Oracle® Health Sciences Data Management Workbench
Study Setup Guide

ORACLE®

Oracle Health Sciences Data Management Workbench Study Setup Guide, Release 3.0

F30622-01

Contents

1   Study configuration basics

2   Set up clinical data models

3   Create libraries of codelists and clinical data models

4 Set up data transformations

5   Create validation checks

6   Create custom programs, listings, and filters, and set up reference data

7   Load data and run jobs

8 Advanced topics

ORACLE

Preface

This guide tells you how to set up a study in Oracle DMW, bringing in data from InForm or another electronic clinical data system, labs, and other file systems and transform and validate the data in Oracle DMW.

- Documentation accessibility
- Related resources
- Access to Oracle Support

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.

Related resources

All documentation and other supporting materials are available on the Oracle Help Center.
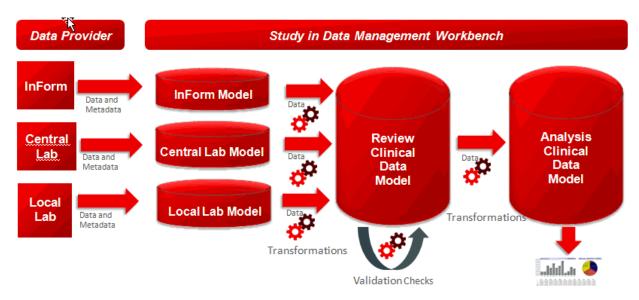
Access to Oracle Support

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.

1

Study configuration basics

The following figure illustrates how you can provide data from an electronic clinical data system (such as InForm), labs, or file systems to transform and validate the data in a study using Oracle Health Sciences Data Management Workbench.

Figure 1-1   Example DMW study design

For details on DMW concepts, see this video: ▶ Video

Table 1-1   High-level steps for creating a study

Task	More information
Begin study creation.	Specify a name and other attributes. See Create a study.
(Optional) Apply a study template.	A study template includes clinical data models, transformations, validation checks, and custom listings. You can modify or delete any of these as needed. See Use a study template for details.
Set up importing data from InForm.	Create an input clinical data model of type InForm and set up the InForm Connector for the study. The Connector creates table metadata exactly as it exists in the InForm database. See Create a clinical data model for InForm data.
Set up importing data from labs and other sources using files.	The system administrator must set up File Watcher for the system and for your study. Then create an input clinical data model of type File. See Create a file input clinical data model.

Table 1-1   (Cont.) High-level steps for creating a study

Task	More information
Create or copy target clinical data models.	A target clinical data model is a collection of tables used together for a purpose such as reviewing or analyzing data. You can: • Create a model by uploading a metadata file or in the user interface. See Create a target clinical data model for transformed data. • Create a model from a library model, which allows you to update the study model whenever the library model is updated. See Create a library clinical data model. • Copy a model from another study.
Create or copy transformations.	A transformation reads data from one or more tables in one or more clinical data models and writes data to a table in a different model. See Set up data transformations.
Create or copy validation checks.	Validation checks, or edit checks, check data for a condition and create discrepancies on faulty data. See Create validation checks.
Create public custom listings and filters. Set up static reference data.	See Create custom programs, listings, and filters, and set up reference data.
Integrate with Thesaurus Management System (TMS) for coding (optional).	• Specify the terminologies to use for the study. See Assign dictionaries for coding (optional). • Map columns to data to be derived from TMS
Load data and run transformations and validation checks on the Home page.	See: • Load data from InForm • Suspend or resume data file loading • Automatic triggering of transformations and validation checks
Upgrade models, transformations, and validation checks to Quality Control (optional) and then Production.	See Validation status and lifecycle stages.

See also:

Advanced topics

• Create a study

• Use a study template

• Assign dictionaries for coding (optional)

Create a study

For details on creating a study, see this video: ▶ Video

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.

   If the study is available as a template, edit the description to help others decide whether to select this study as a template. The maximum length is 2000.

4. **Therapeutic Area (or other category)**: Select the category the study belongs to. The label for this field and the choices available are customizable by your company.

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 value to help determine which partition to use for this study's data in certain cross-study internal tables.

   > 💡 **Tip:**
   >
   > You cannot change this value after saving.

6. Click **OK**.

7. If you are using Oracle Thesaurus Management System as a coding system, click the TMS tab and see Assign dictionaries for coding (optional).

Next: Create clinical data models. Do one or both:

- Use a study template
- Set up clinical data models

Use a study template

A study template is a study that you can copy to another study so that its clinical data models, transformations, and validation check batches become part of the new study. InForm input models are not included in the template.

> ✏️ **Note:**
>
> You can apply only one study template to a study.

In the new study you can:

- Delete or modify template models, tables, transformations, or validation checks.
- In transformations, mark tables and columns in target models as Not Used.
- Copy or create additional models, tables, transformations, and validation checks. If two components of the same type have the same name, the system appends "_1" to the name of the second component added.

1. Click the **Home** page icon 🏠 at the top of any page.

2. In the Home page, select the study you want to copy the template into.

3. Click the **Study Configuration** icon from the navigation bar. Then, select the **Study Template** tab. The system displays a list of study templates with their descriptions.

4. Find the template you want to use.

   • You can filter by entering a value in the blank field above any column. If blank fields are not displayed, click the **Query By Example** icon.

   • To see a template's details, query for it in the Home page, then navigate to its clinical data models, transformations, and validation checks.

5. Select the template and click the **Apply Template to Study** icon.

> ✎ **Note:**
>
> While the system applies the template you cannot use the Study Configuration pages, including Clinical Data Models, Transformations, or Validation Checks. Click the **Refresh** icon to see the updated Job Status.

Back to Table 1-1.

Assign dictionaries for coding (optional)

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

Prerequisite: Dictionaries and TMS domains must be defined in TMS.

1. Go to the **Home** page.

2. Select a study and click the **Modify** icon in the Studies pane.

3. Click the **TMS** tab.

4. For each base terminology (dictionary) needed in the study, select a **TMS Domain** from the drop-down list.

   TMS domains allow you to code terms differently in different studies. Domains also determine TMS system behavior, including whether explicit approval is required for manual classifications.

5. Click **OK**.

Back to Table 1-1.

• Force rederivation of TMS data

Force rederivation of TMS data

Normally, only new or changed data is sent to TMS for processing. Use Force Rederivation to resend all data designated as needing coding in TMS when you have made structural changes such as:

- Adding columns to hold derived data in DMW target tables.

- Updating a dictionary in TMS to a new version with a different structure from the old version.

- Changing domain-related settings in the TMS reference codelist TMS_CONFIGURATION.

Running this job triggers the execution of the downstream transformation.

To force rederivation of all data from TMS:

1. Go to the [home icon] **Home** page.

2. Select a study.

3. Click the [modify icon] **Modify** icon in the Studies pane.

4. Click the **TMS** tab.

5. Click **Force Rederivation** to run Rederivation once, immediately.

6. A confirmation message appears because the job may take a long time. You can still work while it runs. Click **OK**.

> **Note:**
>
> Force Rederivation is not part of setting up TMS in a study. You only run it in a live study, after making the changes described above.
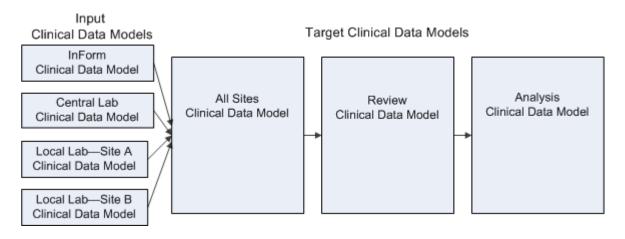
2
Set up clinical data models

A clinical data model is a set of logically related tables. You need one **input** model for each data source. You create **target** models to review, analyze, or report data. You use transformations to standardize and merge data from one or more source models into a target model.

Figure 2-1   Example of clinical data model use in a study

The following table lists the tasks you can perform to set up clinical data models.

Table 2-1   High-level steps

Task	More Information
Create an input clinical data model for every lab.	See Create a file input clinical data model.
Create an input clinical data model for your clinical data system.	If you are using InForm, set up a connection to the InForm study development database and load metadata from there to create the clinical data model. See Create a clinical data model for InForm data. If you are using a different clinical data capture system, see Create a file input clinical data model.
Create target clinical data models and tables as needed.	Create models with tables in the format you need for review and analysis, and any intermediate models required. See Create a target clinical data model for transformed data. You can load metadata from a file using a required syntax, copy a model from another study, create tables manually, or create a study model from a library model, which allows you to update the study model when the library model is updated. To add individual tables, see Add tables.

Table 2-1   (Cont.) High-level steps

Task	More Information
For each table, define additional attributes and columns.	Add constraints to tables . A primary key is required. Set up data blinding in tables . Add columns to support filtering. See Use SDTM identifiers to support important functionality. Set up Unit of Work data processing for tables (optional). Configure table display in the Listings page for columns on the Listings pages (optional). Map columns to data to be derived from TMS (in target models, if you are using TMS).
Add a Subject Visit table.	Add Subject Visit and Subject tables. Each study must have a Subject Visit table with SDTM identifiers in one model.
Install each model.	Install a clinical data model.
Upgrade each model to Quality Control (optional) and then Production.	Validation status and lifecycle stages.

More information on the tasks in this chapter:

- Create a file input clinical data model

- Create a clinical data model for InForm data

- Create a target clinical data model for transformed data

- Add tables

- Add columns to a table manually

- Add constraints to tables

- Set up Unit of Work data processing for tables

- Configure table display in the Listings page

- Set up data blinding in tables

- Add Subject Visit and Subject tables

- Map columns to data to be derived from TMS

- Extract data from a clinical data model

- Modify a clinical data model

- Install a clinical data model

- Unblind and reblind data

- FAQs

Create a file input clinical data model

For details on creating a file input clinical data model, see this video: ⊙ Video

1. On the 🏠 **Home** page, select the study and the Development lifecycle stage.

2. Click the **Study Configuration** icon from the navigation bar. Make sure the **Clinical Data Models** tab is selected.

3. Click the Add icon for Clinical Data Models at the top of the left pane.

4. Enter a name and description. See Naming restrictions for details.

5. For model type, select **Input**.

6. For **Input Type**, select:

   • **File** to load data files from labs.

   • *Name_of_System* to load data files from a clinical data capture system other than InForm.

7. For **Input Data File Type** select either:

   • Text

   • SAS

8. For **Metadata Source** select:

   • **None** to define tables and columns manually.

   • **Load from file**. The system can create tables from:

     – A .zip file that contains one metadata (.mdd) file per table. See Required syntax for metadata files.

     – SAS Transport (CPort or XPort) files, a SAS dataset, or a .zip file that contains SAS datasets *or* text metadata (.mdd) files (not both).
       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 from a metadata file and load data into it from a SAS file. Metadata files are the only automated way to create tables with blinding attributes set.
   >
   > – Tables with a large number of columns may cause problems with data loading, transformations, and display in the Listings pages. The supported maximum ranges from 260 columns if all columns are individually blinded to 370 columns if the table is either not blinded, has only row blinding, or is completely blinded. See My Oracle Support article 2298558.1 for the latest information.

   • **Library Data Model** to create tables from a standard library model. If the library model is updated, you can update the study model.

   > **💡 Tip:**
   >
   > Library models must be checked in to appear in the list. You can only see models you have access to.

- **Study Data Model** to copy tables from a model in another study. Select its project (or other grouping), then its study, and then the model.
  The Copy operation includes transformations that write to the model and validation checks that read from the model.

9. To allow visualization tools access to the data contained in the clinical data model, select the **Business Area** checkbox.

   You can change the default schema name, BA_*model_name*. This is what users of the visualization tool will see. **Limitation**: Use a maximum of 19 characters for the schema name if you plan to create custom listings on the model. Do not use spaces or special characters other than underscore (_).

10. **Data Mart**: A data mart file contains all current data in a clinical data model for export to an external system. Select data mart type(s) to make available.

    Only the types that are available in your environment are displayed.

    - Oracle Export

    - Text Export

    - SAS Export

11. Click **OK**.

Next:

- Configure File Watcher data loading
- Configure the web service location for a source data system

Configure File Watcher data loading

Prerequisite

Your administrator must set up a File Watcher for this study.

1. Click the **Study Configuration** icon ![icon] from the navigation bar. Make sure the **Clinical Data Models** tab is selected.

2. Select **Development** from the Lifecycle drop-down at the top of the page.

   > 💡 **Tip:**
   >
   > In the Development lifecycle context you can configure File Watcher for any lifecycle stage.

3. Select the model and click **Check Out**.

4. Go to the clinical data model's Watcher Configuration tab.

5. In the **Data Source** field, select the name of the lab.

> ✏ **Note:**
>
> If this model is for data from an EDC system other than InForm, this field does not appear.

6. If data will be loaded in SAS files, enter SAS parameter values:

    • **Upload File Type**: Either CPORT, XPORT, or SAS Dataset.

      Datasets must be contained in a .zip file and each have the same name as one target table.

    • **Reported Errors**: The number of errors allowed per dataset before the load fails. No records are loaded if the load fails. See Format checks on files being loaded for more information.

7. Click the **Save** icon.

Next: See "Set text data load parameters" if you will load data in text files. Otherwise, see "Specify a file name for each lifecycle stage."

• Set text data load parameters

• Specify a file name convention for each lifecycle stage

Set text data load parameters

Prerequisite

Select a data source and save. See Configure File Watcher data loading.

1. Click the **Study Configuration** icon ![icon] from the navigation bar. Make sure the **Clinical Data Models** tab is selected.

2. Select **Development** from the Lifecycle drop-down at the top of the page.

> 💡 **Tip:**
>
> In the Development lifecycle context you can configure File Watcher for any lifecycle stage.

3. Select the model and click **Check Out**.

4. In the **Watcher Configuration** tab, click the ▤ **Data Load Parameters** icon.

5. Specify the format to be used in all data files:

    • **Fixed**: 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.

      – **DelimiterChar**: Enter the character to separate column values, for example, a comma (,) or a pipe (|).

–   **Enclosing Character**: If any data value may contain the delimiter character, another character must be added before and after each data value. Enter the character.

6.   Fill in the following fields:

*   **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).

*   **Reported Errors**: The number of errors allowed per file before the load fails. No records are loaded if the load fails. See Format checks on files being loaded for more information.

*   **Rows Before Commit**: The number of rows you want the system to process before committing processed rows to the database.

*   **Date Format**: 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.

> **Tips:**
>
> –   The examples show `mm` for minutes, but it should be `mi`.
>
> –   Use `mm` for a month number (for example, 11) or `mon` for a three-letter month (for example, NOV). Use `yyyy` for a 4-digit year.
>
> –   If you use a 24-hour clock, enter `hh24` for the hour. If you use a 12-hour clock, use `hh` and enter `am` after a space following the time string.
>
> For more examples, see: `https://docs.oracle.com/database/121/ TDDDG/tdddg_globalization.htm#TDDDG255`.

7.   Click **OK**.

Next: Specify a file name convention for each lifecycle stage

Specify a file name convention for each lifecycle stage

This procedures describes how to select the clinical model, name a file specification, set the lifecycle stage, and select the submission mode.

> **Important:**
>
> Make sure you have the privileges to load data in the specified lifecycle stage. If not, **data loads will fail**. When the data load runs, it uses the account that defined or most recently updated the File Specification.

1.   Click the **Study Configuration** icon from the navigation bar. Then click the **Clinical Data Models** tab.

2.   Select **Development** from the Lifecycle drop-down at the top of the page.

> **Tip:**
>
> In the Development lifecycle context you can configure File Watcher for any lifecycle stage.

3. Select the model and click **Check Out**.

4. In the **Watcher Configuration** tab, under File Specifications, click the ➕**Add** icon.

5. Enter a name and description for the **File Specification**.

> **Tip:**
>
> Include the lifecycle stage in the name to make it visible in the Detected Files tab.

6. **File Name**: Enter a regular expression for the names of files to be loaded in each lifecycle stage.

    File name patterns use the POSIX standard Extended Regular Expression syntax. An asterisk (*) in POSIX syntax matches zero or more occurrences of the preceding character. A dot (.) means "any single character." You can use .* to mean "any character or no characters."

    For example, in a study with file input models for three labs, you could use the following file specifications that allow for a date and the lab name:

    *   CentralLab_.*.zip
    *   SpecialLab_.*.cport
    *   LocalLab_.*.zip

> **Tips:**
>
> File names must be unique across all models in the study.
>
> Be careful about case sensitivity. Include the lab or source name.

    For more information, see `https://docs.oracle.com/cd/B28359_01/server.111/b28286/ap_posix001.htm`.

7. **Lifecycle**: Select Development, Quality Control, or Production.

8. Select the **Submission Mode**:

    *   **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**: Within each Unit of Work (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 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:**
>
> 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 Data processing types and modes.
> Use a different file name for incremental/UOW and full loads.

9. **Execution Priority**: The priority for loading data with this File Specification relative to others: Low, Normal, or High.

10. Select the **Dataload Type**:

    - **Immediate**: The Watcher searches for files continuously.

    - **Scheduled**: The Watcher searches at the interval you specify.

11. **Frequency**: If you selected Scheduled, select a frequency for the Watcher to look for a new data file in days, hours, or minutes.

12. **Start Date**: If you selected Scheduled, enter the date and time for file watching to begin.

13. **End Date**: Enter the date and time for file watching to end. You can enter a date far in the future and change it at any time.

14. Click **OK**.

Next: Install a clinical data model or, if the data source is a clinical data capture system other than InForm, Configure the web service location for a source data system.

Configure the web service location for a source data system

If you are using a clinical data capture system other than InForm, you must define its web service location so that Oracle DMW can send discrepancies to it and so that data reviewers can open the other system to view the discrepancy or data there.

1. Click the **Study Configuration** icon from the navigation bar. Then click the **Clinical Data Models** tab.

2. Select **Development** from the Lifecycle drop-down at the top of the page.

> **Tip:**
>
> In the Development lifecycle context you can configure File Watcher for any lifecycle stage.

3. Select the model and click **Check Out**.

4. Go to the *System* **Configuration** tab.

5. For each lifecycle stage, click the Modify icon and enter:

   a. *System* **Lifecycle**: Select the lifecycle stage in the external system that should exchange data and discrepancies with the selected DMW lifecycle stage.

   b. **Output Location**: Select the name of the web service location for the external system, defined by your administrator.

   c. **Base URL**: Enter the URL to use to view discrepancies and data in the external system from the DMW UI.

   d. **URL Listing Prefix**: If required, enter a string to add to the beginning of the base URL to view data in the external system.

   e. **URL Listing Suffix**: If required, enter a string to append to the end of the base URL to view data in the external system.

   f. **URL Discrepancy Prefix**: If required, enter a string to add to the beginning of the base URL to view discrepancies in the external system.

   g. **URL Discrepancy Suffix**: If required, enter a string to append to the end of the base URL to view discrepancies in the external system.

6. Click **OK**.

Back to Set up clinical data models

Create a clinical data model for InForm data

For details on creating a clinical data model for InForm data, see this video: Video

1. On the Home page, select the study and the Development lifecycle stage.

2. Click the **Study Configuration** icon from the navigation bar. Then click the **Clinical Data Models** tab.

3. Click the Add icon for Clinical Data Models at the top of the left pane.

4. Enter a name and description. See Naming restrictions for details.

5. For the model type, select **Input**.

6. For **Input Type**, select **InForm**.

> 💡 **Tip:**
>
> Don't create an InForm input clinical data model by copying it. 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 modifying it as necessary.

7. To allow visualization tools access to the data contained in the clinical data model, select the **Business Area** checkbox.

   You can change the default schema name, BA_*model_name*. This is what users of the visualization tool will see. **Limitation**: Use a maximum of 19 characters for the schema name if you plan to create custom listings on the model. Do not use spaces or special characters other than underscore (_).

   For InForm models, business areas for InForm metadata and operational data models are also created, with "M" and "O" in their names, respectively. For example, **STUDY12345_M_QC** is the name of the metadata model's business area in the quality control lifecycle.

8. **Data Mart**: A data mart file contains all current data in a clinical data model for export to an external system. Select data mart type(s) to make available:

   Only the types that are available in your environment are displayed.

   • Oracle Export

   • Text Export

   • SAS Export

9. Click **OK**.

> ✏️ **Note:**
>
> Tables with more than 339 columns may cause problems with data loading, transformations, and display in the Listings pages. See My Oracle Support article 2298558.1 for the latest information.

Next:

• Configure the InForm Connector

• Select internal InForm tables and views for data transformations

• Load InForm metadata

• Compare DMW and InForm Metadata

Configure the InForm Connector

1. Click the **Study Configuration** icon 🖼 from the navigation bar. Then click the **Clinical Data Models** tab.

2. Select **Development** from the Lifecycle drop-down at the top of the page.

3. Open the InForm clinical data model that you have created. (If necessary, see Create a clinical data model for InForm data for details on creating data models.)

4. If you haven't already checked out the InForm clinical data model, click **Check Out**.

5. In the **InForm Configuration** tab, for each lifecycle stage, in the **Remote Location** field, specify the InForm reporting database from which to load data.

6. **Remote Study Account Name**: Enter the name of the database account that owns the study's InForm reporting database and RDE views for the appropriate DMW lifecycle stage.

> **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 DMW and the InForm metadata at the new location. If there are differences, the metadata comparison report is displayed. If you accept the changes, data loading is suspended. Reload metadata before resuming data loading.
>
> When you reload metadata, **you must redo any blinding and masking** you've done in DMW.

7. **InForm LifeCycle**: Select the InForm lifecycle stage of the study account name.

8. **Webservice Location**: Specify the web service location.

9. **InForm URL**: Enter the URL for the study's InForm website.

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

10. Click the Test URL icon on the far right. If the InForm login page opens, the URL is correct. The new window may open behind the current one.

11. Click the Save icon.

Next: Select internal InForm tables and views for data transformations.

Select internal InForm tables and views for data transformations

1. Click the **Study Configuration** icon from the navigation bar. Then click the **Clinical Data Models** tab.

2. Select **Development** from the Lifecycle drop-down at the top of the page.

3. Select the clinical data model and click **Check Out**.

4. Click the **Select InForm Operational Data and Metadata Tables** icon.

   The system displays all InForm tables and views alphabetically. To sort by type, click the heading of the **Internal Data Model** column. Your administrator makes the default selections. Gray tables and views are required.

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

6. Save.

After you load metadata and data and install, you can see the internal tables and views in the Default Listings page.

Next: Load InForm metadata.

Load InForm metadata

1. Click the **Study Configuration** icon from the navigation bar. Then click the **Clinical Data Models** tab.

2. Select **Development** from the Lifecycle drop-down at the top of the page.

3. Select the InForm clinical data model and click **Check Out**.

4. In the **InForm Configuration** tab, click the **Suspend Data Load** icon.

5. Click the **Load InForm Metadata** icon. In the Development lifecycle, this process includes installing the model.

6. Click the **Refresh** icon and check the Status column.

7. Click the **Resume Data Load** icon.

To load data, see:

• Load InForm data immediately in Development or Quality Control

• Schedule InForm data loads in Production

• Suspend and resume InForm data loading

Compare DMW and InForm Metadata

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

1. Click the **Study Configuration** icon from the navigation bar. Then click the **Clinical Data Models** tab.

2. In the clinical data model **InForm Configuration** tab, click the **Compare DMW and InForm Metadata** icon.

3. In the **Metadata Comparison** window, select the metadata to compare in DMW and InForm.

4. Click **Compare**. The report appears on screen.

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

   If the system finds no differences, a message appears.

For more information, see InForm metadata change detection and synchronization.

Back to Set up clinical data models

Create a target clinical data model for transformed data

For details on creating a target clinical data model for transformed data, see this video:
▶ Video

1. On the 🏠 **Home** page, select the study and the Development lifecycle stage.

2. Click the **Study Configuration** icon ✎ from the navigation bar. Then click the **Clinical Data Models** tab.

3. Click the ➕**Add** icon for Clinical Data Models at the top of the left pane.

4. Enter a name and description. See Naming restrictions for details.

5. For model type, select **Target**.

6. For **Metadata Source** select:

   • **None** to define tables and columns manually.

   • **Load from file**.The system can create tables from a .zip file that contains one (.mdd) file per table. See Required syntax for metadata files.

     SAS Transport (CPort or XPort) files, a SAS dataset, or a .zip file that contains SAS datasets *or* text metadata (.mdd) files (not both).

     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 from a metadata file and load data into it from a SAS file. Metadata files are the only automated way to create tables with blinding attributes set.
   >
   > – Tables with a large number of columns may cause problems with data loading, transformations, and display in the Listings pages. The supported maximum ranges from 260 columns if all columns are individually blinded to 370 columns if the table is either not blinded, has only row blinding, or is completely blinded. See My Oracle Support article 2298558.1 for the latest information.

   • **Library Data Model** to create a study model from a standard library model. If the library model is updated, you can update the study model.

   • **Study Data Model** to copy a model from another study. Select its project (or other grouping), then its study, and then the model.

     The Copy operation includes transformations that write to the model and validation checks that read from the model.

   You can also Add tables manually.

7. To allow visualization tools access to the data contained in the clinical data model, select the **Business Area** checkbox.

You can change the default schema name, BA_*model_name*. This is what users of the visualization tool will see. **Limitation**: Use a maximum of 19 characters for the schema name if you plan to create custom listings on the model. Do not use spaces or special characters other than underscore (_).

8. **Data Mart**: A data mart file contains all current data in a clinical data model for export to an external system. Select data mart type(s) to make available:

   Only the types that are available in your environment are displayed.

   - Oracle Export
   - Text Export
   - SAS Export

9. Click **OK**.

Back to Set up clinical data models

Add tables

The following procedures describe how to copy tables, create tables from a file, and add or modify a table.

- Copy tables
- Create tables from a file
- Add or modify a table manually

Copy tables

Copied tables include:

- Columns, constraints, and blinding and data processing attribute values.
- Validation checks that read from the selected tables, if all the source tables are included in the Copy operation.
- Public custom listings that read from the selected tables, if all the source tables are included in the Copy operation.

1. Click the **Study Configuration** icon from the navigation bar. Then click the **Clinical Data Models** tab.
2. Select the model into which you want to copy tables and click **Check Out**.
3. In the Tables tab, click the **Copy Tables** icon.
4. Select the source: either **Study Data Model** or **Library Data Model**.
5. To specify the tables to copy, start by selecting the study category, then the study (if you selected Study Data Model), then the model.

   - To help find any of these, type part or all of their name in the field above.
   - Click **Clear Filters** to remove all typed text and revert to the full list.

6. Click to select the table or tables to copy.
7. Specify how to handle any copied tables with the same name as existing tables in the model:

- • To replace tables with the same name that already exist in the current model, select **Overwrite the same table names**.
- • If you leave it deselected, 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.

8. Click **OK**.

Back to Set up clinical data models

Create tables from a file

You can create tables from a ZIP file that contains one or more text metadata (.mdd) files, one for each table. For the required syntax, see Required syntax for metadata files.

- • SAS Transport (CPort or XPort) files.
- • A ZIP file that contains one SAS dataset.
- • A ZIP file that contains one or more text metadata (.mdd) files, one for each table. Metadata files are the only way to create tables with all blinding attributes set; see Required syntax for metadata files.

1. Click the **Study Configuration** icon   from the navigation bar. Then click the **Clinical Data Models** tab.

2. Select the model and click **Check Out**.

> 💡 **Tip:**
>
> If the **Check Out** option is not active, you cannot check out the model because it is already checked out or you do not have the required privileges. To see who checked it out, look at the **Checked Out By** value at the left of the gray values near the top of the page. You may need to click the **>>** icon.

3. Click the **Modify** icon in the **Data Model** pane.

4. For **Metadata Source**, select **Load from File**.

5. Click **Choose File**.

6. Click **Open**.

The system creates one table per .mdd file.

Add or modify a table manually

1. Click the **Study Configuration** icon   from the navigation bar. Then click the **Clinical Data Models** tab.

2. Select the Development lifecycle.

3. Select the model and click **Check Out**.

If the **Check Out** option is not active, either the model is already checked out or you do not have the required privileges. To see who checked it out, see the **Checked Out By** value near the top right of the page. You may need to click the **>>** double arrow icon.

> **Note:**
>
> Do not make structural changes to tables in InForm models; see Modify an InForm input model for information on the changes that are and are not allowed.

4. In the Tables tab, click the ➕**Add Table** icon or select a table and click the 🖊**Modify Table** icon.

5. Enter values in the following fields:

   • Enter a **name** and **description** for the column; see Naming restrictions for restrictions.

   • **Oracle Name**: The system enters the value you entered for the name, truncated at 30 characters. See Automatic name truncation.

   • **SAS Name**: The system enters the value you entered for the name, truncated at 32 characters.

   • **SAS Label**: (Optional) The system enters 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 table. 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.

   • **SDTM Identifier**: If this table corresponds to an SDTM standard Subject or Subject Visit table, select its identifier from the list. See Add Subject Visit and Subject tables for information about requirements. Using SDTM identifiers makes automapping more accurate.

   > **Tip:**
   >
   > The SDTM Identifier field is available only when modifying a table.

   • **UOW (Unit of Work) Processing Type**: See Set up Unit of Work data processing for tables.

6. Set blinding attributes; see Set up data blinding in tables .

7. Click **OK**.

8. Finish table details:

   • All clinical data tables must have columns and a primary key. See Add columns to a table manually and Add constraints to tables .

   • You must select a data processing type. See Set up Unit of Work data processing for tables.

- You can change the way the table columns are displayed in the Listings pages.
- To mask data, see Set up data blinding in tables .

Back to Set up clinical data models

Add columns to a table manually

You can add columns to a table as part of creating the table itself when you upload a file; see Create tables from a file.

> ○ **Tip:**
>
> Be sure to add columns with the following SDTM IDs to support filtering in the Listings and Discrepancies windows: SUBJID, USUBJID, VISIT and VISITNUM. See Use SDTM identifiers to support important functionality.

1. Click the **Study Configuration** icon from the navigation bar. Then click the **Clinical Data Models** tab.

2. Select the model and click **Check Out**.

   Caution: Do not make structural changes to tables in InForm models; see Modify an InForm input model for information on the changes that are and are not allowed.

3. Select the table.

4. 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.

   > ✎ **Note:**
   >
   > There is a limit on the number of columns a table can have and still be displayed in the DMW Listings pages. See the note in Required syntax for metadata files for more information.

5. Fill in the following fields:

   - Enter a **Name** and **Description** for the column; see Naming restrictions 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 can also enter a value for **Precision**, which is the total number of digits allowed to the right of the decimal point. 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.

- **Map to Filter**: If the column has the same function and data type as one of the SDTM column identifiers, it is good practice to select it from the list because the system uses this information in several ways; see Use SDTM identifiers to support important functionality.

- **Oracle Name**: By default, the system populates this with the value you entered for the name, truncated at 30 characters.

- **SAS Name**: By default, the system populates this with the value you entered for the name, truncated at 32 characters.

- **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 default data value.

- **Aliases for Automapping**: 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`

- **Nullable**: If selected, having a value in this column is not required. If not selected, all rows must have a value in this column.

- **Codelist** (Optional): If the column should be populated with a limited set of values that are defined in a codelist, select the appropriate therapeutic area (or other category) and then the codelist. You can apply a codelist 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 codelist; see Pivot.

6. Enter blinding attributes; available only if the table has a blinding type of Column; see Specify masking attributes for a column.

7. Click **OK**.

Back to Set up clinical data models

Add constraints to tables

1. Click the **Study Configuration** icon from the navigation bar. Then click the **Clinical Data Models** tab.

2. Select the model and click **Check Out**.

If the **Check Out** option is not active, you cannot check out the model because it is already checked out or you do not have the required privileges. To see who checked it out, look at the **Checked Out By** value at the left of the gray values near the top of the page. You may need to click the **>>** icon.

3. Select the table.

4. In the Constraints tab, click the **Add** icon.

5. 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:

  - **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.

  - **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.

    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.

  - **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.

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

- **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.

> ✎ **Note:**
>
> The Not Null constraint is handled as an attribute called Nullable for each column.

6. **Columns**: Specify the columns in the constraint by selecting them from the list on the left and using an **Arrow** icon to move them to the right.

7. Select **Supports Duplicate** to support inserting records with the same primary key value within a single data load, which 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 key values in a load.

8. Click **OK**.

> ♡ **Tip:**
>
> If you create tables by uploading text files you can define constraints at the same time; see Required syntax for metadata files. InForm tables' constraints are imported as part of a metadata load and cannot be modified in the input model.

Back to Set up clinical data models

Set up Unit of Work data processing for tables

Using Unit of Work (UOW) processing can speed up data loading and transformation execution. See Data processing types and modes for more information.

> ♡ **Tip:**
>
> We recommend always defining tables as Subject Visit UOW if both Subject and Visit are part of the primary key and as Subject UOW if the Subject column is part of the primary key but Visit is not.

1. Click the **Study Configuration** icon 　 from the navigation bar. Then click the **Clinical Data Models** tab.

2. Select the model and click **Check Out**.

3. Select the table.

4. Define a primary key in the Columns pane, Constraints tab, if there is not already a primary key defined. See Add constraints to tables for details.

> 💡 **Tip:**
>
> To quickly see if there is a primary key defined, check the value of the **Process Type** (not UOW Process Type) attribute in the Tables pane. If there is no primary key, the value is **Staging with Audit**. When you define a primary key, the value changes to **Reload**.

5. Click the ✎ **Modify Table** icon to modify the UOW Processing Type attribute.

6. In the Modify Clinical Data Model Table pop-up, select one of the following values from the **UOW Processing Type** drop-down:

    • **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 USUBJID (Unique Subject ID) SDTM Identifier, and must have a primary key that includes that 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 USUBJID (Unique Subject ID) SDTM Identifier and another with the VISITNUM (Visit Number) SDTM Identifier, and both columns must be included in the primary key.

7. Click **OK**.

Back to Set up clinical data models

Configure table display in the Listings page

You can change the way table columns are displayed in the Default Listings page.

1. Click the **Study Configuration** icon ⚙✎ from the navigation bar. Then click the **Clinical Data Models** tab.

2. Select the model and click **Check Out**.

3. Select the table.

4. Go to the **Display Characteristics** tab. Each column name is displayed. If the column has been marked Not in Use in the transformation that writes to the table, the Not in Use column displays a Y. These columns are not displayed in the Listings pages.

5. Click the ✎**Modify** icon.

6. For each displayed column, specify the following:

    • **Displayed?**: Deselect if you do not want to display the column on the Listings pages.

    • **Display Header**: Enter the column header to display.

    • **Display Hover Text**: Enter hover text to display for the column.

7. To change the column order, click the **Reorder** icon. Select a column and then use the arrows to change its order relative to other columns. Columns at the top are displayed on the left.

Back to Set up clinical data models

Set up data blinding in tables

1. Click the **Study Configuration** icon from the navigation bar. Then click the **Clinical Data Models** tab.

2. Select the model and click **Check Out**.

3. Select the table.

4. Set blinding attributes:

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

   - **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.

       > 💡 **Tip:**
       >
       > Enclose the value in single quotes.

       The default masking values are `xxxxx` for character data, `99999` for numeric data, and `15-AUG-3501` for dates.

       To blind data only in cells that meet certain conditions, specify the conditions; see Specify masking attributes for a column.

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

       Blinding Criteria: Click the **Modify Blinding Criteria** icon to specify which rows should be hidden. See Use the Expression Builder for details.

Back to Set up clinical data models

- Specify masking attributes for a column

Specify 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 be masked.

1. Click the **Study Configuration** icon from the navigation bar. Then click the **Clinical Data Models** tab.

2. Select the model and click **Check Out**.

3. Select the table.

4. Select the column and click the ⬚**Modify** icon. In the Modify Clinical Data Model Column window, select the **Blinding Attributes** tab.

5. 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.

6. **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.

    > 💡 **Tip:**
    >
    > Enclose the value in single quotes.

    The default masking values are: `xxxxx` for text, `99999` for numbers, and `15 aug 3501` for dates.

    • Click the ⬚**Modify Masking Value** icon to create an expression to generate multiple values for the system to display; see Use the Expression Builder for details.

7. **Masking Criteria** (for cell-level masking): Click the ⬚**Modify Masking Criteria** icon to specify the criteria for blinding cells in the column; see Use the Expression Builder for details.

Back to Set up clinical data models

Add Subject Visit and Subject tables

Include a Subject Visit table in at least one clinical data model in each study. Associate one and only one Subject Visit table per study with the SUBJECTVISIT table SDTM identifier. The system uses only the Subject Visit table to support filters.

This is required to support filtering data and discrepancies by subject and visit and to support tracking subject visit completeness using flags. See How subject and visit filters work.

> 💡 **Tip:**
>
> DMW includes an SDTM-compatible Subject Visit table and Subject table that you can copy. Alternatively, you can add an SDTM identifier to an existing table; see Use an existing table as the Subject Visit or Subject table.

• Copy the default Subject Visit or Subject table

- Use an existing table as the Subject Visit or Subject table

Copy the default Subject Visit or Subject table

1. Click the **Study Configuration** icon from the navigation bar. Then click the **Clinical Data Models** tab.

2. Select the model and click **Check Out**.

   If the **Check Out** option is not active, you cannot check out the model because it is already checked out or you do not have the required privileges. To see who checked it out, look at the **Checked Out By** value at the left of the gray values near the top of the page. You may need to click the **>>** icon.

3. In the **Actions** drop-down, select **Copy Subject/Visit Table**.

4. 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.

     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 DMW. The table and columns are already associated with the appropriate SDTM identifiers; see Use SDTM identifiers to support important functionality.

5. Select which tables to copy: **Subject**, **Subject Visit**, or **Both**.

6. Click **OK**.

Back to Set up clinical data models

Use an existing table as the Subject Visit or Subject table

Existing tables must comply with the Subject Visit table requirements or Subject table requirements to be designated as the Subject Visit or Subject table.

1. Click the **Study Configuration** icon from the navigation bar. Then click the **Clinical Data Models** tab.

2. Select the model and click **Check Out**.

3. Select the table and click the **Modify** icon.

4. In the Modify Clinical Data Model Table window, select a value from the **SDTM Identifier** drop-down list:

   - SUBJECT
   - SUBJECTVISIT

5. Click **OK**.

- Subject Visit table requirements
- Subject table requirements

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.

Subject 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.

Back to Set up clinical data models

Map columns to data to be derived from TMS

For details on mapping columns to data to be derived from TMS, see this video:
Video

Prerequisites: The study must be assigned to a TMS dictionary domain and your administrator must have defined a TMS Set for the dictionary.

1. Click the **Study Configuration** icon from the navigation bar. Then click the **Clinical Data Models** tab.
2. Select a target model and click **Check Out**.
3. Select the target table that contains data you want to code in TMS.
4. Click the **TMS** tab, then click the Add icon. The **Add TMS Column Association** window appears.
5. 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.
6. 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.
7. In the **Column Name** field, select the table column to map to each TMS Set derived column that you need in your study. You do not need to map all the TMS Set columns.

   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 map the new columns to the TMS Set derived columns.
8. Click **OK**.

Back to Set up clinical data models

Extract data from a clinical data model

If a model has been set up to support data extraction, you can extract all current data in the model into a file. The following sections describe how to view existing data marts, set up data extraction (by generating a data mart program), and run a data mart (by running the data extract job).

- View existing data marts and their run history
- Generate a data mart program
- Run the data extract job

View existing data marts and their run history

1. Click the **Study Configuration** icon ![icon] from the navigation bar. Then click the **Clinical Data Models** tab.

2. Select the model.

3. Click the **Data Extraction** tab.

4. Select a data mart. Its run history appears in the lower pane.

5. In the **Run History** pane, click the link in the relevant column to:

   - View the output.

   - Download the output.

   - View the log file.

Back to Set up clinical data models

Generate a data mart program

1. Click the **Study Configuration** icon ![icon] from the navigation bar. Then click the **Clinical Data Models** tab.

2. Select the model and click **Check Out**.

3. Click the ![icon] **Modify** icon.

4. Select one or more data mart types to generate.

5. Install the model.

After installation completes, go to the **Data Extraction** tab to run the data mart.

Back to Set up clinical data models

Run the data extract job

1. Click the **Study Configuration** icon ![icon] from the navigation bar. Then click the **Clinical Data Models** tab.

2. Select a model.

3. In the **Data Extraction** tab, select a data mart to run.

4. Click the **Execute** icon.

5. Enter values:

    • **For a text data mart:**

        – **Separator**: Select the character to use to separate column values in the output file.

        – **Enclosing Character**: Select the character to use to separate records in the output file. You can select None.

    • **For an Oracle data mart:**

        – **Compress**: Specifies how Export and Import manage the initial extent for table data.

            If set to **Y**, data is flagged for consolidation into one initial extent upon import. If extent sizes are large (for example, because of the PCTINCREASE parameter), then the allocated space will be larger than the space required to hold the data. This is the default value.

            If set to **N**, the export utility uses the current storage parameters, including the values of initial extent size and next extent size. The values of the parameters may be the values specified in the CREATE TABLE or ALTER TABLE statements or the values modified by the database system. For example, the NEXT extent size value may be modified if the table grows and if the PCTINCREASE parameter is nonzero.

        – **Statistics**: Select the type of database optimizer statistics to generate when the exported data is imported. Options are Estimate, Compute, and None. The default value is None.

6. Select the appropriate **Blind Break** option.

    • **Not Applicable**: None of the tables are blinded. All data will be included.

    • **Real (Blind Break)**: If selected, the real, blinded data will be included and the action will be audited. This option is available only if you have special privileges.

    > **✏ Note:**
    >
    > In this release it is not possible to extract masking values. If any data is blinded in the model, you can only extract it if you perform an audited blind break, extracting the real, sensitive data.

    • **Real (Unblinded)**: If available, the real data has been unblinded. If selected, the real, unblinded data will be included. This option is available only if you have special privileges.

7. Select an execution type:

    • **Immediate** to submit the job immediately.

    • **Deferred** to schedule a single execution. Click the Start Date icon and select a date and time.

    • **Scheduled** to create a regular execution schedule. Select a start and end date and time and a frequency number and unit (hours, days, weeks, or months).

For example, with a frequency unit of months and a frequency of 1, the job will run once a month from the specified start date and time to the specified end date and time.

Back to Set up clinical data models

Modify a clinical data model

You can modify an Inform or non-Inform clinical data model. (To add or modify a table manually, see Add or modify a table manually.)

- Modify a non-InForm model
- Modify an InForm input model
- Upgrade a clinical data model to the latest library version
- Roll back changes to the last production version

Modify a non-InForm model

1. Go to the model in the Study Configuration or Library page.

2. Select the model and click **Check Out**.

> 💡 **Tip:**
>
> To modify a target data model, do not check it out directly. Instead, check out the transformation that writes to it, which also checks out the model. Then return to the model and make your changes, which are then synchronized with the transformation.

3. Click the ✎ **Modify** icon in the Data Model pane on the left. In the Description field, describe the changes you are making.

4. 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 Add tables.

     > ✎ **Note:**
     >
     > You can add tables or modify existing ones using a metadata file, but you cannot remove tables using a file.

   - **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 Add columns to a table manually.

   - **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 Add constraints to tables for details.

   - **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** Select this option from the **Actions** drop-down list to change the validation status or to upload a supporting document for the validation status change; see Validation status and lifecycle stages. The system displays this button only if you have the privileges required.

5. **Install** the model. The system uses the old version until the new one is installed. 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.

Back to Set up clinical data models

Modify an InForm input model

The InForm model must have exactly the same table structures as in InForm. You cannot make any structural changes.

> **💡 Tip:**
>
> You can change attributes that apply only in DMW, but if metadata is reloaded from InForm either manually or automatically, your manual changes are lost. See Load InForm metadata for more information.

1. Click the **Study Configuration** icon from the navigation bar. Then click the **Clinical Data Models** tab.

2. Select the model and the Development lifecycle.

3. In the **InForm Configuration** tab, click the **Suspend Data Loading** icon for the Development lifecycle.

4. **Check out** the model.

5. Make your changes. See:

   - Add constraints to tables

   - Set up data blinding in tables

   - Set up Unit of Work data processing for tables

   - Add or modify a table manually for information on adding tables aliases and SDTM identifiers. But do not add or remove tables.

   - Add columns to a table manually for information on adding column aliases, codelists, and SDTM identifiers. But do not add or remove columns.

6. **Install** the model. The system uses the old version until the new one is installed. 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.

   Back to Set up clinical data models

Upgrade 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.

Roll back changes to the last production version

1. Click the **Study Configuration** icon ⚙ from the navigation bar. Then click the **Clinical Data Models** tab.

2. Click the ↺ **Roll Back Clinical Data Models to Production Version** icon in the Data Model pane on the left of the Clinical Data Model page. A window appears.

3. Select one or more clinical data models to roll back.

> ✎ **Note:**
>
> InForm models cannot be rolled back.

4. Click **Rollback**.

5. Click the ↻ **Refresh** icon periodically to see the Job ID, Log, and Job Status.

Install a clinical data model

After creating or modifying a clinical data model, you must *install* it to make it usable. See What happens during installation? and Installation requirements.

To install a clinical data model:

1. Click the **Study Configuration** icon ⚙ from the navigation bar. Then click the **Clinical Data Models** tab.

2. Select the lifecycle in which you want to install the model.

3. If it is an input model, click the ⏸ Suspend Data Load icon.

4. Select the model and click **Check Out**.

5. Select one option from the Install drop-down:

    • **Install** upgrades all tables and does not delete any data.

    • **Full Install** drops and replaces all tables, **deleting all data**. Full installation is not available in the Production lifecycle stage.

> ✎ **Note:**
>
> These options are available only if the object is installable:
>
> – The Version and the Installable Version must be the same.
>
> – See Installation requirements.

6. To see the updated job status in the **Install Status** field, click the ↻ **Refresh** icon. The final status doesn't appear until the job completes and you click **Refresh**.

7. To see the log file, select **Get Install Log** from the **Actions** drop-down list.

Back to Set up clinical data models

Unblind and reblind data

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

A table can be reblinded after being unblinded. Both actions are audited and can be viewed.

The Unblind and Reblind actions are audited.

1. Click the **Study Configuration** icon ⚙ from the navigation bar. Then click the **Clinical Data Models** tab.

2. Go to the Data Models page and select the data model, then the table to unblind.

3. Select **Unblind** from the **Actions** drop-down list.

    If the table is currently unblinded, the **Reblind** option appears instead.

FAQs

• What happens the first time I run the Load Metadata job?

• What happens if there's a protocol change?

• How do InForm and DMW exchange data?

• When I copy a table, sometimes validation checks and custom listings are included in the copy and sometimes not included. Why?

• Can I use Oracle Thesaurus Management System in all three lifecycle stages?

• If I create a new version of a clinical data model in the Development lifecycle, can I revert to the previous Production version?

- Is it possible to get back data that has been deleted?
- Why am I getting error "ORA-00955: name is already used by an existing object"?
- What happens the first time I run the Load Metadata job?
- What happens if there's a protocol change?
- How do InForm and DMW exchange data?
- When I copy a table, sometimes validation checks and custom listings are included in the copy and sometimes not included. Why?
- Can I use Oracle Thesaurus Management System in all three lifecycle stages?
- If I create a new version of a clinical data model in the Development lifecycle, can I revert to the previous Production version?
- Is it possible to get back data that has been deleted?
- Why am I getting error "ORA-00955: name is already used by an existing object"?

What happens the first time I run the Load Metadata job?

The Load Metadata job creates one table for each view in the InForm reporting database that is registered in the RD_DATADICTIONARY table and installs the model. The tables have the same structure as the InForm views plus SKEY columns required in DMW.

What happens if there's a protocol change?

When metadata changes in InForm, DMW detects it:

- During data loading.
- When you promote a model to a higher validation status.
- When you save changes to a remote location or study account.

The load, promotion, or save is not allowed and a message appears that you need to reload metadata.

You can also run the Compare Metadata report by clicking its icon in the InForm configuration tab, and then import metadata.

How do InForm and DMW exchange data?

Data and metadata are imported from the study's InForm reporting database to DMW using a database connection.

Discrepancies are sent almost immediately to InForm as queries, using a web service.

When I copy a table, sometimes validation checks and custom listings are included in the copy and sometimes not included. Why?

When you copy a table, the system checks if you have included all source tables for the validation checks and custom listings in the copy operation. If you have, the validation checks and custom listings are also copied.

If you have not included all the source tables:

- If a validation check is in an ordered batch, you receive a warning.

- For validation checks that are in an unordered batch, and for any public custom listing, the system does not give a warning and does not copy the validation check or custom listing.

If you need the validation check or custom listing, you can copy it separately.

Can I use Oracle Thesaurus Management System in all three lifecycle stages?

In Development and QC you can send data to TMS, derive data for terms that can be automatically coded, and create DMW discrepancies for terms that cannot be automatically coded. These discrepancies will close only if the data is updated in such a way that it can be coded automatically.

Only Production has complete coding functionality. In addition to deriving data:

- TMS creates *omissions* for terms that cannot be automatically coded. TMS users can code these terms manually.

- DMW creates discrepancies that correspond to the omissions and sends them to InForm. You can export discrepancies on lab data to Excel and send the spreadsheet to the lab.

If I create a new version of a clinical data model in the Development lifecycle, can I revert to the previous Production version?

Yes. If, for example, a protocol amendment is canceled after you created a new version of the clinical data model in the Development lifecycle, you can undo all changes and revert to the version being used in the Production lifecycle.

For InForm models, the rolled-back Development model is installed as part of the process. Non-InForm models are not installed during the rollback. The Production version is checked out and made the current Development version.

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.

Is it possible to get back data that has been deleted?

Deleted data is **no longer available in the system**. However, it remains in the database with an end timestamp equal to the date and time of the job when it was deleted.

Why am I getting error "ORA-00955: name is already used by an existing object"?

If you perform an upgrade installation on a clinical data model after modifying a table's primary key (PK) or unique index, you may get the above error from Oracle Warehouse Builder (OWB).

This is because when you create a primary key on a table, the system generates a unique index on the PK columns and maintains a connection between the PK and

the uinque index. If you drop the primary key, the system drops the unique index. However, if you have manually created a unique index on the same columns, the system does not drop the index and the PK and index are no longer connected.

If you subsequently modify a PK column or manually create a unique index, the system generates and runs reorg.sql with the following steps:

1. Drops the PK.

2. Alters the column (or whatever change you have made).

3. Recreates the unique index on the PK columns.

4. Recreates the PK.

This works, but the connection between the primary key and unique index is lost. If you make another manual change, reorg.sql fails with the above error message because it does not drop the unique index in Step 1 due to the disconnect, but does try to create the unique index in Step 3, causing the error.

If you do a Full installation, the system recreates the primary key, the unique index, and a connection between them. However, it also deletes all data in the model's Development lifecycle stage.

3

Create libraries of codelists and clinical data models

Each study group (for example, therapeutic area or project) has a library where you can create standard clinical data models and codelists. These models and codelists are available for use in all studies, but you can use the library to store models and codelists that are appropriate for each study group.

If you want a library of models and codelists that are suitable for use in all studies, your administrator can create a "study group" for that purpose.

- Create a codelist
- Create a library clinical data model

Create a codelist

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

1. Click the Navigation icon [icon] at the top of any page and then click **Library**.

2. Select the **Codelists** tab.

3. Select a project (or whatever study groupings are called in your company) in which to create the codelist.

   > **Tip:**
   >
   > Selecting a grouping is required and can help users find the codelist. But codelists in any grouping can be used in any study.

4. Click the [+] **Add** or [pencil] **Modify** icon. (If you are modifying the codelist, you must first check it out.)

   > **Tip:**
   >
   > Modifying a codelist creates a new version of it. When you assign the codelist to a column, the column uses the current version. Columns that already use the old version will continue to do so unless you explicitly modify the column.
   >
   > - If you want to change all columns to the new version, modify the codelist.
   >
   > - If you want to be able to assign either version to a column, create a whole new codelist with the modified list of values.

5. **Codelist Name** Enter a name.

6. **Description**: (Optional) Enter a description such as its values to help users decide if this is the codelist they need.

7. Click **OK**.

- Add code values
- Remove a codelist

Add code values

1. Select the codelist. In the Codelist Values tab, click the ➕ **Add** or ✏ **Modify Value** icon and enter:

2. **Code**: The allowed clinical data value.

3. **Code Value**: This is the value displayed in the user interface. The system does not evaluate this value.

4. Click **OK**.

5. Repeat for each value.

6. Click **Check In** to make the codelist available for use.

Remove a codelist

1. Click the Navigation icon 🖥 at the top of any page and then click **Library**.

2. Select the **Codelists** tab.

3. Select a project (or whatever study categories are called in your company).

4. Select the codelist.

5. Click **Check Out** if it is not already checked out.

6. Click the ✖ **Remove** icon.

> **✏ Note:**
>
> You cannot remove a codelist if it is associated with a table column.

Create a library clinical data model

You can maintain a library of standard clinical data models and create study clinical data models based on and linked to a library model. When you update the library model, the system displays an Upgrade button in the study model and you can click it to update the study model to the new library version. Any changes you have made to the study model are lost.

- Create a library clinical data model
- Modify a library clinical data model

- Delete a library clinical data model

Create a library clinical data model

1. Click the Navigation icon [icon] at the top of any page and then click **Library**.

2. In the Library Data Models tab, select a study category (Therapeutic Area, Project, or whatever study categories are called in your company) from the drop-down list.

   The clinical data model will be available to studies in this category.

3. Click the [+] **Add** icon. The Add Library Data Model window appears.

4. Enter a name and description for the model. See Naming restrictions for restrictions.

5. Under **Select from source** select one of the following:

   - **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 and blinding attributes set; see Required syntax for metadata files.
     Browse to the file.

6. Click **OK**.

   > **Note:**
   >
   > You must create a primary key for each table. Follow instructions in Add constraints to tables except work in the Library page.

7. If you are creating the tables manually, follow instructions in Add tables except work in the Library page. Complete the table specifications manually:

   - Each table must have a primary key. Follow instructions in Add constraints to tables .

   - Associate SDTM variables with columns whenever possible to enable full functionality in DMW. Follow instructions in Use SDTM identifiers to support important functionality except work in the Library page.

   - Follow instructions in Set up data blinding in tables except work in the Library page.

   - Follow instructions in Set up Unit of Work data processing for tables except work in the Library page.

   - If you are using Oracle Thesaurus Management System (TMS) as your coding system, see Map columns to data to be derived from TMS.

8. 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.

9. Check in the model by clicking **Check In** near the top of the page.

10. To test the library model, navigate to a study in the Study Configuration page and create a study model from the library model, then install it. Check the install log file. If there are any problems, fix them in the library model, upgrade the study model, reinstall, and check the log file again.

Modify a library clinical data model

1. Click the Navigation icon ![icon] at the top of any page and then click **Library**.

2. Select the project (or whatever study categories are called in your company) that includes the model.

3. Under **Filter Library Data Models:**, enter part or all of the model name and press Enter. The system lists all models whose name contains the string you typed.

4. Select the model you want, either by clicking it or by using the Down arrow and then pressing Enter.

5. Click **Check Out** near the top of the page. The system creates a new version of the model.

6. Make changes. If you have the required privileges, you can add, modify, or delete:

   > **✎ Note:**
   >
   > Follow all instructions except navigating to Study Configuration. You can modify these clinical data models in the library.

   - **Tables** by clicking the appropriate icon and making the changes. Follow instructions in Add tables except work in the Library page.

   - **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. Follow instructions in Add columns to a table manually except work in the Library page.

   - **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. Follow instructions in Add constraints to tables except work in the Library page. A primary key is required.

   - **Blinding attribute values**: Follow instructions in Set up data blinding in tables except work in the Library page.

   - **Data Processing attribute value**: Follow instructions in Set up Unit of Work data processing for tables except work in the Library page.

   - **SDTM variable associations**: Follow instructions in Use SDTM identifiers to support important functionality except work in the Library page.

   - **Validation status**: You can upgrade the validation status of a library model to indicate that it is ready for use, but it has no effect.

7. 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.

8. Check in the model by clicking **Check In** near the top of the page.

9. To test the library model, navigate to a study in the Study Configuration page and create a study model from the library model, then install it. Check the install log file. If there are any problems, fix them in the library model, upgrade the study model, reinstall, and check the log file again.

10. To upgrade a study model to the new version of the library model, check out the study model and click the **Upgrade from Library Model** icon.

Delete a library clinical data model

To delete a clinical data model, select it in the Library Data Model pane and You must have the Remove Model privilege.

1. Click the Navigation icon ⌨ at the top of any page and then click **Library**.

2. Select the project (or whatever study categories are called in your company) that includes the model.

3. Under **Filter Library Data Models:**, enter part or all of the model name and press Enter. The system lists all models whose name contains the string you typed.

4. Select a model, either by clicking it or by using the Down arrow and then pressing Enter.

5. Click **Check Out** if it is not already checked out.

6. Click the ✖ **Delete Clinical Data Model** icon.

> **Note:**
>
> It is possible to delete a library model that has been used in a study. This has no effect on the study model.

4
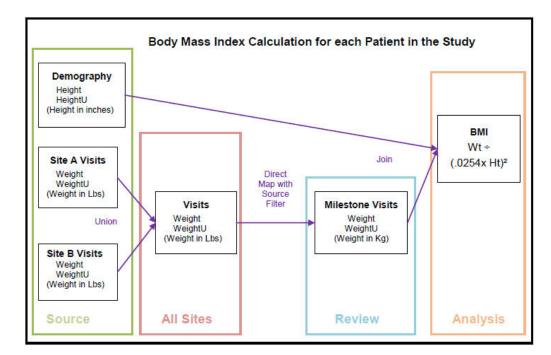
Set up data transformations

This section describes how to map at the following levels:

- Clinical data model

- Table

- Column

It also describes how to cascade blinding and masking, install a transformation, and run transformations and view history. You can also learn how to set automatic triggering of transformations and validation checks, upgrade transformations to synchronize with models, and view validation status and lifecycle stages.

The following diagram shows consecutive clinical data models Source, All Sites, Review, and Analysis. Subjects' weight is collected at each visit and mapped from the source model to each subsequent model, merging data from all sites in a union to the All Sites model, and converting the units from pounds to kilos in the Review model. Subjects' height is collected during the initial visit. To calculate the BMI in the Analysis model, Height is mapped from the source model and Weight is mapped from the Review model. The calculation is performed as an expression on the target column.

Figure 4-1   Transformation Example: BMI Calculation

- Map at the clinical data model level

- Use side transformations

Map at the clinical data model level

For details on mapping at the clinical data model level, see this video: ▶ Video

1. Click the **Study Configuration** icon ⚙ from the navigation bar. Then click the **Transformations** tab.

2. In the Data Models pane on the left, select the clinical data model the transformation will write data into.

   The system displays its tables in both the Target Tables and Source Tables panes so that you can use the target tables as a source for side models.

3. Click the ▦ **Add or Remove Source Model** icon in the Source Tables pane.

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

   > 💡 **Tip:**
   >
   > Reference models are listed after the study models. See Set up reference data.

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

   c. Click **Save**.

   > 💡 **Tip:**
   >
   > If many people are working on the same transformation, Use side transformations.

   Next: Map at the table level.

> **✎ Note:**
>
> A tilde (~) displayed next to an input model means the model was deleted while the transformation was checked in. When you check out the transformation, the deleted model is no longer displayed at all.

Use side transformations

To allow many people to work on a transformation at the same time, each person can create a *side model* with one target table to work on in a corresponding *side transformation*.

- Create a side transformation
- Merge a side transformation with the main transformation

Create a side transformation

Work on one side transformation at a time, then merge it into the main transformation, and delete the side model.

1. Check out the transformation if it is not already checked out.

2. In the Target Tables pane of the Transformation tab, select the target table(s) you want to work on. Use Ctrl+click or Shift+click to select multiple tables.

> **✎ Tips:**
>
> A target table can be in only one side model at a time. An icon next to the table name indicates a table already in a side model. Hover over the icon to see who is working on it.
> You can select a table marked Not Used. It is marked Used when you merge back to the main transformation.

3. Click the ⚘ **Create Side Model** icon. You may need to click the **>>** icon to see it. It is not enabled until you select a table.

4. Enter a name for the side model. Do **not** use the string `$TEMP`.

5. Click **OK**.

6. Select one or more source tables, then select the target table, and then click the 🗒 **Map** icon in the target table's row.

   You can Use Automap before or after mapping manually.

7. Select a **Transformation Type**: Direct, Join, Union, Pivot, Unpivot, or Custom.
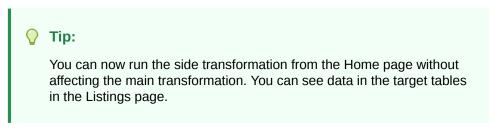
> **Tip:**
>
> Even if you are using a custom program, select the actual type of transformation your program performs: Direct, Join, Union, Pivot, or Unpivot. Use Custom only if your program cannot support data lineage tracing.

8. Map the target table and columns; see Map at the table level and Map at the column level.

9. (Optional) Cascade blinding or define tables and columns as Not Used. When the side transformation is merged back into the main transformation, the blinding and Not Used attributes are applied to the tables in the main transformation.

   Any validation checks or custom listings dependent on tables or columns marked as Not Used are disabled when the side transformation is merged with the main transformation.

10. **Check In** the side model.

> **Tip:**
>
> You can now run the side transformation from the Home page without affecting the main transformation. You can see data in the target tables in the Listings page.

Next: Merge a side transformation with the main transformation.

Merge a side transformation with the main transformation

When you merge the side model with the main model, any blinding and Not Used values are merged with the target model.

1. Go to the main transformation for the target model.

2. Check out the main transformation.

> **Tip:**
>
> If another user has the main transformation checked out, you cannot merge the side model into it. Either wait until you can check out the main transformation or ask the person who has checked it out to merge your side model.

3. Select **Copy from Side Model** from the **Actions** drop-down list.

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. Select the side model and click the **Delete Side Model** icon. You get a warning if it has not been merged with the main transformation.

Map at the table level

For details on mapping at the table level, see this video: ▶ Video

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.

You can work in many ways:

- Copy table transformations
- Use Automap
- Map tables manually
- Mark target tables and columns as Not Used
- Create a self-join
- Authorize target table data for nonprivileged users
- Add expressions on mapped sources
- View source code
- Validate mappings
- Unmap tables

Copy table transformations

If a transformation between the same or very similar source and target tables exists in a different study, you can copy it and do most of the mapping automatically.

1. Select target tables that are the same or similar to the target table mappings you plan to copy.

2. Select **Copy from Another Transformation** from the **Actions** drop-down list.

3. Specify the transformation to copy: Select a project (or other study grouping) from the drop-down list, then select a study, then the model that is the target of the transformation to be copied.

> 💡 **Tip:**
>
> When a transformation is copied from an external transformation or as part of applying a study template, authorized tables are always copied into the new transformation as Not Authorized. If any of the source tables are blinded, the target table must be either blinded or re-*authorized* if it does not contain blinded data.

4. Click **Next**. The system validates the Copy and displays details including 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 I noticed the tables in the target model changed. Why?.

- 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 I noticed the tables in the target model changed. Why?.

- 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.

You can't copy transformations with errors, but you can copy those with warnings.

5. Select mappings to copy, and click **Accept Selected Mappings**.

> **Tip:**
>
> If the source and target tables in the copied transformation have more columns than the tables in the current model, mark those columns Not Used.

Use Automap

The Automap process searches for tables in source models to map to tables in the target model. It creates only direct 1:1 mappings. It does not change existing manual mappings. After running it, you select the mappings you want to use.

For details on the automap process, see this video: ▶ Video and this: ▶ Video

1. After you select a study, click the **Study Configuration** icon ⚙ from the navigation bar to open the Clinical Data Model tab.

2. Click the **Transformation** tab to view the source and target tables.

3. Choose a target data model from the Data Model list in the left panel.

4. Do one of the following:

   - To map all the tables and columns in the data model, click **Map** > **Automap** from the Transformation tab title bar.

   - To map tables and columns you select, select one or more target tables and click **Map** > **Automap Selected Tables** from the option in Target Tables.

   > **Tip:**
   >
   > For the best performance, select 10 tables or less.

5. Wait for the mapping process to complete on the data model. (If you need to work on other data models while the mapping runs on the selected model, you can.) During the mapping process, you see a red status message in the Transformation tab title bar (next to the Check In button). For example, you see an "Automapping in progress" or "Automapping is complete" message in red.

    You also see a notification message open from the notification bubble in the title bar. When done, you see this message: "Automapping suggestions require review and approval."

6. To review and accept the maps, see Accept or reject suggested mappings.

   For more details on the Automap feature, see FAQ How does Automap work?

   • Accept or reject suggested mappings

Accept or reject suggested mappings

Once the automap process completes, a notification opens with: "Automapping suggestions require review and approval." You can accept or reject the suggested mappings. (For more details selecting tables to automap, see Use Automap.)

1. Review all table and column mappings.

   The Type column shows the type of logic used to find the source column: Name Match, Alias Match, Datatype Match, or Partial Name/Alias Match.

2. Clear any maps you do not want.

3. To accept the maps, click **Map** > **Review and Accept maps** from the Transformation tab title bar.

Map tables manually

First, Map at the clinical data model level.

To map at the table level:

1. Select one or more source tables.

2. Select one target table.

3. Click the ⊞ **Map** icon in the target table's row.

4. If you are using a custom program, select the program.

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

   > 💡 **Tip:**
   >
   > Even if you are using a custom program, select the type of transformation your program performs. Use Custom only if your program cannot support data lineage tracing.

6. If you specify a Join, Pivot, or Unpivot, the system displays an icon. Click the icon and supply details.

> 💡 **Tip:**
>
> Keep the primary key intact in the target table. Include SubjID in the primary key.

7. **Authorize**: When one or more source tables contains blinded data, by default the target table is completely blinded and only users with Blind Break privileges can view any data in the table.

    *   To allow users without Blind Break privileges to see some of the table's data but keep other data hidden, modify the target table to mask values in the appropriate columns, rows, or cells. Do this in the Clinical Data Model page; see Set up data blinding in tables for details.

    *   If you are certain that the target table will contain only nonblinded data, select the checkbox in the **Authorize** column to allow users without Blind Break privileges to view all data in the target table. This authorization is audited.

8. **Save**. You must save before mapping columns.

Next: Map at the column level.

Mark target tables and columns as Not Used

See What happens when I mark tables and columns as Not Used? for more information.

*   Explicitly mark as Not Used
*   Cascade Not Used from source to target
*   Mark mapping as Complete

Explicitly mark as Not Used

1. After you select a study, click the **Study Configuration** icon ⚙️ from the navigation bar to open the Clinical Data Model tab.

2. Click the **Transformation** tab to view the source and target tables.

3. Select a single table or column.

4. Select **Mark as Not Used** from the **Mark** drop-down in the table or column pane.

Cascade Not Used from source to target

If the source model contains columns and tables not used, you can extend the Not Used status in the source through to the target tables so they do not appear in the transformation.

1. After you select a study, click the **Study Configuration** icon ⚙️ from the navigation bar to open the Clinical Data Model tab.

2. Click the **Transformation** tab to view the source and target tables.

3. Check if the source has tables or columns not used.

4. Select **Cascade Not Used** from the **Actions** menu in the title bar of the Transformation tab.

Mark mapping as Complete

To mark **all remaining unmapped columns in a single table** as Not Used:

1. Go to the Column Mapping page for a single table.

2. Select **Mark as Complete** from the **Mark** drop-down.

To mark **all remaining unmapped tables** as Not Used:

1. Go to the main transformation page.

2. Select **Mark as Complete** from the **Actions** drop-down in the title bar of the tab.

Create a self-join

1. Select the table in the Source Tables pane and the Target Tables pane and click the **Map** icon.

2. Enter an alias for each occurrence and save.

3. Select a transformation type of Join for the target table.

4. Click the **Join** icon and define the join. See Join.

5. Click the **Map Column** icon for the target table. See Map at the column level.

Authorize target table data for nonprivileged users

If you are certain that one or all target tables do not contain any unmasked blinded data, so users without special blinding privileges should be able to view all data in the table, *authorize* the table(s).

> **Tip:**
>
> Before authorizing, modify all target tables that do contain data that should be hidden by masking values in the appropriate columns, rows, or cells. Do this in the Clinical Data Model page. See Set up data blinding in tables .

To authorize a single table:

1. Select the table.

2. Select its **Authorize** checkbox.

3. Save the transformation.

To authorize all target tables:

1. Select **Authorize Target Tables** from the **Actions** drop-down.

2. Save the transformation.

> 💡 **Tip:**
>
> You cannot *undo* authorization for all tables at once.

To undo authorization:

1. Select the table.

2. Deselect its **Authorize** checkbox.

3. Save the transformation.

For more information, see Data blinding and authorization.

Add expressions on mapped sources

To generate a WHERE clause to filter by column value on the source table(s), restricting the set of source data that participates in the transformation:

1. Select the target table.

2. If needed, add an alias for the source table.

> 📝 **Note:**
>
> The Logical Name field shows alternate names for the table defined in the data model. These values are used in automapping.

3. In the Mapped Sources tab, click the icon in the **Source Filters** column and write an expression, either as free text or using the Expression Builder. See Use the Expression Builder for details.

> 📝 **Important:**
>
> - If you reference a static package or function in free text, you must select it in the **Selected Packages tab**.
>
> - In free text, **use just the column name**, not the table.column format, **unless** you need to use an alias, as in a self-join. In that case the alias.column format is required.

4. If you reference a static package in free text, you must select it in the Selected Packages tab.

View source code

To view generated PL/SQL source code for a table mapping:

1. Select the table.

2. From the **Actions** drop-down, select **View Source**.

Validate mappings

To validate selected mappings:

1. Select one or more target tables.

2. From the Map drop-down in the Target Tables pane, select **Validate Mappings**.

To validate all mappings in the transformation:

- From the Map drop-down at the top of the page, select **Validate Mappings**.

To see error messages, hover over the table or column's mapping status icon. See What does this transformation validation error message mean? for more information.

Unmap tables

1. Select one or more target tables.

2. From the Target Tables **Map** drop-down, select **Unmap**.

Map at the column level

For details on mapping at the column level, see this video: ▶ Video

> **✎ Note:**
>
> Make sure you mapped at the table level before you map at the column level, Map at the table level.

1. In the Target Tables pane, select the target table, then click the **Map Column** icon.

2. (Optional) Select one or more target columns, then select **Automap Selected Columns** from the Map drop-down. The system suggests mappings to matching source columns, if any.

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.

4. To write an expression operating on the target column data as part of the Select clause, do **one** of the following:

   - Enter free text either in the **Expressions** field, using normal SQL syntax (without curly brackets).

> **✎ Important:**
>
> – If you reference a static package or function in free text, you must select it in the **Selected Packages tab**.
>
> – In free text, **use just the column name**, not the table.column format, **unless** you need to use an alias, as in a self-join. In that case the alias.column format is required.
>
> – Create table aliases in the Mapped Sources tab to use in free text expressions.

- Click the ✎**Add or Modify Expression** icon to Use the Expression Builder that generates code using curly brackets.

> **✎ Important:**
>
> Only one syntax style can be supported in a single table mapping.

5. 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.

6. Continue until all columns are either mapped or marked as Not Used.

7. **Save**.

8. From the **Install** drop-down, select either Install or Full Install. See Install a transformation.

To run the transformation, go to the Home page.

Cascade blinding and masking

You can apply the blinding settings of source tables and columns to the target tables and columns they are mapped to using the Cascade Blinding feature for tables and the Cascade Masking feature for columns.

Use the **View** drop-down to display columns in the Source Tables or Columns pane with blinding-related information:

You must complete all the mappings first.

- Cascade blinding to downstream tables
- Cascade masking to downstream columns

Cascade blinding to downstream tables

1. In the Table Mapping pane, click **Cascade Blinding** from the **Actions** drop-down list.

The Cascade Blinded Tables window displays all blinded source tables, the target table they are mapped to, and the type of blinding.

- • If a source table has *column-level* blinding, you can select or deselect individual columns for cascade blinding.

- • If a source table has *row-level* blinding, you must enter blinding criteria for the target table in the Clinical Data Model page. If you don't, the whole target table is blinded.

2. Accept the cascade operation. Either:

- • Select a target table and click **Accept**.

- • Check **Select All** at the top of the window and click **Accept**.

Cascade masking to downstream columns

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

1. In the Column Mapping pane, click **Cascade Masking**.

The Cascade Blinded Columns window displays all blinded source columns, the target column they are mapped to, and the type of blinding.

If a source column has *cell-level* masking, the target column will have *column-level* masking until you specify masking criteria in the Clinical Data Models page.

2. Accept the cascade operation. Either:

- • Select a target table and click **Accept**.

- • Check **Select All** at the top of the window and click **Accept**.

Table transformation types

The system generates program code based on your specifications for the following transformation types (except Custom):

> 📝 **Note:**
>
> For more information on joins, unions, pivots, and unpivots, see *Oracle® Database SQL Language Reference 11g Release 2 (11.2)*.

- • Direct
- • Join
- • Union
- • Pivot
- • Unpivot
- • Custom

Direct

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

1. Select the source and target tables, then select a Transformation Type of **Direct**.

2. Click the **Map Column** icon.

3. For each target column **either**:

   • Select the source column(s) and click the **Map** icon.

   • Click the **Mark as Not Used** icon.

4. Save.

Join

For details on creating a join, see this video: ▶ Video

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.

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.

3. Click the ✚**Add** icon.

4. Select Table 1 (left) and Table 2 (right, also known as the foreign key table) to be joined.

5. Specify if it is to be an outer join on either the left or right side. Leave the checkbox unselected to create an inner join.

6. To join more than two tables, click the ✚**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.

8. Create the join condition for the Where clause by selecting a column from each table and the operator required.

   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**.

10. Click the **Map Column** icon.

11. For each unmapped target column **either**:

    • Select the source column(s) and click the **Map** icon.

    • Click the **Mark as Not Used** icon.

12. Save.

• Join Types

Join Types

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.

Union

For details on creating a union, see this video: ▶ Video

All the data in two or more tables is fed into a single target table that is a superset of all columns in the sources. The source tables should have some overlapping content, such as two versions of a Vital Signs form or two lab vendors providing results for the same subject visit.

1. Select the source tables and target table and click the **Map** icon

2. Select **Union** from the **Type** drop-down.

3. Click the **Map Column** icon.

4. Select each target column in turn and do one of the following:

   - Select the source column(s) and click **Map**.

     > ✏️ **Note:**
     >
     > 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.

5. Save.

Pivot

For details on creating a pivot, see this video: ▶ Video

A pivot converts a source table with a tall, skinny structure of few columns and many rows, 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, based on the value in a pivot column, which must be associated with a codelist.

> **Tip:**
>
> For pivots to work correctly the source table must have a primary key that includes the pivot column plus the **minimum** number of columns required to ensure that each record is unique.
>
> If there are more columns in the key than that, the resulting table may have too many rows with target column values sparsely populated among them rather than having fewer rows with a value in each column.

1. Select the source and target tables and click the **Map** icon, then select a Transformation Type of **Pivot**.

2. Click the **Pivot** icon.

3. Query for and select the pivot column. It must be associated with a codelist.

4. Click **OK**.

5. Click the **Map Column** icon.

6. In the **View** drop-down, select **Columns** and then select **Filter Value**.

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.

8. Save.

Pivots and InForm Repeating Itemsets: In InForm, the primary key includes an internal index column called itemsetindex. In DMW, this internal column must be removed and replaced with the pivot column, which is associated with a codelist. To do this, create an intermediate direct transformation to remove itemsetindex and any other unneeded internal columns and add the pivot column to the primary key. Use the resulting table as the source table for the pivot transformation.

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 4-1   Source table in pivot example

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 codelist 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 4-2   Target Table in Pivot Example

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

For details on creating an unpivot, see this video: ▶ Video

An unpivot converts a source table with a short, fat structure of many columns and few rows into a target table that represents the same data in a tall, skinny structure with more rows and fewer columns, based on the value in an unpivot column, which must be associated with a codelist. Unpivot transformations are used for tables where multiple columns collect the same data, such as the same assessment repeated in each section of a CRF.

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.

3. Query for and select the unpivot column from the target table columns. It must be associated with a codelist.

4. Click **OK**.

5. Click the **Map Column** icon.

6. For each pivoted column, select the target column and all the source columns that will feed data to it, and click **Map**. When appropriate, you can copy a mapping with its filter value. See Step 8 for more information.

   You can use the following pseudo-expressions without quotes:

   • To populate a target column with codelist values, add the following pseudo-expression to the column mapping expression:

   `$UNPIVOT$CODEVALUE`

   > ✏️ **Note:**
   >
   > A column that uses the pseudo-expression $UNPIVOT$CODEVALUE cannot be used as a pivot column.

   • By default, rows with null values for all target columns are suppressed. To include them in the target table, add the following pseudo-expression to the unpivot column mapping expression:

   `$UNPIVOT$INCLUDENULLS`

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.

8. If you have multiple target columns that map to the same source columns and have the same filter value, you can map one of them, then select it and click the **Copy Map To** icon. Select the target column and click **OK**.

9. 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 a blood draw. 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.

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 codelist containing the 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 4-3   Source table columns in unpivot example (short, fat table)

Sub jID	Date	Visit	Sect	Sect 1_Te st	Sect1 _Unit	Sect1_ Value	Sect	Sect 2_Te st	Sect2 _Unit	Sect2_ Value	Sect	Sect3 _Test	Sect3 _Unit	Sect3_ Value
509	0108211 2	1	0hr	Hb	gl	8	1hr	Hb	gl	8	2hr	Hb	gl	9
598	0209211 2	1	0hr	Hb	gl	8	1hr	Hb	gl	9	2hr	Hb	gl	10
613	0211211 2	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 codelist per set of nonpivoted values (SubjID, Date, and Visit) and populates the Section column in the target table with the codelist values as shown in Table 4-4.

Table 4-4   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

Table 4-4   (Cont.) Target table columns in unpivot example (tall, skinny table)

SubjID	Date	Visit	Section	Test	Unit	Value
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 items in the source model that contributed to each data item in the target model.

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. See How the system tracks data lineage.

Use a custom program

For details on creating a transformation using a custom program, see this video:
Video

If the logic you need for a table mapping is too complex to do in the user interface for generated transformations, use a custom program. For example:

- To perform data aggregation, case statements, or complex calculations.

- To use 1-to-many or many-to-many column mappings.

- 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 DMW, and see "Sample Program that Calls the API to Set a Flag" on page 6-9.

> **Tip:**
>
> If you use multiple custom programs with the same name for different target tables in the transformation, the programs must all be in the same location. In other words, they must be the same program. These conflicts can arise if the custom program is part of a template and therefore part of the study created from the template, for example.
>
> If you reference two custom programs with the same name in different locations, installation ends with a warning and the conflicting packages are listed in the installation log file. The validation process, which must be run before installation, marks all mappings with conflicting package names as Invalid in the UI, and gives the full path for each. You must resolve the conflicts and reinstall the transformation before execution can succeed.

You must create custom programs in LSH. See Create custom programs.

1. Select the source and target tables and click the **Map** icon.

2. In the **Type** column, select the type of transformation that the custom program actually performs, if possible: **Direct**, **Join**, **Union**, **Pivot**, or **Unpivot**.

   Select **Custom** *only if* the code does not do one of these. *Data lineage tracing does not work if you select Custom.*

3. Click the **Select Custom Program** icon. Search for the program you want. Use the Query By Example fields above the columns. If they are not displayed, click the **Query By Example** icon.

4. Select a program and click **OK**.

   > 💡 **Tip:**
   >
   > You can open the Static Packages tab and use the drop-down to display either *all* packages, only those that are *already selected* in the transformation, or all *except* those already selected.

Install a transformation

After creating or modifying a transformation, you must *install* it to make it usable.

> ✏️ **Note:**
>
> **Before installing a transformation for the first time**, check if another user has checked out the target model and if so, ask that user or an administrator to check it in. The installation process checks out target tables to add auxiliary columns if they have not already been added, and cannot do so if the model is checked out by a user different from the user who is installing the transformation. The transformation installation fails.

1. In the Study Configuration page, navigate to the transformation, select it, and select one of the following from the Install drop-down:

   • **Install** upgrades all tables and programs without deleting any data.

   • **Full Install** drops and replaces all tables and programs, **deleting all data**. Full installation is not available in the Production lifecycle.

   > 💡 **Tip:**
   >
   > The options are active only if the transformation is installable:
   >
   > – The Version and the Installable Version must be the same.
   >
   > – See Transformation installation requirements.

2. To see the updated job status in the **Install Status** field, click the **Refresh** icon.

3. To see the log file:

    a. Go to the Home page, Transformations tab.

    b. From the **Model** drop-down list, select the source clinical data model.

    c. In the **Target Model** pane, select the model.

    d. Click the icon in the **Install Job Log** column in the same row.

See What happens during installation? for more details.

Upgrade transformations to synchronize with models

If there have been metadata changes in a clinical data model—for example, an increase of column length—that affect a transformation or validation check, the system sets the transformation or validation check to **Upgrade Required**. You must run the upgrade job to synchronize the transformation or validation check with the model.

- If the main transformation is checked in, check it out. This automatically upgrades the main transformation and in some cases side transformations too.

- If either the main transformation or a side transformation is checked out and its Upgrade Required flag is set to Yes, run the Upgrade job. The Upgrade job for the main transformation may automatically upgrade the side transformation or it may set the side transformation's Upgrade Required flag to Yes, in which case you must run the Upgrade job for the side transformation.

- If columns or tables have been removed, mappings may be broken. You must fix these manually.

- After upgrade, install the transformation. You can make other changes before installing.

Run transformations and view history

This section contains the following topics:

- Run a transformation
- View run history

Run a transformation

To run or schedule a transformation:

1. Go to the 🏠 **Home** page, **Transformations** tab.

2. Select a transformation and click the **Submit Job** icon.

> 💡 **Tip:**
>
> The **Submit Job** icon is not active if the selected transformation has not been installed. Check the Install Status column.
>
> If the installation status of the transformation is Warning, you may still be able to run the batch. Check the installation log file.

3. Enter values:

- **Submission Mode**: Select one:

  – **Full** mode includes data deletion. Use Full mode only if you are confident that you are reloading all current data.

  – **Incremental** 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, select:

  – **Full UOW** includes data deletion. Use Full UOW only if you are confident that you are reloading all current data for each subject or subject visit that has any data included.

  – **Incremental UOW** is faster and does not include data deletion.

  > **✎ Note:**
  >
  > Set up regular Incremental loads at frequent intervals and do Full loads at longer intervals. Oracle DMW can load direct transformations (or validation checks) quickly when done incrementally. To do this, Oracle DMW automatically generates your MAP package with additional code. For example, you may see lines with the following format in your MAP program:
  >
  > ```
  > If ( execution mode = 'fast incremental' ) then
  > --- modified logic
  > Else
  > ---original logic
  > End if:
  > ```

- **Force Execution**: Select if you want to run the job even though the source data currency, parameter values, and the version number of the program(s) have not changed since the last run. The system uses **Full mode** regardless of the Submission Mode setting. Full mode includes data deletion.

  If not selected and all the conditions are the same as the last run, the system does not execute the job and returns a status of Success.

- **Submission Type**:

  – **Immediate** Run the job once, as soon as possible.

  – **Scheduled** Set up a regular schedule.

  – **Deferred** Run the job once, at a future time.

- **Trigger Downstream Transformations and Validation Checks**: Select to make this job trigger validation checks on 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 are set up to trigger downstream processes.

View run history

1. Go to the 🏠 **Home** page, **Transformations** tab.

2. To view all jobs, click the ▣ **View Full Job History** icon.

   To view only recent jobs again, click the ▽ **View Recent Jobs Only** icon.

Transformations are displayed by the name of their target clinical data model.

- **To view table transformations**, click a transformation's node.
- **To view run history and pending jobs**, select a transformation in the upper pane.
- **To view log files**, click the icon in the column for the type of job:
  - **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.

FAQs

- Why can't I check out my transformation?
- Why can't I merge my side transformation?
- Why does it say Upgrade Required when I know the data model hasn't changed?
- I noticed the tables in the target model changed. Why?
- What happens when I mark tables and columns as Not Used?
- Can I display mappings in a spreadsheet?
- How does Automap work?
- Where are custom programs stored?
- Can I use a side model like any other model?
- Why doesn't a data load automatically trigger this transformation?
- Why can't I trigger other transformations when I run this one?
- What does this transformation validation error message mean?
- Why can't I check out my transformation?
- Why can't I merge my side transformation?
- Why does it say Upgrade Required when I know the data model hasn't changed?
- I noticed the tables in the target model changed. Why?
- What happens when I mark tables and columns as Not Used?
- Can I display mappings in a spreadsheet?
- How does Automap work?
- Where are custom programs stored?

- Can I use a side model like any other model?
- Why doesn't a data load automatically trigger this transformation?
- Why can't I trigger other transformations when I run this one?
- What does this transformation validation error message mean?

Why can't I check out my transformation?

It may be because its target clinical data model is checked out by a different user.

Why can't I merge my side transformation?

It may be because you haven't checked it in. More likely it's because someone else checked out the model-level transformation and only that person can merge side models/transformations.

Why does it say Upgrade Required when I know the data model hasn't changed?

When a transformation is checked out, its target clinical data model is also checked out. But if the transformation checkout is undone, the model checkout is not automatically undone. To undo the model checkout, go to the model and undo the checkout there. Otherwise the transformation is set to Upgrade Required, and the new model version is no different from the old.

When the transformation is checked in, the target model is not automatically checked in, but the transformation is set to Upgrade Required.

Checking out the model with the transformation enables the Cascade Blinding and Cascade Masking features because blinding and masking are properties of the model and the cascade operation is done from the transformation.

I noticed the tables in the target model changed. Why?

When copying transformations from another study or model, the synchronization job modifies the tables in the current model so that they match the table metadata being copied, including adding, modifying, and removing columns.

What happens when I mark tables and columns as Not Used?

- Any mappings to the table or column are deleted.
- The transformation's status can be Complete without mapping the table or column. Completeness is required for validation and installation.

> ✎ **Note:**
>
> Columns populated by TMS are never included in calculating completeness even if they are marked Used.

- Validation checks and custom listings that are dependent on a table or column marked Not Used are disabled. If you later mark the table or column as Used, you must manually reenable the validation checks and custom listings.

- The table or column is not visible in the Listings pages.

Can I display mappings in a spreadsheet?

Yes.

- For table mappings, select **Export All to Excel** from the **Actions** drop-down in the Target Tables pane.

- For column mappings, click the ⬛**Export All to Excel** icon in the Target Columns pane.

How does Automap work?

Automap uses the following matches to suggest mappings:

- Name

- Alias

- Data type

- Partial name

- Partial alias

If you select a target table that is already mapped, the existing mapping remains.

Where are custom programs stored?

All custom programs for both validation checks and transformations are stored in the DMW_UTILS domain/namespace in the database. There may be subdomains to organize the programs by therapeutic area or other logical grouping.

Can I use a side model like any other model?

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.

- Side models are not visible in the Study Configuration page.

- Side models and transformations cannot have a higher validation status than Development.

Why doesn't a data load automatically trigger this transformation?

Set up the source model(s) to trigger:

1. In the main Transformation page, click the ⬛ **Add or Remove Source Model** icon.

2. Select **Can Trigger** for each, and **Save**.

Why can't I trigger other transformations when I run this one?

Set up triggering in the target model:

1. Navigate to the target model.

2. In the main Transformation page, click the ▤ **Add or Remove Source Model** icon.

3. Select **Can Trigger** for each, and **Save**.

What does this transformation validation error message mean?

The validation 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.

5

Create validation checks

Validation checks, also called *edit checks*, are programs designed to identify flawed data, or *discrepancies* (also called *queries*). 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 that is contained in a single clinical data model. To run a validation check comparing data that originated in both InForm and a lab, create a transformation to put InForm and lab data in one clinical data model.

When you save the validation check, 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 Validation Checks Listings page.

You specify whether or not a validation check automatically closes discrepancies it created when their underlying data item is updated to be valid.

- Create a validation check batch
- Validation checks
- Create a validation check
- Complete a validation check without a custom program
- Complete a validation check using a custom program
- Copy a validation check batch
- Copy a validation check
- Disable or enable a validation check
- Reorder validation checks within an ordered batch
- Install a validation check batch
- Secondary columns
- Run validation check batch and view run history
- View discrepancies created by a validation check
- Upgrade validation checks to synchronize with models
- FAQs

Create a validation check batch

Validation checks are executed in batches. Before you create a validation check you must create a batch for it. Use batches to group validation checks in logical ways:

- Checks that have dependencies on each other so they must be executed in a particular order.
- Standard checks kept together for reuse in many studies.

- Checks on the same clinical data model that should all be triggered by the same event or scheduled for the same frequency.

1. Click the Study Configuration main menu icon ⚙ from the navigation bar. Then, click the **Validation Checks** tab.

2. Select the clinical data model containing the data you want to check.

> ✎ **Note:**
>
> - Models are displayed only if they are installed.
> - The validation checks in the batch reference the latest installed version of the model. If the model is updated after the batch is created, the Upgradeable status changes to Required and you must upgrade the check to reference the latest version of the model. See Why does it say Upgrade Required?.

3. Click the ➕**Add** icon.

4. Enter a name for the batch. See Naming restrictions .

5. Enter a description (optional). The description is displayed where you run the batch.

6. Select **Ordered Execution?** if the validation checks in the batch should be run in a particular order. Validation checks in unordered batches run in parallel.

7. Select **Can be Triggered** to allow the successful completion of a transformation or data load writing to the clinical data model to trigger the execution of the batch.

8. Click **OK**.

Next: Create a validation check.

Validation checks

This window shows any validation checks that have already been created for the selected validation check batch. To make any changes, check out the batch.

- To create a new validation check, click the ➕ **Add** icon. See Create a validation check.

- To copy a validation check from another study, select Copy Checks from the **Checks** drop-down list. See Copy a validation check.

- To modify an existing validation check, select it and click the ✎ **Modify** icon. For an explanation of the fields, see Create a validation check or Complete a validation check using a custom program.

- To delete a validation check, select it and click the ✖ **Delete** icon.

Create a validation check

Before you create a validation check, you must create a batch where you want to execute it; see Create a validation check batch.

For details on creating a validation check, see this video: ⊙ Video

1. Click the Study Configuration main menu icon ⚙ from the navigation bar. Then, click the **Validation Checks** tab.

2. Select the data model where you want to create the validation check from the Data Models panel.

3. Select the validation check batch where you want the new validation check executed.

> ✎ **Note:**
>
> You must create a batch for the validation check before you crate the validation check. If necessary, see Create a validation check batch to create a batch.

4. Click the **Create Validation Check** ➕(plus sign) icon from the top of the Validation Checks for Batch table.

5. In the Name field, enter a name. (See Naming restrictions for details.) Then enter a description in the **Description** field. The name is displayed on the Validation Checks Listings page; the description is not.

   The remaining details affect the discrepancies created by the validation check:

6. **Discrepancy Text**: Describe the problem with the data and/or the action required. Maximum 255 characters.

7. Select **Authorize access to this listing for users without Blind Break rights** if you know that only non-blinded columns will be displayed in the listing. This option is available only if at least one source table contains blinded data and if you have special privileges.

   If you do not select this option and one of the source tables is blinded, the system completely blinds the listing. To see any data, a user with Blind Break privileges must break the blind.

8. **Discrepancy Initial State**: Select a state to be applied to discrepancies created by this validation check: **Open** or **Candidate**. Use Candidate to require manual review for setting the discrepancies to Open.

9. (Optional) Select a **Category**. The validation check applies this category to each discrepancy it creates. Users can filter discrepancies by category.

> **Note:**
>
> If you change the category, all new discrepancies created by the check have the new category, but the category of existing discrepancies remains the same.

10. (Optional) Select the **Initial Discrepancy Action** you want the validation check to immediately perform on new discrepancies, if any.

    • **Needs DM Review** adds this tag to discrepancies with either initial state. Discrepancies must be reviewed in DMW before being sent to InForm or a lab.

    • **Open InForm** immediately sends the discrepancy to InForm as an open query. Available only for data in an InForm clinical data model and only if you selected Open as the initial state.

    > **Note:**
    >
    > Come back and select the **Discrepant Table** and **Discrepant Column** after selecting the tables and columns to be displayed. See Select columns to display in the validation checks listings page.

11. **Execution Order**: (Available only in ordered validation check batches.) 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 lowest, 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.

12. **Continue on Error**: (Available only in ordered validation check batches.)

    • If set to **Yes**, the batch continues to execute subsequent validation checks even after one fails.

    • If set to **No**, execution stops if one of the checks fails.

    Unordered batches always run all validation checks.

13. **Allow Auto Close**: If selected, rerunning the validation check closes any discrepancy it created if the underlying data item is modified so that it no longer meets the criteria of the validation check.

    If not selected, rerunning the validation check changes the state of such discrepancies to Answered so that they can be manually closed after review.

Next: Complete the validation check:

• Complete a validation check *without* a custom program

• Complete a validation check using a custom program

Complete a validation check *without* a custom program

Prerequisites: Create a validation check batch and Create a validation check.

• Select columns to display in the validation checks listings page

- Select packages
- Add table aliases
- Specify validation check criteria
- Generate, test, view, and save source code

Select columns to display in the validation checks listings page

Identify the columns you want to display in the Validation Check Listings page. Include any columns the validation check will operate on. The system creates a SELECT clause for the query based on these selections.

You also select the discrepant table and column that contains the descrepant data that you want checked. In addition to a discrepant column, you can add one or more secondary columns. With secondary columns, Oracle DMW can verify its data against the validation check and check for changes. The action taken on the discrepancy depends on the state of the discrepancy. For more details, see Secondary columns.

> **Note:**
>
> When you view the VC listings, you see the discrepant column header appear with an asterisk (*) and secondary discrepant column headers appear with two asterisks (**) so you can easily identify them.

1. Continuing in the Add Validation Check window, in the Source pane, expand the node for the table or tables with data you want to display in the Validation Checks Listings page.

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

2. Select columns by moving them into the Selected Columns tab. You can work several ways:

   - Drag and drop selected columns or tables.

   - Select columns or tables and click the ➕**Add to Select statement** icon.

   - 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 and highlight it there.

     b. Select the additional columns and click the ➡️ **Use in expression** arrow icon.

3. Select a **Table Alias** for each column used in an expression. If there is only one defined for the table, the system adds it automatically.
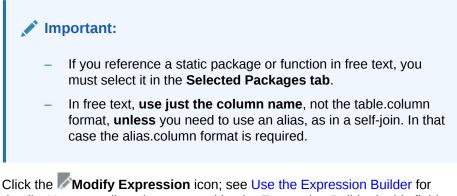
   To add an alias, go to the Table Alias tab. See Add table aliases.

4. Enter the **Column Header for Display**. This header is displayed in the Validation Checks Listings page.

5. **Expression**: Add an expression, if needed, to change data display, for example, to mask blinded data, or to append a string to all values in the column, or to concatenate two column values in a single column. Either:

   For example, to prefix 'Test' before the SUBJID, write: `Select 'Test'+SUBJID`.

   - Enter free text.

   > ✏️ **Important:**
   >
   > – If you reference a static package or function in free text, you must select it in the **Selected Packages tab**.
   >
   > – In free text, **use just the column name**, not the table.column format, **unless** you need to use an alias, as in a self-join. In that case the alias.column format is required.

   - Click the ✏️**Modify Expression** icon; see Use the Expression Builder for details. You can edit code generated by the Expression Builder in this field afterward.

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

   > ✏️ **Note:**
   >
   > If you select a column that contains masked data, write an expression for the column to display masking values. Validation checks can evaluate real data that is masked and create a discrepancy on it. The target table is blinded if the source table supports table/column/cell/row level blinding. Since target tables are auto generated and they are not accessible in DMW other than to see the data in the listing. So, we cannot define row/column/cell level blinding on custom listing and VC's target table. The target table is authorized when the authorize attribute is set for custom listing and VC.

6. **Discrepant Table and Column**: In the upper portion of the window, designate one data item as the one against which discrepancies are created. If the validation check logic processes two or more data items, select one of them.

   > 💡 **Tip:**
   >
   > You must specify Selected Columns before you can specify the primary table or column.

   a. In the **Discrepant Table** field, select the table that contains the discrepant data.

b. In the **Discrepant Column** field, select the column that contains the discrepant data. (When you view a VC listing, the discrepant column appears with an asterisk next to the column name so you can easily identify it.)

7. If you want to add one or more secondary columns to verify its data against the validation check, add the columns you want to use. Then, select the **Secondary Column** check box next to one or more columns that you want to use as secondary columns. Do not select the column already used as the discrepant column as a secondary column. Then click **OK**.

Next: Specify validation check criteria.

Select packages

If you use free text to write an expression and the expression references a static package or function, you must select the package or function in the Select Packages tab.

1. Use the Query By Example fields above the columns to search for the package.

2. Select the package(s) you are using in the validation check.

3. Click **OK**.

Next: Add table aliases.

Add table aliases

Columns used in an expression must have a 3-character table alias.

1. In the **Define Table Alias** tab, select the table from the drop-down list.

2. Enter the alias. **The alias cannot be longer than three characters.**

3. If you need an additional alias, click the ➕**Add Table Alias** icon.

> 💡 **Tip:**
>
> If you modify an alias later, any expressions or criteria that refer to the old alias are updated during the Save operation. If an expression or criteria already saved refers to a deleted alias, update it manually. To work with the expression or criteria before saving, update it manually.

Next: Specify validation check criteria.

Specify validation check criteria

In the Criteria tab, specify the data condition the validation check is looking for. This becomes the WHERE clause.

Either:

• Enter the code directly in the Criteria pane.

• Click the ab⃗ **Add or Modify Criteria** icon to open the Expression Builder. See Use the Expression Builder for details.

> **✎ Important:**
>
> – If you reference a static package or function in free text, you must select it in the **Selected Packages tab**.
>
> – In free text, **use just the column name**, not the table.column format, **unless** you need to use an alias, as in a self-join. In that case the alias.column format is required.

Next: Generate, test, view, and save source code.

Generate, test, view, and save source code

1. Click **Test**. The system generates the code, runs the validation check and displays the records retrieved.

2. **To view the generated PL/SQL source code**, click **View Source**. The button is inactive if the validation check uses a custom program.

3. Click **OK**.

Next: Install a validation check batch.

Complete a validation check using a custom program

Prerequisites: Create a validation check batch and Create a validation check.

1. Continuing in the Add Validation Check window, select **Create VC using a Custom Program**.

2. Click the ✐ **Select a Program** icon.

   a. Use the Query By Example fields above the columns to search for the package.

   b. Select the package.

   c. Click **OK**.

3. In the Select Source Tables tab, move the tables that contain data you want to display, or that your program operates on, to the SelectedTables box.

4. **Discrepant Table and Column**: In the upper portion of the window, designate one data item as the one against which discrepancies are created. If the validation check logic processes two or more data items, select one of them.

   > **💡 Tip:**
   >
   > You must specify Selected Columns before you can specify the primary table or column.

   a. In the **Discrepant Table** field, select the table that contains the discrepant data.

    b. In the **Discrepant Column** field, select the column that contains the discrepant data. (When you view a VC listing, the discrepant column appears with an asterisk next to the column name so you can easily identify it.)

5. If you want to add one or more secondary columns to verify its data against the validation check, select the validation check with the custom program from the Validation Checks for Batch table (if available). (Check the **Custom Program?** column to identify validation checks with custom programs.) Then click the ➕ **Add Validation Check** icon. Select **Create VC using a Custom Program** and click the ✎ **Select a Program** icon. Select the program previously created in Oracle LSH with the secondary column you want to use (if available). Click **OK** to return to the Add Validation Check dialog box. You see the name of the program you selected and the name of the secondary columns (one or more) in the **Secondary Column(s)** field. (You only see the **Secondary Column(s)** field if the program has them.) Select the table from the Available Table list in the Select Sources Table tab. You see the program name, version, latest status, path, and one or more secondary column names (if available).

> **✎ Note:**
>
> When you create a custom program for a validation check in Oracle LSH, you can add secondary columns by adding **$DMWVCSECONDARYVAR$** to the end of any text you add in the **Description** field of the target table column. This identifies it as a secondary column. (You can also just have **$DMWVCSECONDARYVAR$** in the **Description** field.) Do this for every column you want to use as a secondary column. However, you cannot use the discrepant column as a secondary column. For more details on secondary columns, see Secondary columns. For more details on creating a custom program in Oracle LSH, see the *Oracle Life Sciences Data Hub Application Developer's Guide*.

6. Click **OK**.

Next: Install a validation check batch.

Oracle Life Sciences Data Hub

Copy a validation check batch

You can copy a validation check batch from another study or from a different clinical data model in the same study. Disabled validation checks, if any, are included in the copy. No source tables are copied.

> **💡 Tip:**
>
> Copy validation check batches **after** completing the transformation that writes to the model, so that the system can handle them appropriately.

1. In the Study Configuration page, navigate to Validation Checks, and select the clinical data model.

2. Click **Copy Batches** in the Actions drop-down list.

3. Select the project (or other study category), then the study, then the model.

4. Select one or more batches.

5. 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 any of the tables or columns do not exist, the Copy operation fails.

   - If any of the tables are marked Not Used in the transformation that writes to the model, the system copies the validation checks as disabled. If the tables or columns are later marked Used, you can manually enable the validation checks.

   - If any of columns are marked Not Used when you copy a validation check that does not use a custom program to a model, Oracle DMW copies the system validation check as disabled.

   - If any of the discrepant or secondary columns are marked Not Used when you copy a custom validation check to a model, Oracle DMW copies the custom validation check as disabled.

   - If the tables and columns exist and are used, the system copies the validation checks and links them to the appropriate tables and columns.

   Next: Install a validation check batch.

Copy a validation check

1. Click the Study Configuration main menu icon from the navigation bar. Then, click the **Validation Checks** tab.

2. Select the batch where you want to copy a validation check.

3. Select **Copy Checks** from the **Checks** drop-down.

4. Select the project (or other study category), then the study, then the model.

5. Select the batch and validation check.

6. Click **OK**.

Next: Install a validation check batch.

Disable or enable a validation check

1. Click the Study Configuration main menu icon from the navigation bar. Then, click the **Validation Checks** tab.

2. Select the batch that contains the check.

   - To prevent a validation check from being executed when the batch is executed, select it and then select **Disable Check** from the **Checks** drop-down list.

   - To re-activate it, select it and then select **Enable Check** from the **Checks** drop-down list.

For validation checks that use a custom program, Oracle DMW checks if any discrepant and secondary columns are marked as Not Used. It prevents you from enabling a validation check with source tables or columns marked Not Used. For validation checks that do not use a custom program, Oracle DMW checks if any of the columns used by validation check are marked as Not Used. If they are, you cannot enable the validation check. If any of the tables used by validation check are marked Not Used in the transformation, you cannot enable that validation check.

Reorder validation checks within an ordered batch

1. Click the Study Configuration main menu icon from the navigation bar. Then, click the **Validation Checks** tab.

2. Select the batch.

3. Select a check.

4. Select **Reorder Checks** from the **Checks** drop-down list.

   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.

Install a validation check batch

After creating or modifying a validation check batch, you must *install* it to make it usable. See What happens during installation? for details.

Validation check batch installation fails if there are destructive changes in the model such as the removal of a column or table that the validation check reads. See Why does it say Upgrade Required? for details.

1. Click the Study Configuration main menu icon from the navigation bar. Then, click the **Validation Checks** tab.

2. Select one or more validation check batches, then select one of the following from the Install drop-down list:

   • **Install Batch** upgrades all tables and programs and does not delete any data.

   • **Full Install** drops and replaces all tables and programs, **deleting all data**. Full installation is not available in the Production lifecycle.

> **✎ Note:**
>
> The installation options are available only if all selected batches are installable:
>
> – The Version and the Installable Version must be the same.
>
> – See Validation check batch installation requirements for details.

3. To see the updated job status in the **Install Status** field, click the ↻ **Refresh** icon.

4. To see the log file:

   a. Go to the Home page, Validation Checks tab.

   b. From the **Model** drop-down list, select the source clinical data model.

   c. In the **VC Batch Name** column, select the batch.

   d. Click the icon in the **Install Job Log** column in the same row.

Secondary columns

You may want to add secondary columns to ensure the validation check catches changes during different stages of a discrepancy.

For example, you can create a validation check where **Gender** is the discrepant column and **Pregnancy** is the secondary column. The validation check criteria specifies that you want to create a discrepancy if the Gender = "Male" and Pregnancy = "Y" or "Maybe." You manually close the discrepancy when the values are Gender = "Male" and Pregnancy = "Y." Then, with a data reload, the validation check detects that the value of Pregnancy changed from "Y" to "Maybe" after you closed it. Therefore, the validation check creates a new discrepancy and includes details on the history of the changes.

Adding secondary columns allows a validation check to verify the column data and automatically take an action if that data changes depending on the state of the discrepancy. See the following table for details.

Table 5-1   Discrepancy state and action taken with secondary columns changes

Discrepancy State	Action Taken when Secondary Column Data Changes
Manually Closed or Cancelled	Opens a new discrepancy with details on the history of the data in the column. For example, it lists the data the column contained when closed or cancelled and shows what changed (including the dates and times of the previous data and the changed data).

Table 5-1   (Cont.) Discrepancy state and action taken with secondary columns changes

Discrepancy State	Action Taken when Secondary Column Data Changes
Open, Candidate, or Answered and sent to InForm or another external system	Updates the discrepancy with a tag of **VCSecDataChange** so you can locate it in the list of discrepancies. (You can also use it in a filter to search.) Oracle DMW also adds details on the history of the secondary column data changes (including the date and time of the changes).
Open, Candidate, or Answered (not sent)	Updates the discrepancy by changing the state of the discrepancy back to the start state (Open or Candidate) and adds details on the history of the secondary column data changes (including the date and time of changes).
Failed, Pending, or Processing (trying to send to InForm or another external system)	Updates the discrepancy by adding details on the history of the secondary column data changes (including the date and time of the changes).

> **Note:**
>
> You can add one or more secondary columns to a validation check when you first create the validation check (with or without a custom program) or after you create it. For details on adding secondary columns when you create a validation check, see Complete a validation check *without* a custom program or Complete a validation check using a custom program. To add or remove secondary columns after creating a validation check, see the following topic.

- Add or remove a secondary column

Add or remove a secondary column

After you create a batch and a validation check with or without a custom program, you can modify it at any time (after checking it out). For example, you may want to add one or more secondary columns to the validation check or remove them.

> **Note:**
>
> If you created a validation check using a custom program in Oracle LSH, do not follow the steps in this procedure. Instead, locate the custom program in Oracle LSH, check it out, locate the target column you want to use as the secondary column and add **$DMWVCSECONDARYVAR$** to the end of any text included in the **Description** field (to add it to the validation check). If you want to remove a secondary column from the custom program, remove the **$DMWVCSECONDARYVAR$** text from the **Description** field of the target table column. Then update the custom validation check to point to the latest version of the custom program to include the updated secondary columns list.

1. After you select a study, click the Study Configuration main menu icon ![icon] from the navigation bar.

2. From the Clinical Data Model tab, select the model with the batch and validation check you want to modify. Then, click the **Validation Checks** tab.

3. Select a validation check batch from the Validation Check Batches for Model table, then do one of the following:

   • If you selected a checked in batch, click **Check Out** to check them out. Then select one validation check from the Validation Checks for Batch table.

   • If you selected a checked out batch, select one validation check from the Validation Checks for Batch table.

4. Click the ![icon] **Modify Validation Check** icon.

5. Do one of the following:

   • To add secondary columns, select the **Secondary Column** check box next to the columns that you want to use. Do not select the column already used as the discrepant column as a secondary column.

   • To remove secondary columns, clear the **Secondary Column** check box next to the columns you do not want to use.

6. Click **OK**.

7. If ready to install the batch, see Install a validation check batch.

> **Note:**
>
> Discrepant columns appear with a single asterisk in the column header. Secondary columns appear with two asterisks in the column header. This helps you identify them when viewing VC listings.

Run validation check batch and view run history

This section contains the following topics:

• Run a validation check batch

• View run history

Run a validation check batch

You must run validation checks as a batch.

1. Go to the 🏠 **Home** page, **Validation Checks** tab.

2. Select a validation check batch and click the ✅ **Submit Job** icon.

> 💡 **Tip:**
>
> The **Submit Job** icon does not appear if the validation check batch is not installed. Check the Installed Status.
>
> If the installation status of a validation check batch is Warning, you may still be able to run the batch. Check the installation log file. If the status is Warning because one of the source tables is not used in the current study, the batch runs without input from that table.

3. Enter values:

   - **Submission Mode**: Select one:

     - **Full** mode processes all records.

     - **Incremental** mode processes only new and changed records.

       > ✏️ **Note:**
       >
       > Oracle DMW can load validation checks (or direct transformations) quickly when done incrementally. To do this, Oracle DMW automatically generates your MAP package with additional code. For example, you may see lines with the following format in your MAP program:
       >
       > ```
       > If ( execution mode = 'fast incremental' ) then
       > --- modified logic
       > Else
       > ---original logic
       > End if:
       > ```

   - **Force Execution**: Select to run the job even though the source data currency, parameter values, and the version number of the program(s) have not changed since the last run. The system uses **Full mode** regardless of the Submission Mode setting. Full mode includes data deletion.

     If not selected and all the conditions are the same as the last run, the system does not execute the job and returns a status of Success.

   - **Submission Type**: Select:

     - **Immediate** to run the job once, as soon as possible.

     - **Scheduled** to set up a regular schedule.

ORACLE®

- **Deferred** to run the job once, at a future time.

- **Trigger Downstream Transformations and Validation Checks**: Select this checkbox 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.

> ✎ **Note:**
>
> This option appears only if the validation check batch is set up to allow it.

- Click the ↻ **Refresh** icon at any time for an update.

- To check the log file, click the icon in the **Log** column.

View run history

1. Go to the 🏠 **Home** page, **Validation Checks** tab.

2. From the **Model** drop-down list, select the clinical data model.

3. Select a validation check batch in the upper pane.

   The system displays information about its validation checks in the middle pane and information about its run history in the lower pane.

4. To view all jobs, click the ▤ **View Full Job History** icon.

   To view only recent jobs again, click the ▤ **View Recent Jobs Only** icon.

   To view log files, click the icon in the column for the type of job:

   - **Log** (Run History pane) The most recent manually submitted job.

   - **Triggered Job Log** (Run History pane) The most recent triggered job.

   - **Install Job Log** (Validation Check Batch pane) The most recent installation of the validation check batch.

   > ✎ **Note:**
   >
   > A validation check can be *disabled* so that it is not included in the batch execution. To find out if a check was included in the run, check the log file.

View discrepancies created by a validation check

1. Click the Data Management icon 🔍 from the navigation bar and click **Listings** to open the Listings page.

2. Select the source clinical data model.

3. Select **Validation Checks** from the bottom of the left pane.

4. Expand the node for the batch containing the validation check.

5. Select the validation check.

Upgrade validation checks to synchronize with models

If there have been metadata changes in a clinical data model—for example, an increase of column length—that affect a transformation or validation check, the system sets the transformation or validation check to **Upgrade Required**. You must run the upgrade job to synchronize the transformation or validation check with the model.

- Select one or more batches and click the **Upgrade Batch** icon or reinstall the batch(es).

- If columns or tables have been removed, mappings may be broken. You must fix these manually.

- When you next install the batch(es), installation ends with a warning if a validation check refers to a table or column that no longer exists in the source model. You must do the synchronization manually.

FAQs

- Why can't I install my validation check batch?
- Why are some validation checks disabled?
- Do I have to check out the batch to disable a validation check?
- Why does it say Upgrade Required?
- Why isn't the transformation triggering my validation check batch job?
- Where are custom programs stored?
- Why can't I install my validation check batch?
- Why are some validation checks disabled?
- Do I have to check out the batch to disable a validation check?
- Why does it say Upgrade Required?
- Why isn't the transformation triggering my validation check batch job?
- Where are custom programs stored?

Why can't I install my validation check batch?

It may be because the clinical data model is not installed. Install the model and try again.

Why are some validation checks disabled?

Validation checks that read from a table or column marked Not Used are automatically disabled. If the table or column is later marked as Used, you must manually reenable the validation checks to run them.

A person may have manually disabled a validation check to prevent it from running.

To find out why a particular validation check is disabled, hover over the value in the **Disabled Reason** column.

Do I have to check out the batch to disable a validation check?

No. You can disable or enable checks whether the validation check batch is checked in or out.

Why does it say Upgrade Required?

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 system sets the value of the Upgradable column to **Required**. Select the batch, then select **Upgrade Batch** from the Actions drop-down, then install the batch.

See Upgrade validation checks to synchronize with models.

Why isn't the transformation triggering my validation check batch job?

Set this up in two places:

1. In the transformation that writes to the clinical data model that the validation check reads from, set **Can Trigger** to **Yes** in **Add or Remove Source Model**s.

2. In the validation check batch, select **Can Be Triggered**.

Where are custom programs stored?

All custom programs for both validation checks and transformations are stored in the DMW_UTILS domain/namespace. Any other locations listed are subdomains inside that domain.

6

Create custom programs, listings, and filters, and set up reference data

This section describes how to:

Create public custom listings

You can create standard public custom listings for data reviewers to use and modify.

Custom listings display only the columns you specify, making it much easier for reviewers to see the data they need. You also build a WHERE clause to filter the data displayed.

For details on creating a public custom listing, see this video: ▶ Video

If you have a visualization tool integrated with DMW, its users can see custom listings as well as data in the clinical data model they read from.

This section contains the following topics:

Name the listing and mark it Public

1. After you select a study, click 🔍 **Data Management** icon from the navigation bar and select **Custom Listing Manager** from the drop-down menu to open the Custom Listing Manager tab.

2. Click the ➕ **Add** icon from the Custom Listings panel to open the Add Custom Listing dialog box.

3. Enter a name for the custom listing in the **Name** field. Then enter a description of the new custom listing in the **Description** field.

> **Tip:**
>
> Use a naming convention and keep names under 25 characters so data reviewers won't have to scroll to read the name.

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, even though at least one source table contains blinded data. Take care to select columns that do not contain blinded data.
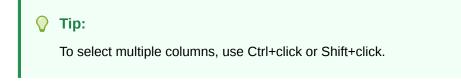
   If any source table is blinded in any way and this setting is not selected, the system blinds the entire target table, so that only users with Blind Break privileges can view any data.

5. Select **Mark as Public** to enable all data reviewers to use this custom listing.

6. Do one of the following:

   • Continue on to Select columns to display to continue defining your custom listing.

   • Click **OK** to close the dialog box and save the custom listing with the details you entered. You can return to this procedure and search for it to continue defining it later.

Select columns to display

Identify the columns to display in the listing and write an expression to change data display if needed; for example to mask blinded data, or to append a string to all values in the column, or to concatenate two column values in a single column.

1. Drag the tables or columns you want to display from the Source pane into the Selected Columns tab.

> **Tip:**
>
> To select multiple columns, use Ctrl+click or Shift+click.

   Tables and columns that are marked Not Used in the transformation that writes to this model are not displayed here.

2. If you need an **Expression** to operate on the column in the SELECT clause, you must create a 3-character **Table Alias** for its source table in the Define table aliases tab, then select it from this drop-down list.

3. To display a different label for the column in the Custom Listings page, enter the label in the **Column Header for Display** field.

4. Enter a **Sort Order** number to determine the column's display order relative to other columns.

5. Select Ascending (**ASC**) or Descending (**DESC**) **Sort Type** for the data display.

6. If you need an **Expression** to operate on the column in the SELECT clause, do one:

   - Enter the expression in the **Expression** field.

   - Click the ab **Modify Expression** icon to Use the Expression Builder for details. You can edit code generated by the Expression Builder in this field afterward.

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

   a. Add one column to Selected Columns and highlight it there.

   b. Select the additional column(s) in the Source pane and click the **Use in Expression** icon in the Source pane.

   > **Note:**
   >
   > If you select a column that contains masked data, write an expression to mask values in the column.

Next: Define table aliases.

Back to Create public custom listings *steps*

Select packages

If you call functions in your code, you must select the package that contains the function in the Select Packages tab. If you use the Expression Builder, you do not need to use the Select Packages tab.

If you need to use a function and you plan to write the expression using the function in free text, open the **Select Packages** tab and select the packages you will use. This enables the system to generate the query code. The system displays all packages in the DMW_UTILS domain that meet certain criteria.

If you use the Expression Builder (reached by clicking the **Modify Expression** icon), you do not need to use the Select Packages tab.

Define table aliases

Table aliases are required only if you are using a self-join or writing a SELECT expression on a table column.

1. In the **Define Table Alias** tab, select the table from the drop-down list.

2. Enter an alias. **The alias cannot be longer than three characters.**

   The system displays the alias in the Selected Columns tab.

3. If you need another alias, click the **Add Table Alias** icon.

4. Click **OK**.

Next: Specify and test criteria.

Back to Create public custom listings *steps*

Specify and test criteria

Specify the data condition the listing will look for.

1. Select the **Criteria** tab.

2. Build the WHERE clause to determine which records appear in the listing. Click the 🔲 **Add or Modify Criteria** icon and Use the Expression Builder .

3. In the custom listing Query Details pane, view and test the generated code:

   • Click **View Source**. The system generates and displays the PL/SQL code.

   • Click **Test**: The system generates PL/SQL code, validates it, and displays either an error message or the records retrieved.

4. Click **OK**.

Next: Save and install a custom listing.

Back to Create public custom listings *steps*

Save and install a custom listing

1. In the Custom Listings pane, select the listing.

2. Click the ⚙ **More Actions** icon, then:

   • 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.

Create public filters

Filters allow data reviewers to specify the data and discrepancies to view, but the user interface is not intuitive and may be difficult for nonprogrammers. You can create a set of public filters that data reviewers can use or modify.

> ✏️ **Tips:**
>
> • Create a set of standard public filters.
>
> • Do not create so many that it is hard for users to find their own filters.
>
> • Name each public filter descriptively so users know if it is useful for their purposes.

• Subject and Visit type filters

• Create a new filter

Subject and Visit type filters

Subject filters allow filtering by subject ID, site, investigator, and country. If you create a public filter for each subject filter type and call them simply Subject ID, Site, Investigator, and Country, data reviewers can set the criteria for a particular site, country, etc.

Filter types that include "Subject" or "Visit" in their name require specifying a "Filter Driver Model" as part of the definition. All filters used in a study should specify the same clinical data model that includes a Subject Visit table. See How subject and visit filters work for more requirements.

Even if a study has a Subject Visit table in more than one model, specify the same model for all filters, or the filters will not work when applied together.

Create a new filter

1. After you select a study, click [icon] **Data Management** icon from the navigation bar. Then do one of the following:

   • Select **Listings** from the drop-down menu to open the listings page. Expand a default, custom, or VC listing, expand a data model, and select a listing. Then click the Filters tab from the left panel.

   • Select **Discrepancies** from the drop-down menu to open the discrepancies for the study you selected. The filter panel appears on the left.

2. In the left panel, do one or more of the following:

   • Click **Advanced Filters** to open a text box. Click inside the box to open a drop-down menu, select a filter item and start building your formula using the appropriate operators. The criteria you select determines what additional items you can access.

   • Click in one or more of the following fields to enter or select the quick filter criteria you want to use (with or without an advanced filter formula). Use the scroll bar to see all your choices:

      a. **Model**: Lists the models in the study (only shown through Discrepancies page).

      b. **Listing**: Lists the listings for the model you selected (only shown through Discrepancies page).

      c. **Country**: Lists the countries of the study or listing (depending if you accessed the filters from Listings or Discrepancies).

      d. **Site**: Lists the location of where the study or listing was done.

      e. **Subject**: Lists the available subject IDs.

      f. **Visit**: Lists the available names of visits.

      g. **Discrepancy State**: Lists the available discrepancy states. You can select Answered, Cancelled, Candidate, Closed, Open.

> **Note:**
>
> By default, you cannot see cancelled or closed discrepancies. You must apply the Discrepancy State of Cancelled or Closed to see them.

h. **Discrepancy Tag**: Lists the available discrepancy tags (for example, ClosedWithAnswer, NeedsDMReview, ClosedAsIs).

> **Note:**
>
> If your account includes the appropriate privileges and you need to show blinded data, select **Show Blinded Data** and click **Yes** when prompted. (The system tracks the number of times you view blinded data.) If you selected a listing, saw a prompt to access blinded data, and clicked Yes, the Show Blinded Data field appears selected already. To stop showing blinded data, clear this field and click **Yes** when prompted.

3. If you want to keep displaying the filtered data, select **Keep the Filter in effect**.

4. Click **Apply**.

5. (Optional) To save the filter criteria for future use, click **Save**, name the filter, and click **Save and Apply**. Use a unique name across all the private, shared, and public filters that you can see in the current study and lifecycle. If you want to use this filter as a template to create a new filter, click **Create New** and repeat these steps.

6. To clear the filters, click **Clear**. To view the data after clearing the filters or creating a new filter, click **Apply**.

Create custom programs

For details on creating a custom program, see this video: ▶ Video

If a transformation or validation check is too complex to create in the user interface, you can use a custom program instead, for example:

• To perform data aggregation, case statements, or complex calculations.

• To call an API to set a flag on records that meet specified criteria see Sample PLSQL program that calls an API to set a flag.

• To use 1-to-many or many-to-many column mappings.

The custom program code must do everything the system does automatically in noncustom transformations and validation checks, including mapping source and target table columns and enabling data lineage tracing when possible; see Enable data lineage tracing in a custom program.

You must define a program in Oracle Life Sciences Data Hub (Oracle LSH). You can write a PL/SQL or SAS program and upload it to the Oracle LSH program. You can reuse custom programs.

> **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* and the *Oracle Life Sciences Data Hub System Administrator's Guide* for instructions.
>
> - Do not modify an Oracle DMW work area directly in Oracle LSH. Oracle DMW work areas are those that are in the DMW_Domain/*project* or other study grouping domain/*studydomain*/*data model* application area. In particular, do not add either tables or programs to these work areas in Oracle LSH. Create custom programs in the DMW_UTILS domain.

This section contains:

- Create a custom program for a transformation
- Create a custom program for a validation check
- Enable data lineage tracing in a custom program
- Sample programs that populate auxiliary columns with the source surrogate key
- Use APIs
- Sample PLSQL program that calls an API to set a flag

Create a custom program for a transformation

You can create the custom program in Oracle LSH before or after creating the transformation.

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 restrictions .

   > **Note:**
   >
   > This is to enable data lineage tracing. See Enable data lineage tracing in a custom program and Sample programs that populate auxiliary columns with the source surrogate key.
   >
   > If your program is performing an operation like aggregation that makes it impossible to support data lineage tracing, you do not need to create or populate these columns.

3. Save and check in the data model. If you start to map to the target table in the user interface before creating the auxiliary columns, you will not see the auxiliary columns in the user interface. To display the columns in the UI, go to the model

and add the columns, then go back to the transformation, check it in, and check it out again.

4. Log in to Oracle LSH, select its Applications tab if it is not already selected, and navigate to the DMW_UTILS domain under DMW_DOMAIN.

   For details, see this video: ▶ Video

5. In the DMW_UTILS domain, navigate to the application area your administrator has set up for storing custom programs of this type, and click **Manage Definitions**. Verify that you are in the right application area.

6. From the **Create** drop-down, select **Program** and click **Go**. Enter a name and description and select the Program Type: PLSQL or SAS. Click **Apply**.

7. For each *source and target* table: 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.

8. In the Search and Select window:

   a. In the **Domain** field, click the 🔍 **Search** icon and select DMW_DOMAIN, the appropriate study grouping, and your study.

   b. Select **Display Table Definitions Under DataModel**. Select the clinical data model and enter the table name if you know it, then click **Go**.

   c. Select the table. The system returns you to the Create Table Descriptor page with the selected table displayed in the Definition Source field. Click **Apply**, then click **Return**.

9. For each *source* table: Click the table name, then click **Update**, change **Is Target** to **No**, click **Apply**, then click **Return**.

   > ✎ **Note:**
   >
   > - Each source table's Oracle name cannot be longer than 25 characters. This is because the target table must contain a column for surrogate key information that contains the source table name.
   >
   > - You can change the table name in the source clinical data model.
   >
   > - Table descriptors allow you to reuse this program in a different study where the source or target tables may have different names and column names but the same structure, by mapping the tables in the Oracle DMW transformation.

10. Write the program. If you have integrated SAS with Oracle LSH you can click the **Launch IDE** button from the program page. You can also upload a program into LSH.

    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.

11. 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 .
>
> - **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.
>
> - 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.
>
> - If required, you can call Oracle LSH or Oracle DMW public APIs from your code; see Use APIs.

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

13. Check in the program in Oracle LSH.

14. In Oracle DMW, navigate to the transformation mapping for the target 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. 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 only if you are performing operations like aggregation that make it impossible to trace data lineage.

    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.)

15. Install the transformation.

Create a custom program for a validation check

To create a custom program for a validation check:

1. Log in to Oracle LSH, select its Applications tab if it is not already selected, and navigate to the DMW_UTILS domain under DMW_DOMAIN.

2. In the DMW_UTILS domain, navigate to the application area your administrator has set up for storing custom programs of this type, and click **Manage Definitions**. Verify that you are in the right application area.

3. From the **Create** drop-down, select **Program** and click **Go**. Enter a name and description and select the Program Type: PLSQL or SAS. Click **Apply**.

4. Create the target table that is displayed on the VC Listings page.

   a. In the Table Descriptors subtab, click **Add Target From New**. The Create Table Descriptors window opens.

   b. Enter values in the required fields. The table name must not be longer than 25 characters.

   c. Add columns. In addition to the columns you need to display the results of the validation check, add one column for each source table with a name like *source_table_*SKEY to enable data lineage tracing.

   > **✎ Note:**
   >
   > Each source table's Oracle name cannot be longer than 25 characters. This is because the target table must contain a column for surrogate key information that contains the source table name.

   d. In the **Actions** drop-down, select **Table Instances from Existing Table Descriptors** and click **Go**.

   e. Select the target table descriptor you just created and click **Create Table Instance**. Confirm.

5. For each source table: 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.

6. In the Search and Select window:

   a. In the **Domain** field, click the 🔍 **Search** icon and select DMW_DOMAIN, the appropriate study grouping, and your study.

   b. Select **Display Table Definitions Under DataModel**. Select the clinical data model and enter the table name if you know it, then click **Go**.

   c. Select the table. The system returns you to the Create Table Descriptor page with the selected table displayed in the Definition Source field. Click **Apply**, then click **Return**.

7. Click the table name, then click **Update**, change **Is Target** to **No**, click **Apply**, then click **Return**.

   > **✎ Note:**
   >
   > Each source table's Oracle name cannot be longer than 25 characters. This is because the target table must contain a column for surrogate key information that contains the source table name.
   >
   > You can change the table name in the source clinical data model.

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

9. 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 for details.
> - **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.

10. Write the program code.

    - Your program must handle populating all target columns, including populating the new auxiliary SKEY columns with the concatenated value of the internal CDR$SKEY (surrogate key) column in each source table. See Enable data lineage tracing in a custom program.
    - If required, you can call Oracle LSH or Oracle DMW public APIs from your code; see Use APIs.

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

12. Check in the program in Oracle LSH.

13. Run public API DME_PUB_XFORM_MAP.populateStaticPackages whenever you add, modify, or remove a custom program. This makes the new or changed program available for use or, in the case of removing a program, makes it unavailable.

14. In Oracle DMW, define the validation check in a batch and enter values for all required fields.

15. 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.

16. 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 area your administrator has set up.

17. Install the validation check batch.

Enable data lineage tracing in a custom program

Custom program code must, if possible, enable data lineage tracing so that the system can:

- For each data point, display information about contributing data points in source models and resulting data points in target models, if any.

- Display each discrepancy on contributing data points on the preferred path in source models and resulting data points in target models.

To enable data lineage tracing in a custom program:

1. Map the CDR$SKEY surrogate key column in the source table to the corresponding auxiliary column in the target table. (The system creates the auxiliary columns in target tables. You do not need to do that.) See How the system tracks data lineage for more information.

2. The program must populate the target table's auxiliary column for each source table with the value of the CDR$SKEY column for each source record, as part of the INSERT INTO statement.

See Sample programs that populate auxiliary columns with the source surrogate key for examples of the SAS and PL/SQL code required to do this.

> **✎ Note:**
>
>   - Data lineage tracing cannot work when it is not possible to trace all contributing data points, as in aggregations. The system displays discrepancies only in the model in which they were created.
>
>   - If a custom program performs a pivot or unpivot, 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.

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
- Example PL/SQL program

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;
/
```

Use APIs

You can call public APIs from custom programs. For more information about APIs, see the *Oracle Health Sciences Life Sciences Warehouse Application Programming Interface Guide*—click the "Help" link from DMW, then click the Books link in the left pane.

Before calling an API or public view, you must call DME_PUB_INITIALIZATION.SetupAPIStudyEnvironment. This procedure checks that the study ID and lifecycle value you pass in are valid and then uses those values and the study's partition ID for the lifecycle stage to set SYS_CONTEXT appropriately. Call it again to change the study or lifecycle context.

Sample PLSQL program that calls an API to set a flag

Your administrator can create flags, each with multiple states, to assign to records to track anything you want.

To use flags to help in the review process of individual subject data records, write a validation check to examine records and apply a particular flag state depending on the data in the record. for example:

- Write a validation check to ascertain that all required fields have a value and set a flag called **Record Complete** to On if they do.

- Write a validation check 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: Validation Incomplete, Validation Complete, or, if the record is locked, Record Locked.

The following 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;
    /
```

Set up reference data

You can develop a library of reference tables in a central location for use in transformations in any study. A study grouping called DMW_REFDATA is shipped with DMW for this purpose. You must create at least one "study" and at least one model within it to store your reference tables. You can create multiple studies or models to sort your tables into categories.

You can create target data models and their transformations in a reference data study the same way you can in clinical studies. You can copy these transformations and models into clinical studies.

All reference data models are displayed after study models in the Add Source Model dialog for transformations. The name of the "study" containing each model is also displayed.

For example:

- Add a table with data on lab normals in DMW_REFDATA, and in a transformation, join it with a study table of lab results to show the lab results along with the normal lab ranges.

- Add a table with US and metric units and SI conversion factors in DMW_REFDATA, and in a transformation, join it with a study table to convert US to metric units.

To use this feature:

1. Create a study in DMW_REFDATA.

2. Create at least one file-type input clinical data model in the study and set up File Watcher for the model.

3. Create at least one reference table in the model and install it.

> **✎ Note:**
>
> DMW_REFDATA models and their tables are visible in other studies only if they are installed.
>
> In addition, updated versions of the model containing reference tables must be installed before the study transformation displays the message that it needs to be upgraded.

4. Load data into the table using File Watcher.

5. Reference the table from a clinical study transformation in the Development lifecycle stage:

   a. In the Add Model dialog, select the model containing the reference table as a source model.

   b. Map the reference table to a target table.

6. *Before* promoting and installing a transformation that uses a reference table to QC or Production, manually promote the reference data model to the appropriate lifecycle, install it, and load data.

> **Note:**
>
> You must manually promote reference data models to QC and Production when they are ready. When transformations that reference tables in the DMW_REFDATA models are promoted to QC or Production, the system automatically changes their reference to the table in the model in the same lifecycle if it exists, but the system cannot automatically promote the model as it does if the model is in the same study.

The following restrictions apply within DMW_REFDATA:

- After a table in a model has been referenced in another study, the model cannot be removed.
- The Oracle name of any object cannot be changed.
- InForm models are not supported.
- Integration with TMS is not supported.
- Data blinding is not supported.

Develop a library of SQL functions

You can develop a library of custom PL/SQL functions, including conversions and derivations. Custom functions are always static references and can reference lookup tables; see Set up reference data.

These custom functions are available for reference from expressions in:

- Transformations
- Validation Checks
- Custom Listings
- Data Masking

> **Note:**
>
> Standard Oracle SQL functions are also available.

You must define each function in Oracle Life Science Data Hub as an Oracle LSH program; see Create custom programs.

Each static reference program must:

- Be created in an Oracle LSH application area in the DMW_UTILS domain.

- Be 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.

Run public API DME_PUB_XFORM_MAP.populateStaticPackages whenever you add, modify, or remove a static package. This makes the new or changed package available for use or, in the case of removing a program, makes it unavailable.

Use the Expression Builder

There are two ways to add an expression, with different advantages:

- **Use the Expression Builder user interface**. This more cumbersome process makes the transformations, validation checks, and custom listings that use it easier to copy and map in the new study.

- **Enter code as free text directly in the Expression Text field**. This is a simpler process for a programmer but results in a less easily reusable transformation, validation check, or custom listing.

> ✏️ **Important:**
>
> – If you reference a static package or function **in free text**, you must select it in the **Selected Packages tab**.
>
> – In free text, **use just the column name**, not the table.column format, **unless** you need to use an alias, as in a self-join. In that case the alias.column format is required.

To use the Expression Builder:

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 phrase in an expression or group smaller units of logic.

   - **Add Item** to add a unit of logic smaller than a group.

2. To add a phrase within a group, click the parentheses ().

   To add a phrase outside a group, click **Expression**.

3. To add an item, in the Expression Item pane select either **Column**, **Function** (for functions written by your company), or **Standard Function** (for Oracle SQL functions).

   To create an expression using column values:

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

b. Click the **Select Column** icon.

In the Select Column window, you can filter above any of the attribute columns to find the table column you want. Select a column and click **OK**.

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

d. If needed, enter a data value. The system encloses the value you enter in single quotes.

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

> 💡 **Tip:**
>
> If you select a conjunction within a group, it appears within the group, at the end. If you need a conjunction outside the group, click **Expression** above, then select the conjunction.

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

> ✏️ **Note:**
>
> You can edit the generated code in the Expression Text pane, but if you do, you cannot continue to build the expression in the user interface. See "**To make a correction in the Expression Builder**" to continue in the user interface.

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

To use a function in your library:

a. For Item Type, select **Function**. The Select Function window appears, displaying a list of Oracle functions.

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

To use a standard 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**.

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

5. Click **Save**.

6. Click **Validate**. The system validates the code and displays any errors or warnings.

To make a correction in the Expression Builder:

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

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

For more information, see:

- [Pass data as input parameter values]
- [Pass constant values]
- [Develop a library of SQL functions]
- [Pass data as input parameter values]
- [Pass constant values]

Pass data as input parameter values

Use curly brackets ("{" and "}") as delimiters and the fully qualified format (`model.table.column`) to indicate input parameter values to SQL functions or custom functions in the expression. The default input is the column value if no metadata is specified after the column name.

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

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

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

Pass 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))
```

FAQ

- [Can I speed up my custom SAS program's runtime?]

Can I speed up my custom SAS program's runtime?

To improve performance, SAS programs can invoke SAS direct load processing if the following conditions are met:

- The target table processing type for SAS program must be Staging, either With or Without Audit.
- The target table descriptor must have "Target as Dataset" set to Yes.
- The administrator must enter the SQL Loader file location in the Detail attribute of the SAS service location in LSH.

7

Load data and run jobs

This section describes how to:

- Load data from a file
- Load data from InForm
- Run transformations and view run history
- Run validation check batches and view run history
- Resend discrepancies that failed to be sent to InForm
- What if...
- FAQs

Load data from a file

The study File Watcher checks a file system location for data files to load.

- Upload data files
- Suspend or resume data file loading
- Verify that a data file loaded
- View data load history
- View data files not processed
- Load data and discrepancies from a non-InForm clinical data system

Upload data files

To load data files:

- Place them on the server in the location your administrator specified for the study, in the correct subdirectory. There is a subdirectory for each lifecycle stage or for each lifecycle stage/file type combination.
- The file name must **exactly** match the file specification defined for the clinical data model; see Specify a file name convention for each lifecycle stage.

  If you are loading data from a clinical data capture system other than InForm, see Load data and discrepancies from a non-InForm clinical data system.

Suspend or resume data file loading

1. After you select a study, click the **Study Configuration** icon ![icon] from the navigation bar. Then click the **Clinical Data Model** tab.
2. Select the clinical data model.
3. In the **Watcher Configuration** tab, select a file specification.

---

4. Click the appropriate icon, either ❚❚ **Suspend** or ▶ **Resume**.

Verify that a data file loaded

1. Go to the 🏠 **Home** page, **Data Loads** tab to see successful data loads.

2. If a data load you are expecting is not there:

   a. Click the **Study Configuration** icon ⚙️ from the navigation bar.

   b. Select the clinical data model.

   c. Select the **Detected Files** tab. It lists all the uploaded data files that File Watcher has detected for a clinical data model.

3. If a file failed to load, you can select it and:

   - Click the ✖ **Delete File** icon to start over with a new or renamed file. The most likely problem is that the file name does not match the File Specification for the model. See Specify a file name convention for each lifecycle stage.

   - Click the 🗒 **Load Data** icon to try again with the same file, for example after fixing the data load parameters.

For each detected file the system displays:

- **File Name** and **File Specification Name** for comparison.
- **Status** The possible statuses are:
  - **DETECTED** The file is in the watched folder but has not been submitted. The scheduled submission time is in the Data Load Date column.
  - **SUBMITTED** The file has been submitted.

    > 💡 **Tip:**
    >
    > This status does not change when the load completes. Select the Data Loads tab on the Home page to see if the load was successful.

  - **MISSING** The file was deleted before or after it was submitted, before the scheduled deletion date.
  - **DELETED** The file was deleted as scheduled.
  - **ARCHIVED** The file was archived as scheduled.
- **File Modified** The modification date of the file on the file system.
- **Data Load Date** The scheduled data load date before the data has been loaded, with an icon to indicate "scheduled," and the actual data load date afterward.
- **Detection Date** The date and time the file was detected, using the date and time in the DMW database.
- **Archive Date** The scheduled archive date before the file is archived and the actual archive date afterward. (Not supported in this release.)
- **Error** Information about a problem, if any.

- **Deletion Date** The scheduled deletion date before the file is deleted and the actual deletion date afterward.

- **Date Missing** If a file is overwritten or removed from the file system before it is archived or deleted, then the Date Missing is stored here.

View data load history

1. Go to the 🏠 **Home** page, **Data Loads** tab.

2. To view all jobs, click the ▤ **View Full Job History** icon.

   To view only recent jobs again, click the ▤ **View Recent Jobs Only** icon.

3. You can filter by entering a value in the blank field above any column. If blank fields are not displayed, click the ▤ **Query By Example** icon.

- Statuses for uncompleted jobs

Statuses for uncompleted jobs

For incomplete jobs, the system displays the job's current status. Click the ↻ **Refresh** icon to update.

- **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 for the clinical data model work area.

- **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. The system does not rerun the job unless the person submitting the job chooses to force reexecution.

View data files not processed

1. Go to the 🏠 **Home** page and select a study.

2. Select the **Files Not Processed** tab.

   The system displays files that were uploaded but could not be loaded into the selected study. A file may not be loadable because:

   - It is misnamed or in nonmatching case, compared to the file specification defined for the clinical data model.

- There is a mistake in the File Specification regular expression defined for the model.

- There is a matching File Specification, but its end date has passed.

- There is a matching File Specification, but its data loading is suspended.

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

3. View unprocessed files.

    Columns include:

    - **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 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 DMW database.

    - **Archive Date** displays the scheduled archive date before the file is archived and the actual archive date afterward.

        > **Note:**
        >
        > Archiving is not supported in this release.

    - **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.

Load data and discrepancies from a non-InForm clinical data system

Clinical data systems other than InForm must load data and discrepancies into Oracle DMW in data files. Each load must include an additional file. The file must:

- Be named exactly as specified in the DISC_LOAD_TBL column of the DME_EXTERNAL_MODEL_ATTRIBS table in your database.

- The file must be formatted the same way as data files (SAS files or for text: fixed or delimited format, and if delimited, either using enclosing characters or not).

- Be in the same .zip file as the data and discrepancy files for text files or the same XPort, CPort, or SAS datasets.

- Include values in three columns for each record as shown in the following table. These values are written to an internal table in the input clinical data model.

Table 7-1   Columns in the Generated Clinical Data Model Forms Table

Order	Column Name	Datatype	Length	Description
1	TABLE_NAME	VARCHAR2	30	The Oracle name of the table in the external system that contains the record.
2	KEY_VALUES	VARCHAR2	4000	The unique key values for the record, separated by tildes (~), without quotation marks, and in column order.
				DMW checks these values against the surrogate key value it generates for each record for use in data lineage tracing.
3	PARTIAL_URL	VARCHAR2	4000	The ID or other string required to uniquely identify the record in the external system.

For more information, see *Using the Generic Connector to Integrate DMW with a Clinical Data System*, My Oracle Support article 2172786.1, or the documentation provided by the integration vendor.

Load data from InForm

- Load InForm data immediately in Development or Quality Control
- Schedule InForm data loads in Production
- Suspend and resume InForm data loading

Load InForm data immediately in Development or Quality Control

In the InForm Configuration tab, click the **Load Data to DMW Development** or **Load Data to DMW Quality Control** icon to immediately load the latest data from InForm into the DMW Development lifecycle stage.

You cannot schedule data loads in Development or QC.

See InForm data isn't loading? for more information (if necessary).

Schedule InForm data loads in Production

1. After you select a study, click the **Study Configuration** icon from the navigation bar. Then click the **Clinical Data Model** tab.
2. Select **Production** from the **Lifecycle** drop-down at the top of the page.
3. Click the **Suspend** icon.
4. Install the model if it has not already been installed in Production.
5. Select **Production** under Study Lifecycle in the table below.
6. Select the **Schedule Production Data Load** checkbox.

> **Note:**
>
> If you do not check this box, it is possible to enter Fetch Frequency and click Save, but the system does not actually schedule the load.

7. Specify a **Fetch Frequency**: the number of minutes between automatic data loads.

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

8. Save.

9. Click the **Resume** icon. The first scheduled data load starts immediately.

See InForm data isn't loading? for more information (if necessary).

Suspend and resume InForm data loading

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

1. Click the Study Configuration icon at the top of any page and then click the **Clinical Data Models** tab.

2. Select the lifecycle from the Lifecycle drop-down at the top of the page.

3. Select the clinical data model.

4. In the **InForm Configuration** tab, 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.

5. Click the ▶ **Resume** icon to allow manual and scheduled data loading.

See InForm data isn't loading? for more information (if necessary).

Run transformations and view run history

1. Go to the 🏠 **Home** page, **Transformations** tab.

2. To view all jobs, click the **View Full Job History** icon.

   To view only recent jobs again, click the **View Recent Jobs Only** icon.

3. See:

   • View transformation job history

   • Run a transformation

- Cancel a pending tranformation job

View transformation job history

Transformations are displayed by the name of their target clinical data model.

- **To view table transformations**, click a transformation's node.
- **To view run history and pending jobs**, select a transformation in the upper pane.
- **To view log files**, click the icon in the column for the type of job:
    - **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.

Run a transformation

To run or schedule a transformation:

1. Go to the **Home** page, **Transformations** tab.

2. Select a transformation and click the **Submit Job** icon.

    > **Tip:**
    >
    > The **Submit Job** icon does not appear if the selected transformation has not been installed. Check the Install Status column.
    >
    > If the installation status of the transformation is Warning, you may still be able to run the batch. Check the installation log file.

3. Enter values:
    - **Submission Mode**: Select one:
        - **Full** mode includes data deletion. Use Full mode only if you are confident that you are reloading all current data.
        - **Incremental** 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, select:

        - **Full UOW** includes data deletion. Use Full UOW only if you are confident that you are reloading all current data for each subject or subject visit that has any data included.
        - **Incremental UOW** is faster and does not include data deletion.

> **Tip:**
>
> Set up regular Incremental loads at frequent intervals and do Full loads at longer intervals.

- **Force Execution**: Select if you want to run the job even though the source data currency, parameter values, and the version number of the program(s) have not changed since the last run. The system uses **Full mode** regardless of the Submission Mode setting. Full mode includes data deletion.

  If not selected and all the conditions are the same as the last run, the system does not execute the job and returns a status of Success.

- **Submission Type**:

  – **Immediate** Run the job once, as soon as possible.

  – **Scheduled** Set up a regular schedule.

  – **Deferred** Run the job once, at a future time.

- **Trigger Downstream Transformations and Validation Checks**: Select to make this job trigger validation checks on 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 are set up to trigger downstream processes.

Click the ↻ **Refresh** icon for an update to the **Job Status**.

Cancel a pending tranformation job

Select a transformation in the Run History pane and click:

- ⊠ **Cancel Job** to cancel the currently running or next pending job.

- ✕ **Cancel All Jobs** to cancel the currently running or next pending job and all future jobs in the schedule for the selected transformation. Available only for scheduled jobs.

  > **Note:**
  >
  > If there is an immediate or deferred job for the same transformation, or another schedule for the same transformation, those jobs are not affected.

- ▤✕ **Cancel Triggered Job** to cancel the currently running or pending job. Available only for jobs that are set up to be triggered by another job.

Run validation check batches and view run history

1. Go to the 🏠 **Home** page, **Validation Checks** tab.

2. From the **Model** drop-down list, select the clinical data model.

3. Select a validation check batch in the upper pane.

    The system displays information about its validation checks in the middle pane and information about its run history in the lower pane.

4. To view all jobs, click the **View Full Job History** icon.

    To view only recent jobs again, click the **View Recent Jobs Only** icon.

    To view log files, click the icon in the column for the type of job:

    •   **Log** (Run History pane) The most recent manually submitted job.

    •   **Triggered Job Log** (Run History pane) The most recent triggered job.

    •   **Install Job Log** (Validation Check Batch pane) The most recent installation of the validation check batch.

    > ✏️ **Note:**
    >
    > A validation check can be *disabled* so that it is not included in the batch execution. To find out if a check was included in the run, check the log file.

    See:

•   [Run a validation check batch](#)
•   [Cancel a validation check batch job](#)

Run a validation check batch

You must run validation checks as a batch.

1. Go to the 🏠 **Home** page, **Validation Checks** tab.

2. Select a validation check batch and click the ✅ **Submit Job** icon.

    > 💡 **Tip:**
    >
    > The **Submit Job** icon does not appear if the validation check batch is not installed. Check the Installed Status.
    >
    > If the installation status of a validation check batch is Warning, you may still be able to run the batch. Check the installation log file. If the status is Warning because one of the source tables is not used in the current study, the batch runs without input from that table.

3. Enter values:

    •   **Submission Mode**: Select one:

- **Full** includes data deletion. Use Full mode only if you are confident that you are reloading all current data.

- **Incremental** is faster and does not include data deletion.

- **Force Execution**: Select to run the job even though the source data currency, parameter values, and the version number of the program(s) have not changed since the last run. The system uses **Full mode** regardless of the Submission Mode setting. Full mode includes data deletion.

  If not selected and all the conditions are the same as the last run, the system does not execute the job and returns a status of Success.

- **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.

- **Trigger Downstream Transformations and Validation Checks**: Select this checkbox 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.

  > **Note:**
  >
  > This option appears only if the validation check batch is set up to allow it.

- Click the ↻ **Refresh** icon at any time for an update.

- To check the log file, click the icon in the **Log** column.

Cancel a validation check batch job

Select a validation check batch in the Run History pane and click:

- ☒ **Cancel Job** to cancel the currently running or next pending job.

- ✖ **Cancel All Jobs** to cancel the currently running or next pending job and all future jobs in the schedule for the selected validation check batch. Available only for scheduled jobs.

  > **Note:**
  >
  > If there is an immediate or deferred job for the same batch, or another schedule for the same batch, those jobs are not affected.

- ▤✗ **Cancel Triggered Job** to cancel the currently running or pending job. Available only for jobs that are set up to be triggered by another job.

Resend discrepancies that failed to be sent to InForm

Validation checks with an Initial Action of Send to InForm immediately send discrepancies they create to InForm as queries. To check the status of this operation:

1. Go to the 🏠 **Home** page, **Validation Checks** tab.

2. Click the 📋 **Failed to Send Discrepancies** icon in the upper right corner.

   The system displays counts for the selected study and lifecycle stage:

   • **Processing for *System*** shows the number of discrepancies currently being processed.

   • **Failed Processing for *System*** shows the number of discrepancies that should have been sent to InForm but weren't.

3. If any discrepancies failed to be sent, click the 📋 **Reprocess Discrepancies** icon to send them again. Click the ↻ **Refresh** icon to see progress.

What if...

• InForm data isn't loading?

• DMW is not sending discrepancies to InForm?

• InForm data isn't loading?

• DMW is not sending discrepancies to InForm?

InForm data isn't loading?

To load data:

• The most current version of the model must be installed.

• In Production, data loading must be resumed, not suspended.

• 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.

DMW is not sending discrepancies to InForm?

DMW sends discrepancies to InForm only if the lifecycle stages match. InForm's UAT lifecycle matches the Quality Control lifecycle in DMW.

If the lifecycle stages match and discrepancies still are not sent, see Resend discrepancies that failed to be sent to InForm.

FAQs

• Should I use Full or Incremental processing?

- Can I load data from any InForm lifecycle into DMW?
- Should I use Full or Incremental processing?
- Can I load data from any InForm lifecycle into DMW?

Should I use Full or Incremental processing?

See Data processing types and modes.

Can I load data from any InForm lifecycle into DMW?

You can load data from any InForm lifecycle into any DMW lifecycle except production, which requires that you assert that you are loading data from an InForm production lifecycle database.

Which InForm data is loaded depends on the setting of the Remote Study Account field. Its lifecycle stage is indicated in the InForm Lifecycle field.

8

Advanced topics

This section includes details on requirements and how to perform advanced tasks (for example, assign user groups to objects). See the following topics for more information.

- Naming restrictions
- Checkout and checkin
- Validation status and lifecycle stages
- Assign user groups to objects
- Snapshot labels
- Automatic triggering of transformations and validation checks
- Required syntax for metadata files
- Use SDTM identifiers to support important functionality
- How subject and visit filters work
- SDTM column equivalents in InForm
- How the system tracks data lineage
- Data blinding and authorization
- Data processing types and modes
- Installation
- InForm metadata change detection and synchronization
- Understanding internal objects and error messages about them

Naming restrictions

Make an object's name descriptive to help other users understand its purpose, but keep it short.

- Avoid special characters and reserved words
- Keep it short
- Automatic name truncation
- Duplicate names: system appends _1
- Naming studies and libraries
- Customizable naming validation package

Avoid special characters and reserved words

Do not use special characters such as ( ) - & @ * $ | % ~ except for underscore (_). Also **do not use Oracle SQL or PL/SQL reserved words**, especially in the Oracle name. For the latest information on reserved words, you can generate a list of all

keywords and reserved words with the V$RESERVED_WORDS view described in the *Oracle® Database Reference*.

> **Note:**
>
> - The system uses the value you enter in the Name field as the default Oracle name.
> - DUPLICATE is a reserved word in DMW for columns. It is used in the Default Listings page to allow filtering on duplicate records.

Keep it short

Short names work better both for display and for technical reasons.

- Keep all table Oracle names to 25 characters or less. This is because target tables contain an internal column for surrogate key information that contains all source table names.

- Windows has a maximum file path length 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.

- Keep container and object names short for integrated development environments

Keep container and object names short for integrated development environments

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 path for source code definitions, table instances, and the SAS runtime script. If the full path exceeds 256 characters, you get an error and cannot open the IDE or run the program. You can use a package to limit name length or display an error when the path is too long; see Customizable naming validation package.

The full path begins with the username of the person who opens the IDE followed by the directory name `cdrwork`. It also includes DMW_DOMAIN, the study grouping domain, and the study. See also Figure 8-3.

- **Source Code definitions path**:
  username>cdrwork>DMW_DOMAIN>*Study_Domain_name*>*Application_Area_name*>*Program_definition_name*>*Source_Code_definition_name*>*version_number*>*fileref*>*source_code_filename*

- **Table instances path**:
  username>cdrwork>DMW_DOMAIN>*Study_Domain_name*>*Application_Area_name*>*Work_Area_name*>*program_ instance_name*>*program_version*>*Table_instance_libname*> *Table_instance_SAS_name*>

- **SAS runtime script path**:
  username>cdrwork>DMW_DOMAIN>*Study_Domain_name*>*Application_Area_name*>*Work_Area_name*>*Program_instance_name*>*version_number*>setup

Automatic name truncation

When you create a new table by uploading a file, the system truncates the Oracle Name to 30 characters and also replaces the last two characters with the number `01` or the next sequential number.

Duplicate names: system appends _1

The system enforces unique naming for each object of the same type in the same container. For example, you cannot have two tables with the same name in the same model.

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), and so on.

Naming studies and libraries

Studies and Libraries in DMW are Oracle LSH domain objects. If you plan to export a domain to another DMW instance, avoid using spaces in its name. Domain names with spaces must be entered with escape characters surrounding them 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 standards. See "Customizing Object Validation Requirements" in the *Oracle Life Sciences Data Hub System Administrator's Guide*.

Checkout and checkin

To modify a clinical data model, table, transformation, or validation check batch, you must check it out. To save your changes, check it in.

If the **Check Out** option is not active, either the object is already checked out or you do not have the required privileges. To see who checked it out, look at the **Checked Out By** value near the top of the page. You may need to click the arrows.

When you check out an object, the system creates a new version of it in the Development lifecycle stage. You can promote the new version to Quality Control or Production. The existing installed version of the object in the QC and/or Production lifecycle area functions as before until you promote the new version to QC or Production and install it there.

Validation status and lifecycle stages

To install an object in the Quality Control or Production lifecycle stage, you must first promote it to the corresponding validation status.

1. On the 🏠 **Home** page, select the study and the Development lifecycle stage.

2. After you select a study, click the **Study Configuration** icon ✎ from the navigation bar.

3. Click the tab for the type of object you want to change the status of:

   - Clinical Data Models

   - Transformations

   - Validation Checks

4. Select the object and select **Modify Validation Status** from the **Actions** drop-down.

   > 💡 **Tip:**
   >
   > The object must be checked in and installed.

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

   - **Development**: Assigned by default to all new and checked out objects.

   - **Quality Control (QC)**: (Optional, corresponds to UAT in InForm.) For objects that are undergoing formal testing.

   - **Production**: For objects that are suitable for use in an active study. The system prevents destructive changes to tables and models in a production environment.

6. (Optional) Click the ➕**Add** icon in the Supporting Documents pane to add a document such as test results, a log file, a requirements document, or a signoff document.

7. Click **OK**.

   > ✎ **Note:**
   >
   > When a transformation is promoted to QC or Production, the system automatically promotes its source and target models to the same lifecycle stage.

8. Install the object in the new lifecycle stage:

   a. At the top of the page, change the lifecycle context to the same as the object's new validation status.

   b. Select and install the object.

> **Tip:**
>
> In QC and Production you can choose to display either latest installed version or an installable version, if any—a newer version that has been installed in a lower lifecycle and promoted to the current lifecycle but not yet installed in the current lifecycle.

Assign user groups to objects

Administrators can assign user groups to study groupings. Study configurators can assign user groups to studies, clinical data models, transformations, or validation check batches.

To have access to any object, a user must be in a user group that is assigned to it. The user's role in the group determines the user's privileges on the object. Studies, clinical data models, transformations, and validation check batches are all objects.

1. Select the object and click the ⚷ **Apply Security** icon.

   The Apply Security window lists all user groups 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 through inheritance.

     When a user group is assigned to a *study grouping*, all studies in the grouping, and all objects in those studies (clinical data models, transformations, and validation check batches), inherit the assignment.

     When a user group is assigned to a *study*, all clinical data models, transformations, and validation check batches in the study inherit the assignment.

   - **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.

   - **Unassigned**: The user group does not have access to the object. Its explicit assignment was undone.

   User groups that never had access to the object are not displayed.

2. In the **Assign To** field, select one:

   - **Metadata**: Required to create or modify studies, clinical data models, tables, validation checks, transformations, and custom listings.

   - **Development**: Required for creating, modifying, or executing the object in the Development lifecycle stage and loading or viewing data in the Development lifecycle stage.

   - **QC**: Required for testing or executing the object in the QC lifecycle stage and loading or viewing data in the QC lifecycle stage.

   - **Production**: Required for executing the object in the Production lifecycle stage and loading or viewing data in the Production lifecycle stage.

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, 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**.

Snapshot labels

You can use Oracle Life Sciences Data Hub to apply snapshot labels to data in a clinical data model in a particular lifecycle stage.

- Apply snapshot labels directly to data in tables
- Apply a snapshot label during job submission

Apply snapshot labels directly to data in tables

1. Log in to Oracle LSH.

2. Expand the Life Sciences Data Hub node in the main menu on the left or from the Navigator drop-down. Select **Applications**.

   Or, if Oracle LSH is already open, go to the Applications tab.

3. Navigate to the tables to which you want to apply a snapshot label:

   a. In the Application Development window, click the 🔍 Search icon next to the **Select Domain** field.

   b. Select Search By: **Domain Name**, enter `DMW_DOMAIN`, then click **Go**.

   c. Click the 📇 Quick Select icon for DMW_DOMAIN.

   d. Expand the node for the study grouping domain that contains the study.

   e. Expand the node for the study.

   f. Click the link for the *lifecycle* application area. A list of work areas, one for each installed clinical data model, appears. The model name is in the work area description.

   g. Click the link for the work area. A list of tables and other objects in the clinical data model appears.

4. In the **Actions** list, select **Manage Snapshot Labels** and click **Go**.

5. Query for the data:

   a. Select either:

      - **Dummy** to label masking values and nonblinded data.

      - **Real** to label sensitive, blinded data and nonblinded data.

    b. Select either:

        - **Most recent timestamp**, and then select a timestamp from the drop-down.

        - **Snapshot label**, and then select a snapshot label from the drop-down.

    c. Click **Search**. The tables that match the criteria are listed below.

6. In the **Snapshot Label to Add or Remove** box, enter a new label or search for an existing one.

7. Select the tables.

8. Click **Remove Snapshot Label**, **Add Snapshot Label**, or **Add/Move Snapshot Label**.

Apply a snapshot label during job submission

You can apply a label to source and/or target tables when you submit a program (transformation or validation check) or load set (data load) in Oracle LSH.

1. Log in to Oracle LSH.

2. Expand the Life Sciences Data Hub node in the main menu on the left or from the Navigator drop-down. Select **Applications**.

Or, if Oracle LSH is already open, go to the Applications tab.

3. Navigate to the tables to which you want to apply a snapshot label:

    a. In the Application Development window, click the 🔍 Search icon next to the **Select Domain** field.

    b. Select Search By: **Domain Name**, enter DMW_DOMAIN, then click **Go**.

    c. Click the 📇 Quick Select icon for DMW_DOMAIN.

    d. Expand the node for the study grouping domain that contains the study.

    e. Expand the node for the study.

    f. Click the link for the *lifecycle* application area. A list of work areas, one for each installed clinical data model, appears. The model name is in the work area description.

    g. Click the link for the work area. A list of tables and other objects in the clinical data model appears. DMW transformations and validation check batches are shown as programs. DMW file loads are shown as load sets.

4. Click the link of the program or load set in the Name column.

5. Click **Submit**.

6. In Apply Snapshot Label, select **Targets**, **Sources**, or **Both**.

7. In the Label field, enter the label.

8. Set all other parameters and click **Submit**.

Automatic triggering of transformations and validation checks

To ensure that DMW data is always as up-to-date as possible:

1. For each transformation, click the ⬚ **Add or Remove Source Models** icon and select **Can Trigger** for each source model.

2. When adding or modifying each validation check batch, select **Can Be Triggered**.

Each time data is loaded into an input data model, it triggers the transformation to the next target model, and then the next, and so on to all target models. In addition, all validation check batches are executed as soon as data is loaded into the model they read from.

Required syntax for metadata 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. This is the only automatic way you can create tables with all constraints and blinding attribute values.

> 💡 **Tip:**
>
> You can create a table initially from a metadata file and then load data into it from a SAS file.

> ✏️ **Note:**
>
> DMW supports a limited number of columns in clinical data model tables. The limit varies depending on whether the model is an inForm input model and on how many columns in a table are blinded.
>
> For file-based input data models and target models:
>
> - 260 if all columns are blinded
>
> - 370 if the table is not blinded, has only row blinding, or is completely blinded
>
> For InForm data models:
>
> - 339 regardless of how many columns are blinded
>
> The limit is due to the system's use of internal columns and the Oracle Database 11 limit of 1000 columns per table. Although it is possible to create tables with more columns, having too many columns may cause issues while loading InForm data, loading text files, transforming data, and displaying data in the Listings pages.

Delimiter: (Optional) Must begin `lsh_delimiter=` . If you do not specify a delimiter, the default delimiter is a comma (,).

Table: (Optional) Must begin `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 the 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. 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. See syntax below.

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 8-1 and Table 8-2.

> **✎ Note:**
>
> If you are using a metadata file to create a table **in LSH**, set attributes *Duplicate_PK_Support_Flag* and *Surrogate_Key_Flag* to `NO` or installation will fail. They are relevant only to DMW.

Table 8-1   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 8-2   Column Attributes with Reference Codelist Values

Attribute	Valid Values
Data Type	VARCHAR2, NUMBER, DATE
Nullable	YES, NO
SDTM Identifier	For valid values, see Use SDTM identifiers to support important functionality.
Blinding Type	TABLE, COLUMN, ROW
Masking Level	COLUMN, CELL

Example 8-1   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 8-2   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}
```

Use SDTM identifiers to support important functionality

In *all* clinical data model tables, it's important to link columns to SDTM identifiers in the **Filter/SDTM** field to support key DMW functionality:

- The **Automap** feature for transformations uses all SDTM identifiers.

- **Filtering** in the Listings pages requires the USUBJID and VISITNUM SDTM identifiers.

- **Filtering** in the Discrepancies pages requires the SUBJID and VISIT SDTM identifiers.

- To display the subject ID and visit name **in TMS** for TMS discrepancies, assign those SDTM IDs to the appropriate columns in any table you map to a TMS Set.

- **Subject Visit unit of work processing** requires the USUBJID and VISITNUM SDTM identifiers.

- **Subject unit of work processing** requires the SUBJID SDTM identifier.

Table 8-3   SDTM Column Identifiers Available for Use in DMW

SDTM ID	Meaning	Data Type	In Default Subject Visit Table?	In Default Subject Table?
~	**Actual Visit Date** (Not an SDTM variable but used as one by the DMW Automap feature.)	date	No	No
COUNTRY	**Country of Investigator Site**	varchar2	No	**Yes**
INVNAM	**Investigator Name**	varchar2	No	No
INVID	**Investigator ID**	varchar2	No	No

Table 8-3   (Cont.) SDTM Column Identifiers Available for Use in DMW

SDTM ID	Meaning	Data Type	In Default Subject Visit Table?	In Default Subject Table?
SITEID	**Site ID**	varchar2	No	**Yes**
~	**Site Name** (Not an SDTM variable but used as one by the DMW Automap feature.)	varchar2	No	No
STUDYID	**Study Identifier** is the unique ID of the study.	varchar2	**Yes**	**Yes**
SUBJID	**Subject ID** unique within study	varchar2	No	No
USUBJID	**Unique Subject ID** unique across all studies for all applications or submissions involving the product.	varchar2	**Yes**	**Yes**
EPOCH	**Visit Cycle** is an interval of time in the planned conduct of a study, associated with a purpose such as screening, randomization, treatment, or follow-up, that applies across all arms of a study.	varchar2	No	**Yes**
VISIT	**Visit Name** is the protocol-defined description of the clinical encounter or description of unplanned visit.	varchar2	**Yes**	No
VISITDY	**Visit Day** Planned study day of visit.	varchar2	**Yes**	No
VISITNUM	**Visit Number** is a numeric version of VISIT, used for sorting. Decimal numbering may be useful for inserting unplanned visits.	number	**Yes**	No
SVSTDTC	**Visit Start Date** (Start Date/Time of Visit) is the start date/time of a subject's visit, represented in ISO 8601 character format.	varchar2	**Yes**	No

How subject and visit filters work

When a user creates a Subject or Visit filter, he or she must specify a *filter driver model*, the clinical data model that contains the Subject Visit table to use. The table can be in any clinical data model in the study, but it must have the SDTM table identifier SUBJECTVISIT and the user must have View privileges on the table.

> 💡 **Tip:**
>
> Associate only one table per study with the SDTM identifier SUBJECTVISIT, so that users have only one choice when they create filters. If they create using different driver models and use them at the same time, in the Discrepancies page the system does not apply either filter.
>
> OR create public filters with the filter driver model set to the same model. Data reviewers can modify public filters instead of creating their own.

The filter logic checks this Subject Visit table and finds each distinct subject/visit combination that meets the criteria of the filter. It then creates a join with the table the user is viewing in the Listings page and displays those records. The filter logic uses different join columns on the Listings page and Discrepancies page, as shown below:

Table 8-4   Join columns required for subject and visit filters

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

Therefore, all tables with data that users may need to view must have columns linked to SDTM identifiers **USUBJID, SUBJID, VISITNUM, and VISIT**. Note that some of these names are used differently in InForm; see SDTM column equivalents in InForm.

In addition, to support all possible filters, the Subject Visit table must have columns linked to the SDTM identifiers **COUNTRY, SITENAME, SITEID, INVNAM, INVID, SUBJID, USUBJID, EPOCH, VISITNUM, VISITDY, VISIT** and **EPOCH**. Users can filter on one of these columns only if it exists in the Subject Visit table and is mapped to the corresponding SDTM identifier.

SDTM column equivalents in InForm

InForm was developed before SDTM and uses different names for equivalent columns.

Table 8-5   SDTM column equivalents in InForm

SDTM/DMW identifier	Equivalent InForm column name
USUBJID	SUBJID
SUBJID	SUBNUMBERSTR
VISNUM	VISID
VISIT	VISITMNEMONIC

How the system tracks data lineage

As data flows from input clinical data models to successive target models, the system stores information linking each data item to the data items that contributed to it from "upstream" input and target models and the data items it contributes to in "downstream" target models. Data reviewers can see a data item's source and target lineage in the Listings pages.

Maintaining this context, or *data lineage*, is required to pass discrepancies back and forth between DMW and its data sources and to recognize a discrepancy as the same discrepancy in all models.

The system uses the following:

- **Mappings**: The system uses the table and column mappings you define as part of a transformation to generate record-level data mappings during transformation and validation check execution.

- **Generated Surrogate Keys**: The system generates a *surrogate key* value for each record by concatenating a generated table identifier and 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 Keys**

  – When a clinical data model is installed, the system adds one auxiliary column named CDR$SKEY to each table to store the surrogate key value for each record in the table.

  – When a transformation program is installed, it adds one auxiliary column to each target table for each of its source tables, to store the surrogate key value of source records.

- Data lineage example

- Data lineage for deleted data

Data lineage example

In this example, multiple data items in several data models contribute to the calculation of each subject's Body Mass Index (BMI).
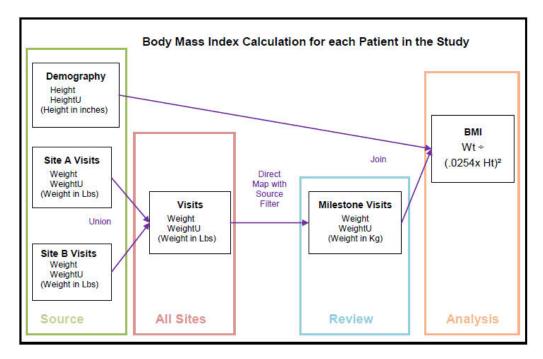
Figure 8-1   Body Mass Index Calculation Example

The calculations shown above result in the following data in table BMI in the Analysis model:

Table 8-6   Data in Table BMI

Study	Subject	Visit	Site	Gender	Age	BMI	BMIU
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. To investigate the source data for Subject 1007, a female aged 42 with a BMI of 33.1, the reviewer selects the BMI data value 33.1 for Subject 1007 in the Listings page and clicks **View Source Data**.

Figure 8-2   Source Data Lineage Display

The Trace Data Lineage window displays the selected data item in the top row, with its sources displayed below. Expand any source to see *its* source. Columns are displayed in the format `data_model.table.column`.

The user can see downstream data by clicking **View Target Data**. The display shows the selected item at the top and the downstream data below.

> ✏ **Note:**
>
> If the system cannot display the entire lineage, it displays an asterisk (*) with a message about possible reasons.

Data lineage for deleted data

When a record is deleted in InForm for which a discrepancy was created on a target data model in DMW, data lineage is broken, and after the target data model is reloaded, the discrepancy is no longer displayed in the source model in the Listings page.

The data reviewer can close or cancel such discrepancies in the Discrepancies page.

Data blinding and authorization

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

You define blinding in input data models either manually or when you create tables using metadata files. You can then cascade blinding and masking attributes to downstream tables as part of defining transformations.

If a transformation reads from a blinded source table, you must either blind the target table or, if you are sure it does not contain any data that should be blinded, authorize it. If a target table has a blinded source table and is not authorized or explicitly blinded, the system completely blinds the target table. Only users with Blind Break privileges can see any data in the table.

- Copying transformations with authorizations
- Authorization and side transformations
- Discrepancies on blinded data

Copying transformations with authorizations

When you copy a transformation from another study or model or as part of applying a study template, authorized tables are copied as **Not Authorized**. You must manually authorize them if appropriate.

Authorization and side transformations

When you create a side model or merge a side model back to the main transformation, the system copies the target table as **Authorized** if:

- There are blinded source tables and you have Blind Break privileges on all of them.
- The target table is not blinded.

Discrepancies on blinded data

At the time a discrepancy is created, the system checks if it should be viewable by all users with access to the table of the underlying data item or only to users with Blind Break privileges, and sets a flag for the discrepancy accordingly. The value of this flag does not change even if the table is subsequently blinded, unblinded, authorized, or unauthorized.

Data processing types and modes

DMW supports two types of data processing: Reload processing and Unit of Work processing (UOW). The unit of work can be either Subject or Subject Visit.

You set the data processing type for a table in its UOW Processing Type attribute in the clinical data model. Possible values include: **Non UOW** (Reload), **Subject**, or **Subject and Visit**. In a target table, this value determines the preferred processing type for transformations writing to the table. For UOW processing to actually occur in transformations, source tables must also be defined with UOW processing.

Both Reload and UOW processing can be run in either Full or Incremental mode. UOW Load mode is available for loading data. 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
- Unit of Work processing
- Data processing during data loading

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
- Incremental

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 UOW Processing Type must be set to either **Subject** or **Subject and 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.

Target tables defined for Unit of Work processing also accept Reload processing.

See Table 8-7.

- Full UOW
- Incremental UOW
- UOW Load

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 an execution set for those records' UOW key (subject or subject visit) and no others. It inserts, updates, and refreshes all records belonging to subjects or subject visits in the execution set and no others. However, if the program has never been run in any mode, all units of work are included in the execution set. 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 8-8.

> **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 8-7   Deletion Behavior in Unit of Work Processing Modes

Unit of Work Mode	Delete within Reloaded Unit?	Delete All Records for Nonreloaded Unit?
UOW Load	Yes	No
Full UOW	Yes	Yes
Incremental UOW	No	No

Data processing during data loading

The system supports loading data from SAS or text files or from InForm:

- Processing data loads from files

- Processing InForm data loads

See also Format checks on files being loaded.

You set up data loading when you define input clinical data models.

- Processing data loads from files

- Processing InForm data loads

- Format checks on files being loaded

- Supporting duplicate primary key values in a load

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.

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.
>
> – 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 8-8   Deletion Behavior in Data Loading Processing Modes

Processing Mode	Uses UOW Logic?	Deletion Behavior
UOW Load	Yes	Deletes nonreloaded records within units of work —subjects or subject visits—with new, changed, or refreshed records.

Table 8-8   (Cont.) Deletion Behavior in Data Loading Processing Modes

Processing Mode	Uses UOW Logic?	Deletion Behavior
Incremental Reload	No	Does not delete any data.
Full Reload	No	Deletes all records that are not reloaded 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 Create a clinical data model for InForm data for more information.

Format checks on files being loaded

In any data load processing mode, the system checks incoming data against the format requirements of target columns, including data type, length, codelist values (if the column is associated with a codelist), and the nullable and check constraints.

When you configure File Watcher for a table, you define the Reported Errors parameter. During a data load, the system counts errors in the file until it reaches the number set in this parameter plus one. It then stops processing the file, reports the errors in the log file, and the data load fails. If the number of errors detected for each file in the text data load is not more than Reported Errors, then the data load completes with a Warning status.

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, another row for the original error, and additional rows for any other errors it finds on the same record. For the purposes of calculating the number of reported errors, the logic treats each record that can't be loaded as one error, even if it contains multiple errors.

Table 8-9 shows the error file entries for two records. Record 1 has two errors and Record 32 has one error. 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.

Table 8-9   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")

Table 8-9   (Cont.) Error file example

TABLE_N AME	FILE_NAME	REC_ NUM	COLUMN_N AME	VALUE	ERROR_MESSAGE
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_E RROR	Captured in the TextLoad.log	ORA-20100: ORA-02290: check constraint (("CDR_W37_1D0156B9.TXT_TV48662730 1 .AGE_CK) violated
DEMOG	DEMOG.txt	32	AGE	05	ORA-02290: check constraint (("CDR_W37_1D0156B9.TXT_TV48662730 1 .AGE_CK) violated

Supporting duplicate primary key values 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 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 How the system tracks data lineage.

Installation

After creating or modifying a clinical data model, transformation, validation check batch, or custom listing, you must *install* it to make it usable.

- What happens during installation?
- Installation requirements

What happens during installation?

When you install a clinical data model for the first time, the installation process creates a database schema for the model. As the following objects are installed, the system adds them to the same schema:

- Clinical data model installation
- Transformation installation
- Validation check batch installation

Clinical data model installation

The installation process:

- Creates or updates a database schema for the model.
- Creates or updates database tables based on DMW table metadata if the most recent version is not already installed.
  - **Full installation** drops and recreates all tables and deletes all data.

    To safeguard your data, full installation is not allowed in the Production lifecycle stage.
  - **Regular installation** upgrades tables and does not delete data unless destructive changes have been made to a table. See Destructive and nondestructive changes.
- Creates or updates a query that selects all columns and rows from all tables in the model, including any masked data. The system uses this view to display data on the Listings pages.
- Checks the compatibility of validation checks and transformations that read from it or write to it. If changes to the model affect a validation check or transformation, the installation process gives a warning. Even if there are no issues, the installation process sets Upgrade to Required for the validation check and transformation to keep all current versions synchronized.

> **Note:**
>
> Clinical data models are also installed when the transformation that writes to the model is installed.

- [Destructive and nondestructive changes](#)

Destructive and nondestructive changes

During a regular, upgrade type of installation of a clinical data model, if there have been destructive changes to any table, all data in that table is deleted.

Destructive changes include:

- Removing a column.
- Decreasing the length of a column.
- Changing the processing type.
- Changing the CREATE_AS_VIEW flag.
- Changing the tablespace name.
- Changing the primary key.
- Changing the "allow duplicate rows" setting.

Nondestructive changes include:

- Increasing the length of a column.
- Changing the data type of a column from number or date to character.
- Changing the Blinding attribute of a table from Yes to No or No to Yes.
- Changes to a table's Support Duplicate attribute from Yes to No or No to Yes.
- Adding a unique constraint.

Transformation installation

The installation process:

- Generates or updates a PL/SQL program for each table-level transformation and corresponding packages in the database.
- Links the transformation programs to the source and target models and upgrade-installs the models if they are not already installed.
- Checks out the target tables and generates the auxiliary columns required to maintain the source system context and data lineage tracing for each record, if they have not already been created. See How the system tracks data lineage.

> **Note:**
>
> Because of this, if the table-level transformation has never been installed, the target model must be either:
>
> – Checked in.
>
> – Checked out by the same user who is installing the transformation.
>
> If not, the installation fails. So if you are installing a table-level transformation for the first time, see if the target model is checked out. If it is checked out by a different user, ask him or her to check it in before you install the transformation.

- Installs the target model in upgrade (regular) mode.

    Even if you select Full installation to install the transformation, the behavior is the same as for regular installation. If and only if a table has had destructive changes, the installation job drops and recreates the table and deletes all its data f rom those tables.See Destructive and nondestructive changes.

Validation check batch installation

The installation process:

- Creates or updates database packages for the generated programs.
- Maps the validation check batch to the latest installed version of the source clinical data model.
- Creates or updates and installs the target tables to store the discrepant data rows.

> **Note:**
>
> Installing a validation check batch does not install the source clinical data model. You cannot install a validation check batch until the source model is installed.

Installation requirements

Objects are not installable until they meet certain requirements.

- Clinical data model installation requirements
- Transformation installation requirements
- Validation check batch installation requirements

Clinical data model installation requirements

- All tables must be installable. A table is not installable if it has no columns.
- All tables must have a primary key constraint.
- All tables identified as blinded must have blinding completely defined.

Transformation installation requirements

- The transformation's status must be Complete. If it is Incomplete, at least one table or column in the target model is neither mapped nor marked Not Used.

- All source and target tables must be installable. (See Clinical data model installation requirements.)

- If it has a custom program, the program must be installable. A program is not installable if it does not have source code or source and target tables.

- All expressions must have valid code.

Validation check batch installation requirements

- It must have at least one validation check.

- Its source tables must be installable.

- If it has a custom program, the program must be installable.

InForm metadata change detection and synchronization

When a protocol change requires changing study metadata (for example, a CRF item is added in Central Designer and then InForm) DMW can detect and propagate the change.

- Compare DMW and InForm metadata on demand
- Automatic metadata change detection

Compare DMW and InForm metadata on demand

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

1. After you select a study, click the **Study Configuration** icon from the navigation bar. Then click the **Clinical Data Model** tab.

2. Select the InForm clinical data model.

3. In the **InForm Configuration** tab, click the **Compare Metadata** icon.

4. In the **Metadata Comparison** window, select the metadata to compare:

   - For the DMW InForm data model: Development, Quality Control, or Production.

   - For InForm: Development, UAT, or Production.

5. Click **Compare**. The report appears on screen.

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

   If the system finds no differences, a message appears.

   To make the changes in DMW, click the **Load InForm Metadata** icon.

Automatic metadata change detection

The system compares metadata during the following processes:

Table 8-10   Automatic Metadata Comparison and Results

Comparison Done During	Result If Significant Differences Exist
Data loading.	Message "Metadata needs to be reloaded" is displayed and data loading is suspended. If it is in the production lifecycle, a message is displayed on the Home page.
Promoting an InForm model to a higher validation status. DMW compares model metadata to metadata in the InForm UAT or Production environment specified in the InForm clinical data model.	A message is displayed and the promotion is not allowed.
Saving changes to a remote location or study account.	Results of the comparison are displayed.
Metadata loading.	See Metadata change detection during metadata loading.

If there are differences and you accept them, click the **Load InForm Metadata** icon in the InForm Configuration tab of the InForm clinical data model to synchronize DMW with InForm.

> **Note:**
>
> The InForm reporting database extract (RDE) views are the source of truth for 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. In that case, they also remain in DMW even after reloading metadata.

• Metadata change detection during metadata 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 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.

Understanding internal objects and error messages about them

Many DMW error messages and log files refer to internal objects. To help you understand these messages, here is a description of the objects. See Figure 8-3 for more information.

- Primary objects
- Container objects
- Secondary objects

Primary objects

Each time you set up a clinical data model, transformation, validation check, or custom program, the system creates a specialized object to do the work.

Each object listed in Table 8-11 is really two objects:

- The **object definition** is located directly in the study domain and includes almost all the object's metadata.

- The **object instance** is located in a work area and contains a reference to the object definition. The object instance is *installed*, creating a database table, view, and/or package.

Table 8-11   Primary internal objects

Object	Purpose	Created when a user...
Table	Holds data.	Creates a clinical data model or table. Also when a user installs a validation check batch, creates validation check target tables.
Load set	Loads data into all tables in one clinical data model.	Creates an input clinical data model. (Load sets are used for file data loading and for InForm internal metadata and operational data models, but not InForm study data or metadata, which are handled by the DMW InForm Connector.)

Table 8-11 (Cont.) Primary internal objects

Object	Purpose	Created when a user...
Program	Contains the source code for a transformation, validation check, or custom listing.	Creates a transformation, validation check, or custom listing.
Data mart	Contains a set of views, one for each table in a clinical data model. When generated, produces a set of files containing all the data in the model.	Creates a clinical data model with the data extract option selected.
Business area	Contains a view to each table in a clinical data model for read-only access for data visualization tools.	Creates or modifies a clinical data model with the Business Area option selected.
Transformation maps	Used for mapping by transformations, validation checks, and custom listings. For transformations, model-level maps contain table-level maps, which contain column-level maps. For validation checks, batch-level maps contain validation check-level maps, which contain column-level maps. For custom listings, listing-level maps contain column-level maps.	Creates a model- or table-level transformation, validation check batch, validation check, or custom listing.
Codelists	Contains a list of valid values that can be assigned to a table column.	Creates a codelist.

Container objects

Some objects are simply containers, or namespaces. See Figure 8-3 for more information.

Table 8-12 Container internal objects

Object	Purpose	Created when a user...
Domain	Contains all other objects, including other domains. The shipped DMW_Domain contains all DMW objects. There is a domain for each study grouping, and a domain for each study inside a study grouping domain. DMW REFDATA and DMW UTILS are both shipped domains. Study domains contain the primary object *definitions* for clinical data models, transformations, validation check batches, and custom listings for their study.	Creates a study or study grouping.
Application area	DMW creates an application area for each lifecycle stage, inside each study domain.	Creates a study.

Table 8-12   (Cont.) Container internal objects

Object	Purpose	Created when a user...
Work area	DMW creates a work area for each clinical data model in each lifecycle stage, inside the lifecycle application area. Contains primary object *instances* for the study lifecycle area, including tables, the load set that writes data to its tables (if it is a file input model), the programs that read from its tables (including validation checks, data marts, and business areas) and the transformation programs that write to its tables (if it is a target model).	Installs a clinical data model.

Secondary objects

The primary objects contain secondary objects.

Table 8-13   Secondary internal objects

Object	Purpose	Created when a user...
Execution setup	Contains parameters used for running an executable object.	Creates an executable object: Load Set (input clinical data model), program (transformation, validation check, custom listing), or data mart (data model with data mart option).
Parameters and Parameter sets	Parameter sets contain parameters and are contained in all executable objects. A parameter contains a reference to a variable and additional metadata.	Creates an executable object.
Table descriptor	Table metadata that is owned by executable objects and used to map them to source and target tables.	Creates an executable object.
Source code	Container for the actual program code for an executable object.	Creates an executable object.
Column	Contains column metadata and a reference to a variable.	Creates a table.
Constraint	Contains constraint metadata.	Creates a constraint.
Variable	Contains metadata used by both columns and parameters.	Creates a table or executable object.

Figure 8-3   Object Ownership