Oracle® Health Sciences Data Management Workbench

User's Guide

Release 2.4

E52292-02

June 2015

ORACLE®

Oracle Health Sciences Data Management Workbench User's Guide, Release 2.4

E52292-02

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 Reviewing and Cleaning Data

10 Using the Home Page

11 Reviewing Data

B Predefined Roles

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 that have purchased support 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

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 — 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 Health Sciences Life Sciences Warehouse 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

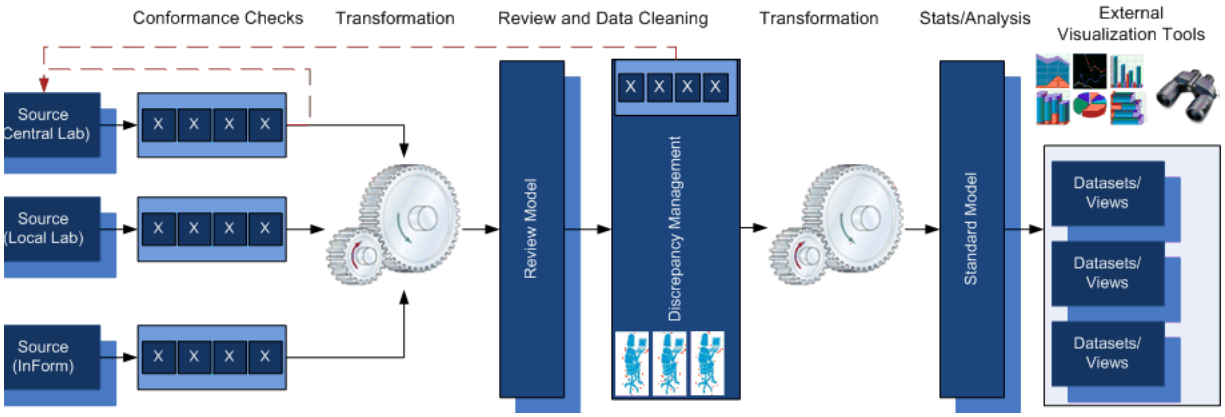
Oracle Health Sciences Data Management Workbench (Oracle DMW) is designed for two basic functions:

It is designed for integration with other systems:

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 Oracle Health Sciences 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

Oracle Health Sciences Data Management Workbench (Oracle DMW) provides a full set of features that allow a data manager to easily review clinical data from various sources in one place.

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 Oracle Health Sciences InForm—is associated with a single data point that has been identified as faulty or possibly faulty. Identifying 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 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**.

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 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.

Your company can set up security so that, for example, data managers have access to the data in the input and Review clinical data 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 pages in the user interface.

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 in the Listings pages.
- 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, and other categories.
- 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 Oracle Health Sciences 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 accord.ing 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.

You must purchase InForm separately.

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 Oracle Thesaurus Management System

You can send data items collected in InForm and labs to Oracle Thesaurus Management System (TMS) for coding to standard terminologies, and derive related information back into Oracle DMW for the coded items (terms).

To use TMS as a coding system, you must purchase it separately.

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 DMW can generate a generic Business Area object for a clinical data model to support third-party visualization tools.

Oracle LSH includes an adapter for Oracle Business Intelligence Enterprise Edition (purchased separately) that works with Oracle DMW as well.

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.
- See "Applying Snapshot Labels" on page 8-23.
- See "Loading Reference Tables" on page 8-24

See the first chapter of the *Oracle Life Sciences Data Hub Implementation Guide* for an overview. See "Object Ownership" on page A-5 for more information on the relationship between Oracle LSH and Oracle DMW objects.

2

Creating Studies and Study Templates

You build up a collection of studies and study components that you can reuse. As you develop study components you validate them and promote them to a higher lifecycle status, optionally with supporting documents.

Creating the First Study

For the first study, you must explicitly create everything you need. Oracle DMW automates some of the tasks.

Table 2–1 Basic Steps Required to Create the First Study

What You Do	Where to Find Information	What Oracle DMW Does For You
Name the study and set attribute values	"Creating a Study" on page 2-1	
Create input data models to hold data in the same format as in the source	"Creating a Study Clinical Data Model" on page 3-2	Imports metadata from InForm, creates tables from files for lab data sources.
Create target data models for reviewing and analyzing data. You may prefer to do this in a library.	"Creating a Study Clinical Data Model" on page 3-2 or "Creating a Library Clinical Data Model" on page 4-1	Creates tables from files; provides SDTM subject and subject visit tables.
Create data transformations from input models to target models	Chapter 5, "Transforming Data to Standard Structures"	Does automapping, provides a query builder, generates joins, unions, pivots, and unpivots, validates.
Create validation (edit) checks	Chapter 6, "Creating Validation Checks"	Provides a query builder.

Creating 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. **Template**: If selected, this study is available as a template for other studies. You can select this setting later, after testing the study.
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; see "Setting Up Library and Study Categories" on page 8-20.
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 Database Partitioning" on page 8-21 for more information.

You cannot change this value after saving.

6. Save.

Associating a Study with a TMS Domain Element

To use Oracle Thesaurus Management System (TMS) to code terms in a study, specify the dictionaries and TMS domains to use.

- **TMS Base Terminology**: The system lists the terminologies (dictionaries) defined in the local TMS instance. Using a terminology is enabled only by selecting a TMS domain.
- **TMS Domain**: Select the base terminology you need, then select the TMS domain through which to use it.

  Domain Terminology: When you select a TMS domain, the system displays the name defined for the rendition of the dictionary used in the domain. The domain terminology may be a virtual dictionary, meaning a version of the base dictionary that is frozen at a particular point in time.

See "Setting Up Coding in TMS" on page 8-14 for more information.

Force Rederivation

Click **Force Rederivation** to run the Force Rederivation job once, immediately. A confirmation message appears because running the job may take a long time.

Normally, transformations send only new or changed data to TMS for processing. Use the Force Rederivation job to process *all* data when you have made structural changes related to TMS in an ongoing study such as:

- Adding columns to target tables to hold derived data.
- Updating a dictionary to a new version with a different structure from the old one.
- Changing domain-related settings in the TMS reference codelist TMS_CONFIGURATION.

Note: The Force Rederivation job has no relation to defining TMS for a study.

Making a Study Available as a Template

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**.

Creating the Next Studies—Reusing Studies and Study Components

As you build up studies and components, you can reuse them in several ways.

Study Templates

Mark a study as a template to reuse it in its entirety. You can delete or mark as Not Used any components you do not need after you apply the template to a new study. The template includes clinical data models, transformations, validation checks, and custom listings. You can build up a set of study templates—for example, for different therapeutic areas or study types.

Study templates include all target clinical data models, all transformations, validation checks, custom listings, and business areas for data visualizations.

Note: You can apply only one study 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 the **Apply Template to Study** icon.

Library Clinical Data Models

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 allowed column values. Code lists remain in the library for reference; they are not copied into studies.

See Chapter 4, "Using Libraries" for more information.

Study Component Copies

You can copy components from other studies or study templates:

- Clinical data models
- Tables
- Transformations
- Validation checks
- Custom Listings

For more information see the chapter on each component.

Cross-Study Objects

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."

Understanding the Validation Lifecycle and Installation

You select a lifecycle area context —Development, Quality Control, or Production—on the Home page in the bottom left corner. You may have access to only one or two lifecycles.

Lifecycle Areas For each study, the system creates three lifecycle areas. Each lifecycle area has a different purpose:

- Use **Development** to create clinical data models, transformation programs, and validation checks either manually or from a library, study, or study template. Load data into the development lifecycle schema and do initial testing there.

  To modify a model, transformation or validation check in Quality Control or Production, you must check it out, creating a new object version, which is always in Development. The existing installed version of the object in the QC and Production areas continues to function as before until you promote the new version to QC or Production and install it there.
- Use **Quality Control** for formal testing of all study components (optional). This is equivalent to the UAT InForm environment.
- Use **Production** to load, review, and clean production study data. The system prevents destructive changes to tables and models in a production environment.

Supporting Documents When you promote an object to the next lifecycle stage, you can associate documents supporting the change in validation status, such as log files or signoff documents, with the object version.

InForm Metadata Comparison When you promote an InForm clinical data model, the system calls the Compare Metadata job to compare the Oracle DMW metadata to the InForm metadata in the lifecycle stage to which the model is being promoted. If there are differences, promotion does not take place. Instead the metadata comparsion report appears.

Installation When you promote an object, it is immediately visible in the new lifecycle area. However, you must **install** it for the changes to take effect.

Within each lifecycle area, each clinical data model has its own database schema. When you upgrade an object to a new lifecycle area, the system clones it from its current schema to the higher lifecycle area schema. Installation creates or updates the actual database tables and the packages for all the model's programs: 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.

The system prevents destructive changes such as dropping or shortening columns in Production. However, if there are destructive changes in Development or Quality Control, the system drops and replaces the table and all its data is deleted. In an InForm model, the system compares the model metadata with the InForm metadata in the higher lifecycle area. If there are differences, the installation does not take place and the comparison report appears.

See "Object Ownership" on page A-5 and the *Oracle Life Sciences Data Hub Application Developer's Guide* for more information.

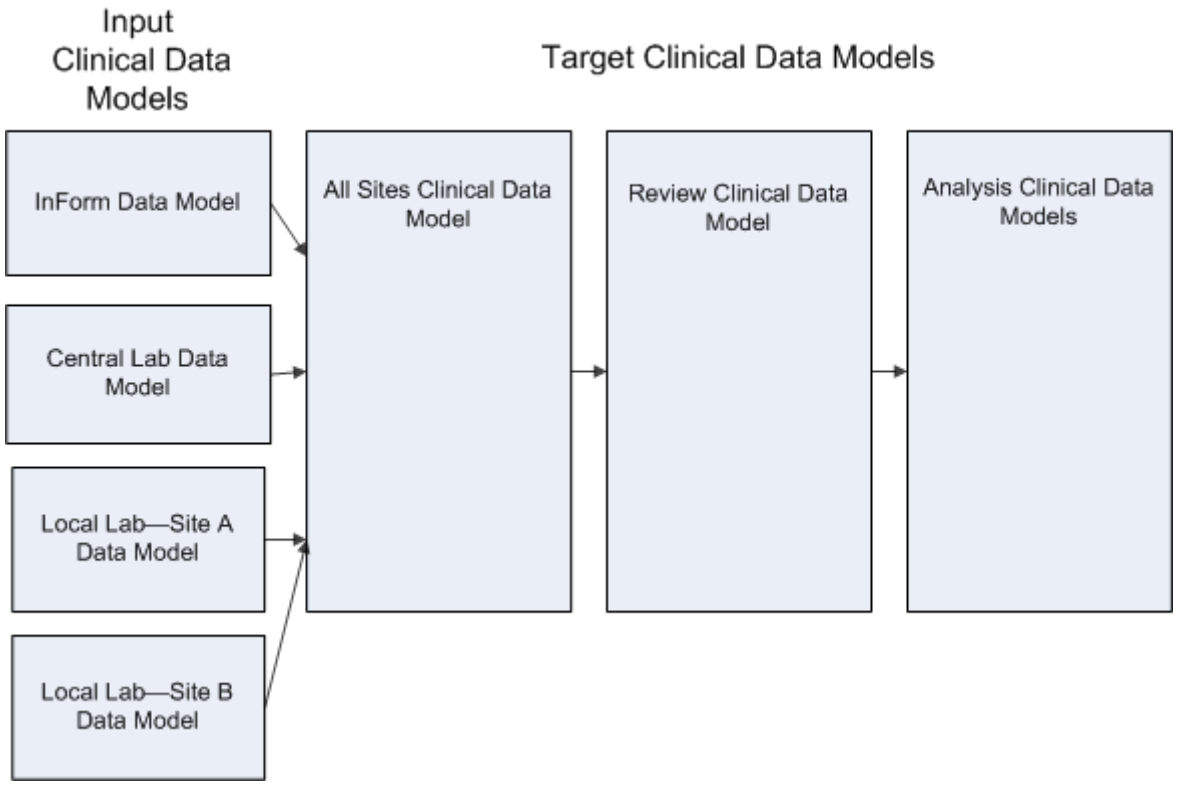
3

Creating 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—Oracle Health Sciences InForm or lab—in each study.

You copy or create additional models with data 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.



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 or study model into the study.
- Applying a study template that contains the models you want.
- Creating the model manually and creating tables by uploading SAS or text files from which the system can read the data structure and create tables to match; see "Required Syntax for Table Metadata Text Files" on page A-3 for more information. You can also add tables and columns manually.

Navigating to 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 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. Press **Enter**.

Creating a Study Clinical Data Model

To create a clinical data model:

1. On the Home page, select the study and the Development lifecycle 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 outside Oracle DMW, from InForm or a lab.
   - **Target** if its data will be derived from another model in the system. Skip to Step 6.
5. If you selected **Input**, select an **Input Type**:
   - **InForm** to load data from InForm. Skip the next steps and see "Configuring the Oracle Health Sciences InForm Connector" on page 3-16.
   - **File** to load data from files, as from a lab. Select a single **File Type** to be loaded into this model: **SAS** or **Text.**

     After creating the model, see "Configuring File Watcher" on page 3-21.
6. Under **Select from source** select:
   - **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 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 and select it.

- **Library Data Model** to copy a library model. Additional fields appear. Select the library model.

  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 is the therapeutic area or whatever other category your company uses. The system displays checked-in models to which you have View access.

  If a library model is updated in the future, you can automatically make the same updates in the study model. However, any local changes you have made are lost.

- **Study Data Model** to copy a model from a study—the current study or a different one. Additional fields appear.

  You can filter by typing all or part of the name of the study you want to copy from, its therapeutic area, or the model you want to copy. The system displays checked-in models to which you have View access.

Note: You cannot create a study's InForm input clinical data model from a copied model; there can be only one input InForm model in a study and its metadata must be imported from InForm. However, you can create a target model by copying an input InForm model—and modify it as necessary.

7. Select **Business Area** to allow visualization tools access to the data contained in the new clinical data model. See "Setting Up a Data Visualization Tool" on page 8-25 for more information.
8. Click **OK**.
9. If the model is an input model, configure its method of importing data:
   - For InForm models, go to InForm Configuration; see "Configuring the Oracle Health Sciences InForm Connector" on page 3-16.
   - For File models, go to Watcher Configuration; see "Configuring File Watcher" on page 3-21.
10. Click the **Install Model** icon. 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. These requirements do not prevent the installation of the model, but you should complete these specifications and then install or reinstall the model.

Adding Tables to a Clinical Data Model

There are several ways to add tables.

Copying Tables

When you copy tables, the system also copies:

- Each validation check that reads from the selected tables, if all its source tables are included in the Copy operation.
- Each public custom listing that reads from the selected tables, if all its source tables are included in the Copy operation.

If you do not select all the tables required by a validation check that is part of an ordered batch, you receive a warning. For other validation checks and custom listings, you do not receive a warning: if all source tables are not selected, the checks and listings are simply not copied. You can copy validation checks and public custom listings separately.

To copy a table:

1. Select the model into which you want to copy tables and click **Check Out**.
2. In the Tables tab, click the **Copy Tables** icon.
3. Select the source: either **Study Data Model** or **Library Data Model**. The system displays all such models to which you have access. You can filter by typing part or all of the name of the Therapeutic Area (or other category), Study (if you selected Study Data Model) and Model.

   Click **Clear Filters** to remove all typed text and revert to the full list.
4. Select a model. The system displays all its tables.
5. Select tables. Use Ctrl+click or Shift+click to select multiple tables.
6. If you want the selected tables to replace tables with the same name that already exist in the current model, select **Overwrite the same table names**. Otherwise the system leaves the existing tables as they are, copies the selected tables and adds `_1` (or an increment of 1) to the name of each copied table that has the same name as an existing table. Any validation checks and custom listings copied with the table are mapped to the copied table.
7. Click **OK**.

Adding Tables from a File

You can upload certain types of files (see below) into a clinical data model to automatically create a table in the same structure as in the file.

1. Select the model and click the **Check Out** icon. (If someone else has checked it out, you cannot work on it.)
2. Click the **Add** icon in the Tables pane. The Add Table window opens.
3. Under **Select from source**, select **Load from file**. A Browse field appears.
4. 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 SAS datasets or .mdd metadata files (one dataset or .mdd file per table).

  You can include data files in the zipped file; either SAS datasets or, with .mdd files, .txt or .csv files in the required format; see "Required Syntax for Table Metadata Text Files" on page A-3.

Notes: All tables that contain clinical data must have a **primary key**.

5. Click **OK**. See "Setting Blinding-Related Attributes" on page 3-9 and "Setting the Data Processing Attribute" on page 3-10.

Adding Subject Visit and Subject Tables

To support filtering data and discrepancies by subject and visit and tracking subject visit completeness using flags and custom programs, you must include one Subject Visit table in at least one clinical data model in each study and associate it with the Subject Visit table SDTM identifier.

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

The system uses only the Subject Visit table to support filters.

Note: To support full filter functionality, add the following columns to the default Subject Visit table and associate each with an SDTM ID: SUBJID, COUNTRY, SITENAME, SITEID, INVNAM, INVID, EPOCH, VISIT, VISITNUM, VISITDY, and SVSTDTC. See "How Filters Work" on page 3-7.

Copying Default Subject Visit or Subject Table

To copy the default Subject Visit or Subject table into a clinical data model:

1. In the Study Configuration page, select the clinical data model, check it out, and click the **Copy Subject/Visit Table** icon in the top row of icons. The Copy Subject/Visit from Library window opens.
2. Select one:
   - **Copy from Library**: Allows you to copy a Subject, Subject Visit, or both tables created in a library clinical data model by your company; see "Subject Visit Table Requirements" on page 3-6.

     Then select the study type—therapeutic area or other category—that contains the library model whose tables you want to copy, and then select the library model.
   - **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 "Default Subject Visit Table Metadata" on page 3-6 and "Default Subject Table Metadata" on page 3-7.
3. Select which tables to copy: **Subject**, **Subject Visit**, or **Both**.
4. Click **OK**.

Subject Visit Table

If you designate an existing table as the Subject Visit table, it must satisfy these Subject Visit Table Requirements. Alternatively, you can copy the shipped Subject table; see Default 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.
- The table must be linked to the SDTM SUBJECTVISIT identifier.
- See "Defining Tables to Support Filtering in the Listings Pages" on page 3-7 for additional requirements.

Default Subject Visit Table Metadata The shipped, or *default*, Subject Visit table follows SDTM specifications.

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

Columns The default Subject Visit table contains these columns.

- 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.

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.

Subject Visit Table Requirements If you designate another table as the SDTM Subject table:

- Its primary key must include the Subject ID and no other columns.
- The Subject ID column must be linked to the SUBJID SDTM identifiers.

Default Subject Table Metadata

Primary Key The shipped Subject table's primary key is the unique subject ID across studies, 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 11-7. To support this functionality, you must define tables as follows:

- "Subject Visit Table" on page 3-8
- "Subject and Visit SDTM Variable Columns in All Tables" on page 3-8

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 3-28.

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 3-15.

Subject and Visit SDTM Variable Columns in *All* Tables

To support full filter functionality, **all tables** users may need to view in the Listings or Discrepancies pages **must contain columns linked to the following SDTM identifiers: SUBJID, USUBJID, VISIT, VISITNUM**.

Filter logic creates a join between the Subject Visit table and the table being viewed on the column shown in the following table.

Table 3–1 Columns Required in All Viewed Tables to Support Filtering

Page	Join Column for Subject Filters	Join Column for Visit Filters
Listings	USUBJID	VISITNUM
Discrepancies	SUBJID	VISIT

Filter logic on the Listings page requires a join of the Subject Visit table with the table being viewed on the USUBJID and VISITNUM columns.

Filter logic on the Discrepancies page requires a join of the Subject Visit table with the table being viewed on the SUBJID and VISIT 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.

Adding and Modifying Tables Manually

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

1. Select the model and click **Check Out**. (If someone else has checked it out, you cannot work on it.)

   Note: Do not make structural changes to tables in InForm models; see "Modifying an InForm Input Model" on page 3-31 for information on the changes that are and are not allowed.

2. In the Tables tab, click the **Add Table** or **Modify Table** icon. The Create (or Modify) Clinical Data Model Table window opens.

3. Enter values in the following fields:
   - Enter a **name** and **description** for the column; see "Naming Objects" on page A-1 for restrictions.
   - **Oracle Name**: By default, the system populates this with the value you entered for the name, truncated at 30 characters; see "Automatic Name Truncation" on page A-2.

     Note: If the Oracle name includes a space, the system replaces the space with an underscore (_) when you save. The system does not replace the spaces in the default SAS name.

   - **SAS Name**: By default, the system populates this with the value you entered for the name, truncated at 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.
   - **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.
   - **UOW (Unit of Work) Processing Type**: See "Setting the Data Processing Attribute" on page 3-10.
   - **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 3-5 for information about requirements. Selecting an SDTM identifier supports filters on the Listings and Discrepancies pages.
4. Set blinding attributes; see "Setting Blinding-Related Attributes" on page 3-9.
5. Click **OK**.

Setting Blinding-Related Attributes

Blinded data is not displayed in Listings pages unless a user with special privileges requests to view it. Such viewing is audited. See "Data Blinding" on page 3-26 for conceptual information.

Set a table's blinding attributes when you add or edit the table:

- **Blinded**: Select if the table may ever contain any sensitive data that should be hidden.

  Note: To unblind data in the table at the end of a study, deselect this attribute. Special privileges are required.

- **Blinding Type** (available only if Blinded is selected): Select one:
  - **Table**: Select to hide all data in the table, then click **OK**.
  - **Column**: Select to mask all values in one or more columns, or in cells where data in the row meets conditions you specify, then click **OK**.

Then select one column, click the **Modify** icon and specify the masking value. If you want to blind data only in rows that meet certain conditions, specify the conditions; see "Specifying Masking Attributes for a Column" on page 3-15.

You can mask values in more than one column.

– **Row**: Select to hide certain rows in their entirety.

Blinding Criteria (available only for row-level blinding): Click the **Modify Blinding Criteria** icon 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: `<, <=, <>, =, >, >=, 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 for Transformations and Blinding Criteria" on page 5-16 for more information.

Setting the Data Processing Attribute

Using Unit of Work (UOW) processing can speed up data loading and transformation execution; see Chapter 9, "Data Processing."

The attribute value of the target table determines the preferred processing type for loads and transformations writing to the table. For Unit of Work processing to actually occur in transformations, source tables must also be defined with the same UOW processing type.

You must define a primary key before you can set the UOW Process Type attribute. When you define a primary key, the UOW Process Type attribute value changes to **Reload**. Modify the table to specify the UOW Process Type value.

Select one of the following:

- **Non UOW**: Jobs writing to the table will use Reload processing.
- **Subject**: Jobs writing to the table will use try to 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 try to 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.
- If a table is defined as UOW the system will use UOW processing if possible; if not it will use Reload processing.

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 5-29. 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 values 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 5-29.

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

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

       Select **Supports Duplicate** 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 values 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.

4. Click **OK**.

Linking a TMS Set to a Table in a Target Clinical Data Model

To code source data to terminology terms using Oracle Thesaurus Management System (TMS), you must link the column (CRF field) to the TMS Set that specifies the information to derive from TMS to Oracle DMW, and add columns to the same table to hold the derived values.

Note: In all three lifecycle areas, you can send data to TMS, derive data for terms that can be automatically coded and send derived data to the source system, and create discrepancies in TMS for terms that cannot be automatically coded. However, only in the Production lifecycle does TMS create omissions that TMS users can code manually. See "Setting Up Coding in TMS" on page 8-14 for more information.

Prerequisites

- The study must be set up for TMS processing, with a TMS domain specified for each terminology, or dictionary to be used for the study; see "Associating a Study with a TMS Domain Element" on page 2-2.
- An appropriate TMS Set must exist; see "Defining TMS Sets" on page 8-14.

To link a TMS Set to a table:

1. Navigate to the data model and select the target table.
2. Click the **TMS** tab, then click the **Add** icon. The **Add TMS Column Association** window appears.

3. Select the **TMS Set** to use. The system displays the TMS Set description, base dictionary, and primary column name, which is the name of the dictionary level defined as the coding level in your TMS installation.
4. Select the **primary column**. The system displays all VARCHAR2 data type columns in the clinical data model table. Select the one whose value you want to have coded in TMS.

   The system displays the derived columns defined for the selected TMS Set.
5. In the **Column Name** field for each one, select the table column to map to each TMS Set derived column.

   If you have not yet added columns to the table to receive the derived data from TMS, click **OK**, then add the columns in the Columns tab, then click the **Edit** icon in the TMS tab and select them for the TMS Set derived columns.
6. Click **OK**.

Adding Columns to a Table

You can add columns to a table as part of creating the table itself when you upload a file; see "Adding Tables from a File" on page 3-4

Adding Columns Manually

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 checks out the table if it is not already checked out.
2. Enter values in the following fields:
   - Enter a **Name** and **Description** for the column; see "Naming Objects" on page A-1 for restrictions.
   - **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 3-15.
- **Oracle Name**: By default, the system populates this with the value you entered for the name, truncated at 30 characters; see "Automatic Name Truncation" on page A-2. See "Naming Objects" on page A-1 for restrictions.

Note: If the Oracle name includes a space, the system replaces the space with an underscore (_) when you save. The system does not replace the spaces in the default SAS name.

- **SAS Name**: By default, the system populates this with the value you entered for the name, truncated at 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). 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—or the reverse—during a transformation, the pivot column must be associated with a code list; see "Pivot" on page 5-14.

---

3. Enter masking attribute values; available only if the table has a blinding type of Column; see "Specifying Masking Attributes for a Column" on page 3-15.
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**.
- SUBJID 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.

1. Select the column and click the **Modify** icon. In the Modify Clinical Data Model Column window, select the **Binding Attributes** tab.
2. Select a **Masking Level**:
   - **Cell**: Masks the real data only in certain rows in this column.
   - **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 values to display instead of the real values:
   - Enter a constant value to be displayed in every row.
   - Click the **Modify Masking Value** icon to create an expression to generate multiple values for the system to display; see "Using the Expression Builder

for Transformations and Blinding Criteria" on page 5-16.

4. **Masking Criteria** (for cell-level masking): Click the **Modify Masking Criteria** icon to specify the criteria for blinding cells in the column; see "Using the Expression Builder for Transformations and Blinding Criteria" on page 5-16.

> **Note:** Define row-level blinding at the table level. See "Setting Blinding-Related Attributes" on page 3-9.

Configuring the Oracle Health Sciences InForm Connector

The system uses a database connection to import data and metadata from the study's InForm reporting database, and a web service to send discrepancies back to InForm as queries. You must specify both for each Oracle DMW lifecycle area.

You can load data from any InForm lifecycle into any Oracle DMW lifecycle except production, which requires that you assert that you are loading data from an InForm lifecycle database. Oracle DMW will send discrepancies back to InForm only if the lifecycle stages match. InForm's UAT lifecycle matches the Quality Control lifecycle in Oracle DMW.

Prerequisite Create the InForm input clinical data model for the study; see "Creating a Study Clinical Data Model" on page 3-2.

> **Note:** You must be working in the Development lifecycle context, selected on the Home page, in order to configure the InForm Connector for any lifecycle area. The current lifecycle context is displayed at the top of the page.

In the InForm Configuration tab of the InForm clinical data model, specify a remote location and web service location to use for each Oracle DMW lifecycle:

1. **Remote Location**: Select the InForm reporting database from which to load data. See "Adding or Modifying a Remote Location" on page 8-6.
2. **Remote Study Account Name**: Enter the name of the database account that owns the study's InForm reporting database and RDE views.

> **Note:** You can change the InForm configuration for any lifecycle to use a different remote location and/or remote study account name. If you do, the system runs a metadata comparison between the InForm model in Oracle DMW and the InForm metadata at the new location. If it finds differences, the metadata comparison report is displayed and you may either cancel the changes or accept them, in which case data loading is suspended. You must reload metadata before resuming data loading.

3. **InForm LifeCycle**: Select the InForm lifecycle stage—Development, UAT, or Production—of the study account name.
4. **Webservice Location**: Select the appropriate web service location. You can add a new web service location if you are working in the Development lifecycle; see "Adding or Modifying a Web Service Location" on page 8-7.
5. **InForm URL**: Enter the URL for the study's InForm website—for example:

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

This is required to support "Link to InForm" actions in the Discrepancies and Listings pages.

After you enter a value, the **Test URL** icon appears. Click it. If the InForm login screen opens, the URL is correct. (The new window may open behind the current one.)

6. **Schedule Production Data Load**: In production only, select the check box to schedule data loads at regular intervals in a production environment. This option is available only if you have data load privileges and the current lifecycle context is Production; see "Scheduling Data Loads in the Production Lifecycle Area" on page 3-18.

   In all lifecycle areas you can load data manually as needed by clicking the **Load Data** icon if you have the required privileges; see "Loading Data" on page 3-18.

7. Save.

Note: To complete the InForm configuration; see:

- "Selecting Internal InForm Tables and Views" on page 3-17
- "Loading InForm Metadata" on page 3-17

Selecting Internal InForm Tables and Views

You can select internal InForm tables and views to be available in this study. You cannot change these selections after you load metadata. Your administrator makes the default selections. You can override the default selections except for tables and views that are required by the system. Their check mark is displayed as gray. See "Available InForm Tables and Views" on page 8-8 for the full list.

You can use selected tables and views as sources in data transformations from one model to another. To view their data, go to the Listings page.

To select tables and views:

1. In the InForm Configuration tab, click the **Select InForm Operational Data and Metadata Tables** icon.

   The system displays all available InForm tables and views alphabetically. To sort by type, click the heading of the **Internal Data Model** column.

2. Select and deselect until you have selected exactly the tables and views you want for this study.

Note: This selection cannot be changed after you save.

---

3. Save.

Loading InForm Metadata

To load metadata, click the **Load InForm Metadata** icon. To check the progress of the job, click the **Refresh** icon and check the Status column.

Note: Select the internal InForm tables you want to use in the study before loading metadata. You cannot change the selection afterward. See "Selecting Internal InForm Tables and Views" on page 3-17

The **Load InForm Metadata** icon is available only in the context of the Development lifecycle (selected on the Home page) and only to users with special privileges for metadata loading; see "InForm Configuration Privileges" on page 3-20.

See "Blinding Data in Input Models" on page 3-26 to set up importing InForm hidden items as blinded.

In its initial run, the Load Metadata job creates one table in the InForm input clinical data model for each view in the reporting database that is registered in the RD_DATADICTIONARY table. The tables have the same structure as the InForm views plus additional columns and column attributes required in Oracle DMW. See "How the System Tracks Data Lineage" on page 5-29 for more information.

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 QC or Production, 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

Click the **Suspend** icon to stop any scheduled data loads and prevent manually loading data.

The system replaces the **Suspend** icon with the **Resume** icon. Click the **Resume** icon to allow manual and scheduled data loading.

You must suspend data loading to load metadata, install the model, or change the remote location, account, or web service for any lifecycle area.

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. Check in and install the model if it has not already been checked in and installed in Production.
2. If the **Status** column says *Data loading may proceed*, click the **Suspend** icon.
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 the **Resume** icon. The first scheduled data load starts immediately.

Comparing Oracle DMW and InForm Metadata

You can compare the metadata—table and column structure—in any Oracle DMW InForm model lifecycle area to any InForm lifecycle database for the same study.

For example, before you promote an InForm clinical data model from Development to Quality Control, compare the Development model's metadata to the InForm UAT lifecycle metadata. When you promote the InForm model to a higher lifecycle, the system runs the same comparison automatically and displays a report.

1. In the InForm Configuration tab, click the **Compare Metadata** icon.
2. In the **Metadata Comparison** window, select the metadata to compare:
   - For the Oracle DMW InForm data model: Development, Quality Control, or Production.
   - For InForm: Development, UAT, or Production.
3. Click **Compare**. The report appears on screen.

   To save the report, click the **Export All to Excel** icon.

   If the system finds no differences, it displays a message to that effect.

Automatic Metadata Change Detection and Synchronization

The system helps keep metadata consistent with InForm metadata in the same lifecycle, in case InForm data structures are changed during the course of the study—for example, CRF fields are added or increased in length to support a protocol amendment. The system detects these changes by querying the IRV_ STUDYVERSION_REVISIONS table for either new versions or revisions without versioning (RWV).

The InForm reporting database extract (RDE) views are the source of truth for Oracle DMW. For example, if a protocol change occurs that results in forms and items being removed from a trial, the RDE views may still retain the views and columns that represent the removed items. If that is the case, they also remain in DMW even after reloading metadata.

Metadata Change Detection During Data Loading

If changes are detected during data loading, the system:

- Prevents promoting the InForm model to a higher validation status.

- Suspends data loading in this lifecycle area where it was detected. If this was Production, it displays a message on the Home page.

You should load metadata and then resume data loading.

Metadata Change Detection During Metadata Loading

During metadata loading, the system:

- Reloads the required static metadata tables: IRV tables and RD_DATADICTIONARY.
- Compares the static metadata tables with the data stored in DME_IA_SRC_TABLES and DME_IA_SRC_COLUMNS to detect significant changes including:
  - New, missing, or changed reference path for an item
  - Changed data type
  - Increased data length for VARCHAR2 columns
  - Changes to column (item) blinding status
- If significant changes are found, updates DME_IA_SRC_TABLES and DME_IA_SRC_COLUMNS, other extended metadata mapping tables, and the InForm input clinical data model tables. Note:
  - Tables are never dropped. If an RDE view is missing, the system marks the corresponding Oracle DMW table as Not Used.
  - Columns are never dropped. If an existing item is not included, it is marked Not Used. If a column data type is changed, a new column is created with the new data type.
  - Updating a column's blinding status may require updating its table's blinding status as well.
- Displays the message "Metadata needs to be reloaded" and suspends data loading until it is done. A user with the required privileges must click the **Load Metadata** button to reload and synchronize metadata.

Metadata Change Detection After Changes to the Remote Location Definition

When you save changes to a remote location or remote study account, the system compares the metadata in the model with the new remote location or schema and displays the results.

If there are differences and you accept them, reload metadata to be sure Oracle DMW is synchronized with InForm.

InForm Configuration Privileges

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

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

Configuring File Watcher

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.

Each study normally has six File Watchers, one for each combination of the two file types—SAS and text— and the three lifecycle stages—Development, Quality Control, and Production. Each of the six study File Watchers watches a single folder to detect files to load. It is also possible to use a three-Watcher setup, with both text and SAS files in the same folder. This is the setup used in the environment hosted by Oracle. See "Registering Folder Locations" on page 8-2 for more information.

If two labs are sending text data to the same study, you have a different clinical data model for each lab, but files from both labs must be placed in the same folder for each lifecycle. To enable File Watcher to match each file with the correct data model, you must specify a unique file name pattern for each lab within the study.

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

Using Data Loads to Trigger Transformations Each data load can 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.

Prerequisite See "Setting Up and Monitoring File Watchers" on page 8-1.

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 for the selected file type:
   - "Setting SAS Data Load Parameters" on page 3-21
   - "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 8-5.
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:

- **Upload 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**: 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 to 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 file type/lifecycle area combination 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 File Specifications, 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, use a naming convention includes the model name or corresponding lab name, but this is not required.

   For example, Study A with file input models for three labs, where labs are expected to include a date just before the extension:

   - CentralLab_.*.zip
   - SpecialLab_.*.cport
   - LocalLab_.*.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 matches zero or more occurrences of the preceeding character. The dot (.) before the asterisk (*) in the examples above means "any character or no characters."

   Be careful about case sensitivity.

   Note: In hosted environments, you must use a file naming convention of starting with the study folder name followed by an underscore. See "Creating and Modifying Study File Watchers" on page 8-3 for more information.

   Note: If you have multiple data files in the same .zip file, the individual files within it must have names that match their target table name.

   ---

3. Lifecycle
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. **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.
6. **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.
7. **Frequency**: If you selected Scheduled, select a frequency for the File Specification to look for a new data file in the specified location in days, hours, or minutes.
8. **Start Date**: If you selected Scheduled, enter the date and time when you want scheduled file watching to begin.
9. **End Date**: 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.
10. 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 currently running, select it and click the **Suspend** icon.
- If the file specification is currently suspended, select it and click the **Resume** icon.

Migrating a Release 2.3 File Watcher to 2.3.1 or Higher

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 "Creating and Modifying Study File Watchers" on page 8-3. 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 the **Force Data Load** or **Delete File** icon. 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 10-1. 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.
  - **ARCHIVED**: The file was archived by File Watcher as scheduled.
- **File Modified**: The modification date of the file on the file system.
- **Data Load Date** The scheduled data load data before the data has been loaded, with an icon to indicate "scheduled," and the actual data load data afterward.
- **Detection Date** The date and time the file was detected, using the date and time in the Oracle DMW database.
- **Archive Date** displays the scheduled archive date before the file is archived and the actual archive date afterward.
- **Error**: Information about the problem.
- **Deletion Date** displays the scheduled deletion date before the file is deleted and the actual deletion date afterward.
- **Date Missing**: If file is overwritten or removed from the file system before it is archived or deleted, then the Date Missing is stored here.

See "Viewing Data Load Information" on page 10-1 for job statuses. To see the data, go to the Listings page.

Installation and Checkout

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 creates or updates a query that selects all columns and rows from all tables in the data model, including any masked data. The system uses this view to display data on the Listings page. This view also serves as the starting point for custom listings.

Whenever you modify a clinical data model or any table contained in it, you must do so in the Development lifecycle area after checking out the model. When you have completed your changes, check in the model and install the new version.

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, and whole rows or cells meeting specified criteria. You can specify masking values for use with blinded columns and cells.

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

Special privileges are required to view blinded data, to unblind data and to view unblinded data. See the *Oracle Life Sciences Data Hub Implementation Guide* for information about blinding-related security user privileges.

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. 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 with blinding attributes 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. By default, all InForm tables are created in Oracle DMW as **not** blinded.

  Your administrator can change a profile value (see "Setting Blinding Behavior for InForm Hidden Items" on page 8-29) so that by default all InForm hidden items are imported to Oracle DMW as blinded, with default masking values:

  – The character data masking value is `xxxxx`.

  – The numeric data masking value is `99999`.

  – The date data masking value is `15-AUG-3501`.

  You can change the blinding and masking values in the individual Oracle DMW tables.

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.

Updating Validation Status

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

1. Select the object and click the **Modify Validation Status** icon. The object must be checked in and installed to be promoted to a higher status. 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. This status corresponds to UAT in InForm.
   - **Production**: This is intended as the status for objects that have passed formal testing and are suitable for use in an active study.

Note: For InForm clinical data models you can click the **Compare Metadata** icon to compare the model's metadata to the metadata in the InForm lifecycle to which you are promoting it. The system also calls the Compare Metadata job during the validation status upgrade process, which fails if there are significant differences.

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 **Add** 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.

After you promote an object, it is visible in the new lifecycle context (change lifecycle context on the Home page to see it). However, you must install the object for the new version to take effect. See "Understanding the Validation Lifecycle and Installation" on page 2-4 for more information.

For InForm clinical data models, the system runs the Compare Metadata Report before promoting the model. If there are differences, the report is displayed and the promotion cannot proceed.

Applying Security

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's parent, or owning, object. The user must also have a role in the user group with the privileges required to create or modify the object type. See "Object Ownership" on page A-5 and Appendix B, "Predefined Roles" for more information.

To view data, a user must:

- Belong to a user group with access to the table containing the data.
- Have a role in the user group with view privileges on tables.
- Have a blinding-related application role that allows viewing nonblinded, unblinded, and/or currently blinded data.

When you assign a user group to an object, the user group assignment is 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. See "Setting Up Security" on page 8-18 for more information.

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 the **Apply Security** icon. 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.

   User groups that never had access to the object are not displayed.
2. In the **Assign To** field, select one:
   - **Metadata**: 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—in its container. See "Predefined Object Security Roles" on page B-3 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

See also "Adding and Modifying Tables Manually" on page 3-8

Manually Modifying a Non-InForm Model

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 3-31.

However, nonstructural changes are allowed; see "Manual Changes Allowed in InForm Models" on page 3-31.

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 and the model has not been checked out by someone else.
3. Make changes. If you have the required privileges, you can:
   - **Add, modify, or delete tables** by clicking the appropriate icon and making the changes; see "Adding Tables to a Clinical Data Model" on page 3-3.
   - **Add, modify, or delete columns** in a table by selecting the table in the upper pane, clicking the appropriate icon in the Columns tab, and making the changes; see "Adding Columns to a Table" on page 3-13.
   - **Add, modify, or delete constraints** in a table by selecting the table in the upper pane, clicking the appropriate icon in the Constraints tab, and making the changes; see "Adding Constraints to a Table" on page 3-11.
   - **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 icon to change the validation status or to upload a supporting document for the validation status change; see "Updating Validation Status" on page 3-27. 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. You can change the attributes that apply only in Oracle DMW, but if metadata is reloaded from InForm either manually or automatically, your manual changes are lost. See "Loading InForm Metadata" on page 3-17 for more information.

Manual Changes Not Allowed in InForm Models

You cannot make any manual structural changes in an InForm clinical data model, including:

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

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 modify an InForm model:

1. Select the Development lifecycle on the Home page, then navigate to the study and model, and then click the InForm Configuration tab.
2. Click the **Suspend Data Loading** icon for the Development lifecycle. The **Check Out** icon appears.
3. Check out the model.
4. Make your changes. See:
   - "Adding Constraints to a Table" on page 3-11
   - "Setting Blinding-Related Attributes" on page 3-9
   - "Setting the Data Processing Attribute" on page 3-10
   - "Adding and Modifying Tables Manually" on page 3-8 for information on adding tables aliases and SDTM identifiers—but do not add or remove tables or columns.
   - "Adding Columns to a Table" on page 3-13 for information on adding column aliases, code lists, and SDTM identifiers—but do not add or remove columns.
   - "Updating Validation Status" on page 3-27

5. Click the **Install Model** icon. 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.

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 from Library Model** icon appears. Click it to synchronize the study model with the library model.

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

Rolling Back Changes to the Last Production Version

If a clinical data model is being used in Production but you have created a new version of it in the Development lifecycle, you can undo all changes and revert to the Production version—for example, if a protocol amendment is cancelled. For InForm models, the rolled-back Development model is installed as part of the process. If the older, Production version has fewer columns or shorter ones, for example, the rollback is "destructive" and all data is deleted from the model in Development.

1. Click the **Roll Back Clinical Data Models to Production Version** icon in the Clinical Data Model pane on the left of the Study Configuration page. A window appears.
2. Select one or more clinical data models to roll back. Use Ctrl+click or Shift+click to select multiple models.
3. Click **Roll Back**.
4. Click the **Refresh** icon periodically to see the Job ID, Log, and Job Status.

Modifying a Table

1. Check out the clinical data model if it is not already checked out.
2. Select the **Edit** icon in the Tables tab.
3. Edit table attributes; see "Adding and Modifying Tables Manually" on page 3-8.
4. When you have made all your changes in the model, **install** the model.

Removing a Clinical Data Model

To delete a clinical data model, select it in the Clinical Data Model pane and click the **Remove** icon. You can remove the whole model, including all its tables, at the same time. You must have the Remove Model privilege to do this.

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

See also "Setting Up Library and Study Categories" on page 8-20.

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. However, if you have made changes to the model in the study, you lose those changes when you upgrade to the new library version.

Libraries correspond to the categories—for example, therapeutic areas—that your company creates for the purpose of grouping studies and library data models; see "Setting Up Library and Study Categories" on page 8-20.

Creating 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 8-20.
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 3-11.

---

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 a Library Clinical Data 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 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 the appropriate icon and making the changes; see "Adding Tables to a Clinical Data Model" on page 3-3.
   - **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 3-13.
   - **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 3-11.
   - **Update Validation Status** The system displays this icon only if you have the privileges required. Click it 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 3-27.
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.

Deleting a Clinical Data Model

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

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 **Add** icon.
2. **Codelist Name**
3. **Description**: (Optional) Enter a description that will help users know what it is, such as a list of its values.
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**: This is the value displayed in the user interface. 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** icon.

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

5

Transforming Data to Standard Structures

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.

- 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 hard-coded 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.

The Automap feature maps tables and columns within mapped source and target models. You 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 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. You can set blinding to cascade downstream.

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 a 1-to-many table mapping—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 "Creating a Custom Program" on page 5-21.

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

Mapping Models, Tables, and Columns

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

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

You can view existing table or column mappings in the user interface and export them to a spreadsheet by clicking the **Export All to Excel** icon in the Target Table or Target Columns pane.

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 Oracle Health Sciences 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 to be populated by data from within the system by a transformation, not by loading 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 both the Target Tables and Source Tables panes in case you want to populate some target tables with data already transferred or transformed into other target tables.
3. Click the **Add or Remove Source Model** icon 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 models 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 Not Used or map it to one or more source tables. See "Marking Target Tables and Columns as Not Used" on page 5-8.

Using Side Models

To allow many people to work on a single model-level transformation at the same time, each person can create a *side model* with selected target tables to work on in a corresponding *side transformation*.

Creating Side Models

1. In the Target Tables pane of the transformation, select the target tables you want to work on. Use Ctrl+click or Shift+click to select multiple tables. The **Create Side Model** icon is enabled when you select the first table.

   An icon between the mapping icon and the table name indicates tables that are already included in side models. Hover over the icon to see which side model includes the table and the user who has it checked out. If you select a table that is already included in a side model, a message appears when you click **Create Side Model**.

   You can select tables marked Not Used. If you map them in the side model, they will be mapped and marked Used when you merge back to the main transformation.

   Notes:

   - To work with a staging table you must create it in the main transformation and then select it for your side transformation; see "Creating Staging Tables" on page 5-7.
   - A table can be the target of only one side transformation at a time. Oracle recommends working on one side model at a time, then merging it into the main transformation and deleting the side model.

2. Click the **Create Side Model** icon. The Create Side Model window opens.
3. Enter a name for the side model. If you enter a name that already exists, the system appends `_1` or higher to the name when you save.
4. Click **OK**. The system displays the side model under the target model in the Transformation pane on the left, with an icon indicating that you have checked it out, and opens a checked-out side transformation with the selected target tables and any existing mappings.

   You can check in, install, and execute the side transformation without affecting the main transformation. You can see data in the target tables in the Listings page and create custom listings on their data.

   The **Upgrade Required** message appears in the side transformation if a source or target model has been modified and the main transformation has been upgraded. (The main transformation must be upgraded before any side transformations.) Click the **Upgrade** icon to upgrade.

   An administrator can check in side models created and checked out by other users so that work can proceed in the absence of the original user.

Merging a Side Transformation with the Main Transformation

1. Navigate to the main transformation for the target model.
2. Check out the main transformation. If another user has it checked out, you cannot copy the side model into it.

3. Click the **Copy from Side Model** icon in the Target Tables pane. A window opens that lists all side models.
4. Select the side model and click **Next**. If any tables have been modified since you created the side model, the system displays a message and you can choose to include the tables in the copy or not.
   - If you continue, the side model tables overwrite the main model tables.
   - If you cancel the copy operation, you can upgrade the side model to reflect changes made in the main model.
5. When you have finished using a side model, select and delete it using the **Delete Side Model** icon. The system warns you if its side transformation has not been copied to the main transformation.

Side Model Limitations Side models are intended for temporary use only and have limitations:

- Side models cannot be used as source models for other transformations.
- If a validation check raises discrepancies on data in a side model, lineage tracing is not enabled and the discrepancies do not appear in the source model.
- You cannot add or remove source models through a side model.
- Side models are not visible in the Study Configuration page.
- Side models and transformations cannot have a higher validation status than Development.

Copying Table-Level Transformations

1. In the Table Mapping pane, select target tables that are the same or similar to the target table mappings you plan to copy.
2. Click the **Copy from Another Transformation** icon. The system displays all target models to which you have access that have checked-in transformations. Type part or all of the name of the Therapeutic Area (or other category), Study, and Model to filter the list.
3. Select a model. The system displays all the tables in the model. Type all or part of the table name to filter.
4. Click **Next**. The system validates and compares the selected target tables in the current model to the selected target tables to be copied and displays the Suggested Mappings window, with details of the transformations to the selected target tables, including, for each target table:
   - The target table's columns. (Expand the table node to see them.)
   - The source tables mapped to the target table.
   - The type of transformation (Direct, Join, Union, Pivot, Unpivot, or Custom).
   - Any errors or warnings detected by the validation:
     - For **pivot** and **unpivot** transformations, if metadata differences exist, the system displays an error. If the differences are on the source side, they must be resolved manually. If the differences are on the target side, you can run synchronization to resolve the differences and then copy. See "Synchronization" on page 5-5.
     - For **union** and **join** transformations, if some of the source tables or columns in the union do not exist in the copied-to study, those mappings

are not included and a warning is displayed. If only one table exists in the copied-to study, the union or join map is converted to a direct map and a warning is displayed.

- For **custom** transformations, if metadata differences exist, the system displays an error. If the differences are on the source side, they must be resolved manually. If the differences are on the target side, you can run synchronization to resolve the differences and then copy. See "Synchronization" on page 5-5.
- For **column mappings that use expressions**, if all the column references do not exist in the current source tables, the missing column references are removed from the expression.

The system prevents you from copying any transformations with errors, but you can copy those with warnings.

5. Review the mappings, select those you want to copy, and click **Accept Selected Mappings**. The system copies the transformation and creates the mappings.

> **Note:** If the source and target tables in the mappings being copied have more columns than the tables in the current model, you can mark those columns as Not Used; see "Marking Target Tables and Columns as Not Used" on page 5-8.

Synchronization The synchronization job modifies the tables in the current model so that they match the table metadata being copied into the current model. Synchronization changes column data type and length, drops additional columns, and creates missing columns.

Using Automap

Automapping can run on all tables and columns in the target model or on up to ten selected target tables for faster results. Use the Automap icon at the top of the page to map all tables, or the icon in the Target Tables pane to map selected tables.

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** icon 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 "Creating a Custom Program" on page 5-21.

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 performs: Direct, Join, Union, Pivot, or Unpivot. Use Custom only if your program cannot support data lineage tracing; see "Creating a Custom Program" on page 5-21.

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 5-11.

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 3-9.

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 an alias to multiple occurrences of the same table to create a self-join; see "Creating a Self-Join in a Transformation" on page 5-7.
- Write an **expression** to operate on the source tables' 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 for Transformations and Blinding Criteria" on page 5-16.

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

Creating a Self-Join in a Transformation

To create a self-join:

1. In the Mapped Sources tab, select the table in the Source Tables pane and the Target Tables pane and click the **Map** icon. The system adds the source table to the Mapped Sources tab again. Repeat if necessary. Enter an alias each occurrence and save.
2. Click the **Join** icon for the target table. The Join window, displays the table names qualified with an alias to differentiate between the multiple occurrences of the same table. Configure criteria for the self-join and save the table map.
3. Click the **Map Column** icon for the target table. The Source Columns pane displays table names qualified with their alias. Select the source column(s) to map to the target column.

Creating Staging Tables

Complex transformation may require multiple steps with 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 the **Create Staging Table** icon.
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 tables 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 **Modify** icon 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 others 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. The system checks out the target clinical data model and adds the staging table to it.

> **Note:** If you need to make changes to the staging table—for example, associating a column with a codelist for use as a pivot column—navigate to the target clinical data model and make the changes, install the model, then return to the transformation.

Marking Target Tables and Columns as Not Used

If you do not need a particular table or column in a target model that you have copied, you can mark it as Not Used. This has several effects:

- They are not visible in the Listings pages, so that the Listings pages display only the tables and columns that are relevant in your study.
- The system disables any validation checks and custom listings that are dependent on a table or column marked Not Used. If you later mark the table or column as Used, you must manually reenable the validation checks and custom listings.
- The system does not consider the table or column when computing the status of the transformation mapping; the status can be Complete without mapping the table or column. Completeness is required for validation and installation.

> **Note:** The system does not include columns that are populated by TMS in the completeness calculation even if they are not marked Used.

To mark a single target table as Not Used select it and click the **Mark as Not Used** icon.

To mark all unmapped tables and columns as Not Used after you have mapped all the target tables you need, click the **Mark as Complete** icon. The system:

- Displays a warning that lists all unmapped tables and columns that will be marked Not Used if you confirm that you want to continue.
- Displays all downstream transformations; those that use this transformation's target model as sources.

  To mark tables and columns in a downstream transformation as Not Used, select it and click the **Cascade Mark as Not Used**. Any table or column that is mapped to a table or column in the current transformation that is marked Not Used is marked Not Used.

> **Note:** Marking a target table or column as Not Used overrides any mappings to it that may exist.

To mark all unmapped columns in a table as Not Used after you have mapped all the target columns you need, click the **Mark as Complete** icon in the Target Columns pane.

To mark a table or column as Used when it is already marked Not Used, select it and click the **Mark as Used** icon.

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 **Map Column** icon. The Map Columns window opens.
2. Click the **Automap** icon (optional). The system 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** icon in the target column's row. You may need to click it twice.

   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): To write an expression operating on the target column data as part of the Select clause, select the target column and click the icon in the Expression tab below. The Expression Builder opens; see "Using the Expression Builder for Transformations and Blinding Criteria" on page 5-16.
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 columns. 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 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 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 source and target models if they are 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

You can undo existing mappings all at once or one at a time.

Undo Multiple Mappings

You can undo multiple mappings at three levels:

- **Model**: To unmap all table and column mappings in a target model, click the **Unmap All** icon at the top of the Transformations page.
- **Table**: To unmap all table and column mappings for selected tables, click the **Unmap** icon in the Target Tables pane.
- **Columns**: To unmap all column mappings for a single table, but not the table mapping, click the **Unmap** icon in the Target Columns pane.

You must confirm before the unmapping proceeds.

Undo a Single Table or Column Mapping

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 **Delete** icon in the Mapped Sources tab.

Cascading Blinding and Masking

As you work on table and column mappings, you can use the View drop-down to display columns in the Source Tables or Columns pane with blinding-related information: Blinding Criteria, Blinding Status, Blinding Type, and Blinded Table.

After you have completed all the mappings, you can cascade source blinding settings to target tables.

Cascading Blinding

1. In the Table Mapping pane, click the **Cascade Blinding** icon. The Cascade Blinded Tables window opens. It displays all blinded source tables and the target tables to which they are mapped, with a description of the type of blinding defined for the source table and the effect on the target table if you do cascade blinding:
   - If a source table has *table-level* blinding, the target table will also be blinded at the table level.
   - If a source table has *column-level* blinding, the target table will also have column-level blinding on the same columns.
   - If a source table has *row-level* blinding, you must enter blinding criteria for the target table. No blinding is cascaded.

   You can expand the node for any source or target table to see blinding details for its columns. For target tables with column-level blinding, you can select or deselect individual columns for cascade blinding.
2. To accept the default blinding for a target table, select it and click **Accept**.

   To accept the default blinding for all target tables, select **Select All** and click **Accept**.

Cascading Masking

Masking substitutes a dummy value for real, blinded, data.

1. In the Column Mapping pane, click the **Cascade Masking** icon. The Cascade Blinded Columns window opens. It displays all blinded source columns and the target columns to which they are mapped, with a description of the type of blinding defined for the source column and the effect on the target column if you do cascade blinding:
   - If a source column has *column-level* masking, the target column will, too.
   - If a source column has *cell-level* blinding, the target column will have *column-level* masking until you specify masking criteria.
2. To accept the default masking for a target column, select it and click **Accept**.

   To accept the default masking for all target columns, select **Select All** and click **Accept**.

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, pivots, and unpivots, see *Oracle® Database SQL Language Reference* 11g Release 2 (11.2).

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 columns that will feed data to it and click the **Map** icon.
   - Click the **Mark as Not Used** icon.
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 tables and target table and click the **Map** icon, 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 check box 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 the **Mark as Not Used** icon.
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 lab vendors providing results.

To create a Union transformation:

1. Select the source and target tables and click the **Map** icon, 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:
    - Select the source columns and click **Map**. See Data Lineage Tracing.

      Note: In a union, if only one source column is mapped to a target column, the system adds a null in the code for the other table(s). For example:

      ```
      insert into tgt_table1
      (col1, col2, col3)
      select col1, col2, col3
      from table1
      union all
      select col1, col2, null
      from table2;
      ```

    - Click the **Mark as Not Used** icon.
4. Save.

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

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 the **Map** icon, 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 **Map Column** icon. 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 the **Map** icon, then select a Transformation Type of **Unpivot**.
2. Click the **Unpivot** icon 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 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 "Creating a Custom Program" on page 5-21 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 Transformations and Blinding Criteria

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.

1. In the Expression Criteria pane, select the following as needed to build the expression from left to right.
   - **Add Group** to add the parentheses () that surround a string in an expression or group smaller units of logic.
   - **Add Item** to add a unit of logic smaller than a group.
2. When you add an item, in the Expression Item pane select either **Column**, **Function**, or **Standard Function**.

   To create an expression using columns:

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

   b. Click the **Select Column** icon. The Select Column window appears, displaying all available columns. You can query (see "Querying By Example" on page 10-7) 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 constant value. The system encloses the value you enter in single quotes.

   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 edit it there.

      Click **Validate** to check the generated code.

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

   To reference a function in your library:

   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 5-18.

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

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

   To use a SQL function:

   a. For Item Type, select **Standard Function**.

   b. Click the **Select Standard Function** icon. A search window appears. To filter, enter all or part of the name in the field above. You can use the wildcard `%`.

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

   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**.

For more information about expressions, see:

- "Passing Data as Input Parameter Values" on page 5-18
- "Passing Metadata as Input Parameter Values" on page 5-18
- "Passing Constant Values" on page 5-18
- "Developing a Library of SQL Functions" on page 5-18

Also see the *Oracle® Database SQL Language Reference* 11g Release 2 (11.2).

If you need to perform a more complex operation on the source data—for example using a lookup table or populating staging tables—see "Creating a Custom Program" on page 5-21.

Passing Data as Input Parameter Values

Use curly brackets ("{" and "}") as delimiters and the format (*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 "Creating a Custom Program" on page 5-21.

Each static reference program must:

- Be created in an Oracle LSH application area in the DMW_UTILS domain, and installed in a Work Area with a Usage Intent of Production.
- 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:

- To validate all mappings in the model, click the **Validate Mappings** icon at the model transformation level.
- To validate mappings for selected tables, select one or more target tables and click the **Validate Mappings** icon at the table level.

To see the error messages for a target table or its columns, hover over the table or column's mapping status icon.

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 tables, 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 a Transformation

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 source and target models and installs the models if they are not already installed.
- 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 5-29.

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 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 **Install Job Log** icon. 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 is checked in, you can 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.

Notes:

- Source models are not required to be installed to be used in the upgrade. They must be checked in.
- On the target side, the upgrade is to the latest checked-out version of the target model—the one the system has locked for the user to work on.
- Side transformations are also upgraded.

Creating 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 Health Sciences Life Sciences Warehouse 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 5-27.
- 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 5-22
2. "Creating a Custom Program in Oracle Life Sciences Data Hub" on page 5-23
3. "Completing the Transformation" on page 5-25

This section also contains:

- "Sample Programs that Populate Auxiliary Columns with the Source Surrogate Key" on page 5-26
- "Sample Program that Calls the API to Set a Flag" on page 5-27

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 5-29.

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 **Add** 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 5-26 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 Oracle DMW_UTILS domain under the Oracle DMW_DOMAIN.
2. In the Oracle 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 8-21. 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 window:
   a. In the **Domain** list, select Oracle 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 columns 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. Oracle 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 DMWOracle DMW public APIs from your code; see the *Oracle Health Sciences Life Sciences Warehouse 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 6-9. If you are defining a transformation, follow instructions in the next section.

Completing the Transformation

If you have not already created auxiliary key columns in the target table, do so now; see "Enabling Data Lineage Tracing in a Custom Program" on page 5-22.

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

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.

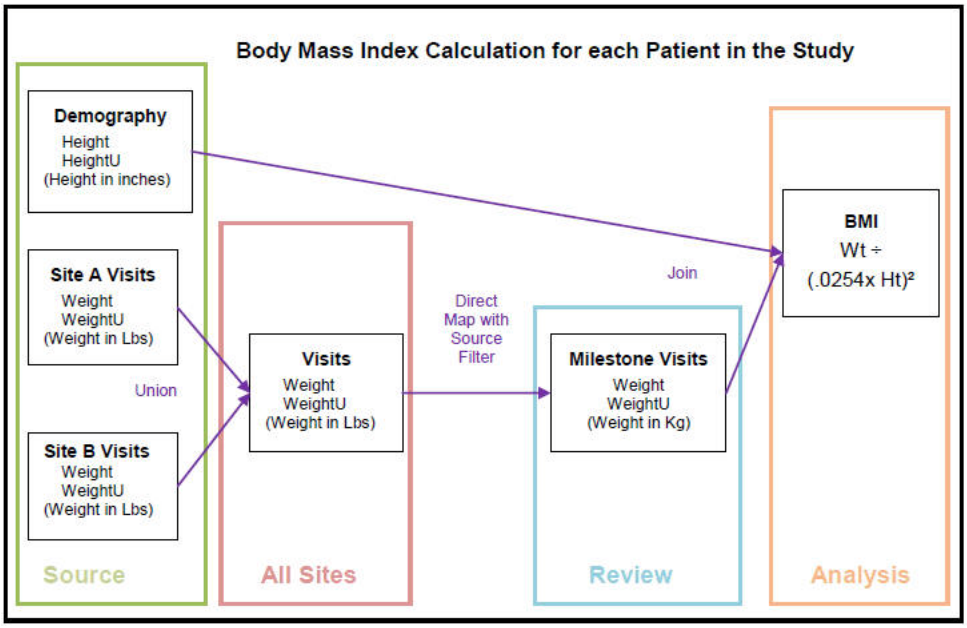
The system uses several mechanisms to maintain context:

- **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.

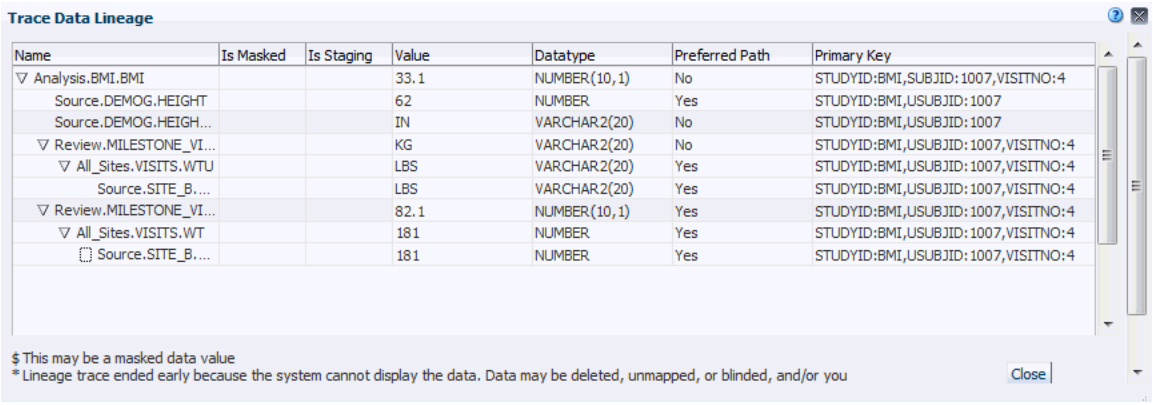
The model Analysis has a table called BMI whose BMI column value is the computation for Body Mass Index using the height from Demography and weight from Milestone Visits.

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

See "Viewing and Running Validation Check Batches" on page 10-4.

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 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 6-9 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 the **Create Batch** icon.
5. Enter a name for the batch. If it is not unique within the study, the system appends "_1" or the next higher number 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 6-3.

Copying Validation Checks

You can copy validation checks from another study or from a different clinical data model in the same study. The system checks if the required source tables are available in the current model.

> **Tip:** Copy validation checks **after** completing the transformation that writes to the model, so that you know which tables are used in the model. If you copy validation checks that read from tables or columns marked Not Used in the transformation, the system copies them as disabled. If the tables or columns are later marked Used, you must manually enable the validation checks.

1. Select the validation check batch into which to copy validation checks. In the Study Configuration page, navigate to Validation Checks, then to the clinical data model, and then to the batch.
2. Click the **Copy Checks** icon in the Validation Checks for *batch_name* pane. The **Copy Validation Checks** window opens.
3. The system displays all validation checks to which you have access. You can type part or all of the name of the Therapeutic Area (or other category), Study, and Model to filter the list.

Click **Clear Filters** to remove all typed text and revert to the full list.

4. Select a model. The system displays all the validation checks associated with the model, with their validation check batch. You can type all or part of the batch or check name to filter.
5. Select one or more validation checks. Use Ctrl+click or Shift+click to select multiple checks.
6. Click **OK**. The system searches the current model for the tables and columns that the validation checks read from, first by Oracle name and then by alias.
   - If the tables or columns do not exist, the Copy operation fails with an error message.
   - If they exist but are marked Not Used in the transformation that writes to the model, the system copies the validation checks as disabled and displays the reason they are disabled on the Validation Checks page.
   - If the tables and columns exist and are used, the system copies the validation checks and links them to the appropriate tables and columns.

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 6-1.
2. Select the batch. If it is not already checked out, click **Check Out**.
3. Click **Add Check**.

Building the Validation Check Query

To create the query for a 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 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 5-23. Then click the **Select a Program** icon to select the custom program.

   If you select **Create VC Using a Custom Program**, select the source tables 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.

> **Note:** The custom program's source tables' Oracle name cannot be more than 25 characters long. This is because the target table must contain a column for surrogate key information that contains the source table name.

4. Specify the columns to display; see "Selecting Columns to Display in VC Listings" on page 6-4. The system creates a SELECT clause based on your specifications.
5. If you need to use a function from a library in the SELECT or WHERE clause, and you plan to write the expression using the function in free text, open the **Select Packages** tab and check the packages you will use. This enables the system to generate the query code and supports the Test function.

   If you use the Expression Builder (reached by clicking the **Modify Expression** icon), you do not need to use the Select Packages tab.
6. If you need to use a self join in the SELECT clause, open the Define Table Alias tab and click the **Add** icon to add another row for the same table. Specify a different aliases for the table in each row. Use the aliases to create a self join in the Criteria pane.

   > **Note:** The table alias must be **three characters or less**. If it is longer it causes problems.

7. Specify query criteria; see "Specifying Criteria" on page 6-5. The system creates a WHERE clause based on your specifications.

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.

   > **Note:** Tables and columns marked Not Used in the transformation are not displayed here, nor are uninstalled tables.

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**, each in its own row.

   To write an expression that operates on multiple columns, add all columns in the expression to the same row:

   a. Move one column into Selected Columns using the **Add** icon and highlight it there.

   b. Select the additional columns in the Source pane and click the **Arrow** icon in the Source pane. The system adds them to the same row.

4. **Table Alias**: If you have defined table aliases for a table in a self join in the Define Table Alias tab, select a column and click the **Select Table Alias** icon to specify which table alias the selected column belongs to.
5. **Alias**: To display a heading for the column different from the column name, enter it in the Alias field. This heading appears in the VC Listings page.

> **Note:** This is different from giving a column or table a database alias, which you do in the data model under Study Configuration.

6. **Expression**: Add the expression, if any, to operate on the column in the SELECT clause. Enter free text or click the **Modify Expression** icon; see "Using the Expression Builder in Validation Checks and Custom Listings" on page 6-7. You can edit code generated by the Expression Builder in this field afterward.

Selecting Packages

If you plan to create expressions in the query using free text, first use the Select Packages tab to identify packages whose functions you will use. This enables the system to generate the query code and supports the test function. The system displays all packages in the DMW_UTILS domain that meet certain criteria; see "Developing a Library of SQL Functions" on page 5-18.

If you use the Expression Builder, you do not need to use the Select Packages tab.

Adding Table Aliases for a Self-Join

To create a self-join:

1. Add the table and column(s) to the Selected Columns tab.
2. Open the **Define Table Alias** tab and click the **Add** icon to add the table as many times as required.
3. Specify a different alias for the table in each row.

> **Note:** The table alias must be three characters or less. If it is longer it causes problems.

4. In the Selected Columns tab, select a column and click the **Select Table Alias** icon to specify which table alias the selected column belongs to.
5. Specify the join criteria in the Criteria pane. See "Using the Expression Builder in Validation Checks and Custom Listings" on page 6-7.

Specifying Criteria

In the Criteria pane, build the WHERE clause.

Click the **Add or Modify Criteria** icon to open the Expression Builder. See "Using the Expression Builder in Validation Checks and Custom Listings" on page 6-7.

You can edit code generated by the Expression Builder or enter code directly in the Criteria pane.

Note: Oracle recommends using the Expression Builder to add columns so that Oracle DMW "knows" what you are using in the expression and can use table and column aliases defined in the clinical data model to help determine if source tables and columns exist when you copy validation checks.

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 the **Check Out** icon.
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 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 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.
   - **Primary Source Column**: Select the column that contains the primary data point.

     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.

- **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.

Using the Expression Builder in Validation Checks and Custom Listings

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 enter free text or edit generated code in the Expression Text pane.

> **Note:** Oracle recommends using the Expression Builder to add columns so that Oracle DMW "knows" what you are using in the expression and can use table and column aliases defined in the clinical data model to help determine if source tables and columns exist when you copy transformations, validation checks, and custom listings.

1. In the Expression Criteria pane, select the following as needed to build the expression from left to right.
   - **Add Group** to add the parentheses () that surround a string in an expression or group smaller units of logic.
   - **Add Item** to add a unit of logic smaller than a group.
2. When you add an item, in the Expression Item pane select either **Column**, **Function**, or **Standard Function**.

   To create an expression using columns:

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

   b. Click the **Select Column** icon. The Select Column window appears, displaying all available columns. You can query (see "Querying By Example" on

page 10-7) 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 constant value. The system encloses the value you enter in single quotes.

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 edit it there.

Click **Validate** to check the generated code.

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

To reference a function in your library:

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 5-18.

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

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

To use a SQL function:

a. For Item Type, select **Standard Function**.

b. Click the **Select Standard Function** icon. A search window appears. To filter, enter all or part of the name in the field above. You can use the wildcard `%`.

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

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**.

For more information about expressions, see:

- "Passing Data as Input Parameter Values" on page 5-18
- "Passing Metadata as Input Parameter Values" on page 5-18
- "Passing Constant Values" on page 5-18
- "Developing a Library of SQL Functions" on page 5-18

Also see the *Oracle® Database SQL Language Reference* 11g Release 2 (11.2).

If you need to perform a more complex operation on the source data—for example using a lookup table or populating staging tables—see "Creating a Custom Program" on page 5-21.

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 10-4.

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 the **Modify Batch** icon and see instructions for "Creating a Validation Check" on page 6-3.

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 and Enabling a Validation Check

To prevent a validation check from being executed when the batch is executed, select it and then click the **Disable Check** icon. To include it in the batch execution after disabling it, select it and then click the **Enable Check** icon.

Note: The system disables validation checks if their source tables or columns are marked Not Used in the transformation that writes to this clinical data model. The system prevents you from enabling a check whose source tables or columns are marked Not Used.

You can change this setting whether the validation check batch is checked in or out. If a check is disabled, the system records who or what disabled it in the **Disabled Reason** column. Hover over the value to see the complete text.

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 the Study Configuration page.
2. Select the batch and click the **Upgrade Batch** icon. The system updates the table and column mappings behind the scenes.

Creating a Custom Validation Check

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 5-23 except for this additional task:

   Each validation check should have a single target table with the following requirements:

   - The table name should be no longer than 25 characters.
   - The 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 5-29.

   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 tables 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 8-21.

7

Configuring Your Discrepancy Workflow

Discrepancies are equivalent to Oracle Health Sciences InForm queries. They can be raised against InForm and non-InForm data in Oracle Health Sciences Data Management Workbench (Oracle DMW).

- 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 or a lab. When sent to InForm, they are created as queries.
- Discrepancy state changes made in Oracle DMW are sent to InForm almost immediately using a web service.

> **Note:** The exception is discrepancies raised in Oracle DMW against items that are hidden in InForm. InForm does not accept queries against hidden items, so these are never sent to InForm. Any resolution must occur within Oracle DMW.

- Changes to queries made in InForm are updated in Oracle DMW in the next data load from InForm for the study.
- Discrepancies raised against lab data can be exported to a spreadsheet and then sent to the source lab. Updates made in the lab are imported during the next data load.

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

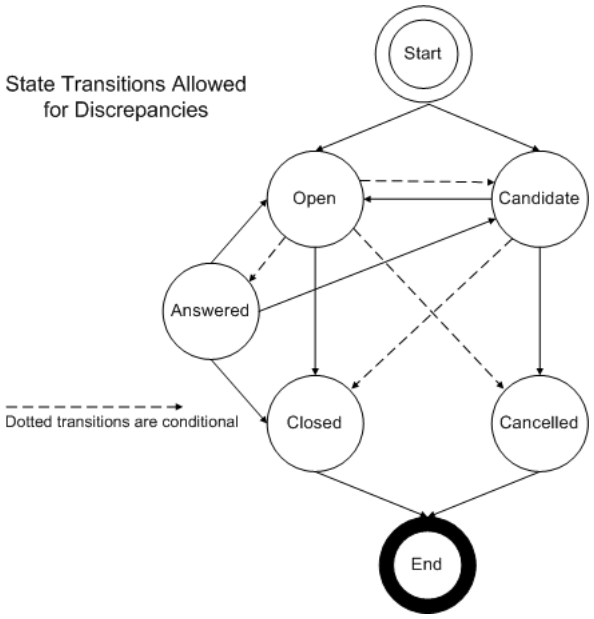
Oracle DMW comes with a set of discrepancy states, allowed transitions between them, and **action**s that change the state of the discrepancy and/or apply a **tag**. You can define custom actions, following the same rules.

When a discrepancy is sent to InForm, the system limits the actions an Oracle DMW user can take on the discrepancy. The same limited actions are allowed whether the discrepancy originated in InForm or in Oracle DMW.

While a discrepancy is in InForm, Oracle DMW users can apply comments and categories to it. These are never sent to InForm.

While a discrepancy is in Oracle DMW, a user can apply an action that changes the state of the discrepancy and must supply a reason for the change. The reason for change is sent to InForm.

Figure 7–1 Discrepancy States and Allowed Transitions for Discrepancies



The diagram shows valid transitions for discrepancies in Oracle DMW:

- 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 always move to Open.
- Candidate discrepancies can always move to Cancelled.
- Candidate discrepancies can move to Closed if they have not been sent to InForm (no matter what kind of data they are on). Also, Candidate discrepancies that ARE in InForm (whether they originated in InForm or DMW) can move to Closed if they were previously in the Open state within InForm—because this matches InForm rules.
- Open discrepancies can move to Answered if they were created in Oracle DMW and have not been sent to InForm (no matter what kind of data they are on).
- Open discrepancies can always move to Closed.
- Open discrepancies can move to Cancelled only if they were created in Oracle DMW and have not been sent to InForm (no matter what kind of data they are on).
- Open discrepancies can move to Candidate if they were created in Oracle DMW and have not been sent to InForm.
- Answered discrepancies can be reopened in either the Open or Candidate state.
- Discrepancies can be always be closed from the Open or Answered state.

Actions

Users apply *actions* to discrepancies to move them from one state or tag label 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.

  Note: Users must select and/or enter a **reason for change** when they apply an action to a discrepancy. You can create a list of valid reasons for users to select by adding values to the DME_DSC_ACTION_REASON lookup.

  See "Setting Lookup Values" on page 8-31..

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.

In some cases the action also triggers a routing action on the discrepancy, either sending it to InForm or exporting it to spreadsheet so it can be sent to a lab.

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 Discrepancy**	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
Open	**Answer Discrepancy**	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)

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

Start State	Action Name	Routing Operation	Result State	Result Tag
Open	**Reset to Candidate**	None	Candidate	None
Open	**Close Discrepancy**	None	Closed	Closed By Data Change (if a data change occurred in the source system)
Answered	**Reopen Discrepancy**	None	Open	None
Answered	**Reopen to Candidate**	None	Candidate	None
Answered	**Close Discrepancy**	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 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.

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 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 **Export All to Excel** 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 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 Oracle 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 and Editing 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.

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

- **Name**: The action's name is displayed in the user interface in the Actions drop-down for the user to select. 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**: The label is required but is not used by the system. It 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 ID** (Optional): If the discrepancy needs to be checked or resolved in the source data system, select the appropriate operation:
  - **Send to InForm**: The system sends the discrepancy to InForm immediately and applies a transition restriction of Limited to it.
  - **Send to Spreadsheet**: 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

Some administration tasks are done partially or entirely in the Oracle Health Sciences Data Management Workbench (Oracle DMW) user interface, while others require logging in to the Oracle E-Business Applications Suite.

Setting Up and Monitoring File Watchers

The File Watcher utility checks a file system location that you specify for data files whose name matches a pattern you specify and loads them into Oracle DMW.

Prerequisites These tasks are documented in the *Oracle Life Sciences Data Hub Installation Guide*:

- Set up the Oracle LSH Distributed Processing (DP) Server and, as part of it, the File Watcher service. The File Watcher service detects files to be loaded.
- Define at least one DP Service Location and SAS and Text for SQL*Loader DP Server services in the Oracle LSH user interface. These services load data from files into the Oracle DMW database. See the *Oracle Life Sciences Data Hub System Administrator's Guide*.
- Create nested directories for files to be loaded. They can be on one or more computers to which the DP Server has access:
  - A root folder on each computer from which files are loaded into Oracle DMW.
  - Either three or six subfolders:

    Three folders, one for each lifecycle. Both SAS and text files are placed in the same folder.

    Six folders, one for each lifecycle/file type combination. Each folder is used for only one file type, either SAS or text.

    See "Registering Folder Locations" on page 8-2.
- Create nested directories for loaded files to be archived:
  - A root folder on each computer where you plan to archive data files. Oracle recommends archiving files on the same computer and same file system for efficiency of moving the files to the archive folders.
  - Either three or six subfolders for archiving. These must correspond to the three or six subfolders you set up for loading data.
- Security for the folders.
- Compatible time zone configuration on each computer.

Tasks

- Register the location of the subfolders for loading and archiving files in Oracle LSH system profiles; see "Registering Folder Locations" on page 8-2.
- For each study, specify a study-specific folder name; see "Creating and Modifying Study File Watchers" on page 8-3.
- For each file data source in each study, create an input clinical data model with a File Watcher file specification; see "Creating a Study Clinical Data Model" on page 3-2 and "Configuring File Watcher" on page 3-21.
- For each data source across all studies, register the data source in the Oracle DMW; see "Viewing and Setting Up Lab Data Sources" on page 8-5.

Registering Folder Locations

1. Log in to Oracle LSH as system administrator. See "How to Log In to Oracle Applications Profile Forms" on page 8-25.
2. Register the root folder locations in Oracle LSH profiles; see "Registering Root Folders for File Watcher Watched Folders" on page 8-27.

Changing the Root Folder Location

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

1. Create new root folders in the new location for both watched and archive folders.
2. Change the value of the LSH profiles to the new location. See "Registering Root Folders for File Watcher Watched Folders" on page 8-27 and "Registering Root Folders for File Watcher Archive Folders" on page 8-28.
3. Restart the Distributed Processing Server that includes the File Watcher service. Instructions are in the *Oracle Life Sciences Data Hub System Administrator's Guide*.
4. Open each study File Watcher in Edit mode and click **Regenerate** to apply the new settings in the study; see "Creating and Modifying Study File Watchers" on page 8-3.

The system then:

- Migrates all existing Study File Watchers to the new root folders.
- Processes any files already submitted for data load at the time of the change.
- Does **not** move existing files to the new location.

The system does not move any files. You must: .

- Determine which files have not been loaded, if any, and move them to the new location.
- Determine which loaded files have not been archived or deleted, if any, and move or remove them.

Selecting the Distributed Processing Server

Specify the Distributed Processing (DP) Service Location to use for loading SAS and text files. These settings apply to all Study File Watchers of the corresponding file type.

Note: The DP Server should have been set up during Oracle DMW installation. See the *Oracle Life Sciences Data Hub Installation Guide* and the *Oracle Health Sciences Data Management Workbench Installation Guide* for more information.

1. In the Watcher Listing pane of the Administration page, click the **DP Servers for File Watcher** icon. The DP Servers for File Watcher window opens.
2. **Text DP Server**: From the drop-down list, select the DP Service Location to use to detect text data load jobs.
3. **SAS DP Server**: From the drop-down list, select the DP Service Location to use to detect SAS data load jobs.
4. Click **OK** to save.

Archiving Data Files

To improve File Watcher performance, minimize the number of data files in the watched folders by either permanently deleting them or moving them to an archive folder automatically.

When you create a study File Watcher you must specify a number of days after the load date to delete data files. If you prefer to archive the files, specify fewer days before archiving them. The system either archives or deletes data files, depending on which number of days is smaller.

When the system moves a file to an archive folder, it appends the current timestamp to the file name to ensure uniqueness.

To set up archiving:

1. Log in to Oracle LSH as system administrator. See "How to Log In to Oracle Applications Profile Forms" on page 8-25.
2. Register the archive root folder locations in Oracle LSH profiles. See "Registering Root Folders for File Watcher Archive Folders" on page 8-28.
3. Enable archiving and specify the number of days after loading data to move the file to an archive folder, using Oracle LSH profiles; see "Registering Root Folders for File Watcher Archive Folders" on page 8-28.
4. For each study File Watcher, specify a larger number of days in the Delete Files After field than in the Oracle DMW:FWR_ARCHIVE_SCHEDULE_DAYS profile.

Note: If you enable archiving after setting up a study File Watcher, open the study File Watcher in Edit mode and click **Regenerate** to apply the new archiving-related profile settings to the study.

Creating and Modifying Study File Watchers

For each study, define a study folder name. The system uses this folder name to create study-specific File Watchers and archive folders in the root folders you create and register; see "Registering Folder Locations" on page 8-2.

To create study file watchers:

1. In the Administration page, 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 get an error if you try to create Study File Watchers before the folder locations are defined in the LSH profiles; see "Registering Folder Locations" on page 8-2.

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 root folder. The Study folder name must be **unique within the Oracle 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 File Watcher Listing pane.

   The system creates the study folders in the locations you specified in "Registering Folder Locations" on page 8-2. If you have enabled archiving and specified locations for archiving files, it creates study-specific archive files as well.
5. **Delete Files After**: To permanently delete files, enter the number of days after the load date to delete each file. To *archive* the files instead of deleting them, enter a higher number here than in the archiving profile value.; see "Registering Folder Locations" on page 8-2.
6. Click **OK**.

   If you have changed File Watcher profile values since you created the study File Watcher, click **Regenerate** to apply the current settings; see "Changing the Root Folder Location" on page 8-2.

Monitoring File Watchers

Click **File Watchers** at the bottom of the right-hand pane in the Administration page to see the File Watcher Listing pane. It displays information about each File Watcher, including:

- **Watcher Status** (Running or Suspended). To change the status, click the **Resume Watcher** or **Suspend Watcher** icon.

  Note: If the Distributed Processing (DP) Server status is Offline, the Watcher cannot run, even if its status is Running. To start the DP Server, follow instructions in the *Oracle Life Sciences Data Hub Installation Guide* or the *Oracle Life Sciences Data Hub System Administrator's Guide* chapter on stopping and starting services and queues.

  You can suspend or resume data loading for a particular data source in the File Specification pane or in the corresponding clinical data model.
- **Exception messages** for errors creating the study or archive folder. Possible problems include:
  - The UNIX user that executes the DP Server process does not have permission to create a directory in the File Watcher root folders.
  - The study folder name is not a valid directory name on the UNIX system.
  - There is a UNIX system issue, such as the file system's running out of space.

To read the error message, drag the column edge to make it wider.

- Data load parameter values for text files for Release 2.3 File Watchers only.

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.

Study Folder Name/File Name Mismatch Report

If you are using Oracle DMW in a hosted environment, the **Study Folder Name/File Name Mismatch Report** icon appears in the File Watcher Listing pane. The report is a listing of files that have not been loaded into any study because File Watcher did not recognize them as belonging to a study. This can occur if:

- The study folder has not yet been created by the File Watcher service.
- The file name does not follow the naming convention that is required in hosted environments of starting with the study folder name followed by an underscore.

> **Note:** This is different from the files listed in the Files Not Processed tab. Those files are associated with a study but do not match the required file specification.

The system places such files in the Unknown_Study subfolder for the relevant lifecycle. The report lists all files contained in the folder, grouped by lifecycle. For each file, the report displays:

- The file name.
- The file modification date.
- The file size, in bytes.

Correct the problem and resubmit the file to the hosted environment. When the file has been loaded successfully, request that the old file be removed from the Unknown_ Study subfolder.

To run the report, click the **Study Folder Name/File Name Mismatch Report** icon and choose to open or save the file.

Viewing and Setting Up Lab Data Sources

Create a list of all the labs that send files to be loaded into the system. This list populates the list of values for input clinical data models.

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 Lab`. This value appears in the list of data sources in the File Watcher configuration tab for clinical data models.
   - **Description**: Enter a description of the data source.
   - **Contact**: Enter the name of the person who should receive discrepancies.

- **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 2.4.

Modifying a Lab Data Source

To change a lab data source definition, click the **Modify Data Source** icon, make changes, and click **OK**. See "Adding a Lab Data Source" for field details.

Deleting a Lab Data Source

To remove a lab data source, select it and click the **Delete Data Source** icon.

Setting Up Oracle Health Sciences InForm Integration

The *Oracle Health Sciences Data Management Workbench Installation Guide* for the first steps required to integrate Oracle Health Sciences InForm with Oracle DMW.

Viewing and Setting Up InForm Data Sources

In the InForm tab under Data Sources on 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 or Modifying a Remote Location

Create a database link to the InForm reporting database for the study:

1. Enter values 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: Do not use spaces.

   - **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 Oracle DMW InForm Connector" in the *Oracle Health Sciences Data Management Workbench Installation Guide* for further details. The system will use this account to connect.

     Note: Do not use spaces or special characters other than underscore (_) in the username or password.

- **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**.

Study configurators add or edit a remote location database link in an InForm model, but only if data loading is not enabled and Development was selected as the current lifecycle area on the Home page.

Adding or Modifying 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`

   - **Authentication 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.
   - **SSO Organization**: If your InForm installation, specifically the InForm Adapter, is hosted by Oracle, you **must** enter the organization associated with the DMW_QUERY user as provisioned for Single Sign-On use. Otherwise you **must not** enter anything in this field as it will cause harm.
   - **Username**: Enter the name of the account set up for authenticating Oracle DMW web service transactions. The default name is Oracle DMW_AUTH. See the section "Create Users in InForm" in the *Oracle Health Sciences Data Management Workbench Installation Guide*.

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

     Note: Do not use spaces or special characters other than underscore (_) in the username or password.

   - **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**.

Study configurators can add or edit a web service location database link for a particular study in the study's InForm clinical data model.

Removing a Remote Location or Web Service Location

To remove a remote location or web service location, select it and click the corresponding **Delete** icon

Modifying a Remote Location or Web Service Location

To modify a remote location or web service location, select it and click the corresponding **Modify** icon. Modify values as required; see "Adding or Modifying a Remote Location" on page 8-6 or "Adding or Modifying a Web Service Location" on page 8-7.

Selecting InForm Tables and Views

You can make InForm metadata and operational tables available in Oracle DMW studies. The tables and views you select in the Administration page become the default selection for all studies. These choices can be overridden at the study level.

The system displays selected InForm tables and views in the Listings pages with their data and in the Transformations page as source tables. The system stores the tables and views in two internal clinical data models named *Operational Data* and *Metadata*.

Some InForm tables and views are required for internal Oracle DMW processing. These are displayed with a gray check mark. You cannot select or deselect them.

To select tables and views:

1. In the Administration page, click **InForm tables and views** under **Data Sources**.

   The system displays all available InForm tables and views alphabetically. To sort by the internal data model they are part of if selected, click the heading of the **Internal Data Model** column.

2. Select the tables and views you want to include by default in Oracle DMW studies.
3. Save.

Available InForm Tables and Views

The system creates two internal clinical data models to store the InForm Metadata and Operational Data tables and views.

Metadata These tables represent the InForm metadata for the study. The metadata loading process uses these tables to define the tables in the InForm input data model.

All available InForm metadata tables and views are listed below. Those that are required in Oracle DMW and cannot be disabled are preceded by an asterisk (*).

IRV_CONTROL_REVS
IRV_FORM_REVS
*IRV_ITEM_REVS
IRV_ITEMSET_REVS
*IRV_MD_FORMS_CONTROLS
IRV_STUDYVERSIONS
IRV_SECTION_REVS
*IRV_STUDYVERSION_FORMS
*IRV_STUDYVERSION_REVISIONS
*IRV_STUDYVERSION_VISITS

IRV_VISIT_REVS
*RD_DATADICTIONARY

In addition, the Metadata model includes the following required Oracle DMW tables:

*DME_IA_SRC_TABLES
*DME_IA_SRC_COLUMNS
*DME_IA_STANDARD_COLUMNS
*DME_IA_TABLE_MAPPING
*DME_IA_COLUMN_MAPPING
*DME_IA_COLUMN_MAPPING_SOURCES

Operational Data These tables contain information collected during the course of the study that supports the clinical data, including InForm queries, form state flags, comments, and subject information. Tables that are required in Oracle DMW and cannot be disabled are preceded by an asterisk (*).

*IRV_COMMENT_REVS
IRV_CUR_COMMENT
IRV_CUR_QUERY
IRV_CUR_SITE
*IRV_CUR_SUBJECT
IRV_CUR_USER
IRV_DIM_SUBJECTVISIT
*IRV_QUERY_REVS
IRV_SV_SUBJECTVISITS
IRV_SUBJECT_REVS
IRV_USERS_SITES
*RT_ACTIVATED_FORMS
*RT_ITEMSTATE

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. Categories allow you to group flags, validation checks, and discrepancies to make them easier to find. A discrepancy, validation check, or flag can have only one category at a time. Users can filter these objects by category.

If a category is applied to a validation check, the validation check applies the category to all discrepancies it creates.

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 2.4 linking categories to model types and subtypes is possible only by using a public API; see the *Oracle Health Sciences Life Sciences Warehouse 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 2.4 you must use public API createModelFlagcatMap to map a category to a clinical data model type or subtype. See the *Oracle Health Sciences Life Sciences Warehouse 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 run 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 7-3 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 or Modifying a Tag

1. Click **Tags** at the bottom of the right-hand pane in the Administration page, then click the **Add** or **Modify** icon.
2. Enter values in the following fields:
   - **Tag Name**: This text is 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.

   Note: **System Tag** indicates whether a tag was created by the system (checked) or by a user (unchecked). You cannot change its value. You cannot modify system tags.
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 a data review tool. Flags can be applied to records (not discrepancies) by validation checks and by users in the Listings pages.

Users can filter records and discrepancies by flag values.

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 6-9 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.

Flag Assignment Auditing The system audits changes in flag assignments to records with the timestamp and user name, by category, in the internal table DME_FLAG_DATA_A. To have a single flag's values audited separately, you can create a separate category for it.

Creating a Flag

To create a flag:

1. Navigate to Administration, then Flags, then click the **Add** 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 and add states, but the flag must have at least one state or it cannot be applied to records.

  Flag State Priority: Select High, Medium, or Low relative to other states of the same flag and other flags. In the Listings pages, the system displays an icon representing the highest priority flag state currently applied to each flagged record.

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

Table 8–1 (Cont.) Flags, Tags, and Categories

Object	Applied To	Applied By	Used as Filter?	Has Multiple States?
Category	Discrepancies, Validation Checks, Flags	Validation check or user	Yes	No

Setting Up Coding in TMS

Oracle Thesaurus Management System (TMS) is a coding tool that you can integrate with Oracle DMW to code source data to terms in dictionaries (terminologies) such as MedDRA and WHO-Drug, so that data can be presented consistently and analyzed correctly.

When Oracle DMW is integrated with TMS, transformation jobs have an additional step: they send source data (*terms*) you specify to TMS and run the TMS autoclassification (automatic coding) job. Terms that cannot be automatically coded must be manually coded in TMS. TMS then codes all future occurrences of the same term the same way within the same domain. See "About TMS Processing" on page 8-16.

You specify which information to derive by creating one or more TMS Sets for each dictionary on the Administration page; see "Defining TMS Sets" on page 8-14. On the Home page you associate TMS domains and their terminologies to a study; see "Associating a Study with a TMS Domain Element" on page 2-2. On the Study Configuration page you associate TMS Set columns with target data model columns; see "Linking a TMS Set to a Table in a Target Clinical Data Model" on page 3-12.

If a TMS user cannot classify the term because, for example, it is really multiple terms, like *nausea and headache*, he or she can send a TMS *action* back to Oracle DMW as discrepancy text.

You must define TMS domains in TMS. These allow you to customize terminologies differently for different studies by creating domain-specific terms and classifications. Domains also determine default TMS system behavior, including whether explicit approval is required for manual classifications.

See the *Oracle Thesaurus Management System User's Guide* for information on setting up dictionaries and TMS domains, and on manually coding (*classifying*) terms. When you create TMS domains, remember that Oracle DMW displays the description, if there is one, for the user to select a TMS domain for use with a particular study. If there is no description, the system displays the TMS domain name.

Defining TMS Sets

Define a TMS Set to specify the information to derive back from a particular terminology (*dictionary)* to Oracle DMW for source system data that is coded in TMS. You can define any number of TMS Sets for a single terminology.

You can derive terms in other *dictionary levels*—for example, from WHO-Drug you could derive Synonym, Drug Constituents, and Anatomical-Therapeutic-Chemical Classification Group. From MedDRA you could derive System Organ Class, Preferred Term, and Special Category. You must define TMS Set columns for each derived item, and the study configurator must map each TMS Set column to actual table columns.

The study configurator does not need to map all the columns in the TMS Set. You can either:

- Define a single TMS Set for each dictionary in TMS with the maximum number of columns that might ever be required, and let the study configurator select and map the columns needed in each study.
- Define multiple TMS Sets for each dictionary, each with exactly the columns required in a particular study. The study configurator can then map all TMS Set columns to target table columns in that study.

In the Administration page, TMS Sets pane, click the **Add** icon. The **Add TMS Set** window appears.

1. Enter a **Name** and **Description** for the TMS Set. Both are displayed on the Study Configuration page.
2. Select a **Terminology** to code against. All terminologies (*dictionaries*) defined in the local TMS instance are listed.
3. Click **OK**. The system displays the new TMS Set in the TMS Set pane.
4. Select the new TMS Set and click the **Add** icon in the TMS Set Columns pane.

Defining TMS Set Columns

Define the information to be derived from TMS to Oracle DMW for each coded term using this TMS Set. These "columns" are mapped to table columns on the Study Configuration page.

1. Select a TMS Set and click the **Add** icon.
2. **Column Designation**: Enter a descriptive name to enable the study configurator to map this TMS Set column to a table column that will store the derived data.
3. **Column Type**: You can only add columns of type Derived.

   Each TMS Set also has one column of type Primary. The primary column must be mapped to the column containing items (*terms*) to be coded and always derives the dictionary term to which the source system term is coded in the level defined as the coding level of the dictionary in TMS.
4. **Dictionary Level Name**: Select the level from which to derive data into the mapped column. The choices include all of the dictionary's defined levels and Dictionary Informative Notes, which may be defined in TMS and apply to the dictionary as a whole.
5. **TMS Field**: The options vary depending on whether you specified a dictionary level or InFormative Notes.

   If you specified a **dictionary level**, the TMS Field choices include:

   - **TERM**: The dictionary term related to the coded term in the selected level.
   - **TERM_UPPER**: The dictionary term related to the coded term in the selected level, in uppercase.

   The choices also include optional, customizable, *level details* that your company may have defined for the dictionary level in TMS:

   - **CATEGORY**: A category value defined by your company in TMS.
   - **DICT_CONTENT_ID**: The unique ID generated by TMS for the dictionary level.
   - **DICT_CONTENT_ALT_CODE**: An alternate ID for the dictionary level; designed to cross-reference a legacy company dictionary.

- **DICT_CONTENT_CODE**: The unique ID of the dictionary level. This column is indexed.
- **VALUE_1 through VALUE_4**: Four customizable fields can store information about the dictionary level.

If you specified text **Dictionary Informative Notes** for Dictionary Level Name, you can select one of several types of Informative Note that your company may have defined for the dictionary as a whole:

- **DICTIONARY VERSION**: The dictionary version as defined in an Informative Note for the base dictionary in TMS.
- **RDC ACTION TEXT**: This data applies only if you are using Oracle Clinical Remote Data Capture Onsite.
- **DICTIONARY NOTE**: Standard text informative note defined in TMS.

6. Click **OK**.

About TMS Processing

TMS processing occurs during table-level transformation post-processing.

New and Updated Data Processed in TMS

TMS processing includes only target tables that have at least one column associated with a TMS Set primary column **and** meet **at least one** of the following conditions:

- New records in the source table.
- Updates to coded terms (data in the primary column) in the source table.
- Updates made in TMS that affect Oracle DMW data.
- Updates in DMW or the source system to TMS-originated discrepancies on data in the source tables.

TMS Autoclassification and Synchronization

Oracle DMW TMS processing first runs the TMS synchronization job, which checks for changes in the TMS repository that impact Oracle DMW terms in TMS, either coded or not yet coded.

TMS processing then loops through all records that meet the criteria in "New and Updated Data Processed in TMS" on page 8-16 and calls autoclassification for each one.

Autoclassification looks for exact matches to source terms in the *coding* dictionary level and, for terms that do not exactly match a dictionary term, the job checks if a TMS user has previously classified another occurrence of the same term to a dictionary term (this is called a *verbatim term assignment*, or VTA), first in the same domain and if not, globally, in TMS.

- For terms with an exact match to a dictionary term or to a VTA, TMS derives and sends information back to Oracle DMW during the same transformation job.
- For terms that do not have a dictionary term or VTA match, TMS creates a discrepancy in Oracle DMW and, if the lifecycle is Production, an omission in TMS.
  - An Oracle DMW user can create a corresponding query in InForm by applying an action to the discrepancy that sends it to InForm or by setting the status to Investigator (INV) Review.

  - A TMS user can classify the term manually, creating a new VTA. During the next transformation, autoclassification finds the new VTA and TMS sends derived information, and answers the discrepancy in Oracle DMW.

If a term that was previously classified is updated to a value that cannot be classified, previously derived information is deleted.

If a source term is deleted in InForm and then Oracle DMW, it calls a TMS API to delete the source term in TMS as well.

> **Note:** TMS autoclassification, synchronization, derivation, actions, and Oracle DMW discrepancy management are the same in all three lifecycle areas. However, TMS omissions are created only in Production, so TMS users can do manual classification only in Production, and Force Derivation works only in Production.

System Interaction and Data Statuses

The same data passes through three systems—InForm, Oracle DMW, and TMS—and conflicts may occur:

- Oracle DMW cannot prevent user changes in InForm. At any time an InForm user may update a TMS-originated discrepancy, changing its text or status, or the underlying data. Those changes are loaded as updates into the Oracle DMW discrepancy system, triggering TMS autoclassification and passing any new discrepancy text to TMS.
- State changes in InForm may conflict with the results of autoclassification in TMS. For example, an InForm user might close a discrepancy thinking that the term is a valid one while TMS does not recognize it. In this case, even though the Closed status is loaded into Oracle DMW, TMS opens a new discrepancy when autoclassification cannot find a match for the term.
- TMS classification might overwrite status changes made in InForm.

Oracle DMW uses four statuses to help coordinate the interaction of the three systems. The first two are set by the system:

- **TMS EVALUATION** is set by Oracle DMW either when a Oracle DMW user adds comment to a TMS-originated discrepancy (that is not marked as an internal comment) or when an InForm user answers a discrepancy (either with or without a data change) and the update is loaded into Oracle DMW. The purpose of TMS EVALUATION is to trigger autoclassification on the data again during the next transformation. Without it, autoclassification would run only if there was a data change or if a TMS user had altered the record.
- **TMS IN PROGRESS** is set by TMS when autoclassification creates or updates an omission. TMS users can manually classify an omission only when its state is TMS IN PROGRESS.

  Oracle DMW users cannot update a discrepancy at status TMS IN PROGRESS. If the discrepancy has previously been sent to InForm, InForm users can update it. However, InForm users will not have new data yet from TMS - until the TMS INV Review status is applied.

TMS users can apply the following statuses during omission management or as defined actions:

- **TMS DM Review** Prevents TMS users from updating. A Oracle DMW user may update with a comment, which sets the discrepancy and omission to TMS

EVALUATION so that TMS runs autoclassification on the term during the next transformation.

- **TMS INV Review** Sends the discrepancy to InForm for Investigator review and prevents TMS users from updating.

Force Rederivation

Click **Force Rederivation** in the TMS tab of the **Create Study** window to run the Force Rederivation job once, immediately. A confirmation message appears because running the job may take a long time.

Normally, transformations process only new or changed data. Use the Force Rederivation job to process all data when you have made structural changes related to TMS in an ongoing study such as:

- Adding columns to target tables to hold derived data.
- Updating a dictionary to a new version with a different structure from the old one.
- Changing domain-related settings in the TMS reference codelist TMS_CONFIGURATION.

Viewing Scheduled Jobs

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.
- Are assigned to a role within that user group that allows the operation on the object.

To view data, users must have the above privileges and also a blinding-related application role that allows them to see the relevant type of data: nonblinded, unblinded, or currently blinded.

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 blinding-related 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 , "Predefined Object Security Roles" for more information.
- **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 3-28 for more information.
- **Assign application roles to users** to allow access to the Oracle DMW user interface. See "Predefined Oracle DMW Application Roles" on page B-1 for more information.
- **Assign blinding-related roles to users** to allow viewing rights to nonblinded, unblinded, or currently blinded data. See "Predefined Blinding-Related Roles" on page B-2 for more information.

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 "Oracle 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 3-28. 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-5 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 8-19.

Setting Up Library and Study Categories

Oracle DMW uses Oracle LSH domain objects as containers to hold studies, study templates, 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 to categorize all studies by drug as well as therapeutic area you must create categories such as "Cardiology Drug 12345" and "Cardiology Drug 678910."
- 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 to classify your studies by therapeutic area, you do not need to set a new label value. To use a different label, you must set profile Oracle DMW_STUDY_CLASSIFICATION_UI_LABEL as described in "Change Study Category User Interface Label" on page 8-27. 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 Oracle 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* at http://docs.oracle.com/cd/E54418_01/doc.24/e54089/dev_ui.htm#CHDHIIHA.
- By running an Oracle LSH API; for instructions see the *Oracle Health Sciences Life Sciences Warehouse Application Programming Interface Guide* at http://docs.oracle.com/cd/E54418_01/doc.24/e53659/domains_api.htm#BABDIJFG.

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 "Creating a Custom Program" on page 5-21. These programs must be created in the Oracle DMW_UTILS domain in Oracle LSH which is created during Oracle LSH post-installation.

You can create application areas within the Oracle DMW_UTILS domain to help users find appropriate custom programs and functions and reuse them, promoting standardization.

The process is similar to creating domains to organize studies and libraries; see "Setting Up Library and Study Categories" on page 8-20.

Note: For performance reasons, Oracle recommends creating application areas instead of domains inside DMW_UTILS to organize programs and other objects.

Configuring Database 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 "Setting Lookup Values" on page 8-31.

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 `Oracle DMW_DOMAIN`. The system displays the Oracle DMW_UTILS domain and the Study and

Library category domains (see "Setting Up Library and Study Categories" on page 8-20).

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 lifecycle 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 `Oracle DMW_DOMAIN`. The system displays domains, including the Oracle DMW_UTILS domain. You may want to load your reference tables into the Oracle 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 Oracle 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 an external data visualization tool so that users can view data graphically and interactively with the protection of Oracle DMW security and blinding access privileges, one clinical data model at a time. If you select the Business Area option when you create a clinical data model, the system generates an Oracle LSH generic type Business Area object in the database with views of all tables in the model. Third-party tools that use generic type Business Areas to work with Oracle LSH will then work with Oracle DMW.

An adapter for the purpose of integrating Oracle Business Intelligence Enterprise Edition (OBIEE) is included with Oracle LSH. You must create an OBIEE-type Business Area in Oracle LSH. Instructions are included in the *Oracle Life Sciences Data Hub Application Developer's Guide*. The *Oracle Life Sciences Data Hub System Administrator's Guide* and the *Oracle Life Sciences Data Hub Installation Guide* have additional information on integrating OBIEE.

You can also create your own adapter to another tool; see the *Oracle Life Sciences Data Hub Adapter Toolkit Guide*.

Setting Profiles

Several profile settings are required in Oracle DMW. In addition, see the *Oracle Life Sciences Data Hub System Administrator's Guide* for information on using profiles to set password requirements.

How to Log In to Oracle Applications Profile Forms

You follow the same basic procedure for setting all profiles.

To log in, do the following:

1. Open your Oracle LSH URL.
2. Log on with the system administrator account. An E-Business Suite screen opens.
3. In the Main Menu pane, expand the **System Administrator** (not System Administration) node, then the **Profile** node, and then click **System**.

   Note: This user interface requires Java 6 Update 7 (1.6.0.0.7). If you do not have it installed, you can download it from `http://www.oracle.com/technetwork/java/javase/archive-139210.html`. To use this feature, you must accept all warnings.

   A new browser screen opens with several windows open and the Find System Profile Values window on top.

   Tip: If you lose the Find System Profiles window at any point, click the **Search** icon in the toolbar or click System Profiles in the Top Ten list.

Use Character Semantics

On each computer where you install Oracle DMW, you must set the Oracle Applications profile **LSH: Use Character Semantics for Workarea Installation** to **Yes**, as instructed in the *Oracle Life Sciences Data Hub Installation Guide*.

If you did not already do this when installing Oracle LSH, do it now:

1. Open your Oracle LSH URL.
2. Log on with the system administrator account. An E-Business Suite screen opens.
3. In the Main Menu pane, expand the **System Administrator** (not System Administration) node, then **Profile**, and then click **System**.

   A new browser screen opens with several windows open and the Find System Profile Values window on top.

   Note: If you lose the Find System Profiles window at any point, click the **Search** icon in the toolbar.

4. In the **Profile** field, enter `LSH: Use Character Semantics for Workarea Installation`
5. Click **Find**.
6. In the **Site** column, select `YES`.
7. In the File menu, select **Save and Proceed**. The system displays a message that the transaction is complete.
8. Click **OK**. The transaction message window disappears.
9. Click the **X** in the upper right corner of the System Profile Values window to close the window.

Increase the Maximum Number of Nested Domains

LSH: Domain Nesting Levels should be set to at least 6 because three levels of domains are predefined for Oracle DMW and you will need more to create categories such as Therapeutic Areas. The maximum allowed value is 9.

Follow instructions in "Use Character Semantics" on page 8-25 except in the Profile field, enter `LSH: Domain Nesting Levels` and in the Site column, enter a number between 6 and 9.

Append Username to Discrepancy Text

This profile determines the default setting for a check box in the Oracle DMW discrepancy creation user interface. Users can override the setting.

If you want the default behavior to be that the check box is not checked, so that by default the username is not added to the discrepancy text, you do not need to set this profile.

This profile determines the default setting for a check box in the Oracle DMW discrepancy creation user interface. Users can override the setting.

To set the default behavior for appending the username to discrepancy text, do the following:

1. Log in. See "How to Log In to Oracle Applications Profile Forms" on page 8-25.
2. In the **Profile** field, enter `Oracle DMW: Append Username to Discrepancy Text`
3. Click **Find**.
4. In the **Site** column, select Yes or No.

- If set to **No**, the default setting is not to append the username of the person creating the discrepancy to the discrepancy text. This is the shipped setting. Note that the system tracks the username in the database regardless of this setting.
- If set to **Yes**, the default setting is to append the username of the person creating the discrepancy to the discrepancy text. When discrepancies are sent to InForm, this information is displayed in InForm.

5. In the File menu, select **Save and Proceed**. The system displays a message that the transaction is complete.
6. Click **OK**. The transaction message window disappears.
7. Click the **X** in the upper right corner of the System Profile Values window to close the window.

Change Study Category User Interface Label

This profile determines the label that appears in the user interface in several places for the category of a study or library. By default, the label is **Therapeutic Area**.

If you prefer to organize studies and libraries in a different way, you can change this label. For example, if you want to organize by Drug Program, change this profile value to "Drug Program" and "Drug Program" will appear in the user interface instead of "Therapeutic Area."

See the chapter on administration in the *Oracle Health Sciences Data Management Workbench User's Guide* for further information.

To change the label, do the following:

1. Log in. See "How to Log In to Oracle Applications Profile Forms" on page 8-25.
2. In the **Profile** field, enter `Oracle DMW_STUDY_CLASSIFICATION_UI_LABEL`
3. Click **Find**.
4. In the **Site** column, enter the label exactly as you want it to appear in the user interface, including upper- and lowercase.
5. In the File menu, select **Save and Proceed**. The system displays a message that the transaction is complete.
6. Click **OK**. The transaction message window disappears.
7. Click the **X** in the upper right corner of the System Profile Values window to close the window.

Registering Root Folders for File Watcher Watched Folders

This set of profiles tells the system where to look for SAS and text files that should be loaded into Oracle DMW. There are nine profiles. You enter values for **either**:

- Three folders, one for each lifecycle. Both SAS and text files are placed in the same folder.
- Six folders, one for each lifecycle/file type combination. Each folder is used for only one file type: SAS or text.

You must create the root folders in the locations you specify here. All studies in this Oracle DMW instance must use the same three or six root folders for their input data files. The system creates a study-specific subfolder in each root folder using the name you specify; see "Creating and Modifying Study File Watchers" on page 8-3. The

study-specific subfolders become the *watched locations* for the study.

> **Note:** If you change these profile values after using them, you must stop and restart the File Watcher service to initialize the new values in the application; see "Changing the Root Folder Location" on page 8-2.

See the sections on File Watcher in the chapters on clinical data models and administration in the *Oracle Health Sciences Data Management Workbench User's Guide* for further information.

1. Log in. See "How to Log In to Oracle Applications Profile Forms" on page 8-25.
2. In the **Profile** field, enter `Oracle DMW:FWR_ROOT_FOLDER%`
3. Click **Find**.
4. In the **Site** column, enter the full path to the subfolder for the profile's file type/lifecycle combination. Oracle recommends using a naming convention so that it is easy to tell which folder should hold which files. Enter values for **either** the first six profiles **or** the last three.

   If you want to use six file type-specific root folders, enter values for these profiles:

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

   If you want to use three folders for both file types, enter values for these profiles:

   - `Oracle DMW:FWR_ROOT_FOLDER_ALL_DEV` for all files in the Development lifecycle.
   - `Oracle DMW:FWR_ROOT_FOLDER_ALL_QC` for all files in the Quality Control lifecycle.
   - `Oracle DMW:FWR_ROOT_FOLDER_ALL_PROD` for all files in the Production lifecycle.
5. In the File menu, select **Save and Proceed**. The system displays a message that the transaction is complete.
6. Click **OK**. The transaction message window disappears.
7. Click the **X** in the upper right corner of the System Profile Values window to close the window.

Registering Root Folders for File Watcher Archive Folders

To archive data files instead of deleting them after loading their data into Oracle DMW, you must create archive folders and register them using this set of profiles.

There are nine profiles, corresponding to the nine profiles that register root folders for watched folders. You must choose the same setup—three or six root folders—for archive folders that you did for watched folders. The system creates a study-specific folder in each root folder using the name you specify; see "Creating and Modifying

Study File Watchers" on page 8-3.

1. Log in. See "How to Log In to Oracle Applications Profile Forms" on page 8-25.
2. In the **Profile** field, enter `Oracle DMW:FWR_ARCHIVE%`
3. Click **Find**.
4. In the **Site** column, enter the full path to the subfolder for the profile's file type/lifecycle combination. Oracle recommends using a naming convention so that it is easy to tell which folder should hold which files. Enter values for **either** the first six profiles **or** the last three, corresponding to the six or three you chose in "Registering Root Folders for File Watcher Watched Folders" on page 8-27.

   If you want to use six file type-specific root folders, enter values for these profiles:

   - `DME:FWR_ARCHIVE_TEXT_DEV` for text files in the Development lifecycle.
   - `DME:FWR_ARCHIVE_TEXT_QC` for text files in the Quality Control lifecycle.
   - `DME:FWR_ARCHIVE_TEXT_PROD` for text files in the Production lifecycle.
   - `DME:FWR_ARCHIVE_SAS_DEV` for SAS files in the Development lifecycle.
   - `DME:FWR_ARCHIVE_SAS_QC` for SAS files in the Quality Control lifecycle.
   - `DME:FWR_ARCHIVE_SAS_PROD` for SAS files in the Production lifecycle.

   If you want to use three folders for both file types, enter values for these profiles:

   - `Oracle DMW:FWR_ARCHIVE_ALL_DEV` for all files in the Development lifecycle.
   - `Oracle DMW:FWR_ARCHIVE_ALL_QC` for all files in the Quality Control lifecycle.
   - `Oracle DMW:FWR_ARCHIVE_ALL_PROD` for all files in the Production lifecycle.

   In addition, you must set the following profile values:

   - `Oracle DMW:FWR_ARCHIVE_ENABLE` Set to `Y` to enable archiving. The default value is `N`.
   - `Oracle DMW:FWR_ARCHIVE_SCHEDULE_DAYS` Enter the number of days to elapse between loading the file and moving it to the archive folder. The default value is 7.

   Note: Be sure to set the file deletion time period for individual study File Watchers to a greater number of days.

5. In the File menu, select **Save and Proceed**. The system displays a message that the transaction is complete.
6. Click **OK**. The transaction message window disappears.
7. Click the **X** in the upper right corner of the System Profile Values window to close the window.

Setting Blinding Behavior for InForm Hidden Items

If this profile is set to `Y`, any tables that contain columns based on InForm hidden items are created with column-level blinding. The columns that are blinded will have a default masking value.

The default value is `Y`.

1. Log in. See "How to Log In to Oracle Applications Profile Forms" on page 8-25.

2. In the **Profile** field, enter `Oracle DMW:BLIND_INFORM_HIDDEN_ITEMS`
3. Click **Find**.
4. In the **Site** column, enter:
   - `Y` so that any tables that contain columns based on InForm hidden items are created with column-level blinding on the hidden items. Blinded columns will have a default masking value.
   - `N` to create all tables with their blinding flag set to N. Users can manually define blinding for tables in the data model in Oracle DMW.
5. In the File menu, select **Save and Proceed**. The system displays a message that the transaction is complete.
6. Click **OK**. The transaction message window disappears.
7. Click the **X** in the upper right corner of the System Profile Values window to close the window.

Make Discrepancy Categories Mandatory

Set this profile to `Yes` to require a manually created discrepancy to have an assigned category. See "Viewing and Creating Categories" on page 8-9.

1. Log in. See "How to Log In to Oracle Applications Profile Forms" on page 8-25.
2. In the **Profile** field, enter `DMW_DSC_CATEGORY_MANDATORY`.
3. Click **Find**.
4. In the **Site** column, enter:
   - `Yes` to require manually created discrepancies to have a category assigned.
   - `No` to make these categories optional.
5. In the File menu, select **Save and Proceed**. The system displays a message that the transaction is complete.
6. Click **OK**. The transaction message window disappears.
7. Click the **X** in the upper right corner of the System Profile Values window to close the window.

Make Discrepancy Actions Mandatory

Set this profile to `Yes` to require a manually created discrepancy to have an assigned action. The action sets the state and tag of the discrepancy. See "Actions" on page 7-3.

1. Log in. See "How to Log In to Oracle Applications Profile Forms" on page 8-25.
2. In the **Profile** field, enter `DMW_DSC_ACTION_MANDATORY`.
3. Click **Find**.
4. In the **Site** column, enter:
   - `Yes` to require manually created discrepancies to have an action assigned.
   - `No` to make these actions optional.
5. In the File menu, select **Save and Proceed**. The system displays a message that the transaction is complete.
6. Click **OK**. The transaction message window disappears.

7. Click the **X** in the upper right corner of the System Profile Values window to close the window.

Hosted Environments

This profile must be set to **Y** if the installation is in an environment hosted by Oracle, to support Single Sign-on functionality. The default value is **N**, which is the required setting for on-premise installations.

1. Log in. See "How to Log In to Oracle Applications Profile Forms" on page 8-25.
2. In the **Profile** field, enter `LSW_SSO_INTEGRATED`
3. Click **Find**.
4. In the **Site** column, enter:
   - `Y` if the installation is in an environment hosted by Oracle, to support Single Sign-on functionality.
   - `N` if the installation is maintained by your company.
5. In the File menu, select **Save and Proceed**. The system displays a message that the transaction is complete.
6. Click **OK**. The transaction message window disappears.
7. Click the **X** in the upper right corner of the System Profile Values window to close the window.

Set Login-Related Profile Values

Follow instructions in the *Oracle Life Sciences Data Hub System Administrator's Guide* control various login and password-related requirements.

Setting Lookup Values

To modify lookup values, do the following:

1. Open your Oracle LSH URL and log in as a user with the LSH Administrator application role.
2. From the Navigator drop-down, select **LSH Setup Admin**, then **Setups**, then **Lookups**. An Oracle Applications window opens with the Oracle Life Sciences Data Hub Lookups window displayed.
3. Press the **F11** key. The window enters Query mode and changes the background color of some of the fields.
4. In the **Type** field, enter the name of the lookup you want to see. You can enter part of the name and use the % wildcard. You can enter `cdr%` to retrieve all Oracle LSH lookups or dme% to retrieve all Oracle DMW lookups—then use the down arrow on the keyboard to view them in alphabetical order.

Note: During its initial development, Oracle LSH was known as CDR. Therefore many internal names contain the string `cdr`. Please think of CDR as a synonym for LSH.

Please think of DME as a synonym for Oracle DMW for the same reason.

---

5. Press **Ctrl+F11** to enter the query. The system displays the lookup and all its current values, including predefined values and those your company has previously added (if any). If your query retrieved more than one lookup, use the Down arrow on your keyboard to view each lookup in turn.

The following information is displayed for each value:

- **Code** is the internal name for the value.
- **Meaning** is the text that appears on the user interface, exactly as it appears here.
- **Description** is an explanation of the value.
- **Tag** is an optional descriptor.
- **Effective Dates**. The **From** column contains the date the value was created in your location. The **To** column may contain a date; if so, this is the date on which the value became or will become unavailable for use. If there is not a date in either column, the value is in effect.

Reasons for Change

To enable users to select from a list of predefined reasons for change to discrepancies, add values to the lookup `DME_DSC_ACTION_REASON`. The access level is **User**. For each value, add a code and a meaning. The *meaning* text appears in the user interface as the reason for change.

If you define meaning text that is the same as Reason for Change text in InForm, when an InForm user applies a Reason for Change that is then imported to Oracle DMW, Oracle DMW recognizes that the text is the same and stores the code as well as the meaning.

Number of Studies per Partition

(`DME_PARTITION_DEVQC` and `DME_PARTITION_PROD`) The access level is **User**. When users define a study in Oracle DMW they must specify whether they expect the amount of data collected in the study to be small, medium, or large relative to other studies. These lookups set the maximum number of small, medium, and large studies' data that can be stored in a single partition of certain cross-study internal tables in the Development, Quality Control and Production lifecycle areas. Development and QC data are stored in the same partition, while Production data is stored in a separate partition. The default values are: 1 large study, 5 medium studies, and 20 small studies for both lifecycle partitions.

You can change the default values in the Meaning column. Each value must be different from the others in the same lookup.

9

Data Processing

To run transformations and validation checks, go to the Home 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 by the system, use Reload processing.

Reload Processing

In Reload processing, the system processes all records in the source tables or table-level files and compares the primary keys of the source records with the primary keys of the records already in the target tables, 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 but its table is included, the system soft-deletes the record—the record still exists in the database but is **no longer available for use**. The audit trail for the record is available.

If an entire table is not included in a data load, full reload does not delete the data in the target table. To delete all data in a target table, load an empty file for the table.

Note: Unless you need to delete data in a particular table, be careful to use Full mode **only** if you are confident that the data being loaded or read is the complete set of current 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 tables 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 tables 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

Table 9–1 (Cont.) Deletion Behavior in Unit of Work Processing Modes

Unit of Work Mode	Delete within Reloaded Unit?	Delete All Records for Nonreloaded Unit?
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 5-29 for more information.

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.

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.

Data Processing in Validation Check Programs

The target tables of validation checks, which are created by the system, use Reload Processing. The user can choose Full or Incremental mode when scheduling or submitting the validation check batch.

See also "Automatically Triggering Transformations and Validation Checks by Upstream Processes" on page 9-5.

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. See "Configuring File Watcher" on page 3-21 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 table-level 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 table-level 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.
Incremental Reload	No	Does not delete any data.
Full Reload	No	Deletes all records that are not reloaded in tables that are 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. See "Configuring the Oracle Health Sciences 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")
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

Reviewing and Cleaning 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 a study 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 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. You can query on most columns to filter the data loads displayed; see "Querying By Example" on page 10-7. Columns include:

- **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 Oracle Health Sciences 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**: For InForm, the job ID of the most recent load for the clinical data model. For File Watcher, the jobId for the most recent data load from the specified file and clinical data model.
- **Last Load**: For InForm, the date and time of the most recent load for the clinical data model. For File Watcher, the date and time for the most recent data load from the specified file and clinical data model.
- **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 the User Interface Display" on page 10-6 and "Querying By Example" on page 10-7.

You set up data loading when you define input clinical data models. See "Configuring the Oracle Health Sciences InForm Connector" on page 3-16 and "Configuring File Watcher" on page 3-21 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 the 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 to see 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 on the Home page. If they are set up to support it, they can be triggered by a job that writes data to any of their source tables.

To run a transformation:

1. Select the transformation you want to run and click the **Submit Job** icon.

   The **Submit Job** icon does not appear if:

   - You have not selected a transformation.
   - The selected transformation has not been installed; check the Install Status column.

2. Enter values in the following fields:

   - **Submission Mode**: Select one:
     - **Full** mode normally takes longer and includes data deletion. Use Full mode only if you are confident that you are reloading all current data.
     - **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

Note: You may want to set up regular Incremental loads at frequent intervals and do 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.

  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 to make this job 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.

  If you are setting up regularly scheduled runs, this setting applies to each job.
- Click the **Refresh** icon at any time for an update.

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.

     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 the **Refresh** icon 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

Select 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 job.

Cancel Triggered Job Click to cancel a triggered job.

Viewing Data Files Not Processed

If the File Watcher service detects a file in a watched location that cannot be loaded because no current, running 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 be loadable because:

- They are misnamed. Note that all letters must be in the same case.
- There is a mistake in the File Specification regular expression; see "Creating File Specifications" on page 3-22.
- There is a matching File Specification, but its end date has passed.

- There is a matching File Specification, but it is suspended.
- There is a matching File Specification, but the data model is not installed.

The Files Not Processed screen displays the following information about each unloadable file:

- **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.
  - **MISSING**: The file was detected but deleted before the scheduled deletion or archive date.
  - **DELETED**: The file was deleted by File Watcher as scheduled.
  - **ARCHIVED**: The file was archived by File Watcher as scheduled.
- **File Type**: Text or SAS.
- **File Modified**: The modification date of the file on the file system.
- **Detection Date**: The date and time the file was detected, using the date and time in the Oracle DMW database.
- **Archive Date** displays the scheduled archive date before the file is archived and the actual archive date afterward.
- **Deletion Date** displays the scheduled deletion date before the file is deleted and the actual deletion date afterward.
- **Date Missing**: If the file is overwritten or removed from the file system before it is archived or deleted, then the Date Missing is store here.
- **File Error**: Information about the problem. For example, if the file matches a current File Specification but the clinical data model has not been installed, an error appears stating that the model needs to be installed.

Changing the 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

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 the up and down arrows 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**, and then select **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:

1. Select **View**, and then select **Columns**. The system displays all available columns, with a check next to those that are currently displayed. 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**, and then select **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.

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** 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, click the **Clear Filters** icon.

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 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 is much faster.

Changing Your Password

To change the password you use to log in to Oracle DMW:

1. Log in to Oracle Life Sciences Data Hub using the URL provided by your administrator and your Oracle DMW password.
2. Click the **Preferences** link in the upper right corner of the My Home screen.
3. Click the **Change Password** link.
4. Enter:
   - **Known As**: Enter your name as you would like it to appear in the welcome banner onscreen.
   - **Old Password**: Enter your current password.
   - **Password**: Enter your new password.
   - **Repeat Password**: Enter your new password again.
5. Click **Apply**.

Known As

Old Password Enter your current password.

Password Enter your new password.

Repeat Password Enter your new password again.

11

Reviewing Data

In the Listings pages you can view clinical data, create discrepancies identifying possibly flawed data, and view existing discrepancies imported from Oracle Health Sciences InForm (queries) or raised by validation (edit) checks or by other people. You can create your own custom listings and view custom listings that other people have made public.

To see data listings:

1. On the Home page, select the study and lifecycle mode and click **Listings**.
2. Under **Listings**, select the name of the clinical data model you want to view. The system lists all models in the current study to which you have security access.
3. In the right pane, select the type of listing you want to view:
   - **Default Listings** to see all study data to which you have access.
   - **Custom Listings** to create your own listings or use those that other people have marked Public.
   - **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 pane.

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. You 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.

Listings Pages

Different listings pages display different data.

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 system always displays the most current data, updating the last modification timestamp each time data is updated.

InForm queries are converted to discrepancies and displayed and tracked along with discrepancies created manually or by validation checks.

You can select multiple records using Ctrl+click or Shift+click at the beginning of each row.

Creating and Viewing Custom Listings

You can view custom listings saved as public and you can create ad hoc listings.

- **To view a saved listing**, select it in the Custom Listing pane. The current data appears on the right.
- **To create your own listing**, click the **Add** icon in the Custom Listing pane; see "Using the Query Builder to Create Custom Listings" on page 11-3.
- **To create a new listing from an existing one** click the **Modify Custom Listing** icon, modify it, and save it with a different name.
- **To copy saved custom listings**, click the **Copy Custom Listings** icon; see "Copying Custom Listings" on page 11-2. You can modify the copy.
- Select **Show disabled custom listings** to view only listings that are not available because a table or column they reference is marked Not Used in this study.
- **To view deleted InForm data** create a custom listing but do not install it. the system displays all records that meet the criteria, including deleted records. After you install it the deleted rows are not displayed.

> **Tip:** If your custom listing displays a row where the Discrepancy_ Exists column says Yes but no cells are highlighted to indicate where the discrepancy is, it is probably because an active discrepancy filter has identified a record with a discrepancy in a column that is not displayed in the listing.
>
> Discrepancy filters apply to discrepancies in all columns of the full records retrieved by the custom listing.

Copying Custom Listings

You can copy custom listings from another study or from a different clinical data model in the same study. The system checks if the required source tables are available in the current model.

1. Navigate to the Listings page, then Custom Listings in the lower left pane.
1. In the Custom Listings pane, click the **Copy Custom Listings** icon. The Copy Custom Listings window opens.
2. The system displays all study clinical data models to which you have access. You can type part or all of the name of the therapeutic area (or other category), study, or model to filter the list.

   Click **Clear Filters** to remove all typed text and revert to the full list.

3. Select a model. The system displays all the custom listings associated with the model. To filter, type all or part of a custom listing name and press Enter.
4. Select one or more custom listings. Use Ctrl+click or Shift+click to select multiple checks.
5. Click **OK**. The system searches the current model for the tables and columns that the custom listings read from, first by Oracle name and then by alias.
   - If the tables or columns do not exist, the Copy operation fails with an error message.
   - If they exist but are marked Not Used in the transformation that writes to the model, the system copies the listings as disabled and displays the reason they are disabled on the Custom Listings page.
   - If the tables and columns exist and are used, the system copies the custom listings and links them to the appropriate tables and columns.

Using the Query Builder to Create Custom Listings

To create a new custom listing:

1. Click the **Add** icon to create a new custom listing. The Add Custom Listing pane appears.
2. Enter a **name** and **description** for the custom listing. The name is displayed on the Listings page but the description is not. The Listings page can display about 25 characters of the name without requiring scrolling to see the whole name. Consider using a naming convention.
3. Select **Mark as Public** to enable users with the required security access to the data model and to the Listings pages to view the custom listing produced by this query. The query is not public until you save it.
4. 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 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.
5. In the Source pane, select the clinical data model whose data you want to view.
6. Specify the columns to display; see "Selecting Columns to Display in Custom Listings" on page 11-4. The system creates a SELECT clause based on your specifications.
7. If you need to use a function from a library in the SELECT or WHERE clause, and you plan to write the expression using the function in free text, open the **Select Packages** tab and check the packages you will use. This enables the system to generate the query code and supports the Test function.

   If you use the Expression Builder (reached by clicking the **Modify Expression** icon), you do not need to use the Select Packages tab.
8. If you need to use a self join in the SELECT clause, open the Define Table Alias tab and click the **Add** icon to add another row for the same table. Specify a different aliases for the table in each row. Use the aliases to create a self join in the Criteria pane.

> **Note:** The table alias must be three characters or less. If it is longer it causes problems.

9. Specify query criteria; see "Specifying Criteria" on page 11-5. The system creates a WHERE clause based on your specifications.
10. See "Viewing and Testing the Generated Code" on page 11-5.
11. Click **OK**.

Selecting Columns to Display in Custom Listings In the Select Columns tab, identify the columns to display for each record retrieved by the query. You can give columns a different header for display and you can combine or write expressions on one or more columns. 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.

> **Note:** Tables and columns marked Not Used in the transformation are not displayed here, nor are uninstalled tables.

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**, each in its own row.

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

   a. Move one column into Selected Columns using the **Add** icon and highlight it there.

   b. Select the additional columns in the Source pane and click the **Arrow** icon in the Source pane. The system adds them to the same row.
4. **Table Alias**: If you have defined table aliases for a table in a self join in the Define Table Alias tab, select a column and click the Select Table Alias icon to specify which table alias the selected column belongs to.
5. **Alias**: To display a heading for the column different from the column name, enter it in the Alias field.
6. **Sort Order**: Enter a number to determine the column's display order relative to other columns in this custom listing.
7. **Sort Type**: Select Ascending (**ASC**) or Descending (**DESC**) for the listing display.
8. **Expression**: Add the expression, if any, to operate on the column in the SELECT clause. Enter free text or click the **Modify Expression** icon to use the Expression Builder; see "Using the Expression Builder in Validation Checks and Custom Listings" on page 6-7. You can edit code generated by the Expression Builder in this field afterward.

Selecting Packages If you need to use a function from a library in the SELECT or WHERE clause, open the **Select Packages** tab and check the package you need. The

system displays all packages in the DMW_UTILS domain; see "Developing a Library of SQL Functions" on page 5-18. The system then selects that package by default when you select Function in the Expression Builder. You can select a different package in the Expression Builder if needed.

Adding Table Aliases for a Self-Join To create a self-join:

1. Add the table and column(s) to the Selected Columns tab.
2. Open the **Define Table Alias** tab and click the **Add** icon to add the table as many times as required.
3. Specify a different alias for the table in each row.

> **Note:** The table alias must be three characters or less. If it is longer it causes problems.
>
> If your alias includes a space, the system replaces the space with an underscore (_) when you save.

4. In the Selected Columns tab, select a column and click the **Select Table Alias** icon to specify which table alias the selected column belongs to.
5. Specify the join criteria in the Criteria pane. See "Using the Expression Builder in Validation Checks and Custom Listings" on page 6-7.

Specifying Criteria In the Criteria pane, build the WHERE clause.

Click the **Add or Modify Criteria** icon to open the Expression Builder. See"Using the Expression Builder in Validation Checks and Custom Listings" on page 6-7.

You can edit code generated by the Expression Builder or enter code directly in the Criteria pane.

> **Note:** Oracle recommends using the Expression Builder to add columns so that Oracle DMW "knows" what you are using in the expression and can use table and column aliases defined in the clinical data model to help determine if source tables and columns exist when you copy custom listings.

Viewing and Testing the Generated Code You can test the query or view the generated code at any time using these buttons:

- **Test**: The system generates PL/SQL code, validates it, and either displays an error message if the code is not valid or displays the records retrieved in the Test Results pane.
- **View Source**: The system generates and displays the PL/SQL code resulting from the query you define in the query builder.

Saving and Installing a Custom Listing Select a Custom Listing in the list on the left and:

- Click the **Save as Query** icon to save the listing for use in another session. For new queries, the Save operation includes installation.
- Click the **Install Custom Listing** icon to install the listing. This is required only for copied queries.

- Click the **Update Status** icon to promote the listing to QC (Quality Control) or Production for validation purposes.

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 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 as configured in the Activities page; see "Viewing and Running Validation Check Batches" on page 10-4.

> **Tip:** If your validation check listing displays a row where the Discrepancy_Exists column says Yes but no cells are highlighted to indicate where the discrepancy is, it is probably because an active discrepancy filter has identified a record with a discrepancy in a column that is not displayed in the listing.
>
> Discrepancy filters apply to discrepancies in all columns, including columns of auxiliary information, of the full records retrieved by the validation check—not just the discrepancies identified by the validation check.
>
> For example, to see only open validation check discrepancies, not validation check discrepancies that have been manually closed for a reason other than data change, you can apply a Discrepancy State filter in the VC Listings page. (Discrepancies that are closed because the underlying data no longer meets the criteria of the validation check are no longer displayed in the VC listing.) However, you will see all records with any open discrepancies, even if the discrepancy raised by the validation check on the record has been manually closed, and even if the column with the open discrepancy is not displayed in the listing.

Actions Available on All Listings Pages

You can do the following on any Listings page:

- "Creating and Using Filters" on page 11-7
- "Using the Find Feature" on page 11-20
- "Creating Discrepancies Manually" on page 11-20
- "Managing Discrepancies" on page 11-21
- "Showing Discrepancies" on page 11-21
- "Showing Flags" on page 11-22
- "Flagging Data" on page 11-23
- "Viewing Data Lineage" on page 11-23

- "Viewing Blinded Data" on page 11-25
- "Exporting to Excel and CSV" on page 11-25
- "Viewing Data in InForm" on page 11-25

Creating and Using Filters

Use filters to limit the data you see in the Listings and Discrepancies pages. You can apply more than one filter at a time and save filter groups. When a filter is applied, the message "Filters On" appears on screen. To see which filters are currently applied, click the **Current Filters** icon.

To remove all current filters, click the **Filter Off** icon.

To open the main Filters window, click the **Filter** icon. Then:

- Optionally, select **Show Filters Shared by Others** to see and use filters others have created and marked as Shared. Shared and private filters you created are always available to you, even if this box is unchecked. Filters marked as Public are always available.
- Select a filter to apply. The drop-down list for each type of filter displays all the filters to which you have access.
- **Create a new filter** by clicking 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.
- **View or edit a filter**: Select the filter from the drop-down list and click the **Modify Filter** icon to see the filter criteria or modify them.

  Note: You cannot modify filters created by someone else and marked as Shared. However, you can copy Shared or Public filters, modify the copy, and save it with a different name. You can modify public filters anyone created if you have special privileges.

- **Delete a saved filter** by selecting a filter and clicking the **Delete Filter** icon. You can delete private and shared filters you created, and you can delete public filters anyone created if you have special privileges. You cannot delete a filter if it is part of a filter group.
- **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, even if you navigate between the Listings and Discrepancies pages. If a particular filter cannot be applied in a new window, a message appears.

  If **Keep Active** is not selected, when you navigate to a different page, the filter is removed the current session, even if it is a named, saved, filter.

  Note: To save a filter for use after you log out, give it a name and click **OK**.

  Filter names must be unique across all private, shared, and public filters that you can see in a particular study and lifecycle.

- Select, create, edit, or delete a **Filter Group**; see "Creating and Using Filter Groups" on page 11-9.
- Click **OK**. Clicking **Reset** changes all filters to Not Set.

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 **Modify Filter** 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.

If you have applied a discrepancy filter in a Listings page, the system displays records that have discrepancies that meet the filter criteria with an additional **star** icon.

See "Defining Tables to Support Filtering in the Listings Pages" on page 3-7 for information on filtering requirements.

Creating and Using Filter Groups

Filter groups apply multiple filters at the same time.

Using a Filter Group To use a filter group, click the **Filter** icon, select the group from the Filter Group list, and click **OK**.

Creating and Modifying a Filter Group **1.** Click the **Filter** icon in the Listings or Discrepancies page, then:

- To create a new filter group, click the Filter Group **Add** icon.

- To create a new group based on an existing group or modify an existing group, select the group from the list and click the Filter Group **Edit** icon.

  To query for a filter group, click **Search** at the bottom of the list. Enter part or all of the name with the wild card `%` and/or enter the class (Public, Private, or Shared). Click **Search**.

2. Select filters for the group.
3. Select **Keep Active** for the filters that you want to remain active if you leave the page.
4. Enter a **Name** and **Description** for a new group. Leave as is to modify a group.
5. **Save As Copy** (Available only if you clicked the **Edit** icon.)
   - Select to save the group with a different name, leaving the original group unchanged.
   - Leave unselected to modify the existing group.
6. Click **OK**. You return to the main Filters window.
7. (Optional) Make the group public or shared.
   - Public filter groups can contain only public filters. Making a group public requires special privileges.
   - Shared filter groups that you create can contain shared filters owned by you and public filters.
   - Private filter groups can contain any filters, including those that are shared and owned by others.
8. Click **OK** to apply the filters and go back to the Listings or Discrepancies page.

Deleting a Filter Group To delete a filter group, select it and click the **Delete** icon.

If you have the privileges to modify a group, you can delete it.

- You can delete private or shared groups that you created.
- You can delete public groups if you have special privileges.

Viewing Filters Currently in Effect

To see which filters are currently applied, click the **Current Filters** icon. The Current Filters window opens, and for each currently applied filter, displays:

- Filter name
- Filter type
- Filter class (private, shared, or public)
- Filter description

To turn all filters off, click the **Filters Off** icon. To selectively turn off, turn on, or modify filters, click the **Filters** icon.

Creating and Editing Subject Filters

You can filter records based on characteristics of subjects, including their country, site, investigator, or ID. Subject filters require a Subject Visit Table table with columns mapped to SDTM identifiers. To create a Subject filter:

1. **Filter Driver Data Model**: Select the clinical data model that contains the Subject Visit table that you want the filtering logic to use. The system uses the columns in this Subject Visit table to determine which Filter Types are available.

   Tip: Whenever possible, use the same model as the Filter Driver Data Model in different filters for the same study. If two filters that use different driver models are applied at the same time and you navigate to the Discrepancies page, the system ignores both filters and gives a warning notifying you that the filters are being ignored.

2. **Filter Type**: Select one:

   - Countries
   - Sites
   - Site IDs
   - Investigator Names
   - Investigator IDs
   - Subject IDs
   - Unique Subject IDs (ID unique across studies)

   The system displays the type you select and all the types above it in the list as columns in the filter window, with one row for each distinct value for the selected filter type.

   Note: If the Subject Visit table in the Filter Driver data model does not have a column mapped to the SDTM identifier for Country, Site, Site ID, Investigator Name, Investigator ID, Subject ID, or Unique Subject ID, the column is not displayed.

   You can enter all or part of a value above any column to filter the rows displayed.

3. Select one or more rows to be the filter value.
4. Check **Exclude Based On Criteria** to filter **out** records that satisfy the criteria. This is the equivalent of adding "Not" to the query logic. Leave unchecked to display **only** the records that satisfy the criteria.
5. **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.
6. **Filter Name**: Enter a descriptive name for the filter to save it to use in another session; that is, after you log out or work in a different study.

   Filter names must be unique across all private, shared, and public filters that you can see in a particular study and lifecycle.
7. **Description**: Enter additional text to help you and other users know if this filter meets their needs. The description is displayed in the Edit Filter window and in the Filter Criteria window that lists all filters currently applied on a page.

8. **Save As Copy**. This option is available only when you are editing an existing, named filter. If this option is selected, you can make changes and save the filter with a different name.
9. Click **OK**.

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. This does not restrict you to using the filter only in that model.
2. **Filter Type**: Select one of the following:
   - Visit Cycle
   - Visit Number
   - Visit Name

   The system displays all three columns with a row for each visit. You can enter 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. Check **Exclude Based On Criteria** to filter **out** records that satisfy the criteria. This is the equivalent of adding "Not" to the query logic. Leave unchecked to display **only** the records that satisfy the criteria.
5. **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.
6. **Filter Name**: Enter a descriptive name for the filter to save it to use in another session; that is, after you log out or work in a different study.

   Filter names must be unique across all private, shared, and public filters that you can see in a particular study and lifecycle.
7. **Description**: Enter additional text to help you and other users know if this filter meets their needs. The description is displayed in the Edit Filter window and in the Filter Criteria window that lists all filters currently applied on a page.
8. **Save As Copy**. This option is available only when you are editing an existing, named filter. If this option is selected, you can make changes and save the filter with a different name.
9. Click **OK**.

Creating and Editing Subject Visit Flags Filters

On the Listings page, use this filter to find records with particular Subject/Visit-type flag states, or values, assigned.

On the Discrepancies page, use this filter to find discrepancies whose underlying data have particular Subject/Visit-type flag states, or values, assigned.

1. **Filter Driver**: Select the clinical data model that contains the Subject Visit table that you want the filtering logic to use. 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.

2. Select whether you want to retrieve data that meet **Any** or **All** conditions.

3. The system lists all flags and flag states, or values. You can enter part or all of a flag or flag state name above the column to filter. For each flag state:

   - Select **Include** to find records with the flag state applied.
   - Select **Exclude** to find records that do not have the flag state applied.
   - If you select **neither Include nor Exclude**, the flag state is not included in the filter criteria at all.

4. **Filter Availability** is **Study**. Saved filters of this type are available for use only in the current study and lifecycle.

5. **Filter Name**: Enter a descriptive name for the filter to save it to use in another session; that is, after you log out or work in a different study.

   Filter names must be unique across all private, shared, and public filters that you can see in a particular study and lifecycle.

   If you do not name the filter, the system calls it *Temporary Filter* and it is available only during your current session.

6. **Description**: Enter additional text to help you and other users know if this filter meets their needs. The description is displayed in the Edit Filter window and in the Filter Criteria window that lists all filters currently applied on a page.

7. **Save As Copy**. This option is available only when you are editing an existing, named filter. If this option is selected, you can make changes and save the filter with a different name.

8. Click **OK**.

Creating and Editing Visit Day Filters

To create a filter based on the visit day number in the study design:

1. In the **Condition** drop-down, select an operator: <, <=, <>, =, >, >=, Between, Is Not Null (has any value), and Is Null (has no value).

   The system displays 0, 1, or 2 fields depending on which operator you select. Enter visit day numbers as required.

2. **Filter Driver Data Model**: Select the clinical data model that contains the Subject Visit table that you want the filtering logic to use. 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. Check **Exclude Based On Criteria** to filter **out** records that satisfy the criteria. This is the equivalent of adding "Not" to the query logic. Leave unchecked to display **only** the records that satisfy the criteria.

4. **Filter Availability** is **Study Level**; filters of this type are available in all data models in the current study and lifecycle.

5. **Filter Name**: Enter a descriptive name for the filter to save it to use in another session; that is, after you log out or work in a different study.

   Filter names must be unique across all private, shared, and public filters that you can see in a particular study and lifecycle.

6. **Description**: Enter additional text to help you and other users know if this filter meets their needs. The description is displayed in the Edit Filter window and in the Filter Criteria window that lists all filters currently applied on a page.

7. **Save As Copy**. This option is available only when you are editing an existing, named filter. If this option is selected, you can make changes and save the filter with a different name.

8. Click **OK**.

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 : <, <=, <>, =, >, >=, Between, Is Not Null (has any value), and Is Null (has no value).

   The system displays 0, 1, or 2 date fields depending on which operator you select. Enter values as required.

2. **Filter Driver Data Model**: Select the clinical data model that contains the Subject Visit table that you want the filtering logic to use. 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. Check **Exclude Based On Criteria** to filter **out** records that satisfy the criteria. This is the equivalent of adding "Not" to the query logic. Leave unchecked to display **only** the records that satisfy the criteria.

4. **Filter Availability** is **Study Level**; filters of this type are available in all data models in the current study and lifecycle.

5. **Filter Name**: Enter a descriptive name for the filter to save it to use in another session; that is, after you log out or work in a different study.

   Filter names must be unique across all private, shared, and public filters that you can see in a particular study and lifecycle.

6. **Description**: Enter additional text to help you and other users know if this filter meets their needs. The description is displayed in the Edit Filter window and in the Filter Criteria window that lists all filters currently applied on a page.

7. **Save As Copy**. This option is available only when you are editing an existing, named filter. If this option is selected, you can make changes and save the filter with a different name.

8. Click **OK**.

Creating and Editing Record Flags Filters

On the Listings page, use this filter to find records with particular Record-type flag states, or values, assigned.

On the Discrepancies page, use this filter to find discrepancies whose underlying data have particular Record-type flag states, or values, assigned.

1. Select whether you want to retrieve data that meet **Any** or **All** conditions.

2. The system lists all flags and flag states, or values. You can enter part or all of a flag or flag state name above the column to filter. For each flag state:
   - Select **Include** to find records with the flag state applied.
   - Select **Exclude** to find records that do not have the flag state applied.
   - If you select **neither Include nor Exclude**, the flag state is not included in the filter criteria at all.
3. **Filter Availability** is **All Studies**. Saved filters of this type are available for use in all studies in the current lifecycle.
4. **Filter Name**: Enter a descriptive name for the filter to save it to use in another session; that is, after you log out or work in a different study.

   Filter names must be unique across all private, shared, and public filters that you can see in a particular study and lifecycle.

   If you do not name the filter, the system calls it *Temporary Filter* and it is available only during your current session.
5. **Description**: Enter additional text to help you and other users know if this filter meets their needs. The description is displayed in the Edit Filter window and in the Filter Criteria window that lists all filters currently applied on a page.
6. **Save As Copy**. This option is available only when you are editing an existing, named filter. If this option is selected, you can make changes and save the filter with a different name.
7. Click **OK**

Creating and Editing Data Change Date Filters

The Data Change Date filter displays records whose most recent update occurred during the time period specified.

1. In the **Condition** drop-down, select an operator to be used to select the dates. The options are: <, <=, <>, =, >, >=, Between, Is Not Null (has any value), and Is Null (has no value).

   Note: Selecting 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.

   The system displays 0, 1, or 2 date fields depending on which operator you select. Enter values as required.
2. Check **Exclude Based On Criteria** to filter **out** records that satisfy the criteria. This is the equivalent of adding "Not" to the query logic. Leave unchecked to display **only** the records that satisfy the criteria.
3. **Filter Availability** is **Study Level**; filters of this type are available in all data models in the current study and lifecycle.
4. **Filter Name**: Enter a descriptive name for the filter to save it to use in another session; that is, after you log out or work in a different study.

   Filter names must be unique across all private, shared, and public filters that you can see in a particular study and lifecycle.

5. **Description**: Enter additional text to help you and other users know if this filter meets their needs. The description is displayed in the Edit Filter window and in the Filter Criteria window that lists all filters currently applied on a page.
6. **Save As Copy**. This option is available only when you are editing an existing, named filter. If this option is selected, you can make changes and save the filter with a different name.
7. Click **OK**.

Creating and Editing Discrepancy Category Filters

The Discrepancy Category filter displays only records that have one or more discrepancies of the categories specified.

1. In the **Discrepancy Categories** drop-down, select one or more categories to be used to filter, or click **All** and then deselect categories.

   Your company defines the categories; see "Viewing and Creating Categories" on page 8-9.
2. Check **Include *(No Category)*** include or exclude discrepancies with no assigned category.
3. Check **Exclude Based On Criteria** to filter **out** records that satisfy the criteria. This is the equivalent of adding "Not" to the query logic. Leave unchecked to display **only** the records that satisfy the criteria.
4. **Filter Availability** is **All Studies**. Saved filters of this type are available for use in all studies in the current lifecycle.
5. **Filter Name**: Enter a descriptive name for the filter to save it to use in another session; that is, after you log out or work in a different study.

   Filter names must be unique across all private, shared, and public filters that you can see in a particular study and lifecycle.
6. **Description**: Enter additional text to help you and other users know if this filter meets their needs. The description is displayed in the Edit Filter window and in the Filter Criteria window that lists all filters currently applied on a page.
7. **Save As Copy**. This option is available only when you are editing an existing, named filter. If this option is selected, you can make changes and save the filter with a different name.
8. Click **OK**.

Creating and Editing Discrepancy Change Date Filters

The Discrepancy Change Date filter displays records that have one or more discrepancies whose most recent update occurred during the time period specified. To create a Discrepancy Change Date filter:

1. In the **Condition** drop-down, select an operator: <, <=, <>, =, >, >=, Between, Is Not Null (has any value), and Is Null (has no value).

   Note: Selecting 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.

The system displays 0, 1, or 2 date fields depending on which operator you select. Enter values as required.

2. Check **Exclude Based On Criteria** to filter **out** records that satisfy the criteria. This is the equivalent of adding "Not" to the query logic. Leave unchecked to display **only** the records that satisfy the criteria.
3. **Filter Availability** is **All Studies**. Saved filters of this type are available for use in all studies in the current lifecycle.
4. **Filter Name**: Enter a descriptive name for the filter to save it to use in another session; that is, after you log out or work in a different study.

   Filter names must be unique across all private, shared, and public filters that you can see in a particular study and lifecycle.
5. **Description**: Enter additional text to help you and other users know if this filter meets their needs. The description is displayed in the Edit Filter window and in the Filter Criteria window that lists all filters currently applied on a page.
6. **Save As Copy**. This option is available only when you are editing an existing, named filter. If this option is selected, you can make changes and save the filter with a different name.
7. Click **OK**.

Creating and Editing Discrepancy State Dates Filters

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.

1. In the **Condition** drop-down, select an operator : <, <=, <>, =, >, >=, Between, Is Not Null (has any value), and Is Null (has no value).

   Note: Selecting 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.

   The system displays 0, 1, or 2 date fields depending on which operator you select. Enter values as required.
2. Check **Exclude Based On Criteria** to filter **out** records that satisfy the criteria. This is the equivalent of adding "Not" to the query logic. Leave unchecked to display **only** the records that satisfy the criteria.
3. **Filter Availability** is **All Studies**. Saved filters of this type are available for use in all studies in the current lifecycle.
4. **Filter Name**: Enter a descriptive name for the filter to save it to use in another session; that is, after you log out or work in a different study.

   Filter names must be unique across all private, shared, and public filters that you can see in a particular study and lifecycle.
5. **Description**: Enter additional text to help you and other users know if this filter meets their needs. The description is displayed in the Edit Filter window and in the Filter Criteria window that lists all filters currently applied on a page.

6. **Save As Copy**. This option is available only when you are editing an existing, named filter. If this option is selected, you can make changes and save the filter with a different name.
7. Click **OK**

Creating and Editing Discrepancy States and Tags Filters

Use this filter to find discrepancies in a particular state, with or without a substate (tag).

- In the Listings page, the filter finds records with discrepancies that meet the criteria.
- In the Discrepancies page, the filter finds discrepancies that meet the criteria.

1. **Discrepancy State and Tag**. The list includes all discrepancy states followed by all possible substates, or tags, for each state. Each state—for example, Open or Candidate—is also displayed without any tags. Selecting that option finds discrepancies with that state and no tags. To filter for all discrepancies in a state, regardless of tag, select all state/tag combinations for the state.
2. **Reopened**. Select **Yes** to view reopened discrepancies only; **No** to view discrepancies that are not currently reopened; or **All** to see discrepancies without regard to whether they are reopened or not.
3. **Data Changed**. Select **Yes** to view discrepancies whose underlying data has been updated since the discrepancy was created; **No** to view discrepancies whose data has not changed since the discrepancy were created; or **All** to see discrepancies without regard to whether their data has changed.
4. Check **Exclude Based On Criteria** to filter **out** records that satisfy the criteria. This is the equivalent of adding "Not" to the query logic. Leave unchecked to display **only** the records that satisfy the criteria.
5. **Filter Availability** is **All Studies**. Saved filters of this type are available for use in all studies in the current lifecycle.
6. **Filter Name**: Enter a descriptive name for the filter to save it to use in another session; that is, after you log out or work in a different study.

   Filter names must be unique across all private, shared, and public filters that you can see in a particular study and lifecycle.

   If you do not name the filter, the system calls it *Temporary Filter* and it is available only during your current session.
7. **Description**: Enter additional text to help you and other users know if this filter meets their needs. The description is displayed in the Edit Filter window and in the Filter Criteria window that lists all filters currently applied on a page.
8. **Save As Copy**. This option is available only when you are editing an existing, named filter. If this option is selected, you can make changes and save the filter with a different name.
9. Click **OK**.

Creating and Editing Discrepancy User Filters

Use this filter to retrieve discrepancies by the person who created or most recently modified a discrepancy.

1. **Filter Type**. Select the way you want to search:

- Created By
- Modified By

2. The system displays all Oracle DMW users, with their user name, full name as entered in the system, and any external systems such as InForm or TMS that they also use. You can enter part or all of any of these values above the column to filter.
3. Check **Exclude Based On Criteria** to filter **out** records that satisfy the criteria. This is the equivalent of adding "Not" to the query logic. Leave unchecked to display **only** the records that satisfy the criteria.
4. **Filter Availability** is **All Studies**. Saved filters of this type are available for use in all studies in the current lifecycle.
5. **Filter Name**: Enter a descriptive name for the filter to save it to use in another session; that is, after you log out or work in a different study.

   Filter names must be unique across all private, shared, and public filters that you can see in a particular study and lifecycle.

   If you do not name the filter, the system calls it *Temporary Filter* and it is available only during your current session.
6. **Description**: Enter additional text to help you and other users know if this filter meets their needs. The description is displayed in the Edit Filter window and in the Filter Criteria window that lists all filters currently applied on a page.
7. **Save As Copy**. This option is available only when you are editing an existing, named filter. If this option is selected, you can make changes and save the filter with a different name.
8. Click **OK**.

Creating and Editing Discrepancy Origin Filters

Use this filter to retrieve discrepancies by the way they were created.

1. **Discrepancy Origins**. Select one or more to search for:
   - **All** returns all discrepancies, regardless of origin.
   - **InForm** returns discrepancies created as queries in InForm.
   - **Manual** returns discrepancies created manually in Oracle DMW.
   - **TMS** returns discrepancies created as omissions in Oracle Thesaurus Management System (TMS). This option applies only if you have TMS integrated with Oracle DMW.
   - **Validation Check** returns discrepancies created by validation checks in Oracle DMW.
2. Check **Exclude Based On Criteria** to filter **out** records that satisfy the criteria. This is the equivalent of adding "Not" to the query logic. Leave unchecked to display **only** the records that satisfy the criteria.
3. **Filter Availability** is **All Studies**. Saved filters of this type are available for use in all studies in the current lifecycle.
4. **Filter Name**: Enter a descriptive name for the filter to save it to use in another session; that is, after you log out or work in a different study.

   Filter names must be unique across all private, shared, and public filters that you can see in a particular study and lifecycle.

If you do not name the filter, the system calls it *Temporary Filter* and it is available only during your current session.

5. **Description**: Enter additional text to help you and other users know if this filter meets their needs. The description is displayed in the Edit Filter window and in the Filter Criteria window that lists all filters currently applied on a page.
6. **Save As Copy**. This option is available only when you are editing an existing, named filter. If this option is selected, you can make changes and save the filter with a different name.
7. Click **OK**.

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 each searchable column 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 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): To add criteria for a column already used, you can add another field for the column and specify another operator and value—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. You can select multiple data points with the same issue and create a discrepancy against all of them at the same time.

After a discrepancy is created, it is visible everywhere its underlying data point is visible, including in other data models and listings.

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.
2. Click the **Create Discrepancy** icon.
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 "Oracle 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.

     Whether or not a category is required depends on a profile setting. See "Make Discrepancy Categories Mandatory" on page 8-30.
   - **State**: Select **Open** or **Candidate**, according to your company's policy. Candidate discrepancies require manual review before being moved to 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 to make it 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 profile setting. See "Append Username to Discrepancy Text" on page 8-26.
   - **Actions**: Select an action from the list—for example, Send to InForm. The system applies the action to the discrepancy when you click **OK**.

     Whether or not an action is required depends on a profile setting. See "Make Discrepancy Actions Mandatory" on page 8-30.
4. Click **OK**.

Managing Discrepancies

To act on a discrepancy, select it and click the **Manage Discrepancies** icon. The Manage Discrepancies window opens, and you can work on that one discrepancy just as in the Discrepancies page. When you are finished, close the window to work in the Listings page again with the same data displayed.

You can also select a discrepancy and click the **Go to Discrepancies** icon to go to the Discrepancies page, but when you finish working there and return to the Listings page, you have to reset the context.

Showing Discrepancies

Records with one or more discrepancies have a yellow rectangle on the left and the word **Yes** in the DISCREPANCY_EXISTS column. In the cell for each data point in the row with a discrepancy, the system displays the icon for the discrepancy's state. If a

data point has multiple discrepancies, the system displays the icon for the state that requires the most work to resolve. The order is:

- Open
- Candidate
- Answered
- Cancelled
- Closed

In addition, if you have applied a discrepancy filter, the system displays data points that have the discrepancies that meet the filter criteria with an additional **Star** icon.

To see all discrepancies associated with a particular data point, select the data point and click the **Show Discrepancies** icon. For each discrepancy the system displays:

- Icon showing the current state of the discrepancy; see "Discrepancy Display" on page 12-2 for an explanation of each icon.
- **Discrepancy ID**.
- **Discrepancy**: Text that describes the data problem 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 or click the **Manage Discrepancies** icon.

Showing Flags

To show all flags currently applied to a record, select it and click the **Show Flags** icon. The system displays each flag name, its applied value (state) and its priority, as well as the timestamp of the data when the flag was applied. You can click a column header to sort the flags by name, value, or priority.

Records that have at least one flag assigned have an icon displayed to the left of the row that represents the highest priority flag state currently applied to the record:

-  High priority flag state.
-  Medium priority flag state.
-  Low priority flag state.

Flags provide information about the state of a record. Record flags track the data review process. Subject Visit flags track data completeness. You can filter on any flag. InForm CRF form and section states are imported as flags, and your administrator can create flags for use within Oracle DMW (see "Creating and Using Flags" on page 8-11). A flag has one or more states, or values, each with a priority relative to other values of the same flag and other flags. A record can have multiple flags applied at the same time, each with a single value.

InForm CRF 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**

Flagging Data

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 the **Assign Flags** icon.
2. Select the flag to assign.
3. Select the flag state value to assign.

   The system compares the record's displayed timestamp with the last data update and if there is more recent data—a data load has completed while you were viewing the page—a message appears telling you to refresh the page and the flag is not assigned. Click the **Refresh** icon.

   If you the most recent data is already displayed, the system applies the flag state.

> **Note:** The **Assign Flags** icon 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 the **Assign Flags** icon.
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 5-29 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.

   The Trace Data Lineage window appears:

Trace Data Lineage Display

The 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.

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*.
- **Is Masked**: If a dollar sign ($) is displayed, the displayed data point may be a masked value instead of real data. Its table or column contains masked data, but this particular data point may not be masked.
- **Is Staging**: If a **#** is displayed, the data point displayed is in a staging table used in a transformation, not in a clinical data model table.
- **Value**: The data value.
- **Link to InForm**: If the data originated in InForm, click the **View Data in InForm** icon to open InForm and see the data in context there.
- **Incomplete Lineage**: If an asterisk (*) is displayed after the column name in the Name column, 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**: 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 the data point is not on the preferred one.
- **Primary Key**: 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, you can choose to see the real data when you first open the Listings page. You can use the padlock icon to toggle between viewing blinded data and viewing the masking values (or, in the case where a whole table or row is blinded, not seeing the table or row at all). The system records each blind break in a non-modifiable audit trail.

Users who do not have the privileges required to see blinded data do not see either the message upon opening the page or the padlock icon.

Exporting to Excel and CSV

There are four icons for exporting data:

- **Export All to Excel** generates an .xls file that includes all data that satisfies the current filters (if any).
- **Export Current Page to Excel** generates an .xls file that includes the currently visible data.
- **Export All to CSV** generates a comma-delimited text file that includes all data that satisfies the current filters (if any).
- **Export Current Page to CSV** generates a comma-delimited text file that includes the currently visible data.

Viewing Data in InForm

Select a data point and click the **View Data in InForm** icon. The InForm Login page opens. After you log in, you see the data in context, with access to comments and the audit trail.

If the selected data point has more than one source data point, the system displays the one designated as the preferred data source.

Note: The InForm URL must be defined in the InForm configuration for the current data model and lifecycle.

12

Cleaning Data

A discrepancy is associated with a data point identified as incorrect or possibly incorrect. A discrepancy may originate as an Oracle Health Sciences 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 discrepancies manually, 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 pane.

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:

- **Actions**: Select the action you want to apply from the drop-down list. You must then select and/or enter a reason for change. If the discrepancy is sent to InForm, the reason is visible there.

  Where Is the Action I Want to Apply?

- **Add Comment**: Enter information that may be helpful in the review process and click **OK**.

  Internal Comment? is used only for discrepancies that originated in Oracle Thesaurus Management System (TMS). If it is **not** selected and you add a comment to a discrepancy that was raised in TMS, the system changes the discrepancy's status to **TMS EVALUATION** and TMS autoclassification runs on the source term during the next transformation job.

- **Export All to Excel**: When you click the **Export All to Excel** the system generates an .xls file of all discrepancies that meet the current filter conditions (if any) and puts it in the location you specify.

Modifying a Single Discrepancy

1. Select one discrepancy and click the **Edit** icon 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 filter discrepancies by category.
   - **Action**: Select an appropriate action to apply to the discrepancy. You must then select and/or enter a reason for change. If the discrepancy is sent to InForm, the reason is visible there.

     Where Is the Action I Want to Apply?
   - **Allow 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, select and/or enter a reason for the change.
2. Click **OK**.

Where Is the Action I Want to Apply?

If you selected multiple discrepancies, 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 and Editing Actions" on page 7-6.

If you selected a single discrepancy and the discrepant data point is derived from multiple data values from different sources, the system uses the data source identified as Preferred in the transformation to determine which routing action to enable. If no action is available to send the discrepancy to the source you need, you may need to enter the discrepancy against the data point in the source data model.

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.

Click the **Refresh** icon to show any changes that have occurred since you opened the window. Other users may be working on the discrepancies you are viewing.

The icon displayed indicates the current state of the discrepancy. The state is also displayed as text in the State column and in the Detail pane.

Table 12–1 Discrepancy Icons


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 Spreadsheet or **Sent to InForm**—Normally for the purpose of sending to the source lab. No further action is allowed.

Managing TMS Discrepancies

If your study is set up to use Oracle Thesaurus Management System (TMS) for coding source data to standard terminologies, or dictionaries, TMS processes designated items that have new or changed data during transformations. If TMS is able to automatically code a designated source term to a dictionary term, it does so and derives specified data back to Oracle DMW.

TMS creates a discrepancy when it cannot automatically code a designated source term to a dictionary term. A TMS user must examine the data and either:

- Classify it manually—possible only in the production lifecycle. TMS then closes the discrepancy and derives data to Oracle DMW.
- Send text to Oracle DMW as discrepancy text describing a problem with the data or an action required. For example, if the item is "headache and nausea" the message could be "Split the term."

You can look at a discrepancy's history to see all past discrepancy texts. If TMS autoclassification was unable to code the term, the discrepancy has standard text, "TMS unable to classify term."

Oracle DMW uses four TMS-related system categories, displayed in the Categories column, to help process these discrepancies:

- **TMS in Progress** prevents updates in Oracle DMW while the item (*term*) awaits manual classification in TMS.
- **TMS Evaluation** means that the item or discrepancy has been updated either in the source system or in Oracle DMW, and triggers processing the term in TMS during the next transformation.
- **TMS DM Review** means that the data manager should review the discrepancy in Oracle DMW.
- **TMS Inv Review** sends the discrepancy to InForm as a query. The system also sends any subsequent updates to the discrepancy to InForm. The investigator should review the query.

  You cannot change the status of TMS discrepancies, but the status can be changed in InForm as well as TMS. This may cause conflicts. For example, an InForm user

might close a discrepancy thinking the term is a valid one while TMS does not recognize it. In this case, even though the Closed status is loaded into Oracle DMW, TMS opens a new discrepancy when autoclassification fails.

See "Setting Up Coding in TMS" on page 8-14 for more information.

Viewing Data and Queries in InForm

In the Details pane, click the **View Data in InForm** icon or the **View Query in InForm** icon. The InForm Login screen opens. After you log in, the system displays the selected query or its underlying data in context.

Note: The InForm URL must be defined in the InForm configuration for the current data model and lifecycle.

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 the **Edit** icon.
- **Table**: The database table that contains the discrepant value.
- **Model**: The clinical data model that contains the data value against which the discrepancy was originally created.

Note: If the model is *InForm*, 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 in the Discrepancy Management pane by selecting a value from the **Actions** drop-down list.
- **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. You can change the category in the Discrepancy Management pane by clicking the **Edit** icon.
- **Created**: The username of the person who created the discrepancy and the date it was created. If the user created the discrepancy as a query in InForm, the InForm username is displayed and "InForm" is displayed.
- **Modified**: The username of the person who most recently modified the discrepancy and the date it was modified. If the user modifed the discrepancy as a query in InForm, the InForm username followed by "(InForm)" is displayed.
- **Tag**: The tag assigned to the discrepancy, if any. Some actions apply tags.

- **New Data**: If **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 or someone else clicks **Display Full Record**, the system considers that you have reviewed it and sets this value to **No**. You can also set it to **No**.
- **View Query in InForm**: When you click this icon, the InForm Login screen opens. After you log in, the system displays the selected query in context.
- **View Data in InForm**: When you click this icon, the InForm Login screen opens. After you log in, the system displays the underlying data in context.

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 the User Interface Display" on page 10-6.

Select **Refresh** to view any changes that have occurred since you selected the discrepancy or last refreshed.

Displaying the Full Record

The Display Full Record pane displays all column values for the 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$SKEY contains the primary key values of the record separated by tildes (~). This value is used internally to trace data lineage; see "How the System Tracks Data Lineage" on page 5-29.
- CDR$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.

Creating and Using Discrepancy Filters

Use filters to show only discrepancies that meet certain criteria. You can name and save filters to use later, use filters other people have created and made public or shared, copy existing filters, group filters, or just use a filter during your current session; see "Creating and Using Filters" on page 11-7.

The system continues to apply filters set in the Listings pages in the Discrepancies page, except:

- Data Changed filters are ignored.
- A Subject or Visit filter set to apply only to a specific data model is interpreted as an additional filter restricting discrepancies to just those in that data model.
- If different Subject and Visit filters specify different data models as the driving data model and they are applied at the same time, the system ignores all Subject and Visit filters.

A message appears when any of these situations occur.

Discrepancy filters include:

- "Creating and Editing Location (Tables and Models) Filters" on page 12-6
- "Creating and Editing Data Source Filters" on page 12-6
- "Creating and Editing Discrepancy States and Tags Filters" on page 11-18
- "Creating and Editing Discrepancy User Filters" on page 11-18
- "Creating and Editing Subject Visit Flags Filters" on page 11-12
- "Creating and Editing Record Flags Filters" on page 11-14
- "Creating and Editing Discrepancy Origin Filters" on page 11-19

Creating and Editing Data Source Filters

Use this filter to retrieve discrepancies by their original data source or data source type—for example, a lab or InForm.

1. **Filter Type**. Select the way you want to search:
   - **Datasource** The system lists InForm and all labs used in the current study.
   - **Datasource Type** The system lists Labs (for all labs used in the current study), InForm, and Unknown.
2. Select the data source or type to search. You can enter part or all of a model or table name above the column to filter.
3. Check **Exclude Based On Criteria** to filter **out** records that satisfy the criteria. This is the equivalent of adding "Not" to the query logic. Leave unchecked to display **only** the records that satisfy the criteria.
4. **Filter Availability** is **All Studies**. Saved filters of this type are available for use in all studies in the current lifecycle.
5. **Filter Name**: Enter a descriptive name for the filter to save it to use in another session; that is, after you log out or work in a different study.

   Filter names must be unique across all private, shared, and public filters that you can see in a particular study and lifecycle.

   If you do not name the filter, the system calls it *Temporary Filter* and it is available only during your current session.
6. **Description**: Enter additional text to help you and other users know if this filter meets their needs. The description is displayed in the Edit Filter window and in the Current Filters window that lists all filters currently applied on a page.
7. **Save As Copy**. This option is available only when you are editing an existing, named filter. If this option is selected, you can make changes and save the filter with a different name.
8. Click **OK**

Creating and Editing Location (Tables and Models) Filters

Use this filter to retrieve discrepancies by the table or data model against which they were originally raised.

1. **Filter Type**. Select the structure you want to search:

- **Model** The system lists the models in the current study.
- **Table** The system lists the models and tables in the current study.

2. Select one or more models or tables. You can enter part or all of a model or table name above the column to filter.
3. Check **Exclude Based On Criteria** to filter **out** records that satisfy the criteria. This is the equivalent of adding "Not" to the query logic. Leave unchecked to display **only** the records that satisfy the criteria.
4. **Filter Availability** is **Study**. Saved filters of this type are available for use only in the current study.
5. **Filter Name**: Enter a descriptive name for the filter to save it to use in another session; that is, after you log out or work in a different study.

   Filter names must be unique across all private, shared, and public filters that you can see in a particular study and lifecycle.

   If you do not name the filter, the system calls it *Temporary Filter* and it is available only during your current session.
6. **Description**: Enter additional text to help you and other users know if this filter meets their needs. The description is displayed in the Edit Filter window and in the Current Filters window that lists all filters currently applied on a page.
7. **Save As Copy**. This option is available only when you are editing an existing, named filter. If this option is selected, you can make changes and save the filter with a different name.
8. Click **OK**

Exporting All to Excel

Click **Export to Excel** to generate an .xls file that includes all discrepancies that satisfy the current filters (if any).

Adding a Comment

Select one or more discrepancies and click the **Add Comment** icon to add the same comment to all selected discrepancies.

The system uses the **Internal Comment?** check box only for discrepancies that originated in Oracle Thesaurus Management System (TMS). If it is **not** selected and you add a comment to a discrepancy that was raised in TMS, the system changes the discrepancy's status to **TMS EVALUATION** and TMS autoclassification runs on the source term during the next transformation job.

Actions

The **Actions** drop-down lists all valid actions for the selected discrepancies. To apply an action, select it from the list. A window appears.

1. In the window, either:
   - Select a standard reason for change.
   - Enter your own reason.
   - Select a reason and enter additional information.

2. Click **OK**.

Part III

Appendixes

This section contains the following topics:

A

Reference Information

This section contains reference information.

Naming Objects

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.

> **Note:** The system uses the value you enter in the Name field for the default Oracle name.

For information on 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.

> **Note:** In addition, DUPLICATE is a reserved word in Oracle DMW for columns. It is used in the Default Listings page to allow filtering on duplicate records.

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 Oracle DMW_DOMAIN itself, the study, and any libraries or other subdomains you create. (The maximum number of subdomains is 9. This number is configurable using a profile in Oracle LSH; see the *Oracle Life Sciences Data Hub System Administrator's Guide*.) See also "Object Ownership" on page A-5.

- **Source Code definitions path**: *username*>cdrwork>*Domain_name*>*Subdomain_name(s)*>*Application_Area_name*>*Program_definition_name*>*Source_Code_definition_name*>*version_number*>*fileref*>*source_code_filename*
- **Table instances path**: *username*>cdrwork>*Domain_name*>*Subdomain_name(s)*>*Application_ Area_name*>*Work_Area_name*>*program_ instance_name*>*program_ version*>*Table_Descriptor_libname*> *Table_Descriptor_SAS_name*>
- **SAS runtime script path**: *username*>cdrwork>*Domain_name*>*Subdomain_name(s)*>*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 sequential number (in this case, 02).

Handling of Duplicate Names: System Appends _1

The system enforces unique naming for each object type in the same container. The system enforces unique names only within the immediate container. For example, you cannot have two tables with the same name in the same model, but you can have tables with the same name in different models in the study.

If you try to create a second object of the same type and name in the same container, the system creates the object but appends an underscore and the number one (_1) to

the name. If you add a third object of the same type and name, the system increments the number (_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 .mdd file must have the syntax described here, in the order given. 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 then load data into it from a SAS file.

Delimiter (Optional) Must be `lsh_table= delimiter`. If you do not specify a delimiter, the default delimiter is a comma (,).

Table (Optional) Must begin with `lsh_table=` . If you do not specify a table name in the file, the system uses the file name (without the extension) as the table name and follows Oracle LSH default behavior for the attribute values.

Columns The system expects a set of column attribute values, one column per row in the file, optionally preceded by a row identifying the delimiter and a row defining Table attribute values, each of which must begin with a key word. Column position is determined by the order in which the column rows in the file are processed. Oracle LSH default behavior for the attribute values applies.

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.

Comments A row beginning with two dashes is treated as a comment.

Syntax

```
--This is a comment.
lsh_delimiter = ,
lsh_table= Name, Description, Oracle Name, SAS Name, SAS Label, Process Type,
Allow Snapshot?, Blinding Flag?, Blinding Status, SAS Library Name, Is Target?,
```

```
Target as Dataset?, SDTM Identifier (SUBJECT/SUBJECTVISIT), Table Alias, Blinding
Type (TABLE/COLUMN/ROW), Blinding Criteria
--Column details:
Name, Data Type, Length, Precision, Oracle Name, SAS Name, SAS Format,
Description, SAS Label, Nullable?, Default Value, Date Format, SDTM Identifier,
Column Alias, Masking Level (COLUMN/CELL), Masking Value, Masking Criteria
--Constraints must start with string "CONSTRAINTS". Requirements for each type:
CONSTRAINT,Name,Description,PRIMARYKEY,Duplicate_PK_Support_Flag (YES/NO),
Surrogate_Key_Flag (YES/NO),{delimited_list_of_columns_in_key]
CONSTRAINT,Name,Description,UNIQUE,,,[delimited_list_of_columns_in_key]
CONSTRAINT,Name,Description,NONUNIQUE,,,[delimited_list_of_columns_in_key]
CONSTRAINT,Name,Description,BITMAP,,,[delimited_list_of_columns_in_key]
CONSTRAINT,Name,Description,CHECK,,,[column_name],{delimited_list_of_values}
```

Note that all constraints require square brackets ([]) around the column name(s). In addition, the check constraint requires curly brackets ({}) around the list of values.

See Table A–1, " Table Attributes with Reference Codelist Values" andTable A–2, " Column Attributes with Reference Codelist Values".

Example A–1 Metadata File

```
lsh_delimiter = |
--This section is for 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?, SDTM Identifer (SUBJECT/SUBJECTVISIT), Table Alias, Blinding Type
(TABLE/COLUMN/ROW), Blinding Criteria
lsh_table=S_QS|S_QS Table|S_QS|S_QS|S_QS|Staging with
Audit|Yes|Yes|Blinded|Target|Yes|Yes||qs|ROW|(VISITDY < 100)
--
--This section is for columns
--Name, Data Type, Length, Precision, Oracle Name, SAS Name, SAS Format,
Description, SAS Label, Nullable, Default Value, Date Format, SDTM Identifier,
Column Alias, Masking Level(COLUMN/CELL), Masking Value, Masking Criteria
--
STUDYID|VARCHAR2|12||STUDYID|STUDYID|$12.||Study Identifier|Yes||
USUBJID|VARCHAR2|11||USUBJID|USUBJID|$11.||Unique Subject Identifier|Yes||
QSTESTCD|VARCHAR2|7||QSTESTCD|QSTESTCD|$7.||Question Short Name|Yes||
VISITNUM|NUMBER|||VISITNUM|VISITNUM|8.||Visit Number|Yes||
DOMAIN|VARCHAR2|2||DOMAIN|DOMAIN|$2.||Domain Abbreviation|Yes||
QSSEQ|NUMBER|||QSSEQ|QSSEQ|8.||Sequence Number|Yes||
QSTEST|VARCHAR2|40||QSTEST|QSTEST|$40.||Question Name|Yes||
QSCAT|VARCHAR2|70||QSCAT|QSCAT|$70.||Category for Question|Yes||
QSSCAT|VARCHAR2|26||QSSCAT|QSSCAT|$26.||Sub-Category for Question|Yes||
QSORRES|VARCHAR2|20||QSORRES|QSORRES|$20.||Finding in Original Units|Yes||
QSORRESU|VARCHAR2|7||QSORRESU|QSORRESU|$7.||Original Units|Yes||
QSSTRESC|VARCHAR2|4||QSSTRESC|QSSTRESC|$4.||Character Result/Finding in Std
Format|Yes||
QSSTRESN|NUMBER|||QSSTRESN|QSSTRESN|8.||Numeric Finding in Standard Units|Yes||
QSSTRESU|VARCHAR2|7||QSSTRESU|QSSTRESU|$7.||Standard Units|Yes||
QSBLFL|VARCHAR2|1||QSBLFL|QSBLFL|$1.||Baseline Flag|Yes||
QSDRVFL|VARCHAR2|1||QSDRVFL|QSDRVFL|$1.||Derived Flag|Yes||
VISIT|VARCHAR2|19||VISIT|VISIT|$19.||Visit Name|Yes||
VISITDY|NUMBER|||VISITDY|VISITDY|8.||Planned Study Day of Visit|Yes||
QSDTC|VARCHAR2|10||QSDTC|QSDTC|$10.||Date/Time of Finding|Yes||
QSDY|NUMBER|||QSDY|QSDY|8.||Study Day of Finding|Yes||
--
--This section is for constraints
--
CONSTRAINT|pk_1|pk|PRIMARYKEY|Yes|YES|[STUDYID|USUBJID]
```

Example A–2 Constraint Metadata

```
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}
```

Table A–1 Table Attributes with Reference Codelist Values

Attribute	Valid Values
Processing Type	UOW, Reload
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. You can safely use the default value (YES).
Target as dataset	YES, NO. You can safely use the default value (NO).
SDTM Identifier	For tables: SUBJECT, SUBJECTVISIT
Blinding Type	TABLE, COLUMN, ROW

Table A–2 Column Attributes with Reference Codelist Values

Attribute	Valid Values
Data Type	VARCHAR2, NUMBER, DATE
Nullable	YES, NO
SDTM Identifier	For valid values, see "Using SDTM Identifiers for Columns" on page 3-15.
Blinding Type	TABLE, COLUMN, ROW
Masking Level	COLUMN, CELL

Object Ownership

In order to understand object security, you must understand the object ownership hierarchy. The following section describes the hierarchy and Figure A–1, "Object Ownership" represents it graphically.

Object Hierarchy

Oracle DMW uses Oracle LSH object types and a few additional object types created for use in Oracle DMW. The following diagram shows only primary and instance 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 shown in Figure A–1, "Object Ownership", is described in the following sections.

DMW Domain

The DMW_DOMAIN domain is an LSH container object of type Domain. It contains all user-created objects in Oracle DMW.

DMW_DOMAIN is created within the Oracle LSH instance domain during product installation.

DMW Utilities Domain

The DMW_UTILS domain is created during product installation inside the Oracle DMW domain.

The DMW_UTILS domain is designed to hold user-defined custom programs, functions, and reference tables for use with the programs. You can create application areas within it to organize the programs you create, using the Oracle LSH user interface or public APIs. Create programs and tables in the Oracle LSH user interface or with public APIs.

Note: For performance reasons, Oracle recommends creating application areas instead of domains inside DMW_UTILS to organize programs and other objects.

Study Category Domains

Define additional domains in the Oracle DMW domain to organize studies into categories and to hold library clinical data models and code lists for each category. By default the categories are called therapeutic areas in the user interface, but you can change this; see "Setting Up Library and Study Categories" on page 8-20

You create these domains in the Oracle LSH user interface or with public APIs.

Study Domains

A 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 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.

Each lifecycle area has a work area for every installed clinical data model, validation check batch, and custom listing. Transformation maps are installed in their target clinical data model's work area.

Clinical Data Models

When you create a clinical data model, the system puts the model's metadata directly in the study domain. When you install the model, the system:

- Creates a work area object in the current lifecycle area.
- Creates a database schema for the model 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 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 in the input model's work area.

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 in the study domain and installed in the relevant work area.

Transformation map programs are installed in the target model's work area.

Transformation Maps

Transformation map metadata is contained 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.

Validation Check Batches and Validation Checks

Validation check batches are created internally as model-level transformation maps, and validation checks are created as child table-level transformation maps. They in turn contain validation check details.

Custom Listings

Saved custom listings are created internally as table-level transformation maps.

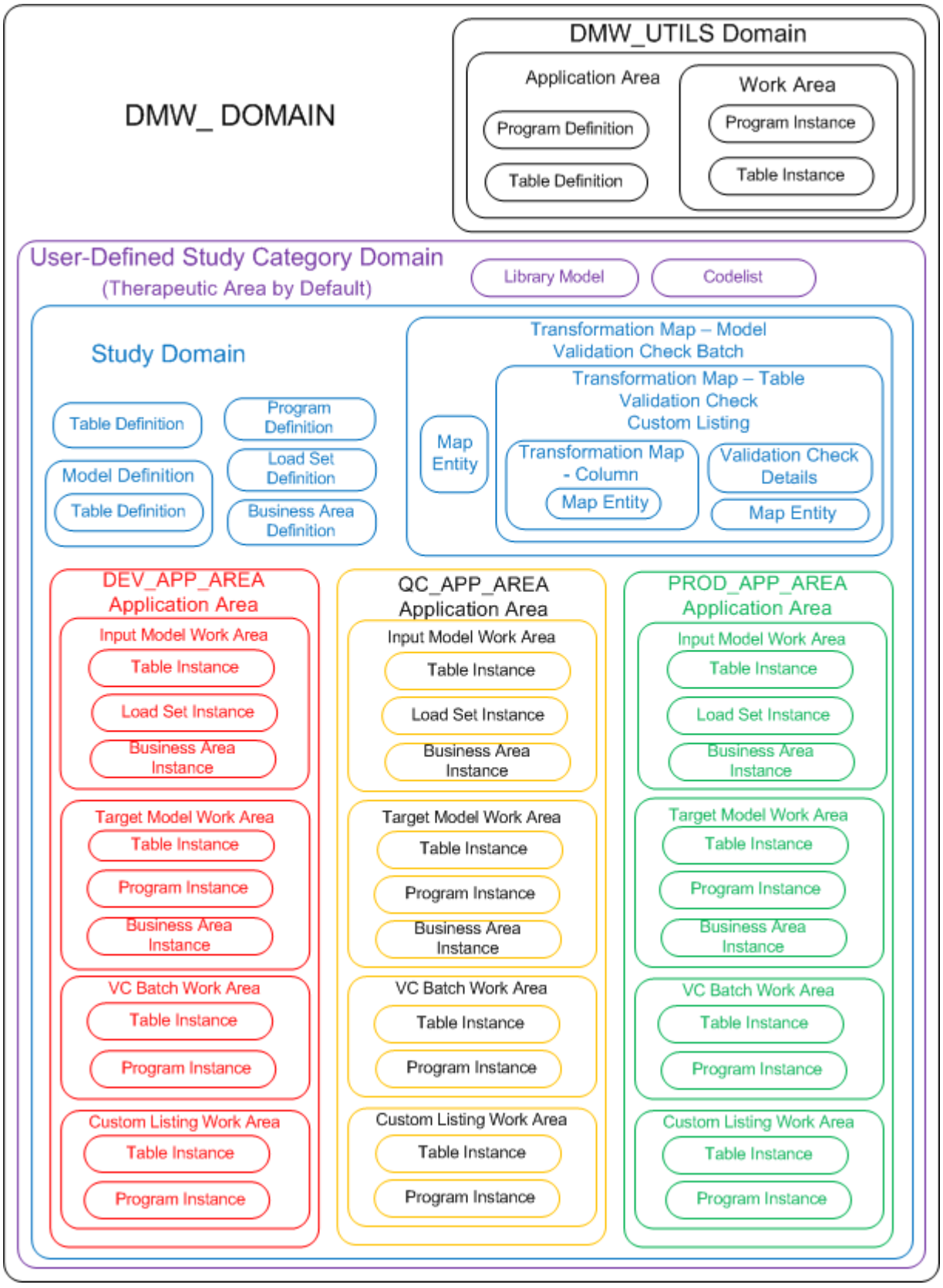
Business Areas

When you create a study, if you select the option to create a business area, the system does so. 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 8-25

Data Marts

To 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.

Figure A–1 Object Ownership



Effects of User Group Assignment to Objects

A user group assigned to an object is 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 definitions.

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 program contained in the transformation or validation check batch.

Only after installation can tables actually hold data, and transformations and validation checks read data from tables and write data to tables.

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.

B

Predefined Roles

You can use shipped, predefined object security and application roles to simplify security setup.

See also "Object Ownership" on page A-5.

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

Oracle 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

Oracle 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 Oracle DMW_STUDY_MANAGER and Oracle DMW_STUDY_CONFIG.

DMW_STUDY_CONSUMER

Oracle 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

Oracle 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

Oracle 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 or to 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 program that reads from currently blinded tables—Tables with a Blinding Status of Blinded—and operates on the real data, not the masked or dummy data (also requires an object security role with the Blind Break operation on Table instances).
- View the output generated by a program run on real, blinded data (also requires an object security role with the Blind Break operation on outputs).
- View currently blinded data in the user interface (requires an object security role with the Blind Break operation on Table instances).
- View currently blinded data through a visualization tool (requires an object security role with the Blind Break operation on Table instances).

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) 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.

Oracle DMW_STUDY_DEVELOPER

The Oracle DMW_STUDY_DEVELOPER object security role is intended for users who 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 Oracle DMW_STUDY_MANAGER or Oracle DMW_STUDY_CONFIG application role. If they need to create library data models they should also have the Oracle DMW_LIB_ADMIN application role.

Oracle DMW_STUDY_DEVELOPER allows users to perform the following operations on the specified objects.

Clinical Data Models and Query Builder

Users assigned the Oracle 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 Oracle 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 Oracle 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 Oracle 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 Oracle 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 Oracle 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: Assign Flags

Validation Checks

Users assigned the Oracle 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 Oracle 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 Oracle 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

Oracle DMW_STUDY_QC

The Oracle DMW_STUDY_QC object security role is intended for users who 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 Oracle DMW_STUDY_MANAGER or Oracle DMW_STUDY_CONFIG application role.

Oracle DMW_STUDY_QC allows users to perform the following operations on the specified objects.

Clinical Data Models and Query Builder

Users assigned the Oracle 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 Oracle 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 Oracle 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 Oracle 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 Oracle 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 Oracle 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 Oracle 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 Oracle 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 Oracle 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 Oracle 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

Oracle DMW_STUDY_PROD

The Oracle DMW_STUDY_PROD role is intended for users who 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 Oracle DMW_STUDY_MANAGER application role.

Oracle DMW_STUDY_PROD allows users to perform the following operations on the specified objects.

Clinical Data Models and Query Builder

Users assigned the Oracle 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 Oracle 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 Oracle 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 Oracle 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 Oracle 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 Oracle 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 Oracle 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 Oracle 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 Oracle 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 Oracle 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

Oracle DMW_STUDY_ADMIN

The Oracle DMW_STUDY_ADMIN object security role is intended for superusers. It includes the object-related roles that Oracle DMW_STUDY_DEVELOPER and Oracle 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 Oracle DMW_STUDY_MANAGER or Oracle DMW_SYS_ADMIN application role.

Oracle DMW_STUDY_ADMIN allows users to perform the following operations on the specified objects.

Clinical Data Models and Query Builder

Users assigned the Oracle 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 Oracle 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 Development/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 Oracle DMW_STUDY_ADMIN role have access to the following operations for flags.

Oracle DMW Object Operations

Table Instance: Assign Flags

Filters

Users assigned the Oracle 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 Oracle 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 Oracle 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 Oracle 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 Oracle 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 Oracle 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

Oracle DMW_STUDY_INST_ACCESS

The Oracle 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 Oracle DMW_STUDY_CONSUMER application role.

The Oracle 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 Oracle 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 Oracle 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 Oracle 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 Oracle 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 Oracle 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 Oracle 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

Oracle 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 Oracle 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 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 Oracle Health Sciences 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 Oracle Health Sciences 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 loads the data into the system.

InForm

A clinical trial data collection and management application; full name is Oracle Health Sciences InForm.

Listings

A page in the Oracle DMW user interface 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 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

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W