Oracle Health Sciences Data Management Workbench

Study Builder Recommendations

Release 3.3
F74688-01
February 2023

ORACLE

Oracle Health Sciences Data Management Workbench Study Builder Recommendations, Release 3.3

F74688-01

Contents

1 General recommendations

Create and modify study objects in the DEV lifecycle only

You should only create and modify study object in the Development lifecycle as described:

- **Recommendation**: Use the Development lifecycle to build your study, then promote objects to QC and Production.
- **Rationale**: Underlying LSH functionality supports this usage model. QC is the only place where you have a reasonable ability to verify how objects will behave when you promote them to Production. In addition, when you check an object out of the QC or Production lifecycle, you need to re-promote it to QC and back to Production.
- **Additional information:** If you check out a model, transformation or validation check in QC or Production, the existing installed version of the object in the QC and Production lifecycles continues to function until you promote the new version to QC or Production and install it there.

We recommend that you use each lifecycle in the following ways:

- **Development**: Create clinical data models, transformation programs, and validation checks. You can create them manually or from a library, a study, or a study template. Load data into the Development lifecycle schema, and do initial testing there.
- **QC**: Formally test study components (optional). This is equivalent to the UAT InForm environment.
- **Production**: Load, review, and clean production study data. The system prevents destructive changes to tables and models in a production environment.

Use the DMW user interface to work with objects in the DMW_DOMAIN

- **Recommendation**: When working with the DMW_DOMAIN and the DMW_UTILS sub-domain, do only the following in the LSH user interface:
  - Create therapeutic areas as sub-domains of the DMW_DOMAIN
  - In the DMW_UTILS sub-domain:
    - Create custom functions.
    - Store custom program definitions.
    - Create LSH tables and programs that you want to use in your DMW studies.

For all other tasks, use the DMW user interface.

- **Rationale**: Working outside of the DMW user interface to update underlying LSH objects or create objects in DMW's hierarchy is not supported. Doing so can cause:
  - Synchronization issues between DMW and the underlying database objects.
  - Instability in the LSH/DMW environment.
- **Additional information:** Never create any objects, including data marts, load sets, tables, programs workflows or business areas in DMW_DOMAIN. Provided with the correct security, data stored in DMW is still accessible to domain hierarchy outside the DMW_DOMAIN that can establish Views to DMW_DOMAIN tables. As a result, data marts and other LSH objects can be used against the data that's stored and cleaned in DMW.

  DMW creates all necessary application areas, work areas, programs, tables and additional objects to function as described in the Oracle documentation.

Do not perform tasks in the DMW_DOMAIN work areas if check out required

In general, do not perform a task in the DMW_DOMAIN work areas if the task requires you to check out a work area as described:

- **Recommendation**: The exception to this rule is any task that requires you to check out a work area that is used for a custom function, and the work area exists in an application area in the DMW_UTILS sub-domain.
- **Rationale**: DMW cannot anticipate its underlying database objects being checked out from outside its application. The unexpected object states cause system instability.
- **Additional information:** Only perform actions in the DMW_DOMAIN under direct instruction from Oracle Support or Oracle Product Engineering when they determine it is necessary for the resolution of a Service Request.

2
InForm integration

After a LIVE deployment, do not remove forms or items from your InForm study

- **Recommendation**: Do not remove forms or items after you deploy your InForm study to LIVE.
- **Rationale**: When you remove a form or item after a LIVE deployment, the database column that corresponds to the removed form or item is removed from the reporting schema for the study. This causes destructive changes to the InForm data model in the DMW Production lifecycle.
- **Additional information:** Instead of removing a form:
  - Add a global condition to the study, and include a triggering condition for the form that can't be met. As a result, the form is not triggered to appear for additional subjects.

    For example:

    ```
    this.<visitrefname>.<formrefname>.<sectionrefname>.<itemrefname>.Empty && 1==2
    ```
  - Make sure that the refpath that you use is for a valid item in the study so that the expression is properly validated and attached to the study design.

  Instead of removing an item:

  - Create an in-place study revision and configure the following properties for the item:
    * **Display Override** - Hidden
    * **SDV Required** - False
    * **Required** - False

After a LIVE deployment, do not modify form or item RefNames

- **Recommendation**: Do not modify form or item RefNames.

- **Rationale**: If you modify a study object's RefName, its column is removed from the reporting database schema, and a new column is created for the new RefName. This is a destructive change to the integrated InForm data model in DMW.
- **Additional information:** Instead of modifying a form or item RefName in the study design, rename its corresponding column in the target data model in DMW.

Re-load study data after each round of testing

- **Recommendation**: If you integrate InForm UAT or TRN studies with DMW in the Development or QC lifecycles, re-load study metadata after each round of testing to ensure that you are always using the latest study version.
- **Rationale**: InForm UAT or TRN studies might contain columns that don't exist in the Production study, or column properties that differ substantially. If there is a mismatch between the metadata in DMW and the metadata in the study's database, DMW doesn't allow you to promote the study to the Production lifecycle.
- **Additional information:** You can run the Meta Data Difference Report to determine if there are differences between the InForm Production database and the InForm data model in the DMW QC lifecycle.

Keep the InForm data model in sync between the Development and QC lifecycles

- **Recommendation**: Refrain from promoting your InForm data model until you are ready to QC all data models and transformations. After you promote, make sure to continue promoting all changes so that the study in each lifecycle is consistent.
- **Rationale**: It is important to keep the InForm study versions as consistent as possible between lifecycles. Interim changes that are not promoted to higher lifecycles can cause deprecated object references, which creates objects in invalid states in the InForm QC data model.
- **Additional information:** If you add or remove forms during review rounds, each lifecycle that has a defined data model should see the different metadata versions in order to properly manage the changes in each lifecycle.

Do not promote standard libraries or templates to Production

- **Recommendation**: Use standard libraries and templates in Development and QC lifecycles only.
- **Rationale**: When you promote a study to the Production lifecycle, you can no longer make destructive changes to it. However, because standards are updated often, there is a reasonable expectation that you might want to make changes that aren't allowed in Production. Maintaining these libraries and templates in QC provides you with the freedom to make those significant changes.
- **Additional information:** Standard libraries are not intended to collect Production data, so this recommendation should not limit the functionality of those studies.

Run a full data model installation after an InForm UAT study reset

- **Recommendation**: If you perform an InForm UAT study reset, we recommend that you run a full installation for the InForm data model.
- **Rationale**: An InForm UAT reset wipes out metadata and clinical data from the UAT study so that a new study design can be tested. In DMW, when you run a full installation, DMW drops and recreates all tables, and their data. As a result, you can make sure that objects and data that were removed in InForm are removed in DMW.
- **Additional information:** Good InForm SDLC dictates that UAT should be a simulation of the Live InForm study configuration. In order to keep this accurate, you must reset the InForm study in between rounds of testing so that only Central Designer deployments that were deployed or will potentially be deployed to Production are present and installed in the UAT study. These InForm resets wipe all data from the InForm study between rounds of testing, and when you redeploy, new unique IDs are created for metadata columns like VISITID, FORMID, ITEMSETID, etc.

  In DMW, tables in the InForm data model and in the InForm operational views do not drop data that was deleted in InForm after the study reset. The only way to make sure DMW data accurately reflects the current state of the InForm study is to run a full install to load the latest metadata has been loaded. The correct procedure for this is as follows:

  1. Reset the InForm UAT study.
  2. Deploy packages from Central Designer to InForm.
  3. In DMW, reload metadata changes to the Development lifecycle.
  4. Run a Full Installation in the Development lifecycle.
  5. Import clinical data to the Development lifecycle.
  6. Promote the study to the QC lifecycle.
  7. Run a Full Installation in the QC lifecycle.
  8. Import clinical data to the Development lifecycle.

  Note that you only need to promote the changes to QC and perform that Full install if you have already promoted InForm to QC before the change. This recommendation also assumes that both DEV and QC lifecycles are integrated to the same InForm UAT study.

3
Clinical One integration

Work with two-section form data in DMW

When you create a two-section form in Clinical One and have it flow from Datahub to DMW, the data is condensed into a single table, viewable in DMW on the listings interface.

Lab and two-section forms can be comprised of a flat section and a repeating section. The flat section of the form has a set number of questions with corresponding answers. The repeating part of the form allows whoever is entering the data to enter multiple instances of the question. For example, a two-section form collecting data about adverse effects to medicine may have a flat section containing questions about the medication name and dosage while the repeating part of the form allows a user to add several different items relating to the symptoms.

Since the data is flattened into a single grid in the listings view, it's important for you to know how to interpret the data correctly, as creating a discrepancy or a validation check using wrong type of form data could cause a discrepancy to end up in an irreversible purgatory between Clinical One and DMW.

> **Tip:**
>
> The NONREPEATING column is a quick way to check whether you're working with flat section data or repeating form data. The flat section form data rows are marked as "Y" in the NONREPEATING column and repeated form data rows are marked as "N".

Guidelines for working with two-section and lab forms

We recommend that you adhere to the following guidelines to ensure clean data and an uninterrupted flow between Clinical One and DMW:

- To create a discrepancy on a piece of data collected from the flat section of a form, make sure that the discrepancy is created on a row marked with Y in the NONREPEATING column. Flat section data is duplicated to the repeating section rows so it can be easy to create a discrepancy on the wrong record!
- Make sure you create discrepancies for the repeating form data on the combined record by looking for the N in the NONREPEATING column.
- Create separate Validation Checks for creating discrepancies on flat section vs. repeating section columns. Make sure to include NONREPEATING = Y/N in the Validation Check criteria accordingly.

- Take all of the above into consideration when you design transformations or custom joins for downstream data models. For example, if you create a downstream transformation data model which only has combined records (NONREPEATING column is marked as N) but is used to create discrepancies on flat section columns, the discrepancies would be internally linked to the repeating row. If the repeated row is subsequently deleted, the discrepancy would automatically close, even though it was created on the flat section column.

What if I want to create a discrepancy on flat section column data but accidentally select a record in a combined (NONREPATING = N) row?

A discrepancy created on a flat column but in a repeating row will display correctly in the DMW and Clinical One interfaces. However, the discrepancy will be internally linked to the repeating row record. If the repeating row record is deleted in Clinical One, the discrepancy is automatically closed.

What if I want create a discrepancy on data in the repeating column but end up selecting a record in a flat-only (NONREPEATING =Y) row?

A discrepancy created on a repeating column but in a flat-only record will only be visible in DMW. The discrepancy will only appear on the right-hand side panel of Queries listed on the subject visit page in the Clinical One interface.

4

External data loads

Set up file watchers and load data to Development and QC before you promote

- **Recommendation**: Follow the workflow of loading data to the Development and QC lifecycles before you promote to Production.
- **Rationale**: Loading data into all source models and performing a fully qualified system test in the QC lifecycle is the most effective way to determine whether a study is ready for Production.
- **Additional information:** The QC lifecycle is meant to be used as a validation lifecycle, and allows you to verify working objects beyond initial setup within the Development lifecycle.

Create input data models for new data sources by loading an externally-created MDD file

- **Recommendation**: When you add a data source whose metadata hasn't previously been used in a specified DMW environment, create the source data model by loading an externally-created MDD file.
- **Rationale**: It's quicker and less error prone to define columns and other data model properties in an MDD. In addition, it's relatively easy to modify and re-load an existing MDD file if you need to make changes.
- **Additional information:** Oracle Health Sciences Consulting has worked with customers to develop several different MDD generation macros and utilities to further increase the ease of working with MDDs outside of DMW.

Accurately define data sources

- **Recommendation**: Make sure to define data sources and associate them with the correct vendor.
- **Rationale**: Defining data sources for file-watched data, and assigning the correct vendor to the files they provided improves DMW's filtering functionality discrepancy handling.

- **Additional information:** You can select discrepancies with a common data source, and select Send to Spreadsheet to produce a file that lists the discrepancies for a single vendor. Properly defining data sources for input data models prevents cross-contamination of data issues from different vendors and the manual effort that is required to distribute discrepancies to the correct recipients.

5
Clinical data models

In DEV and QC, run a full data model installation only to revert destructive changes

- **Recommendation**: When you install data models in the DEV or QC lifecycle, run a full installation only to revert destructive changes.
- **Rationale**: A regular data model installation modifies existing structures, and a full installation performs a drop and recreate action on the structures.

  DMW does not allow you to make destructive changes to data models in the Production lifecycle. As a result, you should not need to run a full installation in the Production lifecycle.
- **Additional information:** When you're working in the Production lifecycle, we do not recommend using the full data model installation. However, you might need to run a full data model installation if you made several changes, deletions, or modifications to the data structure in the DEV or QC lifecycle.

  When you run a regular installation to install data models, DMW updates or upgrades updated tables. When you run a full installation to install data models:

  – DMW drops and recreates all tables.
  – DMW does not drop and recreate the data model schema.
  – LSH drops and recreates the data model schema.
  – LSH drops and recreates Business Area (BA) objects.

Install and promote data models after every change

- **Recommendation**: Each time you update a data model (for example by adding a table or column), run a regular data model installation, then promote and install to the QC lifecycle.

- **Rationale**: Installing and promoting data models after each update ensures that changes are installable, both individually and incrementally.
- **Additional information:** Installing and promoting ensures that the same versions of the data models exist in both the DEV and QC lifecycles. Keeping the lifecyles in sync can help to improve performance and reduce errors that can occur if you promote cumulative data model changes at one time.

Check out the associated transformation when you update a target data model

- **Recommendation**: When you update a target data model, check out its associated transformation as well.
- **Rationale**: When you update a target data model, if check out its associated transformation, DMW automatically upgrades the transformation to use the latest version of the data model. As a result, you do not need to manually upgrade the transformation, and you can easily ensure that the transformation uses the most up-to-date version of the target data model.
- **Additional information:** You are still required to upgrade maps if you modify the transformation's source data models.

Limit data model size

- **Recommendation**: Create data models with 100 tables or fewer.
- **Rationale**: Data models with more than 100 tables can cause performance issues.
- **Additional information:** If you need to, you can create smaller data models and leverage transformations to move the data. If your InForm study's source data model contains more than 100 tables:

  1. Divide the InForm Buffer Model so that each direct transformation between InForm and the Buffer handles fewer than 100 tables.

     This configuration allows several jobs to share the load and results in improved performance.

  2. Create a Union for common data sets (for example., the InForm ECG or PK sample forms) to a smaller number of target tables.

     This allows you to consolidate the multiple Buffer models into a single Aggregation layer.

     For more information about buffer models and aggregation layers, see the data model plan recommendations in the *Guidance for DMW Implementation and Configuration* document (Document ID: 2469980.1) on My Oracle Support.

Check out a data model to remove or update its business areas

- **Recommendation**: If you need to modify a business area (BA):

1. Check out its associated data model in the Development lifecycle.
2. Update the data model in the Development lifecycle.
3. Install the changes in the Development lifecycle.
4. Promote and install to QC and Production, as needed.

- **Rationale**: A business area is a property of a data model, and you can only change it by creating a new version of the data model.
- **Additional information:** Avoid modifying the business area schema in the LSH user interface. This is consistent with recommendations In DEV and QC, run a full data model installation only to revert destructive changes and Install and promote data models after every change.

Include a SUBJECTVISIT table

- **Recommendation**: To include a SUBJECTVISIT table in your study:
  1. Select one table in one of your study's data models, and set its SDTM Identifier property to SUBJECTVISIT.
  2. In the SUBJECTVISIT table, select a column to use for each of the following SDTM Identifiers:
     - USUBJID
     - VISITNUM
     - SUBJID
     - VISIT
  3. In the SUBJECTVISIT table, set the SDTM Identifier property for as many additional columns as you can.
- **Rationale**: The SUBJECTVISIT table is required to support discrepancy handling, filtering, and Unit Of Work processing.
- **Additional information:** We do not recommend defining more than one SUBJECTVISIT table per study, due to potential Unit Of Work considerations.

Use SDTM Identifiers for as many columns as possible

- **Recommendation**: For each table, assign SDTM Identifiers to as many columns as possible.
  - We recommend assigning USUBJID and SUBJID for subject-level tables.
  - We recommend assigning USUBJID, VISITNUM, SUBJID, and VISIT for subject-visit-level tables.
- **Rationale**: DMW/LSH indexes are based on the identifiers.
- **Additional information:** Mapping columns to SDTM Identifiers increases the filtering capability in the DMW Data Management user interface.

6
Transformations

Use the Regenerate program definitions option to rebuild program definitions in a transformation

- **Recommendation**: We recommend that you select Regenerate program definitions when you run a regular transformation installation if you want DMW to rebuild program definitions based on the latest version of the source and target data models in the transformation.
- **Rationale**: If you select Regenerate program definitions when you run a regular transformation installation, DMW does the following:
  - Rebuilds the program definitions based on the latest version of the source and target data models.
  - Regenerates portions of the program that were missing prior to the modifications.
- **Additional information:** If you are running a full transformation installation, we recommend that you only select Regenerate program definitions if absolutely necessary.

Remove a source model from its transformations before deleting it

- **Recommendation**: If you need to remove a data model from a study:
  1. Check out transformations that use the data model as a source model.
  2. For each transformation, click Add or Remove Source Models.
  3. In the Add Model dialog box, deselect the source model that you want to delete.
  4. Delete the data model from the study.

Avoid using the Undo Checkout feature for transformations

- **Recommendation**: Instead of using the Undo Checkout option for a transformation, check the transformation in and then check it out again.
- **Rationale**: When you use the Undo Checkout option, DMW might remove custom functions associated with the transformation. In addition, other issues related to expressions could occur.
- **Additional information:** We recommend tracking all changes as new versions in DMW. You can always roll objects back to their most recent Production version.

Use unique aliases

- **Recommendation**: When you use the auto-map feature, we recommend that you select a single target table. We do not recommend using the auto-map feature to map at the data model level.
- **Rationale**: Poor performance occurs when you auto-map more than one table at a time, and is worsened when you map to 10 or more tables at a time.
- **Additional information:** To ensure the best performance when you auto-map, use the Column Mapping user interface, and use the Shift+Select keyboard shortcut to run auto-mapping only for columns in a single table that have known map matches.

Do not copy a transformation that needs an upgrade

- **Recommendation**: When you copy a transformation from one study to another using the Copy from Another Transformation option, make sure that the transformation you want to copy does not require an upgrade.
- **Rationale**: A transformation that requires an upgrade references old data model definitions. As a result, if you copy it, you'll likely need to perform manual updates to the copy.
- **Additional information:** To further reduce the likelihood that you'll need to make manual changes to the copied transformation, we recommend that you copy transformations that were created by templates or study libraries. However, we also recommend that you manually confirm that the transformation doesn't contain maps that need to be upgraded.

Wait to create staging tables

- **Recommendation**: Create staging tables only when the Aggregation Layer and the desired Consumer model are defined, and the transformation is known.

  Avoid defining staging tables on the fly within the target data model.
- **Rationale**: The Create Staging Table functionality has limitations. As a result, we do not recommend using it.
- **Additional information:** The Guidance for DMW Implementation and Configuration document contains details for creating staging data models that partition the intermediate steps from the target models that are intended for consumption.

7

Custom listings

Do not add and remove columns simultaneously in a single modify custom listing save operation

Recommendation: Do not add and remove columns simultaneously while modifying public or private custom listings in a single save operation. Instead, custom listings should be modified in a two-step process. For example, first remove the desired columns and save the custom listing. Then, you can go back and modify the listings to add any desired columns.

Rationale: Custom listings become unreliable when columns added and removed simultaneously from one or more tables as part of a single modify custom listing operation.

Additional information: We recommend that you verify the modified columns along with the data when you use the Test Data feature.

Do not use staging tables as a source for custom listings

- **Recommendation**: When you create a custom listing, do not include columns from staging tables as its source.
- **Rationale**: Staging tables are intermediate steps in a data transformation. They are not intended to be used for data review.
- **Additional information:** If you use staging tables, we recommend that you use them to explicitly define the intermediate tables in a separate staging model, as specified in the Guidance for DMW Implementation and Configuration document.

  We don't recommend using staging tables due to the limitations of the feature.

8
Validation checks

Make sure to select a relevant discrepant column

- **Recommendation**: The discrepant column should always be selected as a column that is present in the listing.
- **Rationale**: When you select the Discrepant Table and Column for a validation check, DMW allows you to select any table and column, not just the ones that make sense in the context of the validation check.

Avoid renaming columns in VC listings

- **Recommendation**: In a VC listing, do not rename the columns. Instead, use the default Oracle name of the underlying data model column for the column names.
- **Rationale**: Renaming columns in a VC listing can lead to validation check performance issues, and the inability to modify the validation checks.
- **Additional information**: You might need to rename columns if:
  - Column names are not unique.
  - You need to handle self-referencing queries.

  In these cases, renaming is allowed.

Verify the validity of a validation check batch before you copy it

- **Recommendation**: Before you copy a validation check batch, to make sure that it's valid, verify that the columns in the source data model exist in the target data model.
- **Rationale**: DMW allows you to copy a validation check batch that does not have valid columns in the destination data model. We recommend that you ensure that the validation check is valid before you copy it.
- **Additional information:** We recommend that you only copy validation checks from approved templates or study libraries. In addition, if a validation check is deprecated, we

recommend that you put in place a documented process to make sure that they aren't propagated to other studies.

Copy valid validation checks

- **Recommendation**: Avoid copying validation checks that are disabled or that require an upgrade.
- **Rationale**: Validation checks that require an upgrade might not include necessary columns from the data models, or have other un-resolvable issues.
- **Additional information:** We recommend that you only copy validation checks from approved templates or study libraries. In addition, if a validation check is deprecated, we recommend that you put in place a documented process to make sure that they aren't propagated to other studies.

Copy blinded validation checks to blinded tables

- **Recommendation**: Copy blinded validation checks to blinded tables and non-blinded validation checks to non-blinded tables.
- **Rationale**: Columns in a VC listing retain the blinding properties of their underlying data model. If there is a mismatch between the blinding of the validation check and its tables, DMW will report errors.
- **Additional information:** We recommend that you define libraries with a blinded and unblinded validation check definition. You can then copy the validation check with the appropriate blinding setting for your purposes.

Do not remove a source table from a validation check

- **Recommendation**: If you need to remove all of the columns in a table from a validation check, do not modify the validation check. Instead, disable the existing validation check and create a new validation check that excludes the table.
- **Rationale**: The validation check program does not gracefully remove source tables when you update the validation check.

Do not use staging tables as a source for validation checks

- **Recommendation**: When you create a validation check, do not include columns from staging tables as its source.
- **Rationale**: Staging tables are intermediate steps in a data transformation. They are not intended to be used to raise discrepancies.
- **Additional information**: If you use staging tables, we recommend that you use them to explicitly define the intermediate tables in a separate staging model, as specified in the Guidance for DMW Implementation and Configuration document.

  We don't recommend using staging tables due to the limitations of the feature.

9

Custom programs and functions

Avoid using custom programs

- **Recommendation**: Whenever possible, define a transformation using the DMW user interface, including standard Oracle SQL and custom functions, as opposed to custom programs.
- **Rationale**: If you use custom programs, you are responsible for the coding. This isn't a scalable approach. In addition, custom programs are not as re-usable as transformations that you define in the DMW user interface.
- **Additional information**: For more information about using custom functions, see the Guidance for DMW Implementation and Configuration document.

  The only scenario in which we recommend using a custom program is when defining the Filter Drive table, which is marked with the table SUBJECTVISIT SDTM Identifier. In an ideal study design, that is the only custom program used.

To create table descriptions for a custom program, copy existing table definitions in LSH

- **Recommendation**: To create table descriptors for a custom program, in LSH, copy the study's existing target and source table definitions to the local application area. Do not use the existing definitions from the study application area for the custom program.

  > **Note:**
  >
  > You do not need to do this if the source and target tables are the same table definition.

- **Rationale**: The study's existing target and source table definitions typically provide the most accurate description of the tables. In addition, following this guidance allows for a consistent approach when underlying table definitions are updated.

- **Additional information**: When you reference a study-specific table to create a table descriptor, make sure that the table descriptor does not reference the study, and that you copy the definition to the local application area.

  This method is a reliable, efficient, and repeatable way to create table descriptors. If you need to update or recreate them, you can recreate the exact process and the definition is a reliable description of the circumstance in the study.

  If you used the same definition for the source and target tables, load an MDD file that describes the table as the table descriptor for both source and target.

Use cloning to create study-specific custom programs

- **Recommendation**: To create a study-specific custom program, clone an existing custom program to a study-specific application area in the DMW_UTIS domain, and update the clone with the study-specific information.
- **Rationale**: When you clone a custom program, you can update table descriptors that are specific to a particular study without modifying the initial custom program or creating one from scratch.
- **Additional information:** Do not use one application area as a common repository for all studies. Instead, use LSH cloning to duplicate the core program definition for specific studies.

Include only columns for the Listing user interface in a custom program for a validation check

- **Recommendation**:
- **Rationale**: When you define a custom program for a validation check, include only the columns that you want to appear in the Listings user interface. Additional columns can cause processing issues.
- **Additional information:** You must include PK and _SKEY columns in the custom program.

Specify a static reference for secondary source code

- **Recommendation**: When you add secondary source code to a custom program in LSH, make sure to select Yes for the Static Reference field.

  > **Note:**
  >
  > Yes is not selected by default.

- **Rationale**: If you do not specify a static reference, LSH defines a new definition, which is typically not the intent of secondary source code.