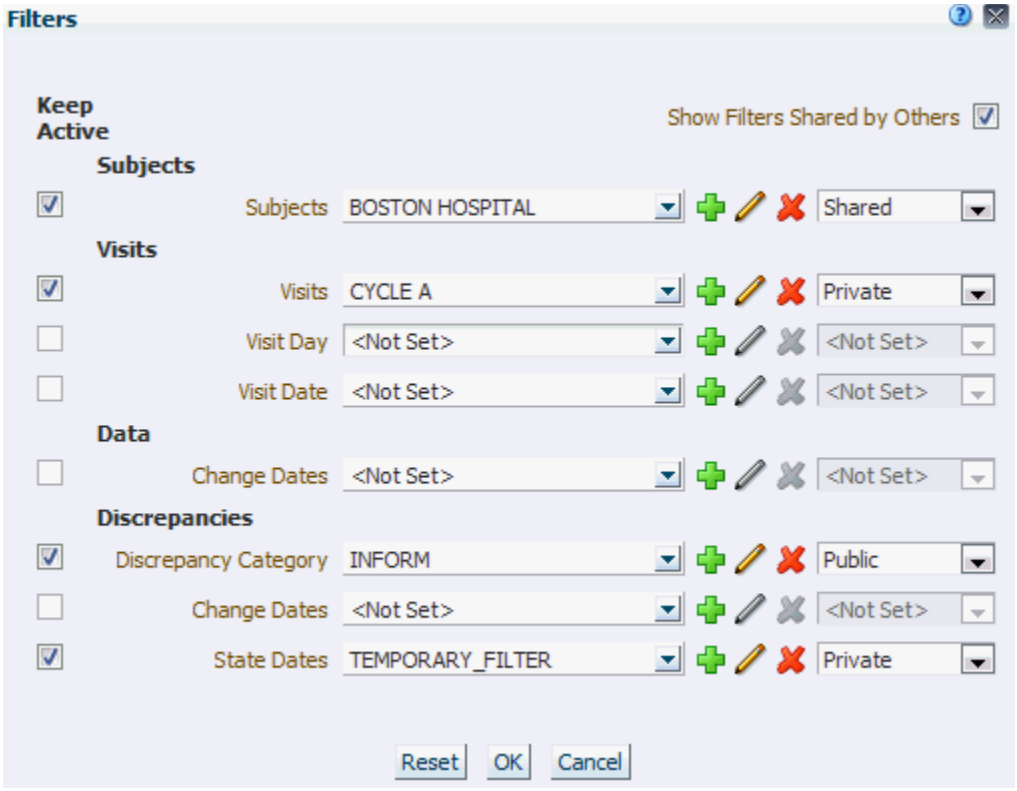
- Click **OK** to save your changes for the session, **Reset** to change all filters to Not Set, or **Cancel** to leave filters in the state they were when you opened the window.

The screenshot below shows the main Filters window with four filters selected. Two are private filters created by the current user, one is a shared filter created by another user, and one is a public filter created by another user with special privileges:

- The shared Subjects filter named BOSTON HOSPITAL filters for subjects in the Boston Hospital site.
- The private Visits filter named CYCLE A filters for records collected in visits that are part of Cycle A. This private filter has a name and is saved for future use.
- The public Discrepancy Categories filter called INFORM filters for discrepancies on data that originated in InForm.
- The private Discrepancy State Dates filter is unnamed and unsaved. It filters for records whose discrepancy state changed between specific dates that you can see if you click the Edit icon.

All these filters are applied to the data at the same time, so that the system displays only data that satisfies the criteria for all filters. In the case of the discrepancy filters, the system displays only records with discrepancies that meet the criteria for state change date range and have the InForm category applied.

All filters in the example are set to Keep Active so that as long as the current user stays in the current study and lifecycle, the filters apply, if possible. If the user views data in a table that does not have a column required for the filter—such as a Visit column or Site column in this example—the system ignores that filter.



The following sections describe how to create and edit each type of filter.

- "Creating and Editing Subject Filters" on page 11-8
- "Creating and Editing Visit Filters" on page 11-9
- "Creating and Editing Visit Day Filters" on page 11-10
- "Creating and Editing Visit Date Filters" on page 11-11
- "Creating and Editing Data Change Date Filters" on page 11-11
- "Creating and Editing Discrepancy Category Filters" on page 11-12
- "Creating and Editing Discrepancy Change Date Filters" on page 11-12
- "Creating and Editing Discrepancy State Date Filters" on page 11-13

See "Defining Tables to Support Filtering in the Listings Pages" on page 3-16 for information on filtering requirements.

Creating and Editing Subject Filters

Subject filters can be defined only in the context of a data model with a Subject Visit Table table. To create a Subject filter:

1. **Filter Driver Data Model**: (Required) Select the clinical data model that contains the Subject Visit table that you want the filtering logic to use.
2. **Filter Type**: The system displays as many of the following as are included in the Subject Visit table in the selected Driver Data Model. Select one:
   - **Countries**—includes or excludes all subjects in one or more countries
   - **Sites**—includes or excludes all subjects at one or more clinical sites

- **Site IDs**—includes or excludes all subjects at one or more clinical sites
- **Investigator Names**—includes or excludes all subjects based on their Investigator's name
- **Investigator IDs**—includes or excludes all subjects based on their Investigator's ID
- **Subject IDs**—includes or excludes one subject based on an ID that is unique within the current study
- **Unique Subject IDs**—includes or excludes one subject based on an ID that is unique across all studies

The system displays the column you select and all the columns above it in the list, with the selected column's distinct values.

You can type in all or part of a value above any of the columns to filter the choices displayed.

3. Select one or more rows to be the filter value.
4. Enter additional values as required:
   - Check **Exclude Based On Criteria** if you want to filter **out** records that satisfy the criteria rather than displaying **only** the records that satisfy the criteria. This is the equivalent of adding a "Not" to the query logic.
   - **Filter Availability**:
     - **Study Level** makes the filter available in all data models in the current study and lifecycle.
     - **Data Model Level** makes the filter available only in the current data model and lifecycle.
   - **Filter Name**: Enter a name for the filter if you want to save it to use in another session; that is, after you log out or work in a different study. Give it a descriptive name so that you will know if you want to use it or not.
5. Save. You must save the filter even if you want to use it temporarily.

Creating and Editing Visit Filters

Visit filters can be defined only in the context of a data model with a table that meets all requirements for a Subject Visit Table. To create a filter for a single visit:

1. **Filter Driver Data Model**: (Required) Select the clinical data model that contains the Subject Visit table that you want the filtering logic to use.
2. Select a filter subtype. The options are:
   - Visit Cycle
   - Visit Number
   - Visit Name

   The system displays all three columns with a row for each visit.

   You can type in all or part of a value above any of the columns to filter the choices displayed; for example, type the name of a visit cycle to see only visits in that cycle.
3. Select a row to be the filter value.
4. Enter additional values as required:

- Check **Exclude Based On Criteria** if you want to filter **out** records that satisfy the criteria rather than displaying **only** the records that satisfy the criteria. This is the equivalent of adding a "Not" to the query logic.
- **Filter Availability**:
  - **Study Level** makes the filter available in all data models in the current study and lifecycle.
  - **Data Model Level** makes the filter available only in the driving data model and lifecycle.
- **Filter Name**: Enter a name for the filter if you want to save it to use in another session; that is, after you log out or work in a different study. Give it a descriptive name so that you will know if you want to use it or not.

5. Save. You must save the filter even if you want to use it temporarily.

Creating and Editing Visit Day Filters

To create a filter based on the visit day in the study design:

1. In the **Condition** drop-down, select an operator to be used to select the visit days. The options are:
   - < Less than
   - <= Less than or equal to
   - <> Less than or greater than
   - = Equal to
   - > Greater than
   - >= Greater than or equal to
   - Between—a time duration between two dates
   - Is Not Null—has any date value
   - Is Null—has no date

   The system displays 0, 1, or 2 fields depending on which operator you select. Enter visit day numbers as required.
2. Select a clinical data model that contains a Subject Visit table. This does not restrict you to using the filter only in that model. As you browse data in different models and tables, if the current table does not have a Visit column, the system ignores this filter.
3. Enter additional values as required:
   - Check **Exclude Based On Criteria** if you want to filter **out** records that satisfy the criteria rather than displaying **only** the records that satisfy the criteria. This is the equivalent of adding a "Not" to the query logic.
   - **Filter Availability**: **Study** is the only option. The filter is available in all clinical data models that support it in the study, in all life cycle modes.
   - **Filter Name**: Enter a name for the filter if you want to save it to use in another session; that is, after you log out or work in a different study. Give it a descriptive name so that you will know if you want to use it or not.
4. Click **OK** to save. You must save the filter even if you want to use it temporarily.

Creating and Editing Visit Date Filters

To create a filter based on the actual visit date:

1. In the **Condition** drop-down, select an operator to be used to select the visit dates. The options are:
   - < Less than
   - <= Less than or equal to
   - <> Less than or greater than
   - = Equal to
   - > Greater than
   - >= Greater than or equal to
   - Between—a time duration between two dates
   - Is Not Null—has any date value
   - Is Null—has no date

   The system displays 0, 1, or 2 date fields depending on which operator you select. Enter values as required.
2. Select a clinical data model that contains a Subject Visit table. This does not restrict you to using the filter only in that model. As you browse data in different models and tables, if the current table does not have a Visit column, the system ignores this filter.
3. Enter additional values as required:
   - Check **Exclude Based On Criteria** if you want to filter **out** records that satisfy the criteria rather than displaying **only** the records that satisfy the criteria. This is the equivalent of adding a "Not" to the query logic.
   - **Filter Availability**: **Study** is the only option. The filter is available in all clinical data models that support it in the study, in all life cycle modes.
   - **Filter Name**: Enter a name for the filter if you want to save it to use in another session; that is, after you log out or work in a different study. Give it a descriptive name so that you will know if you want to use it or not.
4. Click **OK** to save. You must save the filter even if you want to use it temporarily.

Creating and Editing Data Change Date Filters

Available only on the Default Listings page, the Data Change Date filter displays records whose most recent update occurred during the time period specified. To create a Data Change Date filter:

1. In the **Condition** drop-down, select an operator to be used to select the dates. The options are:
   - < Less than
   - <= Less than or equal to
   - <> Less than or greater than
   - = Equal to
   - > Greater than
   - >= Greater than or equal to

- Between—a time duration between two dates
- Is Not Null—retrieves all records; there is always a value for last-changed-date. If the record has not been updated, it is the same as the creation date.
- Is Null—retrieves no records

The system displays 0, 1, or 2 date fields depending on which operator you select. Enter values as required.

2. Enter additional values as required:
   - Check **Exclude Based On Criteria** if you want to filter **out** records that satisfy the criteria rather than displaying **only** the records that satisfy the criteria. This is the equivalent of adding a "Not" to the query logic.
   - **Filter Availability**: **Study** is the only option. The filter is available in all clinical data models that support it in the study, in all life cycle modes.
   - **Filter Name**: Enter a name for the filter if you want to save it to use in another session; that is, after you log out or work in a different study. Give it a descriptive name so that you will know if you want to use it or not.
3. Click **OK** to save. You must save the filter even if you want to use it temporarily.

Creating and Editing Discrepancy Category Filters

Available only in the Default Listings page, the Discrepancy Category filter displays only records that have one or more discrepancies of the categories specified.

To create a Discrepancy Category filter:

1. In the **Discrepancy Categories** drop-down, select one or more categories to be used to filter, or click All and then deselect categories you do not want to use.

   Your company defines the categories that are available.
2. Check **Include *(No Category)*** if you want the filter to include or exclude discrepancies with no assigned category.
3. Enter additional values as required:
   - Check **Exclude Based On Criteria** if you want to filter **out** records that satisfy the criteria rather than displaying **only** the records that satisfy the criteria. This is the equivalent of adding a "Not" to the query logic.
   - **Filter Availability**: **All Studies** is the only option. The filter is available in all clinical data models that support it in all studies, in the current life cycle mode.
   - **Filter Name**: Enter a name for the filter if you want to save it to use in another session; that is, after you log out or work in a different study. Give it a descriptive name so that you will know if you want to use it or not.
4. Click **OK** to save. You must save the filter even if you want to use it temporarily.

Creating and Editing Discrepancy Change Date Filters

Available only in the Default Listings page, the Discrepancy Change Date filter displays records that have one or more discrepancies whose most recent update ocurred during the time period specified. To create a Discrepancy Change Date filter:

1. In the **Condition** drop-down, select an operator to be used to select the dates. The options are:
   - < Less than
   - <= Less than or equal to
   - <> Less than or greater than
   - = Equal to
   - > Greater than
   - >= Greater than or equal to
   - Between—a time duration between two dates
   - Is Not Null—retrieves all records; there is always a value for last-changed-date. If the record has not been updated, it is the same as the creation date.
   - Is Null—retrieves no records

   The system displays 0, 1, or 2 date fields depending on which operator you select. Enter values as required.
2. Enter additional values as required:
   - Check **Exclude Based On Criteria** if you want to filter **out** records that satisfy the criteria rather than displaying **only** the records that satisfy the criteria. This is the equivalent of adding a "Not" to the query logic.
   - **Filter Availability**: **All Studies** is the only option. The filter is available in all clinical data models that support it in the study, in all life cycle modes.
   - **Filter Name**: Enter a name for the filter if you want to save it to use in another session; that is, after you log out or work in a different study. Give it a descriptive name so that you will know if you want to use it or not.
3. Click **OK** to save. You must save the filter even if you want to use it temporarily.

Creating and Editing Discrepancy State Date Filters

Available only in the Default Listings page, the Discrepancy State Date filter displays only records that have one or more discrepancies whose most recent state change occurred during the time period specified. To create a Discrepancy State Date filter:

1. In the **Condition** drop-down, select an operator to be used to select the dates. The options are:
   - < Less than
   - <= Less than or equal to
   - <> Less than or greater than
   - = Equal to
   - > Greater than
   - >= Greater than or equal to
   - Between—a time duration between two dates
   - Is Not Null—retrieves all records; there is always a value for last-changed-date. If the record has not been updated, it is the same as the creation date.

- Is Null—retrieves no records

The system displays 0, 1, or 2 date fields depending on which operator you select. Enter values as required.

2. Enter additional values as required:
   - Check **Exclude Based On Criteria** if you want to filter **out** records that satisfy the criteria rather than displaying **only** the records that satisfy the criteria. This is the equivalent of adding a "Not" to the query logic.
   - **Filter Availability**: **All Studies** is the only option. The filter is available in all clinical data models that support it in the study, in all life cycle modes.
   - **Filter Name**: Enter a name for the filter if you want to save it to use in another session; that is, after you log out or work in a different study. Give it a descriptive name so that you will know if you want to use it or not.
3. Click **OK** to save. You must save the filter even if you want to use it temporarily.

Using the Find Feature

On the Listings pages you can use the Find feature to search on values in the columns of the current table.

1. Click the Find icon:  . The Find window opens, displaying an enterable field for all searchable columns in the table currently displayed in the Listings page.
2. Enter or select the values you want to search for, and click one:
   - **Any**: finds records that satisfy at least one Find criterion.
   - **All**: finds only records that satisfy all Find criteria.

   Alternatively, click **Advanced**. Any values you have already entered remain in effect. Advanced Find allows you to use operators on each column:
   - **Number operators**: Equals, Does not equal, Less than, Less than or equal to, Greater than, Greater than or equal to, Between, Not between, Is blank, Is not blank.
   - **Character (alphabetic) operators**: Starts with, Ends with, Equals, Does not equal, Less than, Less than or equal to, Greater than, Greater than or equal to, Between, Not between, Contains, Does not contain, Is blank, Is not blank.
   - **Date operators**: Equals, Does not equal, Before, After, On or before, On or after, Between, Not between, Is blank, Is not blank.

   Then enter a value or values to search on.
3. Click:
   - **Search** to find the criteria you have specified.
   - **Reset** to remove all criteria.
   - **Add Fields** (available in the Advanced window only): If you want to search for additional values in the same column, you can add another listing for the column and specify an operator and value in both the originally displayed field and the new one; for example, Site Country contains `US` or (Any) Site Country contains `CANADA`.

Creating Discrepancies Manually

If you find faulty or questionable data, you can create a discrepancy against it in any of the Listings pages. If you build a query that detects multiple data points with the same flaw, you can select all of them and create a discrepancy against them all at the same time.

Once created, a discrepancy is visible everywhere its underlying data point is visible, including in other data models and listings. Discrepancies against lab data can be exported to a file and sent to the lab.

To create a discrepancy:

1. In the Listings page, select the record or records against which you want to create a discrepancy. Select records with the same type of problem so that the discrepancy text can be the same for all selected records.
2. Click **Create Discrepancy**.
3. Enter values:
   - **Discrepancy**:Enter text that describes the problem with the data or the action required; for example, "Diastolic blood pressure is not within the specified range. Please confirm or correct."

     The system generates an ID beginning with "DMW" followed by numbers and adds it to the beginning of this text when you save. This ID is required for internal processing.
   - **Category**: Select from the list, according to your company's policy. You can use the category as a filter in the Discrepancies page.
   - **State**: Select **Open** or **Candidate**, according to your company's policy. Candidate discrepancies require manual review before being moved to an Open status and then being resolved.
   - **Data Model**: The system displays the current data model.
   - **Discrepancy Count**: They system displays the number of discrepancies currently open on the same data point.
   - **Append User Name**: Select the check box to append your username to the discrepancy text, which ensures that it is visible in InForm and at labs. Your username is stored in the database regardless of this setting and is displayed in the History pane of the Discrepancies page.

     This check box may be inactive, depending on a systemwide profile setting; see the *Oracle Life Sciences Data Hub System Administrator's Guide*.
4. Click **OK**.

Showing Discrepancies

To see all discrepancies associated with a particular data point, select the data point and click **Show Discrepancies**. For each discrepancy the system displays:

- Icon showing the current state of the discrepancy:
- **Discrepancy ID**
- **Discrepancy**: Text that describes the problem with the data or the action required.
- **Tag**: The tag currently applied to the discrepancy, if any.

- **Latest Comment**: People can communicate with comments about a discrepancy. The latest one is displayed.

To see the complete discrepancy history and to take action on discrepancies, go to the Discrepancies page.

Showing Flags

Records that have at least one flag assigned have a Flag icon displayed to the left of the row:

To see all flags assigned to a record and their values, select the record and click **Show Flags**. You can click the Name column header to order the flags alphabetically by name or the Value column header to order the flags by value.

InForm form and section states are imported as flags. You cannot change these flag assignments in Oracle DMW. Each InForm flag has two states: Yes and No (Y and N). They have the same names that they do in InForm plus the prefix "Inf_":

- **Inf_Started**
- **Inf_SDVReady**
- **Inf_SDVPartial**
- **Inf_SDVComplete**
- **Inf_Locked**
- **Inf_Frozen**
- **Inf_Signed**
- **Inf_Complete**
- **Inf_MissingItem**
- **Inf_HasQueries**
- **Inf_HasComments**
- **Inf_HasData**
- **Inf_NotDone**
- **Inf_Deleted**
- **Inf_DeletedDynamicForm**

Flags that do not being with "Inf" are custom flags created by your company. You can apply these flags to records as required. Validation checks can also apply them.

Flagging Data

Use flags to filter data and to track data states. You can apply any number of flags to a record, but only one flag value at a time. Your company creates flags as needed; see "Creating and Using Flags" on page 8-11.

Assign Flag To assign a flag to a record:

1. Select a record and click **Assign Flags**.
2. Select the flag to assign.

3. Select the flag state value to assign.

The system does not preserve a history of flag assignments.

> **Note:** The Assign Flag button is active only if flags have been defined for the current data model type and if you have the privileges required to assign flags to data.

Remove Flag To remove a flag:

1. Select a record and click **Assign Flags**.
2. Select the flag to remove.
3. Select **Clear Flag** at the bottom of the list of flag states.

Viewing Data Lineage

A single data point—for example, a subject's weight—appears first as loaded from InForm or a lab and then in subsequent *downstream* clinical data model. The column name may change from one model to the next—for example, from WT to WEIGHT—and the value may be converted to different units and used to derive other values, such as Body Mass Index (BMI). You can trace each data point's trail; for example, from the WEIGHT value in kilos in the Review model you can see the same value as WT in pounds in the InForm input model *upstream* and the BMI in the Analysis model downstream.

> **Note:** You can create a discrepancy against a data point in any clinical data model and the system will immediately display the discrepancy against the corresponding data points upstream and downstream. See "How the System Tracks Data Lineage" on page 5-24 for more information.

To view data lineage of a data point:

1. Select the data point.
2. Select one:
   - **View Source Data** to see upstream data that contributed to the selected data point. The system traces back as far as the input data model that receives data in the same structure as in the source system.
   - **View Target Data** to see downstream data that the selected data point contributes to.
   - **View Preferred Path** to see upstream data designated as the preferred path for sending related discrepancies back to the source system. If there are multiple source data points, one must be designated as preferred when the discrepancy is created. The system displays the discrepancy on the data points in the preferred path.
3. Click **Trace Data Lineage**. A pop-up window displays the selected data point in the top row. Expand its node to see the immediate source (upstream) or target (downstream) data point. Expand that node to see its immediate source or target data point, and so on.

If there are multiple immediate source or target data points, they are all displayed at the same level of indentation.

Trace Data Lineage Display For each data point in the path, the window displays:

- **Name**: The item or column name preceded by its table name and data model name: *data_model_name.table_name.column_name*.
- **Unlabeled Blinding Column**: If a dollar sign ($) is displayed, the data point may show mask values instead of real data.
- **Unlabeled Incomplete Lineage Column**: If an asterisk (*) is displayed, the system cannot display a data point and the rest of its path. This may be because:
  - The data point has been deleted or unmapped.
  - You do not have the privileges required to view the data point, either because you do not have access to view its table at all or because the table is fully or partially blinded and you do not have the required blinding-related privileges.
- **Data Type**: show the item's data type
- **Preferred Path**: **Yes** if the data point is on the preferred path or the only path; **No** if there are multiple paths and this data point is not on the preferred one.
- **Primary Key**: displays the primary key values of the record, in column order as *column1_name*:*column1_value*,*column2_name*:*column2_value*,*column3_name*:*column3_value* and so on.

Viewing Blinded Data

Sensitive data in certain columns, rows, cells, or whole tables may be blinded and require special privileges to view. If you have the required privileges, the **View Sensitive Data** button is active. If you click the button you see the real data. The system records each such blind break in a non-modifiable audit trial.

Exporting All to Excel

When you select **Export All to Excel** the system generates an .xls file including all records currently displayed, and puts it in the location you specify. You can send the file to the lab as is or open it first in Microsoft Excel.

Sort and filter records as required onscreen before exporting.

Note: In the Listings page, Export All to Excel includes all data currently displayed. This is different from the Discrepancies page, where the Export to Excel button includes all discrepancies.

Exporting All to CSV

When you select **Export All to CSV** the system generates a .csv (comma-separated values) file including all records currently displayed, and puts it in the location you specify. You can send the file to the lab as is or open it first in Microsoft Excel.

Sort and filter records as required onscreen before exporting.

12

Cleaning Data

This section contains the following topics:

A discrepancy is associated with a data point identified as incorrect or possibly incorrect. A discrepancy may originate as an InForm query or be identified in Oracle Health Sciences Data Management Workbench (Oracle DMW) by a validation (edit) check or manual query; see "Reviewing Data" on page 11-1.

Use the Discrepancies page to review discrepancies, add comments, apply an action to change their state and/or apply a tag to route them to other reviewers or to InForm or to a spreadsheet to send to the lab where lab data originated.

Your company can set up custom actions, states, categories, and tags to effectively create a custom workflow for discrepancy management. See "Configuring Your Discrepancy Workflow" on page 7-1 for information.

> **Note:** To review all data and raise manual discrepancies, go to the Listings page.

To view data discrepancies in a study:

1. Select the study and a lifecycle mode on the **Home** page.
2. Select **Discrepancies**.

Managing Discrepancies

In the Discrepancy Management pane you can review discrepancies, filter the display, and take action on one or more discrepancies.

A discrepancy is associated with its data point in all models, through both upstream and downstream transformations, but the system recognizes it as a single discrepancy. The table and column displayed for each discrepancy are those of the data point in the

model in which the discrepancy was created. You can see the model name in the Details section.

Acting on Multiple Discrepancies

You can select one or more discrepancies and do the same thing to all of them if the action is valid for all the selected discrepancies. You can use Ctrl+Click or Shift+Click to select multiple discrepancies and:

- **Set State**: Select the action you want to apply from the Set State drop-down list.

  The system displays only actions that apply a valid next state for all the selected discrepancies. If you select discrepancies that do not share a valid action, the system lists no actions. You must change your selection to a set of discrepancies that are in the same state. They may also need to have the same tag applied, depending on how your company uses the system; see "Creating a Custom Workflow by Creating Custom Actions" on page 7-6.

- **Add Comment**: Enter information that may be helpful in the review process and click **OK**.

  If the comment is intended for other people using Oracle DMW, select **Internal Comment** and click **OK**. The comment is then visible only within the system.

  If the comment is intended for lab personnel or InForm users, leave **Internal Comment** deselected and click **OK**. The comment is then included with the discrepancy when it is sent to its source: InForm or a lab.

- **Export to Excel**: When you select **Export to Excel** the system generates an .xls file of all discrepancies, and puts it in the location you specify.

Editing a Single Discrepancy

Select one discrepancy and click **Edit** to change:

- **Discrepancy**: Edit the text describing the problem with the data or action required.
- **Category**: Select a category to apply to the discrepancy. You can then use the category to filter for the discrepancy. A discrepancy can have only one category at a time.
- **Action**: Select an appropriate action to apply to the discrepancy. The system displays only valid actions. Actions can change the discrepancy's state and tag. Your company can set up its own actions.
- **Unread**: This field helps track whether an Oracle DMW user has viewed this discrepancy. Discrepancies created by InForm or validation checks start out with their Unread field checked. The system unchecks it when a user selects the discrepancy and then Display Full Record. Discrepancies created manually start out with their Unread field unchecked.
- **Auto Close**: If checked, validation checks with the required code can close this discrepancy. If unchecked, this discrepancy must be closed manually even if a validation check that would close it is executed.
- **Reason for Action**: If you apply an action to the discrepancy, enter a reason for the change.

Save or **Cancel** your changes.

Discrepancy Display

Most of the columns displayed across the Discrepancy Management pane are the same as those listed under Detail when a single discrepancy is selected; see "Viewing Discrepancy Details" on page 12-4 for descriptions.

You can reorder columns in the page by dragging them.

Icons

The icon displayed indicates the current state of the discrepancy. The state is also displayed in the State column and in the Details pane.

Table 12–1 Discrepancy Icons and Their Meaning


Candidate—Requires manual review before opening.


Open—Requires resolution.


Answered—Additional information or a data change is available. Requires manual review and resolution.


Closed—No further action is allowed.


Cancelled—No further action is allowed


Sent to InForm or a Spreadsheet (for a Lab) —No action is currently allowed on InForm discrepancies but all actions are allowed on lab data.

Refresh

Click to show any changes that have occurred since you opened the window. Other users may be working on the discrepancies you are viewing.

Filtering Discrepancies

You can filter discrepancies two ways:

- Filtering by Category
- Filtering by Multiple Criteria

Filtering by Category

Click a single category in the Categories pane on the left. The system displays only discrepancies assigned that category.

Filtering by Multiple Criteria

All current and closed or cancelled discrepancies are displayed by default, but you can apply filters to display only a subset of data.

Expand the Filter section on the left of the screen and choose to filter on all or any of the available options.

- If you select **All**, the system displays only discrepancies that meet all the filter conditions.
- If you select **Any**, the system displays all discrepancies that meet one or more filter conditions.

Then set one or more filter conditions:

- **Subject**: Select a single subject ID from the list.
- **Visit**: Select a single visit from the list.
- **Model**: Select the clinical data model in which the discrepancy was raised.
- **Table**: Select the table that contains the discrepant data you want to examine.

> **Note:** All models, tables, and items, and all states and tags are always available to select, regardless of your previous selections, so it is possible to select an invalid combination. If you select an invalid combination, the system displays no data.

- **Item**: Select the item (table column) that you want to examine.
- **State**: All discrepancies have a current assigned state—Candidate, Open, Cancelled, Answered, or Closed. You can specify a single state to view; for example, if your task is to determine whether to set Candidate discrepancies to Open or Cancelled, select **Candidate** as the filter value for State.
- **Tag**: Tags may serve as substates or as extra identifiers.
- **Category**: Select the category of discrepancies you want to view. Categories are set up by your company specifically to support filtering discrepancies.

Click **Search** to display discrepancies that meet your conditions or **Reset** to delete all conditions.

> **Note:** The system continues to apply your filters until you change them, even if you move to a different page.

Viewing Discrepancy Details

The Details pane displays all current information about a discrepancy, including the data value. You cannot make changes here.

- **Subject**: Subject ID.
- **Visit**: Name of the visit.
- **Discrepancy**: Discrepancy text. You can edit this in the Discrepancy Management pane by clicking Edit.
- **Table**: The database table that contains the discrepant value.
- **Model**: The clinical data model that contains the table.

> **Note:** If the model is "InForm" it means that the discrepancy originated as a query in InForm.

- **Item**: The table column that contains the discrepant value.
- **Value**: The discrepant data point value
- **State**: The state currently applied to the discrepancy: Candidate, Open, Cancelled, Answered, or Closed. You can change the state using either **Set State** or **Edit**, then **Action**—when appropriate.

- **Days in State**: The number of days the current state has been applied to the discrepancy.
- **Category**: The category assigned to the discrepancy, if any. You can filter by category. A discrepancy can have only one category at a time.
- **Created**: The date the discrepancy was created.
- **Modified**: The most recent date the discrepancy was modified.
- **Tag**: The tag assigned to the discrepancy, if any.
- **New Data**: If **Yes** ($YESNO$YES), the data value has been changed in the source system and the discrepancy has not been reviewed in Oracle DMW since then. When you move the focus to another discrepancy, the system considers that you have reviewed it and sets this value to **No**($YESNO$NO).

Viewing Discrepancy History

The History pane displays the history of the discrepancy—by default in reverse chronological order, with the most recent change at the top. The user who made the change and the date of the change are displayed. You cannot make changes to the data but you can change the display; see "Changing User Interface Display" on page 10-7.

Select **Refresh** to view any changes that have occurred since you selected the discrepancy or last refreshed.

Displaying the Full Record

This pane displays all column values for record containing the discrepant data point, including the current value and the original value, if it is different. The discrepant item is always listed at the top.

Two internal columns are displayed at the bottom:

- CDR$SKEY contains the primary key values of the record separated by tildes (~). This value is used internally to trace data lineage; see "How the System Tracks Data Lineage" on page 5-24.
- CDR$DUP_NUM is normally null, but can contain an integer that indicates the record's place relative to other records loaded from the same source with the same primary key; see "Supporting Duplicate Primary Keys in a Load" on page 9-7.

Select **Refresh** to view any changes that have occurred since you selected the discrepancy or last refreshed.

Exporting to Excel

Click **Export to Excel** to generate a .xls file with all discrepancies. You can rename the file and save it to your local computer.

Adding a Comment

Select one or more discrepancies and click **Add Comment** to add the same comment to all selected discrepancies.

Part III

Appendixes

This section contains the following topics:

A

Reference Information

This section contains the following topics:

Naming Objects

This section contains the following topics:

Make an object's name descriptive to help other users understand its purpose, but keep it short. Develop naming conventions; see "Customizable Naming Validation Package" on page A-3.

Avoid Special Characters and Reserved Words

To avoid problems, **do not use special characters** such as (& @ * $ | % ~) in object names, except for underscore (_). Also **do not use Oracle SQL or PL/SQL reserved words** in object names, especially the Oracle name, which defaults from the generic name. For lists of reserved words, see:

Oracle® DatabaseSQL Language Reference at
http://download.oracle.com/docs/cd/E11882_01/server.112/e17118.pdf

Oracle® Database PL/SQL Language Reference at
http://download.oracle.com/docs/cd/E11882_01/appdev.112/e17126.pdf

For the latest information, you can generate a list of all keywords and reserved words with the V$RESERVED_WORDS view, described in the *Oracle® Database Reference* at http://download.oracle.com/docs/cd/E11882_01/server.112/e17110.pdf.

Name Length: Keep It Short

Although many name fields allow 200 characters, short names work better both for display in the user interface and for technical reasons. For example, the Validation

Check (VC) Listings page displays columns using the format *data_model_name>table_name>column_name* in a single UI column. The shorter the name of the data model, table, and column, the easier it is for the user to see the whole value.

In addition, Windows has a maximum length of a file path of 256 characters that may be a problem for users who develop custom programs in Oracle LSH; see "Keep Container and Object Names Short for Integrated Development Environments" on page A-2.

Oracle and SAS names also have much smaller limits; see "Automatic Name Truncation" on page A-2.

Keep Container and Object Names Short for Integrated Development Environments

Although names can contain up to 200 characters, the maximum length of a file path is 256 characters in Windows. When you open an integrated development environment (IDE) such as SAS or run a SAS program on your personal computer, the system uses the actual full file path for source code definitions, table instances, and the SAS runtime script. If the full file path exceeds this length you get an error and cannot open the IDE or run the program. You can use a package to limit object name size or display an error when an object's file path is too long; see "Customizable Naming Validation Package" on page A-3.

The full file path begins with the username of the person who opens the IDE followed by the directory name `cdrwork`. It also includes the DMW_DOMAIN itself, the study, and any libraries or other subdomains you create. (The maximum number of subdomains is 9. This number is configurable in Oracle LSH; see the *Oracle Life Sciences Data Hub System Administrator's Guide*.)

- **Source Code definitions**. The path for Source Code definitions is: *username*>cdrwork>*Domain_name*>*Subdomain_name(s)* (0-9)>*Application_ Area_name*>*Program_definition_name*>*Source_Code_definition_nameversion_number*>*fileref*>*source_code_filename*
- **Table instances**. The path for Table instances is: *username*>cdrwork>*Domain_name*>*Subdomain_name(s)* (0-9)>*Application_ Area_name*>*Work_Area_name*>program instance name>program version>*Table_Descriptor_libname*>*Table_Descriptor_SAS_name*>
- **SAS runtime script**. The path for x is: *username*>cdrwork>*Domain_name*>*Subdomain_name(s)* (0-9)>*Application_ Area_name*>*Work_Area_name*>*Program_instance_name*>*version_number*>setup

Automatic Name Truncation

Names can contain up to 200 characters. However, the system populates the values for Oracle Name and SAS Name with the value you enter for Name stored in uppercase, truncating the **Oracle Name to 30 characters** and the **SAS Name to 32 characters**. You can change the Oracle and SAS Names.

When you create a new object by uploading a table, view, data set, or column, the system truncates the Oracle Name and also replaces the last two characters with the number 01 or, if an object of the same type with the same name already exists in the same container, the next highest number (in this case, 02).

Handling of Duplicate Names

The system enforces unique naming for each object type in the same container, whether the container is an organizational object—Domain, Application Area or Work Area—or a Report Set or Workflow. (Report Sets contain Programs. Workflows may

contain Programs, Load Sets, Report Sets, and/or Data Marts.) The system enforces unique names only within the immediate container; for example, you cannot have two Program instances named Program_A in the same Workflow, but you can have a Program_A in a Workflow and another Program_A directly in the Work Area that contains the Workflow.

If you try to create a second object of the same type and name in the same container, the system creates the object but automatically appends an underscore number to the name. If you add a third object of the same type and name, the system increments the number. For example, if you create a Program called Merge in Application Area 12345, and then create another Program called merge in the same Application Area, the system names the new Program Merge_1. If you create a third Program called Merge in the same Application Area, the system names the new Program Merge_2.

Naming Studies and Libraries

Studies and Libraries in Oracle DMW are Oracle LSH domain objects. If you plan to export a domain to another Oracle DMW instance, you may want to avoid using spaces in its name. Domain names with spaces must be entered with escape characters surrounding it in the Import Export Utility; for example: `\" domain name\"`. See the *Oracle Life Sciences Data Hub System Administrator's Guide* for more information.

Customizable Naming Validation Package

Object creation and modification code includes a call to a predefined validation package from every object name field. By default, this package performs no validation and returns a value of TRUE, allowing users to enter any name in the field. However, you can customize the package to enforce your own naming conventions, full path length, or other object attribute standards. See "Customizing Object Validation Requirements" in the *Oracle Life Sciences Data Hub System Administrator's Guide*.

Required Syntax for Table Metadata Text Files

You can define tables in a data model by uploading a .zip file that contains one .mdd file per table, each of which must have the required syntax described here. This is the only way you can automatically create tables that include all constraints and blinding attribute values.

> **Note:** You can create a table initially from a metadata file and subsequently load data into it from a SAS file.

The system uses the file name (without the extension) as the table name.

The system expects a set of column attribute values, optionally preceded by a row identifying the delimiter and a row defining Table attribute values, each of which must begin with a key word. A row beginning with dashes is treated as a comment. For example:

```
--This is a comment.
```

Additional information on each section of the required syntax follows Example A–1, "Metadata File".

Example A–1 Metadata File

```
lsh_delimiter = |
```

```
--This section is for the Table attributes
--Name, Description, Oracle Name, SAS Name, SAS Label, Process Type, Allow
Snapshot ?, Blinding Flag ?, Blinding Status, SAS Library Name, Is Target?, Target
as Dataset ?
lsh_table= AE|Adverse Events|AE|AE|AE|Reload|Yes|Yes|Blinded|target|yes|yes

--This section is for columns
--Name, Data Type, Length, Precision, Oracle Name, SAS Name, SAS Format,
Description, SAS Label, Nullable, Default Value,date format
Study|varchar2|15||Study|Study|$15.||Study|yes||
DCMNAME|varchar2|16||DCMNAME|DCMNAME|$16.||DCM Name|yes||
DCMSUBNM|varchar2|18||DCMSUBNM|DCMSUBNM|$18.||DCM Subset Name|yes||
SUBSETSN|number|3||SUBSETSN|SUBSETSN|3.||DCM Subset Number|yes||
DOCNUM|varchar2|20||DOCNUM|DOCNUM|$20.||Document Number|yes||
INVSITE|varchar2|10||INVSITE|INVSITE|$10.||Site|yes||
INV|varchar2|10||INV|INV|$10.||Investigator|yes||
PT|varchar2|10||PT|PT|$10.||Patient|yes||
ACCESSTS|date|||ACCESSTS|ACCESSTS|$datetime20.||Accessible TS|yes|1|MM-DD-YYYY
LOGINTS|date|||LOGINTS|LOGINTS|$datetime20.||Login TS|yes|1|DD-MON-YYYY
LSTCHGTS|date|||LSTCHGTS|LSTCHGTS|$datetime20.||Last Change TS|yes|1|MON-DD-YYYY
VISIT_NUMBER|number|10||VISIT_NUMBER|VISIT|10.||Visit|yes||
QUALIFYING_VALUE|varchar2|70||QUALIFYING_VALUE|QUALIFYINGV|$10.||Qualifying
Value|yes||

--This section is for constraints
--Name, Description,Constraint Type,Comma separated list of columns,Comma
separated list of values for check const,duplicate support for PK flag ,surrogate
key flag
CONSTRAINT|pk_1|pk|PRIMARYKEY|YES|YES|[Study]
CONSTRAINT|uk|uniq|UNIQUE|||[DCMNAME]
CONSTRAINT|pk_22|nuq|NONUNIQUE|||[DOCNUM]
CONSTRAINT|bmap_invsite|bitmap on INVSITE|BITMAP|||[INVSITE]
CONSTRAINT|check_inv|check on INV|CHECK|||[INV]|{1|2|3|4}
```

Delimiter The first row defines the delimiter used in the file. If not specified, Oracle LSH treats it as a comma delimited file. The delimiter row must begin with: `lsh_delimiter=`

Table Attributes The second row lists the table attributes required in the file. The Table attribute row must begin with: `lsh_table=`

If the second row is not present or contains null values, the system assumes that the file name (without extension) is the Table Name and follows the normal Oracle LSH default behavior for the attribute values. The attributes and their required order in the file are: `Name, Description, Oracle Name, SAS Name, SAS Label, Process Type, Allow Snapshot?, Blinding Flag?, Blinding Status, SAS Library Name, Is Target?, Target as Dataset?`

Some attributes have associated reference codelists and allow either the actual values for the associated reference codelist (RC) columns or the decode values defined in the "Meaning" attribute of the _RC lookup. For example, "select meaning from cdr_lookups where lookup_code='< RC>' so that **YES** or **Yes** or **$YESNO$YES** are acceptable values.

The table below outlines the applicable values for each attribute that has an associated reference codelist. For more information, see the *Oracle Life Sciences Data Hub Application Developer's Guide*.

Note: Processing types that require audit keys are not supported.

Table A–1 Attributes with Reference Codelist Values

Attribute	Values
Processing Type	UOW, Reload, Staging with Audit, Staging without Audit, Transactional High Throughput, Transactional without Audit
Allow Snapshot	Yes, No
Blinding Flag	Yes: The table may contain sensitive data at some point in time. No: The table will never contain blinded data.
Blinding Status	If Blinding Flag is set to Yes, the data may have a status of either Blinded or Unblinded. (Users can set Blinding Status to Authorized but not in the table itself.) If Blinding Flag is set to No, the data must have a Blinding Status of Not Applicable.
Is Target	Yes, No
Target as dataset	Yes, No
Data Type	Date, Number, Varchar2
Nullable	Yes, No

Columns Subsequent rows must contain the column, or variable, attributes with each represented by a new row in the text file with attributes. The position is determined by the order in which the column /variable rows are processed. For example, in a comma delimited file: `Name, Data Type, Length, Precision, Oracle Name, SAS Name, SAS Format, Description, SAS Label, Nullable, Default Value, Date Format`

Oracle LSH validation rules apply to the column or variable attributes. The operation uses Oracle LSH default values if invalid values are provided for any of the attributes.

Constraints The final section is for constraints. Each constraint must have its own row starting with the string `CONSTRAINT` followed by values you supply for the constraint name, description, and constraint type, followed by other values depending on the constraint type.

The required values for Constraint Type are, with the remaining values required for each type:

- `PRIMARYKEY` | Duplicate PK Support Flag (`YES` or `NO`) | Surrogate Key Flag (`YES` or `NO`) | Comma-separated list of columns in key

  For example: `CONSTRAINT|pk_1|pk|PRIMARYKEY|YES|YES|[SUBJID`
- `UNIQUE` | Comma-separated list of columns in key
- `NONUNIQUE` | Comma-separated list of columns in key
- `BITMAP` | Comma-separated list of columns in key
- `CHECK` | Comma-separated list of allowed values

Example A–2 Constraint Metadata

```
--Name, Description,Constraint Type,Comma separated list of columns,Comma
separated list of values for check const,duplicate support for PK flag ,surrogate
```

```
key flag
CONSTRAINT|pk_1|pk|PRIMARYKEY|YES|YES|[Study]
CONSTRAINT|uk|uniq|UNIQUE|||[DCMNAME]
CONSTRAINT|pk_22|nuq|NONUNIQUE|||[DOCNUM]
CONSTRAINT|bmap_invsite|bitmap on INVSITE|BITMAP|||[INVSITE]
CONSTRAINT|check_inv|check on INV|CHECK|||[INV]|{1|2|3|4}
```

Object Ownership

Figure A–1, "Oracle DMW Object Ownership" shows the hierarchy of object containership in Oracle DMW.

Effects of User Group Assignment to Objects

A user group assigned to an object is automatically assigned by *inheritance* to all objects contained in it. You can explicitly revoke the user group from any owned object, which also revokes the assignment from objects contained in that object.

When you assign a user group to an object, you must select either Metadata, Development, QC, or Production. The last three choices provide access to the three lifecycle areas, shown as their actual object names in the diagram: DEV_APP_AREA, QC_APP_AREA, and PROD_APP_AREA or to an object within the selected lifecycle area. *Metadata* provides access to object definition(s).

As in Oracle LSH, object definitions are pure metadata. They must be installed as object instances in a Work Area schema to be used:

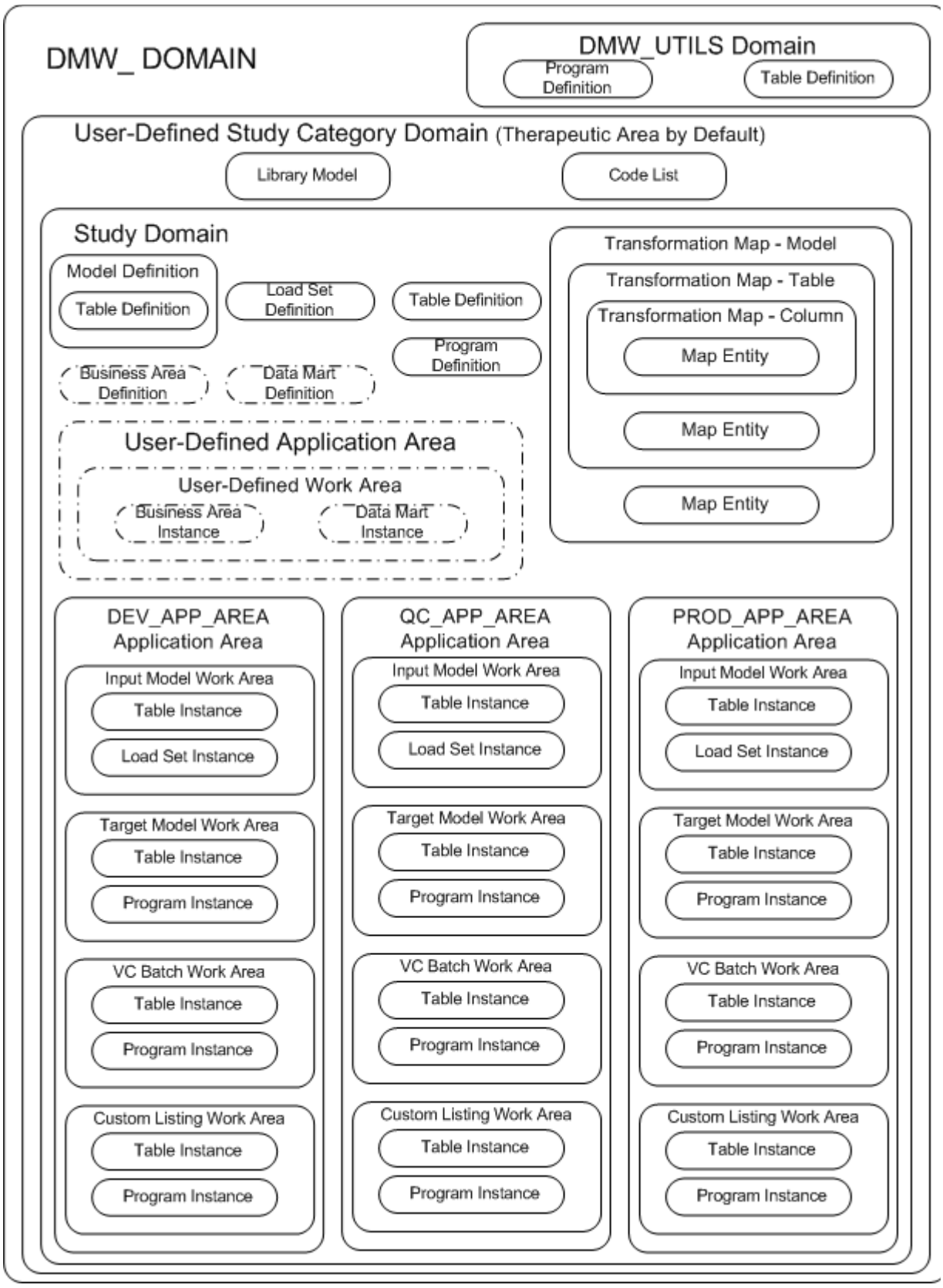
- When you install a clinical data model in the Development lifecycle area, the system creates a database schema for the model and creates a database table for each table defined in the model.
- When you install a transformation or a validation check batch in the Development lifecycle area, the system creates a PL/SQL package for each programs contained in the transformation or validation check batch.

Only then can tables actually hold data, transformations and validation checks read and write data.

Users must have access to the objects in the lifecycle Work Areas to install and execute objects. To modify objects such as models, transformations, and validation checks, they must have access to the appropriate lifecycle Work Area *and* the object metadata.

Note: The actions a user in a user group can take on an object to which the user group is assigned depend on the privileges defined for the role the user has in the group. See "Predefined Object Security Roles" on page B-3.

Figure A–1 Oracle DMW Object Ownership



Oracle DMW Object Hierarchy

Oracle DMW uses Oracle LSH object types and a few new object types created for use in Oracle DMW. The following diagram shows only primary objects; the relationships with secondary, or component, objects such as parameters, source code, variables, and columns is the same as in Oracle LSH and is documented in the *Oracle Life Sciences Data Hub Application Developer's Guide*.

The hierarchy used in Oracle DMW, shown in Figure A–1, "Oracle DMW Object Ownership", is described in the following sections.

DMW Domain

Within the Oracle LSH instance domain, one of which is automatically created in each installation of Oracle LSH, the DMW domain is also automatically created during product installation. It contains all user-created objects in Oracle DMW.

The DMW domain is an LSH object of type Domain.

DMW Utilities

The DMW_Utils domain is automatically created during product installation inside the DMW domain. It is designed to hold all custom programs and functions created for use with Oracle DMW by customers. You can create domains within it to organize the programs you create, using the Oracle LSH user interface or public APIs.

You create programs in the Oracle LSH user interface or with public APIs.

Study Category Domains

You define additional domains in the DMW domain to categorize studies and to hold library clinical data models and code lists for each study category. By default the categories are called therapeutic areas, but you can change this; see Section , "Setting Up Library and Study Categories."

You create these domains in the Oracle LSH user interface or with public APIs.

Study Domains

An Oracle DMW study is a domain. When you create a study in the Oracle DMW user interface, the system creates a domain inside the study category domain you indicate.

Lifecycle Areas

When you create a study, the system automatically creates three lifecycle areas; one each for Development, Quality Control, and Production. These are LSH objects of type Application Area and their names are DEV_APP_AREA, QC_APP_AREA, and PROD_APP_AREA respectively.

Clinical Data Models

When you create a clinical data model, the system puts its object definitional metadata directly in the study domain. When you install the model, the system creates a Work Area object in the current lifecycle area and a database schema for it containing a database table for each table definition in the model definition.

Load Sets

When you create an input clinical data model of type File, the system automatically creates an Oracle LSH Load Set object which it then uses to actually load data from a

file to the model in Oracle DMW. When you install the model, the system also installs the Load Set.

Programs

When you create a custom program for a transformation or validation check, or when the system generates a program for a transformation or validation check, the Program definition is contained directly in the study and installed in the relevant clinical data model work area:

Transformation Maps

Transformation map metadata is contained directly within the study domain. Each transformation map has three levels: the model level contains all table-level maps, and the table-level maps contain column-level maps.

Business Areas

If you want to use a data visualization tool to view Oracle DMW data, create an Oracle LSH Business Area object and link it to the tables you want to view. See "Setting Up a Data Visualization Tool" on page 8-19

Data Marts

If you want to use export Oracle DMW data to a file, create an Oracle LSH Data Mart object and link it to the tables whose data you want to export. See the *Oracle Life Sciences Data Hub Application Developer's Guide* chapter on data marts for information.

B

Predefined Roles

You can use shipped, predefined object security and application roles to simplify security setup.

See also "Object Ownership" on page A-6.

Predefined Oracle DMW Application Roles

Each user must have at least one of the following application roles to access the Oracle DMW user interface.

DMW_STUDY_MANAGER

DMW_STUDY_MANAGER is intended for users who run data loads, transformations, and validation checks. It provides access to the Home, Study Configuration, Listings, and Discrepancies pages.

Users with this role can create, modify, and remove studies in the Home page.

DMW_STUDY_CONFIG

DMW_STUDY_CONFIG is intended for users who set up studies by defining models, transformations, and validation checks. It provides access to the Home, Study Configuration, Listings, and Discrepancies pages.

Users with this role can create, modify, and remove studies in the Home page.

> **Note:** There is no functional difference between the application roles DMW_STUDY_MANAGER and DMW_STUDY_CONFIG.

DMW_STUDY_CONSUMER

DMW_STUDY_CONSUMER is intended for users who need to review data and raise discrepancies. It provides access to the Home, Listings, and Discrepancies pages.

DMW_LIB_ADMIN

DMW_LIB_ADMIN is intended for users who create and modify library models and code lists. It provides access only to the Library page.

DMW_SYS_ADMIN

DMW_SYS_ADMIN is intended for users who do administrative tasks including setting up data sources and defining objects used across studies, such as categories, flags, and tags. It provides access only to the Administration page.

Predefined Blinding-Related Roles

Some Oracle DMW users should have blinding-related privileges in order to view blinded data and/or unblind data. This requires both a blinding-related application role and blinding-related object security privileges through a role in a user group assigned to the object, plus normal object security access to the object.

Since the Predefined Object Security Roles all include object-level access to blinded data, it is very important that you assign these application roles **only** to users who require them.

For more information about blinding data in Oracle LSH, see the *Oracle Life Sciences Data Hub Implementation Guide* and the *Oracle Life Sciences Data Hub Application Developer's Guide*.

LSH Data Blind Break User

Only the LSH Data Blind Administrator can assign this role to users. This role does not provide access to any parts of the user interface.

Users with this application role can do the following, if they have normal object security access and the blinding-related object security privilege noted:

- Run a job on Tables with a Blinding Status of Blinded that displays the real data, not the dummy data (also requires an object security role with the Blind Break operation on Table instances)
- View the output(s) generated by a job run on real, blinded data (also requires an object security role with the Blind Break operation on outputs)

LSH Data Unblind User

Only the LSH Data Blind Administrator can assign this role to users. This role does not provide access to any parts of the user interface.

Users with this application role can do the following, if they have normal object security access and the blinding-related object security privilege noted:

- Permanently unblind data in Table instances (also requires an object security role with the Unblind operation on Table instances)
- Change the status of an output from Blinded to Unblinded (also requires an object security role with the Unblind operation on outputs)

> **Note:** The LSH Data Unblind User application role is not required in conjunction with the Read Unblind operation on either Table instances or outputs. Users can have an object security role that includes the Read Unblind operation on Table instances (to run a job on unblinded Table instances) and/or outputs (to view the results of such a job) without having the LSH Data Unblind User application role as well.

Predefined Object Security Roles

In this appendix, each set of permissions is organized by the application for which the object type was originally created: Oracle LSH or Oracle DMW. Each permission granted by the role is presented in the format **object type: operation**.

> **Note:** Each role includes all blinding-related privileges. However, in order to perform blinding-related operations users must have blinding-related application roles as well; see "Predefined Blinding-Related Roles" on page B-2.

The roles are:

- DMW_STUDY_DEVELOPER
- DMW_STUDY_QC
- DMW_STUDY_PROD
- DMW_STUDY_ADMIN
- DMW_STUDY_INST_ACCESS
- DMW_STUDY_INFORM_CONFIG

DMW_STUDY_DEVELOPER

The DMW_STUDY_DEVELOPER object security role is intended for users who will configure studies by creating data models and creating and running transformations and validation checks. To test their work, they also need the ability to load data into the development lifecycle area from InForm and files, create custom listings, create and act on discrepancies, and assign flags to records. Blinding-related privileges are also included in this role.

Users with this role should have the DMW_STUDY_MANAGER or DMW_STUDY_CONFIG application role. If they need to create library data models they should also have the DMW_LIB_ADMIN application role.

DMW_STUDY_DEVELOPER allows users to perform the following operations on the specified objects.

Clinical Data Models and Query Builder

Users assigned the DMW_STUDY_DEVELOPER role have access to the following operations for clinical data models and the query builder.

Oracle LSH Object Operations

Domain: Create Data Model
Domain: Create Load Set
Domain: Create Variable
Domain: View
Application Area: Create Work Area
Application Area: View
Load Set: View
Load Set: Modify
Load Set: Modify Val Status to QC
Load Set: Modify Val Status to PROD
Variable: Modify

Variable: Modify Val Status to QC
Variable: Modify Val Status to PROD
Table: View
Table: Modify
Work Area: Create Table Instance
Work Area: Create Load Set Instance
Work Area: Modify
Work Area: Modify Val Status to QC
Work Area: Modify Val Status to PROD
Work Area: Delete
Work Area: Deploy (Install)
Work Area: View
Load Set Instance: Modify
Load Set Instance: Modify Val Status to QC
Load Set Instance: Modify Val Status to PROD
Table Instance: Modify
Table Instance: Remove
Table Instance: Modify Val Status to QC
Table Instance: Modify Val Status to PROD

Oracle DMW Object Operations

Data Model: Modify
Data Model: Delete
Data Model: Publish Dev (used to install the Development Work Area)
Data Model: Modify Supporting Info
Data Model: Modify Val Status to QC
Data Model: Modify Val Status to PROD
Data Model: Create Private Listing
Data Model: Create Public Listing

Blinding

Users assigned the DMW_STUDY_DEVELOPER role have access to the following operations for blinded data.

Oracle LSH Object Operations

Table Instance: ReadData
Table Instance: BlindBreak
Table Instance: Unblind
Table Instance: ReadUnblind

Filters

Users assigned the DMW_STUDY_DEVELOPER role have access to the following operation for filters.

Oracle DMW Object Operations

Filter: Mark as Public in Dev

Data Lineage Tracing

Users assigned the DMW_STUDY_DEVELOPER role have access to the following operations for data lineage tracing.

Oracle DMW Object Operations

Table Instance: View CTX Data Lineage

Transformations and Query Builder

Users assigned the DMW_STUDY_DEVELOPER role have access to the following operations for transformations and the query builder.

Oracle LSH Object Operations

Application Area: Create Work Area
Execution Setup: Modify Val Status to PROD
Execution Setup: Modify Val Status to QC
Execution Setup: Submit
Domain: Create Program
Domain: Create Variable
Program: Modify Val Status to PROD
Program: Modify Val Status to QC
Program: Modify
Program: Remove
Program Instance: Modify Val Status to PROD
Program Instance: Modify Val Status to QC
Program Instance: Create Execution Setup
Program Instance: Modify
Program Instance: Remove
Table: Modify Val Status to PROD
Table: Modify Val Status to QC
Table Instance: Modify Val Status to PROD
Table Instance: Modify Val Status to QC
Table Instance: BlindBreak
Table Instance: ReadData
Table Instance: ReadUnblind
Table Instance: Remove
Table Instance: Unblind
Table Instance: Modfiy
Table: Modify
Table: Remove
Table: View
Work Area: Create Program Instance
Work Area: Create Table Instance
Work Area: Deploy (Install)
Work Area: Modify
Work Area: Delete
Work Area: Modify Val Status to PROD
Work Area: Modify Val Status to QC

Oracle DMW Object Operations

Domain: Create Transformation Map
Data Model: Load Data in Development
Transformation Map: Modify
Transformation Map: Delete
Transformation Map: View
Transformation Map: Modify Validation Status QC
Transformation Map: Modify Validation Status Production
Transformation Map: Modify Supporting Info
Transformation Map: Deploy (Install) in Development
Transformation Map: Modify Security

Transformation Map: Classify

Discrepancies and Flags

Users assigned the DMW_STUDY_DEVELOPER role have access to the following operations for discrepancies and flags.

Oracle DMW Object Operations

Table Instance: Create Manual Discrepancy
Table Instance: Close Discrepancy
Table Instance: Send Discrepancy to InForm
Table Instance: Send Discrepancy to Spreadsheet
Table Instance: Reopen Discrepancy
Table Instance: Answer Discrepancy
Table Instance: Cancel Discrepancy
Table Instance: Add Comment to Discrepancy
Table Instance: Edit Discrepancy
Table Instance: Show Discrepancies
Table Instance: Create Data Amendment
Table Instance: Modify Data Amendment
Table Instance: Cancel Data Amendment
Table Instance: Assign Flags

Validation Checks

Users assigned the DMW_STUDY_DEVELOPER role have access to the following operations for validation checks.

Oracle LSH Object Operations

Execution Setup: Modify Val Status to PROD
Execution Setup: Modify Val Status to QC
Execution Setup: Submit
Domain: Create Program
Program: Modify Val Status to PROD
Program: Modify Val Status to QC
Program: Modify
Program: Remove
Program Instance: Modify Val Status to PROD
Program Instance: Modify Val Status to QC
Program Instance: Create Execution Setup
Program Instance: Modify
Program Instance: Remove
Table: Modify Val Status to PROD
Table: Modify Val Status to QC
Table: Modify
Table: Remove
Work Area: Create Program Instance

Oracle DMW Object Operations

Data Model: Create VC Batch
VC Batch: Modify
VC Batch: Delete
VC Batch: View
VC Batch: Modify Validation Status QC
VC Batch: Modify Validation Status Production

VC Batch: Modify Supporting Info
VC Batch: Deploy (Install) in Development
VC Batch: Submit in Development
VC Batch: Submit Individual VC in Development
VC Batch: View Listings in Development

InForm

Users assigned the DMW_STUDY_DEVELOPER role have access to the following operations for InForm clinical data models.

Oracle LSH Object Operations

Load Set Instance: Create Execution Setup
Program Instance: Create Execution Setup
Application Area: Create Program
Program: Modify
Program Instance: Modify

Oracle DMW Object Operations

Data Model: Load Data in Development
Data Model: Load InForm Metadata
Web Service Location: Modify Web Service Location
Web Service Location: Create Connection

File Watcher

Users assigned the DMW_STUDY_DEVELOPER role have access to the following operations for the File Watcher.

Oracle DMW Object Operations

Data Model: File Watcher Config
Data Model: Data Load DEV

DMW_STUDY_QC

The DMW_STUDY_QC object security role is intended for users who will conduct formal testing of study objects including data models, transformations, and validation checks. They also need the ability to load data into the Quality Control lifecycle area from InForm and files, create custom listings, create and act on discrepancies, and assign flags to records. Blinding-related privileges are also included in this role.

Users with this role should have the DMW_STUDY_MANAGER or DMW_STUDY_CONFIG application role.

DMW_STUDY_QC allows users to perform the following operations on the specified objects.

Clinical Data Models and Query Builder

Users assigned the DMW_STUDY_QC role have access to the following operations for clinical data models and the query builder.

Oracle LSH Object Operations

Work Area: Create Table Instance
Work Area: Create Load Set Instance
Work Area: Modify

Work Area: Modify Val Status to QC
Work Area: Modify Val Status to PROD
Work Area: Deploy (Install)
Table Instance: Modify
Table Instance: Remove
Table Instance: Modify Val Status to QC
Table Instance: Modify Val Status to PROD
Load Set Instance: Modify
Load Set Instance: Modify Val Status to QC
Load Set Instance: Modify Val Status to PROD

Oracle DMW Object Operations

Data Model: View
Data Model: Publish QC (used to install the QC Work Area)
Data Model: Modify Supporting Info
Data Model: Modify Val Status to QC
Data Model: Modify Val Status to PROD
Data Model: Create Private Listing
Filter: Mark Filter Public QC

Blinding

Users assigned the DMW_STUDY_QC role have access to the following operations for blinded data.

Oracle LSH Object Operations

Table Instance: ReadData
Table Instance: BlindBreak
Table Instance: Unblind
Table Instance: ReadUnblind

Discrepancies

Users assigned the DMW_STUDY_QC role have access to the following operations for discrepancies.

Oracle DMW Object Operations

Table Instance: Create Manual Discrepancy
Table Instance: Close Discrepancy
Table Instance: Send Discrepancy to InForm
Table Instance: Send Discrepancy to Spreadsheet
Table Instance: Reopen Discrepancy
Table Instance: Answer Discrepancy
Table Instance: Cancel Discrepancy
Table Instance: Add Comment to Discrepancy
Table Instance: Edit Discrepancy
Table Instance: Show Discrepancies
Table Instance: Create Data Amendment
Table Instance: Modify Data Amendment
Table Instance: Cancel Data Amendment

Filters

Users assigned the DMW_STUDY_QC role have access to the following operation for filters.

Oracle DMW Object Operations

Filter: Mark as Public in QC

Flags

Users assigned the DMW_STUDY_QC role have access to the following operations for flags.

Oracle DMW Object Operations

Table Instance: Assign Flags

Data Lineage Tracing

Users assigned the DMW_STUDY_QC role have access to the following operations for data lineage tracing.

Oracle DMW Object Operations

Table Instance: View CTX Data Lineage

Validation Checks

Users assigned the DMW_STUDY_QC role have access to the following operations for validation checks.

Oracle LSH Object Operations

Execution Setup: Modify Val Status to PROD
Execution Setup: Modify Val Status to QC
Execution Setup: Submit
Program: Modify Val Status to PROD
Program: Modify Val Status to QC
Program Instance: Modify Val Status to PROD
Program Instance: Modify Val Status to QC
Program Instance: Create Execution Setup
Table: Modify Val Status to PROD
Table: Modify Val Status to QC
Work Area: Create Program Instance

Oracle DMW Object Operations

VC Batch: View
VC Batch: Modify Validation Status QC
VC Batch: Modify Validation Status Production
VC Batch: Modify Supporting Info
VC Batch: Deploy (Install) in Quality Control
VC Batch: Submit in Quality Control
VC Batch: Submit Individual VC in Quality Control
VC Batch: View Listings in Quality Control

InForm

Users assigned the DMW_STUDY_QC role have access to the following operations for InForm clinical data models.

Oracle LSH Object Operations

Load Set Instance: Create Execution Setup

Program Instance: Create Execution Setup
Application Area: Create Program
Program: Modify
Program: View
Program Instance: Modify
Program Instance: View
Work Area: Clone
Program Instance: Modify Validation QC
Program: Modify Validation QC

Oracle DMW Object Operations

Data Model: Load Data in QC

File Watcher

Users assigned the DMW_STUDY_QC role have access to the following operations for the File Watcher.

Oracle DMW Object Operations

Data Model: FWR Configuration
Data Model: Data Load QC

Transformations and Query Builder

Users assigned the DMW_STUDY_QC role have access to the following operations for transformations and the query builder.

Oracle LSH Object Operations

Execution Setup: Modify Val Status to PROD
Execution Setup: Modify Val Status to QC
Execution Setup: Submit
Program: Modify Val Status to PROD
Program: Modify Val Status to QC
Program Instance: Modify Val Status to PROD
Program Instance: Modify Val Status to QC
Program Instance: Create Execution Setup
Table: Modify Val Status to PROD
Table: Modify Val Status to QC
Table Instance: Modify Val Status to PROD
Table Instance: Modify Val Status to QC
Table Instance: BlindBreak
Table Instance: ReadData
Table Instance: ReadUnblind
Work Area: Create Program Instance
Work Area: Create Table Instance
Work Area: Deploy (Install)
Work Area: Modify
Work Area: Modify Val Status to PROD
Work Area: Modify Val Status to QC

Oracle DMW Object Operations

Data Model: Load Data in Quality Control
Transformation Map: View
Transformation Map: Modify Validation Status QC

Transformation Map: Modify Validation Status Production
Transformation Map: Modify Supporting Info
Transformation Map: Deploy (Install) in Quality Control
Transformation Map: Modify Pvt Listing
Transformation Map: Delete Pvt Listing
Transformation Map: View Public Listing

DMW_STUDY_PROD

The DMW_STUDY_PROD role is intended for users who will work in the Production environment, including: promoting and installing clinical data models, transformations, and validation checks, loading production data from InForm and files, running transformations and validation checks on production data, reviewing data and creating custom listings, and creating and acting on discrepancies. Blinding-related privileges are also included in this role.

Users with this role should have the DMW_STUDY_MANAGER application role.

DMW_STUDY_PROD allows users to perform the following operations on the specified objects.

Clinical Data Models and Query Builder

Users assigned the DMW_STUDY_PROD role have access to the following operations for clinical data models and the query builder.

Oracle LSH Object Operations

Work Area: Create Table Instance
Work Area: Create Loadset Instance
Work Area: Modify
Work Area: Modify Val Status to QC
Work Area: Modify Val Status to PROD
Work Area: Deploy (Install)
Table Instance: Modify
Table Instance: Remove
Table Instance: Modify Val Status to QC
Table Instance: Modify Val Status to PROD
Load Set Instance: Modify
Load Set Instance: Modify Val Status to QC
Load Set Instance: Modify Val Status to PROD

Oracle DMW Object Operations

Data Model: View
Data Model: Publish Prod (used to install the Production Work Area)
Data Model: Modify Supp Info
Data Model: Modify Val Status to QC
Data Model: Modify Val Status to PROD
Data Model: Create Private Listing
Filter: Mark Filter Public PROD

Blinding

Users assigned the DMW_STUDY_PROD role have access to the following operations for blinded data.

Oracle LSH Object Operations

Table Instance: BlindBreak
Table Instance: Unblind
Table Instance: ReadUnblind
Table Instance: ReadData

Discrepancies

Users assigned the DMW_STUDY_PROD role have access to the following operations for discrepancies.

Oracle DMW Object Operations

Table Instance: Create Manual Discrepancy
Table Instance: Close Discrepancy
Table Instance: Send Discrepancy to InForm
Table Instance: Send Discrepancy to Spreadsheet
Table Instance: Reopen Discrepancy
Table Instance: Answer Discrepancy
Table Instance: Cancel Discrepancy
Table Instance: Add Comment to Discrepancy
Table Instance: Edit Discrepancy
Table Instance: Show Discrepancies
Table Instance: Create Data Amendment
Table Instance: Modify Data Amendment
Table Instance: Cancel Data Amendment

Filters

Users assigned the DMW_STUDY_PROD role have access to the following operation for filters.

Oracle DMW Object Operations

Filter: Mark as Public in Prod

Flags

Users assigned the DMW_STUDY_PROD role have access to the following operations for flags.

Oracle DMW Object Operations

Table Instance: Assign Flags

Data Lineage Tracing

Users assigned the DMW_STUDY_PROD role have access to the following operations for data lineage tracing.

Oracle DMW Object Operations

Table Instance: View CTX Data Lineage

Validation Checks

Users assigned the DMW_STUDY_PROD role have access to the following operations for validation checks.

Oracle LSH Object Operations

Execution Setup: Modify Val Status to PROD
Execution Setup: Modify Val Status to QC
Execution Setup: Submit
Program: Modify Val Status to PROD
Program: Modify Val Status to QC
Program Instance: Modify Val Status to PROD
Program Instance: Modify Val Status to QC
Program Instance: Create Execution Setup
Table: Modify Val Status to PROD
Table: Modify Val Status to QC
Work Area: Create Program Instance

Oracle DMW Object Operations

VC Batch: View
VC Batch: Modify Validation Status QC
VC Batch: Modify Validation Status Production
VC Batch: Modify Supporting Info
VC Batch: Deploy (Install) in Production
VC Batch: Submit in Production
VC Batch: Submit Individual VC in Production
VC Batch: View Listings in Production

InForm

Users assigned the DMW_STUDY_PROD role have access to the following operations for InForm data.

Oracle LSH Object Operations

Load Set Instance: Create Execution Setup
Program Instance: Create Execution Setup
Application Area: Create Program
Program: Modify
Program Instance: Modify
Work Area: Clone
Program Instance: Modify Validation PROD
Program: Modify Validation PROD

Oracle DMW Object Operations

Data Model: Load Data in Production

File Watcher

Users assigned the DMW_STUDY_PROD role have access to the following operations for the File Watcher.

Oracle DWM Object Operations

Data Model: FWR Configuration
Data Model: Data Load PROD

Transformations and Query Builder

Users assigned the DMW_STUDY_PROD role have access to the following operations for transformations and the query builder.

Oracle LSH Object Operations

Execution Setup: Modify Val Status to PROD
Execution Setup: Modify Val Status to QC
Execution Setup: Submit
Program: Modify Val Status to PROD
Program: Modify Val Status to QC
Program Instance: Modify Val Status to PROD
Program Instance: Modify Val Status to QC
Program Instance: Create Execution Setup
Table: Modify Val Status to PROD
Table: Modify Val Status to QC
Table Instance: Modify Val Status to PROD
Table Instance: Modify Val Status to QC
Table Instance: BlindBreak
Table Instance: ReadData
Table Instance: ReadUnblind
Work Area: Create Program Instance
Work Area: Create Table Instance
Work Area: Deploy (Install)
Work Area: Modify
Work Area: Modify Val Status to PROD
Work Area: Modify Val Status to QC

Oracle DMW Object Operations

Data Model: Load Data in Production
Transformation Map: View
Transformation Map: Modify Validation Status QC
Transformation Map: Modify Validation Status Production
Transformation Map: Modify Supporting Info
Transformation Map: Deploy (Install) in Production
Transformation Map: Modify Pvt Listing
Transformation Map: Delete Pvt Listing
Transformation Map: View Public Listing

DMW_STUDY_ADMIN

The DMW_STUDY_ADMIN object security role is intended for superusers. It includes the object-related roles that DMW_STUDY_DEVELOPER and DMW_STUDY_QC have, plus administrator privileges including creating studies and study categories, setting up connections to InForm and File Watcher locations, and assigning user groups to objects such as data models, transformations, and validation checks. Blinding-related privileges are also included in this role.

Users with this role should have the DMW_STUDY_MANAGER or DMW_SYS_ ADMIN application role.

DMW_STUDY_ADMIN allows users to perform the following operations on the specified objects.

Clinical Data Models and Query Builder

Users assigned the DMW_STUDY_ADMIN role have access to the following operations for clinical data models and the query builder.

Oracle LSH Object Operations

Domain: Create SubDomain
Domain: Modify

Domain: Delete
Domain: Manage Security
Domain: Create Load Set
Domain: Create Variable
Domain: Create Program
Domain: Create Data Model
Domain: Create Transformation Map
Table Instance: Manage Security
Application Area: Manage Security
Application Area: Create Work Area
Application Area: Create Program
Work Area: Manage Security
Work Area: Create Table Instance
Work Area: Create Load Set Instance
Work Area: Modify
Work Area: Modify Val Status to QC
Work Area: Modify Val Status to PROD
Work Area: Delete
Work Area: Deploy (Install)
Work Area: Create Program Instance
Variable: Modify
Variable: Modify Val Status to QC
Variable: Modify Val Status to PROD

Oracle DMW Object Operations

Data Model: Create Private Listing
Data Model: Create Public Listing
Data Model: Modify
Data Model: Delete
Data Model: Deploy (Install) Model in Dev/QC/Prod
Data Model: Modify Supporting Info
Data Model: Modify Val Status to QC
Data Model: Modify Val Status to PROD
Data Model: Create VC Batch
Data Model: Load Data in Dev/QC/Prod
Data Model: Manage Security
Data Model: Configure InForm
Data Model: Load InForm Metadata
Filter: Mark Filter Public Dev
Filter: Mark Filter Public QC
Filter: Mark Filter Public Prod

Discrepancies

Users assigned the DMW_STUDY_ADMIN role have access to the following operations for discrepancies.

Oracle DMW Object Operations

VC Batch: Modify
VC Batch: Delete
VC Batch: View
VC Batch: Modify Validation Status Production
VC Batch: Modify Supporting Info
VC Batch: Deploy (Install) in Development/QC/Prod
VC Batch: Submit in Developmene/QC/Prod

VC Batch: Submit in Individual VC in Development/QC/Prod
VC Batch: View Listings in Development/QC/Prod
Table Instance: Create Manual Discrepancy
Table Instance: Close Discrepancy
Table Instance: Send Discrepancy to InForm
Table Instance: Send Discrepancy to Spreadsheet
Table Instance: Reopen Discrepancy
Table Instance: Answer Discrepancy
Table Instance: Cancel Discrepancy
Table Instance: Add Comment to Discrepancy
Table Instance: Edit Discrepancy
Table Instance: Show Discrepancies
Table Instance: Create Data Amendment
Table Instance: Modify Data Amendment
Table Instance: Cancel Data Amendment

Flags

Users assigned the DMW_STUDY_ADMIN role have access to the following operations for flags.

Oracle DMW Object Operations

Table Instance: Assign Flags

Filters

Users assigned the DMW_STUDY_ADMIN role have access to the following operations for filters.

Oracle DMW Object Operations

Filter: Mark as Public in Dev
Filter: Mark as Public in QC
Filter: Mark as Public in Prod

Data Lineage Tracing

Users assigned the DMW_STUDY_ADMIN role have access to the following operations for data lineage tracing.

Oracle DMW Object Operations

Table Instance: View CTX Data Lineage

Validation Checks

Users assigned the DMW_STUDY_ADMIN role have access to the following operations for validation checks.

Oracle LSH Object Operations

Program: Manage Security
Program Instance: Manage Security
Program: Modify Val Status to PROD
Program: Modify Val Status to QC
Program: Modify
Program: Remove
Program Instance: Modify Val Status to PROD

Program Instance: Modify Val Status to QC
Program Instance: Create Execution Setup
Program Instance: Modify
Program Instance: Remove
Execution Setup: Manage Security
Table: Manage Security
Table: View
Table: Modify
Table: Modify Val Status to PROD
Table: Modify Val Status to QC
Table: Remove

Oracle DMW Object Operations

VC Batch: Manage Security
VC Batch: Modify
VC Batch: Delete
VC Batch: View
VC Batch: Modify Validation Status Production
VC Batch: Modify Supporting Info
VC Batch: Deploy (Install) in Development/QC/Prod
VC Batch: Submit in Developmene/QC/Prod
VC Batch: Submit in Individual VC in Development/QC/Prod
VC Batch: View Listings in Development/QC/Prod

InForm

Users assigned the DMW_STUDY_ADMIN role have access to the following operations for InForm.

Oracle LSH Object Operations

Adapter Area: Create Remote Location
Remote Location: Modify
Adapter Area: Create Web Service Location
Remote Location: View
Remote Location: Create Connection
Connection: Modify
Remote Location: Delete
Connection: Delete

Oracle DMW Object Operations

Web Service Location: Modify Web Service Location
Web Service Location: Create Connection
Web Service Location: Delete Web Service Location
Web Service Location: View
Data Model: Configure InForm
Data Model: Load InForm Metadata

File Watcher

Users assigned the DMW_STUDY_ADMIN role have access to the following operations for the File Watcher.

Oracle DMW Object Operations

Data Model: FWR Configuration

Transformations and Query Builder

Users assigned the DMW_STUDY_ADMIN role have access to the following operations for transformations and the query builder.

Oracle LSH Object Operations

Domain: Manage Security
DataModel: Manage Security
Table Instance: Manage Security
Table Instance: Modify
Table Instance: Remove
Table Instance: Modify Val Status to QC
Table Instance: Modify Val Status to PROD
Table Instance: ReadData
Table Instance: BlindBreak
Table Instance: Unblind
Table Instance: ReadUnblind
Application Area: Manage Security
Work Area: Manage Security
Program: Manage Security
Program: Manage Security
Program: Modify Val Status to PROD
Program: Modify Val Status to QC
Program: Modify
Program: Remove
Program Instance: Modify Val Status to PROD
Program Instance: Modify Val Status to QC
Program Instance: Create Execution Setup
Program Instance: Modify
Program Instance: Remove
Execution Setup: Manage Security
VC Batch: Manage Security
Table: Manage Security
Execution Setup: Modify Val Status to PROD
Execution Setup: Modify Val Status to QC
Execution Setup: Submit

Oracle DMW Object Operations Transformation Map: Modify Private Listing
Transformation Map: Delete Private Listing
Transformation Map: Modify Public Listing
Transformation Map: View Public Listing
Transformation Map: Delete Public Listing
Transformation Map: Manage Security for Public Listing
Transformation Map: Modify Val Stat QC for Public Listing
Transformation Map: Modify Val Stat PROD for Public Listing
Transformation Map: Modify Supporting Info
Transformation Map: Modify
Transformation Map: Delete
Transformation Map: View
Transformation Map: Modify Validation Status QC
Transformation Map: Modify Validation Status PROD
Transformation Map: Modify Supporting Info
Transformation Map: Deploy (Install) in Development/QC/PROD
Transformation Map: Modify Security
Transformation Map: Classify

DMW_STUDY_INST_ACCESS

The DMW_STUDY_INST_ACCESS role is intended for users who require only View access to data and discrepancy listings. Blinding-related privileges are also included in this role.

Users with this role should have the DMW_STUDY_CONSUMER application role.

The DMW_STUDY_INST_ACCESS role allows users to perform the following operations on the specified objects.

Clinical Data Models and Query Builder

Users assigned the DMW_STUDY_INST_ACCESS role have access to the following operations for clinical data models and the query builder.

Oracle LSH Object Operations

Domain: View
Data Model: View
Work Area: View
Table Instance: View

Blinding

Users assigned the DMW_STUDY_INST_ACCESS role have access to the following operations for blinded data.

Oracle LSH Object Operations

Table Instance: ReadData
Table Instance: BlindBreak
Table Instance: Unblind
Table Instance: ReadUnblind

Discrepancies

Users assigned the DMW_STUDY_INST_ACCESS role have access to the following operations for discrepancies.

Oracle DMW Object Operations

Table Instance: Show Discrepancies

Data Lineage Tracing

Users assigned the DMW_STUDY_INST_ACCESS role have access to the following operations for data lineage tracing.

Oracle DMW Object Operations

Table Instance: View Data Lineage Tracing

Validation Checks

Users assigned the DMW_STUDY_INST_ACCESS role have access to the following operations for validation checks.

Oracle DMW Object Operations

VC Batch: View

VC Batch: View Listings in Development

Transformations

Users assigned the DMW_STUDY_INST_ACCESS role have access to the following operations for transformations.

Oracle LSH Object Operations

Table Instance: BlindBreak
Table Instance: ReadData
Table Instance: ReadUnblind

DMW_STUDY_INFORM_CONFIG

This role has the privileges required to configure the link to InForm for an input clinical data model.

InForm

Users assigned the DMW_STUDY_INFORM_CONFIG role have access to the following operations for InForm clinical data models.

Oracle LSH Object Operations

Adapter Area: Create Remote Location
Remote Location: Modify
Adapter Area: Create Web Service Location
Remote Location: Create Connection
Connection: Modify

Oracle DMW Object Operations

Domain: Create Data Model
Data Model: InForm Config
Data Model: Modify Data Model
Data Model: Delete Data Model

Glossary

action

A mechanism for moving a discrepancy from one state to another; may also apply a **tag**.

audit trail

A complete record of changes made to study data or study objects such as validation checks and tables.

blinding

Hiding data that might divulge which patients were receiving the study drug and which were not. An entire table may be blinded, in which case no data in the table is visible to users without special blinding-related privileges, or certain columns, rows, or cells may have mask values visible to users without special blinding-related privileges.

category

1. A metadata attribute equivalent to a label that can be applied to discrepancies—either by a validation check or by a user directly in the Discrepancies page—or to validation checks or flags. If a category is applied to a validation check, the validation check automatically applies the category to all discrepancies it creates. Users can filter discrepancies by category.

2. Companies must categorize, or group, studies in Oracle DMW to use a common library for the study group. The default categorization is by therapeutic area.

clinical data model

A reusable set of database tables and related objects including data transformation programs, validation checks, and security that are assembled to address a specific need during a trial. For example, an input model contains tables with exactly the same structure as the files loaded from external systems such as InForm or labs; a review model can contain tables with a structure suitable for review and analysis, plus the transformation programs required to put input source data into the review structure; and a reporting data model can contain tables in SDTM or ODM CDISC structure.

conformance check

Oracle SQL Loader checks incoming data against formatting requirements such as data type and code list values.

Data Management Workbench

See **Oracle DMW**.

discrepancy

Potential error detected either manually or programmatically in clinical data. Tied to a data point. Called a *query* in InForm.

edit check

See **validation check**.

flag

A review metadata attribute associated with a data record. Flags are applied to a data record, can have any number of states, and are local to **Oracle DMW**. They apply to the record only in the clinical data model in which they are applied.

File Watcher

A feature that checks for new data files in a location defined by an administrator at specified intervals and automatically loads the data into the system.

InForm

A clinical trial data collection and management application; full name Oracle Health Sciences InForm.

Listings

A page in the Oracle DMW user interfaces that displays data records in the current clinical data model. There are three Listings pages: Default Listings, which displays all records in the current model (dependent on the user's security privileges), Validation Check Listings, which displays records specified by a single validation check at a time; and Custom Listings, which displays records specified by a user's query.

manual discrepancy

A discrepancy that a user, usually a data manager, creates in Oracle DMW during the review process.

Oracle DMW

Oracle Health Sciences Data Management Workbench.

patient

See **subject**.

preferred path

If a data point has multiple source data points, one must be marked by a user or the transformation program that created the data point as the preferred source data point. If a target data point has only one source data point, the system automatically marks it as the preferred source. The upstream or downstream path from one preferred source data point to the next is called the preferred path.

The system associates any discrepancies with the preferred path when routing them to the source system.

For example, for a Body Mass Index (BMI) value one source value—either Weight or Height—must be designated the preferred source data point for the purpose of routing a discrepancy back to the source system. In InForm the system creates a query on the preferred source data point.

Query Builder

User interface feature to facilitate building valid PL/SQL queries.

staging table

A table created for use with a transformation program to hold data at an intermediate stage during transformation execution. Source and target data lineage displays may include staging tables, but they are not otherwise visible in the user interface.

subject

A candidate accepted into a study; sometimes called a *patient*.

surrogate key (SK)

System-generated record ID based on the primary key identifiers for the table. The first token in the Surrogate Key is the table instance ID (tableRef obj_id). This is followed by each value from the primary key columns, in column order. The tokens are separated by a tilde (~).

tag

A label for discrepancies that can be applied with an **action** and used for filtering discrepancy display, creating substates, or directing communication among teams while a discrepancy remains in a single state.

validation check

Program that can be executed to check data and create discrepancies; also called an *edit check*.

Index

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W