CA Gen

Host Encyclopedia Version Control User Guide

Release 8.5

CA Technologies Product References

This document references the following CA Technologies products:

- CA Gen

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Contents

Chapter 1: Host Encyclopedia Version Control

Host Encyclopedia Version Control

Host Encyclopedia Version Control provides a basic set of tools to share objects between models. Version control provides a way to maintain multiple models that represent the same system at different stages of development, testing, and production.

When an object is changed in one model, version control can transfer the changed object to other copies of the model. For example, an object that is changed in a development model can be transferred to the test and production copies of the model.

The version control lets you apply changes to the production model easily because you can copy only the directly changed objects without having to copy the entire model.

To implement production enhancements, control is maintained over Construction objects, as in traditional source program maintenance, and Planning, Analysis, and Design objects too. Using version control, production enhancements are as easy to maintain as a traditional source program (which simply copies updated programs from library to library).

Version control makes the coordination of large development efforts possible. Multiple models contain identical versions of objects. Changed the objects in one model can be copied to other models so that object definitions remain the same.

Because you can use multiple models in the development effort, more analysts can participate with less chance of causing contention problems for any one model or object.

You cannot transfer objects between two model families in two separate Encyclopedias.

Version Control Definitions

A model is a collection of objects that defines an information system.

A model family defines the set of models that are related. Typically, all the models in a family support one production model.

Migration, the most common use of version control, is the transfer of one or more objects from one model to another. Models not in the same model family are not related, and objects cannot be migrated between them.

Adoption of a model moves it from one model family to another. Adoption is required before migration if the models involved in the migration belong to different model families. Adoption within related models is occasionally required to synchronize a logical object that has been created independently in two models.

Model Unadoption lets you correct adoption errors, such as the accidental reversal of model names at the time of adoption, or remove all common ancestry before readopting a model into its family. Model unadoption removes the adopted model from the family into which it was mistakenly adopted and moves it into a new family.

An aggregate object is a collection of related objects and the smallest selectable unit in version control functions.

An aggregate set is a collection of aggregate objects that is created and saved for convenient reuse.

Common ancestry occurs when two objects in a model family have the same Original Object ID.

Version Control Functions

Version control consists of four major functions:

- Migration
- Adoption
- Model Unadoption
- Reports

Version Control Procedures

Version control provides these procedures and reports:

- Rename a Model Family
- Migrate Aggregate Objects to a New Model
- Migrate Aggregate Objects to an Existing Model
- Adopt Aggregate Objects Using Related Model
- Unadopt Model from Existing Family
- Compare Aggregate Objects Between Two Models

- Trial Migrate Aggregate Objects to an Existing Model
- Trial Adopt Aggregate Objects Using Related Model
- Aggregate Object Where Exists Report

Version control provides procedures for creating and maintaining aggregate sets:

- Retrieve Aggregate Set
- Add Aggregate Set
- Modify Aggregate Set
- Copy Aggregate Set
- Rename Aggregate Set
- Delete Aggregate Set
- Aggregate Set Object Report

Basic Concepts

Using version control requires an understanding of these basic concepts:

- Model family
- Aggregate the objects
- Aggregate the sets
- Common ancestry

Model Family

A model family defines a set of models that are related. Typically, all the models in a family support one production model.

Creating a Model Family

To create a model family, add a new model to the Host Encyclopedia or unadopt an existing model from its family. The encyclopedia creates a model family with the same name as the new model.

Adding New Members

You can then add new members to a model family by any one of the following methods:

- Model Copy to create a copy of a complete model
- Create Model from Subset to create a new model that includes only part of its parent model
- Migrate the objects to a new model (using Migrate Aggregate Objects to a New Model).
- Adopt a model from one family into another
- Unadopt a model from its family into a new family

A new model family member can be created from any existing member of the family-not only from the model that originally created the family. Model family members need not complete copies of each other. However, sometimes they differ only by whether they represent current production, development, or user testing.

Aggregate Objects

Using aggregate objects lets you identify objects that have changed and must be transferred between family members. Each aggregate object in a model represents a collection of related objects in much the same way that a scoping object does in subsetting. Although similar in some ways, aggregate objects and subset scoping objects have important differences.

Version control can define smaller, more specific expansions than subsetting. For example, version control can migrate an object as small as a single attribute, whereas to check out an attribute, subsetting must scope the entity type that owns the attribute.

The ability to define smaller, more specific expansions is an example of granularity. Version control is more granular than subsetting. The context requirement of version control is less stringent because the sole purpose is to transfer objects from one model to another.

For a list of aggregate objects that can be migrated, see Aggregate Objects for Migration *(see page 58).*

An aggregate object can have three parts:

- Component objects
- Companion objects
- Enabling the objects

Component Objects

Component objects are part of the basic definition of the aggregate object. For example, attributes are component objects of an entity type. Component objects are always migrated with an aggregate object. If an aggregate object is used for migration, its component objects must be migrated so that the aggregate object will have the same definition and context in the destination model as in the source model.

Component objects are similar to directly related objects in subsetting. In some instances, a component object of an aggregate object is also an aggregate object in its own right. For example, an attribute is a component of an entity type and also an aggregate object itself.

If the aggregate object in the source model has fewer component objects than in the destination model, migration deletes component objects from the destination model that do not exist in the source model.

Companion Objects

Companion objects are similar to neighborhood objects in subsetting. They are not part of the definition of the aggregate, but if the referencing aggregate object is migrated, companion objects are automatically migrated if they do not already exist in the destination model. However, you can also specifically request their migration.

The purpose of companion objects is to maintain the context in which the aggregate object exists in the source model. If an aggregate object's companion objects already exist in the destination model, they are not replaced. However, any companion objects missing from the destination model are migrated to maintain context.

Enabling Objects

Enabling the objects are objects in the destination model that make a successful migration possible. Likewise, the absence of enabling objects causes a migration to fail. An enabling object can:

- Have a parental relationship with the aggregate object

  For example, you cannot migrate an entity type unless its parent subject area is present in the destination model.

- The aggregate object references

  For example, you cannot migrate an action block if any of its views reference entity types that are absent from the destination model.

Common Ancestry

Two objects in different models shares common ancestry within a model family if the objects have the same Original Object ID. When a new object is initially uploaded from the toolset, it is assigned a unique identifier as its Current Object ID and this same identifier as its Original Object ID. This identifier is used internally to identify the object regardless of any changes that are made to it, including name changes. When a new object is created through migration, that object is assigned a unique Current Object ID but inherits the Original Object ID from the object on which it was based.

You can determine an object's Original Object ID by using the WalkEncy utility for that object and examining the ORG_ID column.

Migration propagates common ancestry. Consider an object that is created in Model 1, then migrated to Model 2, then migrated to Model 3. The following table defines the object's Original Object ID and current object ID after each migration:

Object IDs	Model 1	Model 2	Model 3
Original Object ID	123	123	123
Current Object ID	123	456	789

Aggregate Sets

An aggregate set is a user-defined collection of aggregate objects that has been saved for later use. Any aggregate object can be a member of any aggregate set without affecting its status within the model.

An aggregate set is built from the aggregate objects in one particular model. Once built, the set can be used with any model in the family. For example, an aggregate set that is built from Model A could later be used to migrate objects from Model B to Model C.

Create an aggregate set for:

- Trial migration-Modify the set to obtain the preferred migration result. Use the set when making the final migration.
- Migration-Use the same set later for migration to other models.

Migration between two projects models-Use the set periodically to update the destination project model with new development work.

Chapter 2: Perform Difference Analysis

Difference Analysis

Difference analysis determines:

- Objects to select for migration
- That two objects with common ancestry are different
- Objects that do not exist in the other model

Difference Analysis Steps

Perform the following steps for difference analysis:

1. Identify the aggregate objects that you want to compare.
2. Run the Compare Aggregate Objects Report for each object that is identified in Step 1, or run the report for all aggregate objects.

   For details about interpreting the Aggregate Objects Compare Report, see Evaluate the Compare Report *(see page 24).*
3. Once the collection of aggregate objects that are associated with a change has been identified, it is necessary to determine the exact difference between those aggregates in different family members before performing any migrations. You can analyze detail differences by:
   - Generating standard system reports or custom reports that are based on the Public Interface Export and using guide or automated compare techniques.
   - Downloading appropriate subsets from both models and using the workstation toolset to review the differences.

Session of Change Comparison

Session of change comparison is an essential part of difference analysis. This section explains the concepts underlying change comparison and provides examples of the most common outcomes.

Changes occur to objects:

- At the toolset

  These changes are uploaded to the Host Encyclopedia.

- On the Host when users:

  Rename or delete objects using the Host Encyclopedia's Rename or Delete Object utility

  Package modules or generate code or databases

An object undergo one or more changes. If an object does change, the change is associated with the session object created by the changing process-using the DIRCHGD association in the Host Encyclopedia.

A session is an object that is used by version control to track changes. Every aggregate object is associated with a session. If the object changes again, a new session is created; the object is associated with the new session and disassociated from the old session.

Session properties include:

- Unique session ID
- Date and time of the change session (also known collectively as the change timestamp)

The change timestamp and session ID constitute a session of change that uniquely records the last change to the aggregate objects associated with the session.

An occurrence of a session object remains associated with an aggregate object until the aggregate object changes again.

In its lifetime, an aggregate object can be associated with many different sessions-but it is associated with only one session at a time using the DIRCHGD association-the session that reflects the last update to the object.

Note: Migration carries the session object from the source model to the destination model.

Difference Analysis Reports

The Difference Analysis Reports are essential in preparing for a migration. They are:

- Aggregate Object Where Exists Report
- Compare Aggregate Objects Report

Generate the Aggregate Object Where Exists Report

The Aggregate Object Where Exists report displays the models within the model family where selected aggregate objects exist. Common ancestry with the selected object is used to identify the existence of the object in other models.

To generate this report, you must have at least Read Only authorization on all models in the family.

Steps to Execute Where Exists Report

Follow these steps:

1. Access the CA Gen Main Menu.
2. Select options 1.2.9. Press Enter.
3. The Specify Model Name for Where Exists Report panel appears.
4. Specify the name of the model containing the aggregate objects.
5. Type **/** (slash) next to the execution option you want:
   - Online
   - Batch

   Press Enter. The Aggregate Set Retrieval panel appears.
6. To use a set, type the name of an aggregate set, type **Y** in the Retrieve Aggregate Set field and press Enter. To continue without using a set, leave the set name blank, the Retrieve Aggregate Set field at N, and press Enter.

   The Select Aggregate Object Types for Where Exists Report panel appears.
7. Type **/** (slash) by the aggregate object types to be expanded. To reset and begin the selection again, type CANCEL and press Enter.
   a. Press Enter. A message appears.
   b. Press Enter. A selection list appears with the occurrences of the first object type appears.
   c. Type **S** next to the aggregate object occurrences you want to select. Type **E** to expand an occurrence. Press Enter. A message appears.
   d. Repeat until all object types have been processed. The Confirm Aggregate Objects Selected for Where Exists panel appears.
   e. Type **/** (slash) next to any individual occurrences you want to remove.
   f. Press Enter.

      The Confirm Selected Aggregate Objects panel appears.

8. Type ACCEPT, SAVE, END, or CANCEL.

   ACCEPT or SAVE stores the set and ends processing. ACCEPT processes but does not save. SAVE processes and saves the set.

9. Press Enter:

   - If you selected online execution, the Where Exists report appears in Browse mode for review.
   - If you selected batch execution, the Batch JOB card maintenance panel appears. Modify the JOB card as necessary and SUBMIT the Job.

Generate the Compare Aggregate Objects Report

The Compare Aggregate Objects report compares the aggregate objects that are selected from the source model (and all their subordinate aggregate objects) to equivalent aggregate objects and subordinates in the destination model.

The comparisons have four possible results:

- The objects are the same.
- The objects are different.
- No equivalent object exists in the destination model.
- The object exists in the destination model but no equivalent object exists in the source model. This can occur when you select Compare All or when you select a high-level aggregate object that has more components in the destination model than in the source model.

To generate this report, you must have at least Read Only authorization on both the source model and the destination model. For comparison, the destination model must be in the same family as the source model.

The following three procedures provide step-by-step instructions for:

- Comparing all objects in two models
- Comparing selected objects and creating an aggregate set of those objects that meet the criteria you select, where options include:
  - Objects that are different
  - Objects that do not exist in the destination model
  - Objects that do not exist in the source model
- Comparing the objects in an existing aggregate set and, optionally, updating the set with the results of the comparison

Compare All Objects Without Aggregate Set Input or Output

The simplest Compare Report to run compares all objects in the source model with those in the destination model. Generating this type of report does not involve use of aggregate sets.

Use this option to create a single-use baseline report on all objects in the two models.

Steps to Generate a Compare Report on All Objects

Follow these steps:

1. Access the CA Gen Main Menu. Select options 1.2.6. and press Enter.
2. On Compare Source Model Name panel, type the name of the source model and press Enter.

   Alternatively, tab to the field and press the prompt key for a list of valid choices. Select a model from the list by typing any character except period (.) next to one name and press Enter again.
3. Complete the Compare Destination Model Name panel.
   a. Type the destination model name.
   b. Type **/** (slash) next to All Objects
   c. Type **/** (slash) next to the execution option you want.
      - Online displays the report on your terminal.
      - Batch prints the report on the specified output device.
   d. Press Enter.

      The Compare Report Aggregate Set Create/Modify panel appears.
   e. Accept all defaults to bypass this feature, and press Enter.
4. Evaluate the Confirm Model Names for Compare All Aggregate Objects. If the source and destination model names and execution mode are correct, press Enter.
   - If you selected All Objects and Online execution, the Compare Report appears for browsing. After the browsing, press F3 to display the Report Print Options panel. Complete this panel to print the report.
   - If you selected All Objects and Batch execution, the Batch JOB card maintenance panel appears. Modify the JOB card as necessary and SUBMIT the Job.

Note: The Compare Aggregate Objects report lists objects in the following three categories. Objects that are the same in both models are not reported.

- The following objects do not exist in the source model.
- The following objects do not exist in the destination model.

- The following objects are different.

Select Objects to Compare and Create Aggregate Set

When you expect to perform multiple version control functions with the same set of objects or when you expect to perform one function multiple times with slight modifications, it is more efficient to save your object selections in an aggregate set than to reselect for each usage. To create an aggregate set automatically from Compare the results saves you the time of creating the set from the objects that are listed in one or more sections on the report. Even if more objects are reported than you want to use, it is frequently easier to delete those than to create the set from scratch.

Steps to Create an Aggregate Set from Compare

Follow these steps:

1. Access the CA Gen Main Menu. Select options 1.2.6. Press Enter.
2. On Compare Source Model Name panel, type the name of the source model and press Enter.

   Alternatively, tab to the field and press the prompt key for a list of valid choices. Select a model from the list by typing any character except period (.) next to one name and press Enter again.
3. Complete the Compare Destination Model Name panel.
4. Type the destination model name.
   a. Type **/** (slash) next to Selected Objects for the range of objects.
   b. Type **/** (slash) next to the execution option you want.
      - Online displays the report on your terminal.
      - Batch prints the report on the specified output device.
   c. Press Enter.

      The Compare Report Aggregate Set Create/Modify panel appears

   Note: When you specify Selected Objects, the report includes a list of all objects that are the same in both models. If you select All Objects, the reports will not include objects that are the same.

5. Create an aggregate set composed of the objects that are identified by the Compare Report in an aggregate set:
   - Change the Save in aggregate set default N to Y
   - Enter a unique name for the new aggregate set
   - Specify the aggregate set option by typing / next to the option describing objects to save in the set.
   - Use the following tips to help in the process:
     - If creating an aggregate set to migrate object changes that are made in the source model to equivalent objects in the destination model, select Objects in both but different.
     - If creating an aggregate set to migrate new objects that are created in the source model to the destination model, include the option Objects in Source but not in Destination.
     - If creating an aggregate set to migrate new objects that are created in the destination model to the source model, select only the option Objects in Destination but not in Source.

     Note: When you migrate, you specify as the source model the model that was the destination for this compare.
   - Select Take No Action to avoid overwriting an existing set in the event the name you entered is not unique. (You reset this to Replace or Merge when using or modifying this set; otherwise no update to the set is made. To avoid this consequence, you can select Replace now to ensure that this setting is in force the next time that you use this feature.)
   - The Aggregate Set Retrieval panel appears with the name of the set you are creating.
6. If you are creating an aggregate set of objects in the destination model but not in the source, leave no as the retrieval option and remove the aggregate set name. Otherwise, change the retrieval option to yes and leave the aggregate set name so that the objects meeting your criteria can be retrieved from the source model. Press Enter.

   The Select Aggregate Object Types panel appears. If you specified set retrieval on the previous panel, the following message appears, Number of aggregate objects retrieved: 0 out of 0.
7. Type **/** (slash) by the aggregate object types to which the occurrences you want to select belong and press Enter. The slashes that you enter appear as S's. Press Enter.

   The Object Occurrences panel for the first aggregate object type you selected appears.

8. Complete the Object Occurrences panel.
   a. Type **S** next to the aggregate object occurrences you want to select. Type **E** to expand an occurrence. Press Enter.
   b. Repeat until all object types have been processed.

      The Confirm Comparison List panel appears.
9. Review the list. Type **/** (slash) next to any individual occurrences you want to remove. Press Enter twice.
10. On the Confirm Selected Aggregate Objects panel, type one of the following commands:
    - ACCEPT
    - SAVE
    - END
    - CANCEL

    ACCEPT processes but does not save the set, then ends processing. (This is the default, which is used if you press Enter.) SAVE processes and saves the set, then ends processing.
11. Press Enter:
    - If you chose the Online execution, the Browse Report File displays the Compare Aggregate Objects report.
    - If you chose the Batch execution, the Batch JOB card maintenance panel appears. Modify the JOB card as necessary and SUBMIT the Job.

Compare Objects from an Existing Aggregate Set

Use this procedure to update or replace an aggregate set in any of the following situations:

- If you have migrated objects using the set, you created with the initial compare and you now want to replace the aggregate set with any objects still to be migrated.
- If you have not yet migrated the objects composing the existing aggregate set, but you have made some changes that are based on the trial migrate report. You can merge such changes into the existing set. If your set includes all objects that are compared as different, but you want it to also include all objects that are compared as existing in the source only. Such a set would enable you to migrate both new and changed objects in the same migration session. The Merge option makes this possible.

You can use a set without replacing it or merging more objects into it with the Take No Action option.

Steps to Generate Compare from Aggregate Set

Follow these steps:

1. Access the CA Gen Main menu. Select options 1.2.6. Press Enter.
2. On Compare Source Model Name panel, type the name of the source model. Press Enter.
3. Complete the Compare Destination Model Name panel.
   a. Type the destination model name.
   b. Type / (slash) next to Selected Objects
   c. Type / (slash) next to the execution option you want. Online displays the report on your terminal. Batch prints the report on the specified output device.
   d. Press Enter. The Compare Report Aggregate Set Create/Modify panel appears.
   e. To use an existing set and, optionally, modify the criteria:
      - Change the Save in aggregate set default N to Y.
      - Enter the name of the aggregate set to use.
      - Specify aggregate set options by typing / next to each option describing objects to save in the set.
   f. Select one of the following options from the choices under-If the set already exists:
      - Take No Action-Select Take No Action to use the set as is, but not update it (this option would be appropriate for a set you plan to delete right away).
      - Merge Sets-Select Merge Sets to update the existing set with any additions generated with the current compare criteria. This option is useful when you want to add more objects to those previously compared or when you want to add objects meeting a different criteria than that used to create the existing set.
      - Replace Set-Select Replace Set to replace the objects in the set with those that now meet the specified criteria. If you want to delete objects from the set, replace rather than merge the set. If you used this set to migrate, this use replace that set with objects now eligible for migration-objects that compare as the same will not be included in the replaced set.
      - The Aggregate Set Retrieval panel appears.
4. Press Enter to retrieve the aggregate set you selected.

   The Select Aggregate Object Types panel appears with a message stating the number of objects that are retrieved out of the number of objects in the set. You add or remove any selected object types or accept the set as is. Press Enter twice.

5. Review the Object Occurrences panel for each selected object type.
   a. Type **S** next to the aggregate object occurrences you want to select. Type **E** to expand an occurrence. Press Enter.
   b. Repeat until all object types have been processed.

      The Confirm Comparison List panel appears.
6. Review the displayed objects and type / (slash) next to any individual occurrences you want to remove. Press Enter.

   The Confirm Selected Aggregate Objects panel appears.
7. Type one of the following commands:

   ACCEPT, SAVE, END, or CANCEL

   ACCEPT processes but does not save the set, then ends processing**.** SAVE processes and saves the set, then ends processing.
8. Press Enter.
   - If you chose the Online execution, the Browse Report File displays the Compare report.
   - If you chose the Batch execution, the Batch JOB card maintenance panel appears. Modify the JOB card as necessary and SUBMIT the Job.

Evaluate the Compare Report

The Compare report lists compared objects in four ways: those that are the same in both models, those that exist in both models but are different, and those that are unique to one model or the other.

Deciding What to Migrate

The messages of the Compare Aggregate Object report helps you decide what to migrate:

- Aggregate the objects that are listed as same do not require migration and as different is migrated.
- Aggregate the objects not in the destination model can be migrated as needed.

Same or Different?

To determine whether objects are the same or different, the report first looks for common ancestry. Without common ancestry, the object does not exist in the other model. With common ancestry, they are said to be versions of the same object.

Objects that have common ancestry (the same Original Object ID) can be the same version of the same object or different versions of the same object.

For Objects to Be	The Objects in Both Models Have
Same	The same Original Object ID and Association using DIRCHGD to a Session object that has the same: Change date, Change time and Session ID properties
Different	The same Original Object ID and Association using DIRCHGD to a Session object that does NOT have the same: Change date, Change time and Session ID properties

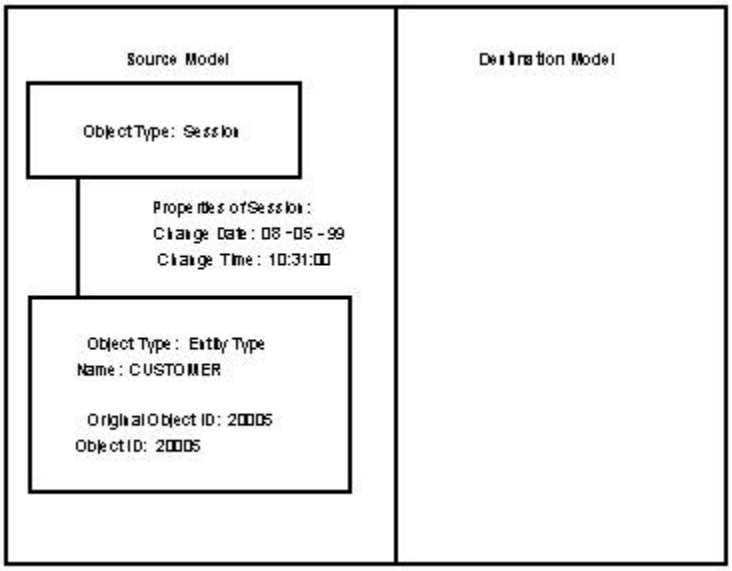
Examples of Difference Analysis

The following scenario illustrates how changes to objects in two models in three consecutive sessions are reflected in the Compare Aggregate Objects Report. The second and third examples illustrate how Session objects determine the difference or sameness of objects.

1. CUSTOMER entity type is created in Source Model. Compare Report is run.
2. CUSTOMER entity type is migrated from Source Model to Destination Model. Compare Report is run.
3. CUSTOMER is updated in Source Model. Compare Report is run.

Aggregate Object Not in Destination Model

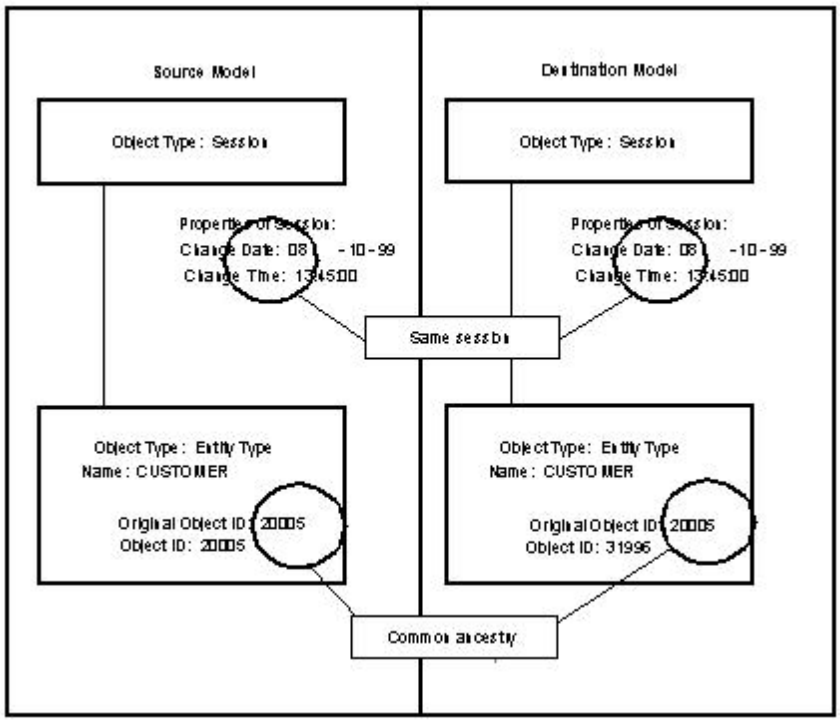
The aggregate object CUSTOMER was created in the source model. It exists only in the source model; it does not exist in the destination model.



Important! The Compare Aggregate Object Report indicates that the entity type CUSTOMER does not exist in the destination model because no object in the destination model has the same Original Object ID.

Aggregate Objects Listed As Same

Assume that during this session, CUSTOMER was migrated from the source model to the destination model. (The same Compare Report result would occur for CUSTOMER if the destination model were created by copying the source model and the entity type CUSTOMER existed in the source model at the time of the copy. That is, CUSTOMER would be reported as being the same in both models.)



Note: Compare Report uses the Change Date and Change Time of the session object to determine whether equivalent objects are the same version. If equivalent objects were last changed in the same session, the objects are reported as being the same.

Report Results**:** The Compare Aggregate Object Report indicates that entity type CUSTOMER is the same in the source and destination models. Objects with the same Original Object IDs are evaluated as the same when their Session Change Date and Change Time are alike.

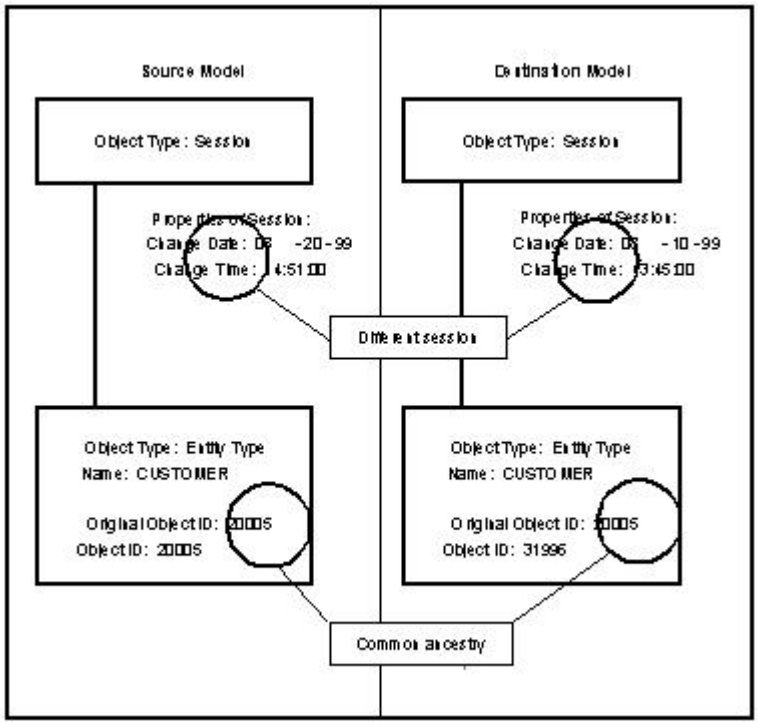
If you run the report with the Compare All option, the report does not list objects that are the same. To report on objects that are the same, select objects individually or specify an aggregate set.

Aggregate Objects Listed As Different

Assume that the entity type CUSTOMER is updated in the source model.

When the change occurs, a new session is created and the entity type in the source model is now associated with that new session.

An example of aggregate objects existing in both models but in different versions is as follows:



Report Results: The Compare Aggregate Objects Report indicates that entity type CUSTOMER in the source model is different from entity type CUSTOMER in the destination model. Objects with the same Original Object IDs are evaluated as different when their Session Change Date and Change Time are different.

Chapter 3: Maintain Aggregate Sets and Model Families

Maintain Aggregate Sets

An aggregate object is a collection of related objects and the smallest selectable unit in version control functions.

An aggregate set is a collection of aggregate objects that is created and saved for convenient reuse.

A model family defines the set of models that are related. Typically, all the models in a family support one production model.

Create an Aggregate Set

Follow these steps:

1. Access the CA Gen Main Menu.
2. Select options 1.5.1. Press Enter.

   The Add Aggregate Set panel appears.
3. Specify the model that contains the objects you want for the new aggregate set.
4. Type a name for the new set. Press Enter.

   The Aggregate Set Retrieval panel appears. Press Enter to bypass.

   The Aggregate Object Type Selection panel appears.
5. Type **/** (slash) by the aggregate object types to be expanded. To reset and begin the selection again, type **CANCEL** and press Enter.
6. Press Enter. A message appears.

   Input has been accepted; press Enter or enter “accept” command to proceed.
7. Press Enter.

   The Aggregate Object Occurrence Selection List appears with the occurrences of the first object type selected.

8. Type **S** next to the aggregate object occurrences you want to select. Type **E** to expand an occurrence. Press Enter.

   Repeat until all object types have been processed. The Confirm Selection of Aggregate Objects panel appears.

9. Type **/** (slash) next to any individual occurrences you want to remove.

   a. Press Enter.

   b. The Confirm Selected Aggregate Objects panel appears.

   c. Type ACCEPT, SAVE, END, or CANCEL; then press Enter.

      ACCEPT or SAVE stores the set and ends processing.

Modify an Aggregate Set

Follow these steps:

1. Access the CA Gen Main Menu.
2. Select options 1.5.2. Press Enter.

   The Modify Aggregate Set panel appears.

3. Specify the model that contains the objects to use in modifying the aggregate set.
4. Type the name of the set to be modified. Press Enter.

   The Aggregate Object Type Selection panel appears.

5. Type **/** (slash) by the aggregate object types to be expanded. To reset and begin the selection again, type **CANCEL** and press Enter.

   a. Press Enter. A message appears.

   b. Press Enter.

      The Aggregate Object Occurrence Selection List appears with the occurrences of the first object type selected.

   c. Type **S** next to the aggregate object occurrences you want to select. Type **E** to expand an occurrence. Press Enter.

      Repeat until all object types have been processed. The Confirm Selection of Aggregate Objects panel appears.

   d. Type **/** (slash) next to any individual occurrences you want to remove.

   e. Press Enter.

   f. The Confirm Selected Aggregate Objects panel appears.

   g. Type ACCEPT, SAVE, END, or CANCEL; then press Enter:

      ACCEPT *or* SAVE stores the set and ends processing. ACCEPT processes but does not save. SAVE processes and saves the set.

Copy an Aggregate Set

Follow these steps:

1. Access the CA Gen Main Menu.
2. Select options 1.5.3. Press Enter.

   The Copy Aggregate Set panel appears.
3. Specify the aggregate set to copy.
4. Type the name of the new aggregate set. Press Enter.

   The message All processing is completed normally appears.

Rename an Aggregate Set

Follow these steps:

1. Access the CA Gen Main Menu.
2. Select options 1.5.4. Press Enter.

   The Rename Aggregate Set panel appears.
3. Specify the aggregate set to rename.
4. Type the new name for the set. Press Enter.

   The message All processing is completed normally appears.

Delete an Aggregate Set

Follow these steps:

1. Access the CA Gen Main Menu.
2. Select options 1.5.5. Press Enter.

   The Delete Aggregate Set panel appears.
3. Specify the aggregate set to delete. Press Enter.

   The Aggregate Set Statistics panel appears.
4. Press Enter to complete the deletion.

   The message All processing is completed normally appears.

Generate Aggregate Set Reports

Report on Aggregate Set Objects for One Model

Follow these steps:

1. Access the CA Gen Main Menu.
2. Select options 1.5.6. Press Enter.

   The Aggregate Set Reports panel appears.
3. Select option 1. Press Enter.

   The Aggregate Set Object report appears.
4. Specify model name.
5. Type the aggregate set name or request Prompt.
6. Type **/** (slash) next to the execution option you want.
   - Online
   - Batch
7. Press Enter.
   - If you selected the Online execution, the Aggregate Set Definition report appears in Browse mode.
   - If you selected the Batch execution, a JCL stream appears in edit mode.
8. Edit the JCL if necessary.
9. Type SUBMIT or SUB and press Enter.

   The message appears: JOB jobname (jobnumber) submitted.

Report on Aggregate Set Objects for all Models

Follow these steps:

1. Access the CA Gen Main Menu.
2. Select options 1.5.6. Press Enter.

   The Aggregate Set Reports panel appears.
3. Select option 2. Press Enter.

   The Display Aggregate Set Objects for all Models in a Family appears.
4. Type an aggregate set name or request Prompt.
5. Type / (slash) next to the execution option you want.
   - Online
   - Batch

6. Press Enter.
   - If you selected the Online execution, the Family Aggregate Set Definition report appears in Browse mode.
   - If you selected the Batch execution, a JCL stream appears in edit mode.
7. Edit the JCL, if necessary.
8. Type SUBMIT or SUB and press Enter.

   The message appears: JOB jobname(jobnumber) submitted.

Report on Aggregate Sets for a Family

Follow these steps:

1. Access the CA Gen Main Menu.
2. Select options 1.5.6. Press Enter.

   The Aggregate Set Reports panel appears.
3. Select option 3. Press Enter.

   The Display Aggregate Sets in a Family panel appears.
4. Specify the name of the Model family.
5. Type **/** (slash) next to the execution option you want.
   - Online
   - Batch
6. Press Enter.
   - If you selected the online execution, the Aggregate Set Definition report appears in Browse mode.
   - If you selected the batch execution, a JCL stream appears in edit mode.
7. Edit the JCL, if necessary.
8. Type SUBMIT or SUB and press Enter.

   The message appears: JOB jobname(jobnumber) submitted.

Rename a Model Family

Rename a Model Family lets you change the name of a model family. When a new model family is created, it receives the same name as the new model during:

- Upload of a new model
- Generate new model

- Cross Encyclopedia copy
- Public Interface import

Rename the model family so that the family name is different from that of any model. To rename a model family, own at least one model in the model family or be the encyclopedia administrator.

Renaming a Model Family

Follow these steps:

1. Access the CA Gen Main Menu.
2. Select options 1.2.1. Press Enter.

   The Rename Model Family menu appears.
3. Type the current model family name.
4. Type the new model family name. Press Enter.

   The following message appears: All processing is completed normally.

   The model family is renamed.

Chapter 4: Perform Adoption or Unadoption

Understanding Adoption

You can adopt both entire models and individual aggregate objects. Adoption is sometimes required before you can migrate objects.

Adoption Restrictions and Requirements

You can adopt, trial adopt, or compare objects between any two models with supported schemas.

A model cannot be adopted if it is checked out. However, if subsets of the model are checked out, the model can be adopted.

Model Adoption

Before you can migrate objects between two models, the models must be in the same model family. If the models belong to different families, first adopt one of the models into the other's family.

Model Adoption Terms

Model adoption changes a model from membership in one model family to membership in another model family.

The related model is the model already in the required model family.

The adopted model is the model outside the required model family-the one to be adopted.

Model Adoption Process

The Model Adoption process:

- Compares aggregate objects (and some sub-aggregates) in both models, looking for matches by using a rule table.

  The most important criterion in establishing a match is identical object name value. See General Name Equivalence Rules.

- Changes the Original Object ID of the aggregate object in the adoptee model to the Original Object ID of the aggregate object in the related model. (This change occurs only when a match is found.)

  After the change in Original Object ID, the aggregate objects have common ancestry, and the adoptee aggregate objects are considered equivalent in migration.

Results of Model Adoption

Model Adoption makes the adoptee a member of the family of the related model. Model Adoption does not change the appearance of the adoptee model or add any new objects to it. Model Adoption does not change the related model in any way.

Unlike migration, adoption succeeds even when one or more selected objects cannot be adopted. The partial adoptions leave the model in a valid state, while partial migrations do not.

Aggregate Object Adoption

Even if models belong to the same family, you sometimes adopt aggregate objects within related models. To obtain a single definition of the object, you can adopt objects between the two models to establish the necessary common ancestry for the objects before replacing one by migration.

Adoption Between Models in Different Families

When the adoption occurs between models in different families, the adoptee is moved from its current family to the family of the related model. Adoption can also occur between two models already in the same family and already containing objects with common ancestry.

Adoption Between Models Within the Same Family

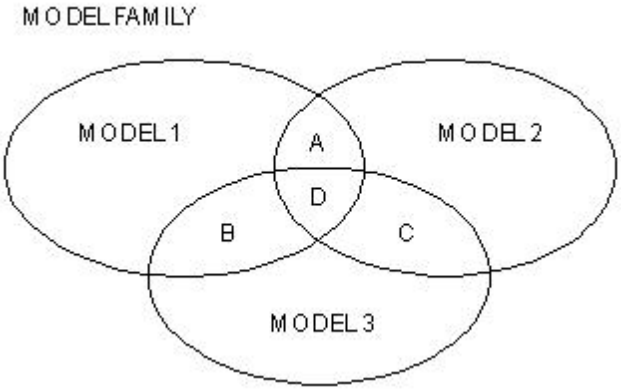
Adoption between models in the same family increases the number of objects that have common ancestry in the two models. An adoption does not mean that every object in the adopted model can be migrated to another family member with the expectation of a migrate, delete, or replace.

Note: As long as two models have not adopted each other, they have objects in common that do not have common ancestry.

One reason to adopt within a family is that two developers can create the same logical object independently in two models. Upon realizing the situation, someone can adopt between the two models to establish common ancestry for the object that was created redundantly.

Another reason to adopt within a family is models that overlap.

In the following illustration, Models 1, 2, and 3 represent Project models in the same family. They have been adopted using the production model as the related model, creating common ancestry among the three models for only those objects they all share with the production model (area D). The models have not adopted each other.



Intersection Area	Represents
D	The set of objects for which common ancestry has been established among all three models (Model 1, Model 2, and Model 3).
A	The additional set of objects for which common ancestry would be established, if an adoption were performed between Models 1 and 2.
B	The additional set of objects for which common ancestry would be established, if an adoption were performed between Models 1 and 3.
C	The additional set of objects for which common ancestry would be established, if an adoption were performed between Models 2 and 3.

Objects in area D can be migrated to the Production model or any of the other three models with a delete/replace result. Objects in area A can be migrated to Model 3 since they are in the same family, but the objects would be created as new objects because Models 1 and 3 have not adopted each other.

To perform adoptions among Models 1, 2, and 3. If you want to migrate objects between these models in the normal delete/replace manner, the adoptions are required.

Aggregate Objects for Adoption

Most aggregate objects that can be migrated can also be adopted, but some cannot. For a complete list of aggregate objects that can be selected for adoption, see the following table:

Aggregate Object for Adoption	Description
ATTRIBUTE	Attribute of entity type or subtype
BUSINESS AREA	Business area
BUSINESS SYSTEM	Business system
COMMAND	Command
COMMON ACTION BLOCK	Common, Default, and Derivation ADs
COMPONENT IMPLEMENTATION	Component implementation
COMPONENT MODEL	Component model
COMPONENT SPECIFICATION	Component specification
CONFIGURATION INSTANCE	Configuration Instance
CRITICAL SUCCESS	Critical success factor
CURRENT DATA	Current data base or data store
CURRENT INFO. SYSTEM	Current information system
CUSTOM PROXIES	Custom Proxies
CUSTOM VIDEO PROPERTY	Custom video property
DATA CLUSTER	Natural data store
DATA COLUMN	Data column definition
DATA TABLE	Data table definition
DATABASE	Database definition
DENORMALIZED COLUMN	Denormalized column definition
DIALECT	Dialect
DIALOG FLOW	Dialog flow
ENTITY	Entity type
ENVIRONMENT	Environment
EVENT	External event
EXIT STATE	Exit state
EXTERNAL OBJECT	External object

Aggregate Object for Adoption	Description
FACILITY	Computing or communication facility
FOREIGN KEY COLUMN	Foreign key column definition
GOAL	Goal
INDEX	Index definition
INFORMATION NEED	Information need
INTERFACE TYPE	Interface type
LINK TABLE	Link table definition
LOCATION	Location of business assets
NAVIGATION DIAGRAM	Navigation diagram
OBJECTIVE	Objective
ONLINE LOAD MODULE	Online load module
OPERATIONS LIBRARY	Operations library
ORGANIZATIONAL UNIT	Non-root organizational unit
PERFORMANCE MEASURE	Performance measure
PROCEDURE	Procedure
PROCEDURE STEP	Procedure step
PROCESS	Process definition
RELATIONSHIP	Relationship membership
SERVER MANAGER	Server manager
SPECIFICATION TYPE	Specification type
STORAGE GROUP	Storage Group
STRATEGY	Strategy
TABLESPACE	Tablespace definition
TACTIC	Tactic
TEMPLATE	System screen template
TRANS OPERATION	Transaction operation
TYPEMAP	Typemap
USER CLASS	User defined object class
USER FUNCTION	Function definition
USER MATRIX	User defined matrix

Aggregate Object for Adoption	Description
USER OBJECT	User defined objects
USER SUBJECT AREA	Subject area
WEB SERVICE DEFINITION	Web service definition
WINDOW LOAD MODULE	Window load module
WORK ATTRIBUTE	Attribute of work attribute set
WORK ATTRIBUTE SET	Work attribute set
z/OS LIBRARY	z/OS library

Adoption Rules

The rules for the adoption of subordinate objects are the same as the rules for the migration of subordinate objects.

General Name Equivalence Rules

The adopted objects are matched based on the following categories:

- Objects of the same type and name
- Objects of the same type and name but within parents of common ancestry
- Objects with parents that have common ancestry
- Objects that cannot be selected for an adoption but can be adopted if its parent is adopted

The objects in each category are shown in the following lists.

Objects of the same type and name; this is the most common type of matching:

- BAA Common Action Blocks
- BSD Common Action Blocks
- Business Systems
- Component Implementations
- Component Models
- Component Specifications
- Configuration Instances
- Current Data Stores

- Databases
- Dialects
- Entity Types
- Events
- Exit States
- External Objects
- Functions
- Interface Types
- ISP objects
- Navigation Diagrams
- Processes
- Specification Types
- Storages Groups
- Subject Areas
- Typemaps
- Work Attribute Sets

Objects of the same type and name but within parents of common ancestry:

- Attributes
- Commands
- Configuration Instance
- Custom Proxies
- Custom Video Property
- Online Load Modules
- Operations Libraries
- Procedures
- Procedure Steps
- Server Managers
- Tablespaces
- Templates
- User-defined Objects

- Web Service Definition

  Note: First 32 characters of the name must be same.

- Window Load Modules
- Work Attributes
- z/OS Libraries

Objects with parents that have common ancestry. This matching occurs when parent and child objects have a one-to-one association:

- Data Columns
- Data Tables
- Primary identifying indexes

Objects that cannot be selected for an adoption but can be adopted if its parent is adopted:

- Batch Jobs
- Batch Job Steps
- Default Edit Patterns
- Dialog boxes
- Entity Views
- Group Views
- Identifiers
- Implementation Units
- Packaging for procedure steps
- Permitted Values
- RI Trigger Action Blocks
- Procedure Step Action Blocks
- Process Action Blocks
- Screen Implementations
- Screens
- Subtypes
- System-defined PF keys
- Technical Systems
- Windows

Specific Name Equivalence Rules

Some objects have specific rules for name equivalence.

The aggregate objects that are adopted if they meet specific name equivalence rules follow:

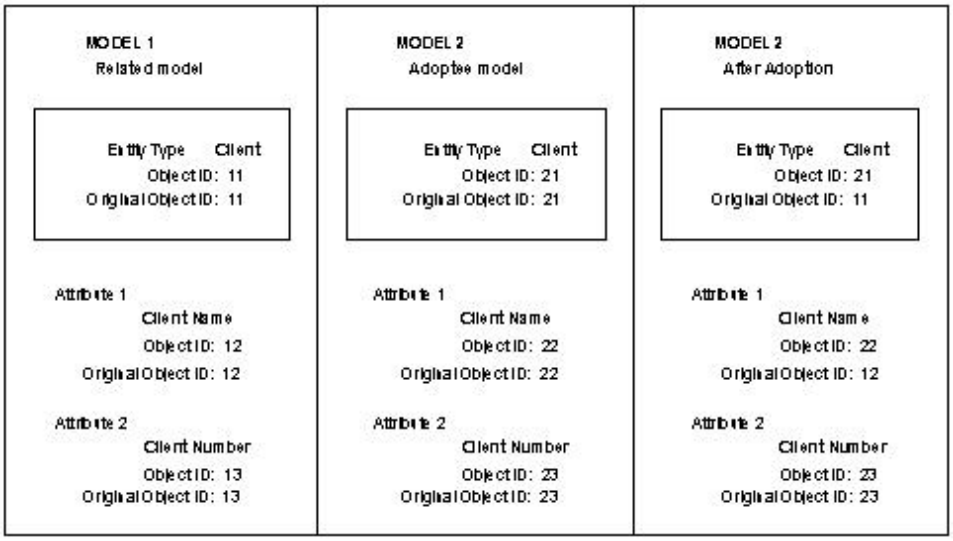
This Type of Aggregate Object	Is Adopted If:
DIALOG FLOW	The source and destination procedure steps have common ancestry and at least one of the associated exit states have common ancestry
DENORMALIZED COLUMN	The parent tables, associated attributes, and the associated relationships have common ancestry.
FOREIGN KEY COLUMN	The parent table's associated attributes and the associated relationships have common ancestry.
INDEX (non-identifier-based)	The associated data tables with common ancestry that implement relationships with common ancestry or the associated data tables with common ancestry and index with same name and associated database with same name and do not implement relationships.
LINK TABLE	The associated data tables and the associated relationships have common ancestry.
RELATIONSHIP	They have the same source and destination names and the associated entity types or subtypes have common ancestry.
TRANSACTION OPERATION (delegating)	They have the same name and are associated to entity type (or work attribute sets) and delegated to transactions with common ancestry
TRANSACTION OPERATION (non-delegating)	They have the same name and are associated to entity types and procedure steps with common ancestry
USER MATRICES	They have the same name and the associated x and y user or system-defined classes have common ancestry.

Adoption Examples

The following examples show adoptions accepted, not performed, and overwritten.

Example 1

In the first adoption example, Model 1 is the Related Model, and Model 2 is the Adoptee Model (in another family).



Results: Before the adoption, the Original Object ID of Entity Type Client is 11 in Model 1 and 21 in Model 2. Because the object type and names match, the Original Object ID in Model 2 is changed to 11. Now the objects have common ancestry.

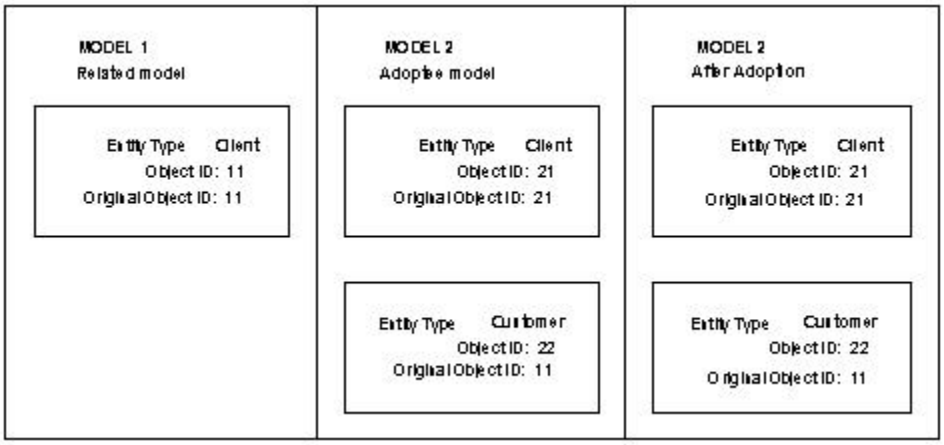
Note: The entity types have both an Object ID and Original Object ID and that the Object ID did not change for either.

The Attribute Client Name is also successfully adopted. This could not have occurred if the parent object, Entity Type Client, had not been adopted.

Because the Name properties of Attributes Client Number and Client Num are not the same, the Original Object ID (23) of Client Num remains unchanged after the Adoption.

Example 2

In the second adoption example, Entity Type Client is not adopted because Entity Type Customer in Model 2 already owns Original Object ID 11:



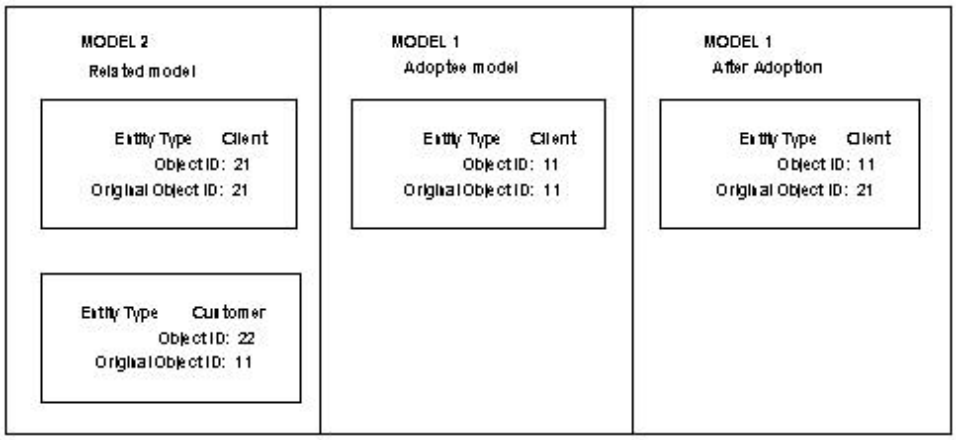
Because no two objects within the same model can have the same Original Object ID, the adoption of Client is not performed.

See the message IGNORED in Evaluate Adoption Report Messages.

Example 3

If an adoption is not performed, you may not need to adopt. For example, if you reverse the adoptee and related models in the preceding example showing adoption that is not performed, Model 2 becomes the related model, Model 1 becomes the adoptee model, and Entity Type Client is adopted.

Important! Performing the adoption shown in the following illustration could cause you to lose common ancestry with objects in other models for the Entity Type Client. This approach is not recommended but is shown to illustrate another aspect of adoption.



In this case, duplicate Original Object IDs will not be created for Client, so the adoption succeeds. The link between Client in Model 1 and Customer in Model 2 is broken. Because of the name match, a new link would be established between the two Client Entity Types.

Perform Trial Adoption and Adoption

When you perform trial adoption or adoption, you select one of the following options:

- All objects-Adopts all objects within the adoptee model that can be adopted. Will not fail because of the failure of a single adoption step, as migration would, but perform what it can.
- Selected objects-Adopts only the user-selected objects and their subordinate objects. Not all aggregate objects are directly adoptable. If a particular object is missing from the panel, the object is adopted when its parent is adopted or it is a system-defined object.
- System-defined objects-Adopts only the system-defined objects within the adoptee model. The system-defined objects are not user-accessible. They are system objects that are used for internal control.

The list of objects are as follows:

- Pcroot object for the model
- Root Subject Area
- $IEF System Work Attribute Set
- System-supplied Work Attribute Set

- System-supplied Functions, their work attribute sets, and their views
- System-supplied Object Classes
- System-defined Matrices
- Generics that is associated to system-defined object classes
- Default Dialect
- Scroll Amount Values
- Technical Design
- Root Function
- Root Organizational Unit

Note: Any adoption causes the system-defined objects to be adopted. The system-defined objects option is the equivalent of creating an empty model in the related model's family using the Migrate To New option and not selecting any aggregate object to migrate, then merging the adoptee model into the empty model. Skip this option when the adoptee and related model are already in the same family.

Trial Adopt Aggregate Objects Using Related Model

Trial Adopt Aggregate Objects Using Related Model generates a report of anticipated adoption results without actually applying updates.

To perform trial adoption, you must have Read Only authorization on the Related Model and on the Adoptee Model.

Note: Trial adoption and trial migration take longer than adoption and migration, respectively, because of database rollbacks.

Steps to Trial Adopt a Model

Follow these steps:

1. Access the CA Gen Main menu.
2. Select options 1.2.8. Press Enter.

   The Trial Adoption Adoptee Model Name menu appears.
3. Specify the name of the adoptee model. Press Enter.

   The Trial Adoption Related Model Name panel appears.
4. Type the name of the related model.

   Alternatively, press Tab to go to the field and press the prompt key for a list of valid choices. Select a model from the list by typing any character except period (.) next to one name and press Enter again.

5. Type **/** (slash) next to the object range you want.
   - All objects
   - System-defined objects
   - Selected objects
6. Type **/** (slash) next to the execution option you want.
   - Online
   - Batch
7. Press Enter.
   - If you chose Selected Objects, the Aggregate Set Retrieval panel appears. Continue with the next step.
   - If you selected All objects or System-defined objects and Online processing, the Confirm Models and Families for Trial Adoption panel appears.

   Press Enter to continue. The Trial Adopt Model/Objects report appears in Browse mode.

   If you selected All objects or System-defined objects and Batch processing, the Batch JOB card maintenance panel appears. Modify the JOB card as necessary and SUBMIT the Job.
8. To use a set, type the name of an aggregate set, type **Y** in the Retrieve Aggregate Set field and press Enter. To continue without using a set, leave the set name blank, the Retrieve Aggregate Set field at N, and press Enter.

   The Trial Adoption Aggregate Object Type Selection panel appears.
9. Type **/** (slash) by the aggregate object types to be expanded. To reset and begin the selection again, type CANCEL.
   a. Press Enter. A message appears.
   b. Press Enter. A selection list appears with the occurrences of the first object type selected.
   c. Type **S** next to the aggregate object occurrences you want to select. Type **E** to expand an occurrence. Press Enter.
   d. Repeat until all object types have been processed. The Confirm Aggregate Objects for Trial Migration panel appears.
   e. Type **/** (slash) next to any individual occurrences you want to remove.
   f. Press Enter. The Confirm Selected Aggregate Objects panel appears.
10. Type ACCEPT, SAVE, END, or CANCEL; then press Enter.

    ACCEPT *or* SAVE stores the set and ends processing. ACCEPT processes but does not save. SAVE processes and saves the set.

Adopt Model, Selected Objects, or System-Defined Objects

To adopt, you must have at least Read Only authorization on the Related Model and must Migrate To on the Adoptee model.

Adopt Aggregate Objects Using Related Model changes the family membership of one model to another family. The process can also be used with two models in the same family to establish or synchronize the common ancestry of objects in a model.

The adoption function displays a report indicating:

- Objects that were adopted
- Objects could not be adopted because of conflicts

Steps to Adopt a Model

Follow these steps:

1. Access the CA Gen Main menu.
2. Select options 1.2.4. Press Enter.

   The Adoption Adoptee Model Name menu appears.
3. Type the name of the adoptee model. Press Enter.

   The Adoption Related Model Name panel appears. Alternatively, tab to the field and press the prompt key for a list of valid choices. Select a model from the list by typing any character except period (.) next to one name and press Enter again.
4. Type the name of the related model.

   Alternatively, tab to the field and press the prompt key for a list of valid choices. Select a model from the list by typing any character except period (.) next to one name and press Enter again.
5. Type **/** (slash) next to the object range you want.
   - All objects
   - System-defined objects
   - Selected objects
6. Type **/** (slash) next to the execution option you want.
   - Online
   - Batch

7. Press Enter.
   - If you selected System-defined objects or All objects and Online processing, the Confirm Models and Families for Adoption panel appears. Press Enter to continue.
   - The Adopt Model/Objects report appears in Browse mode.
   - If you selected System-defined objects or All objects and Batch processing, the Batch JOB card maintenance panel appears. Modify the JOB card as necessary and SUBMIT the Job.
   - If you chose the Selected objects, the Aggregate Set Retrieval panel appears.
8. To use a set, type the name of an aggregate set, type **Y** in the Retrieve Aggregate Set field and press Enter. To continue without using a set, leave the set name blank, the Retrieve Aggregate Set field at N, and press Enter.

   The Adoption Aggregate Object Type Selection panel appears.
9. Type **/** (slash) by the aggregate object types to be expanded. To reset and begin the selection again, type **CANCEL**.
   a. Press Enter. A message appears.
   b. Press Enter.

      A selection list appears with the occurrences of the first object type selected.
   c. Type S next to the aggregate object occurrences you want to select. Type **E** to expand an occurrence. Press Enter.

   Repeat until all object types have been processed. The Confirm Aggregate Objects Selected for Adoption panel appears.
10. Type **/** (slash) next to any individual occurrences you want to remove.
11. Press Enter.

    The Confirm Selected Aggregate Objects panel appears.
12. Type ACCEPT, SAVE, END, or CANCEL; then press Enter.

    ACCEPT or SAVE stores the set and ends processing. ACCEPT processes but does not save. SAVE processes and saves the set.

Evaluate Adoption Report Messages

The adoption report contains the following messages:

- ADOPTED-IDs are modified to have common ancestry. The labels are also the same in both models.

- ADOPTS-IDs are modified to have common ancestry. The labels are not the same in both models. Therefore, both labels are displayed.

  For example, data tables are not required to have the same label to be adopted. Possible message:

  “Data Table CUSTOMER adopts Data Table CLIENT.”

- IGNORED-IDs are **not** modified because of conflict with existing IDs. An object already exists in the Adoptee model with the same Original Object ID. If adopted, the IDs would not be unique. The labels are also the same in both models.

- IGNORES-IDs are **not** modified because of conflict with existing IDs. An object already exists in the Adoptee model with the same Original Object ID. If adopted, the IDs would not be unique. The labels are *not* the same in both models. Therefore both labels are displayed.

  Possible message for data tables:

  “Data Table CUSTOMER ignores Data Table CLIENT.

Model Unadoption

Model unadoption lets you correct adoption errors or remove all common ancestry with any other model. For example, if you inadvertently reverse model names at the time of adoption, model unadoption can remove the adopted model from the family into which it was mistakenly adopted. It can also be used to disassociate a model from its current family.

Model unadoption does not change the appearance of the model.

Unadopt Model from Existing Family creates a model family and moves the unadopted model to this new family. The unadopted model no longer has common ancestry with models in its prior family. (The unadoption process replaces all Original Object IDs with unique IDs.)

Note: Unadoption of large models can cause DB2 to escalate its locks to the tablespace level, creating contention with other users. The iIncremental commits are not taken during the unadopt function.

When to Use

Use the model unadoption when:

- You want to remove a model from its model family and create a new model family.
- You want to create one or more new objects by migration into a model with common ancestry.

- The objects to be migrated originated in the destination model.

How to Use

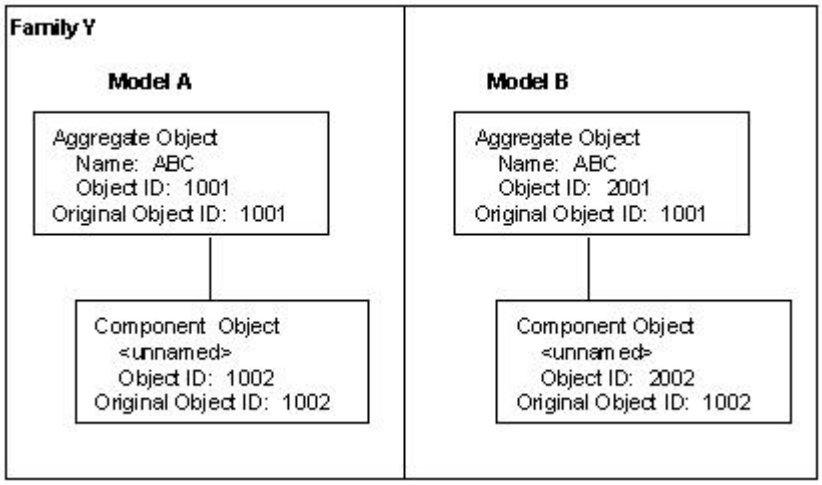
To migrate an object into a model of common ancestry as a new object:

- Unadopt the model that contains the object you want to migrate.
- Rename the object, if necessary.
- Adopt the model back into the family.
- Migrate the object from the unadopted model into a model in the prior model family.

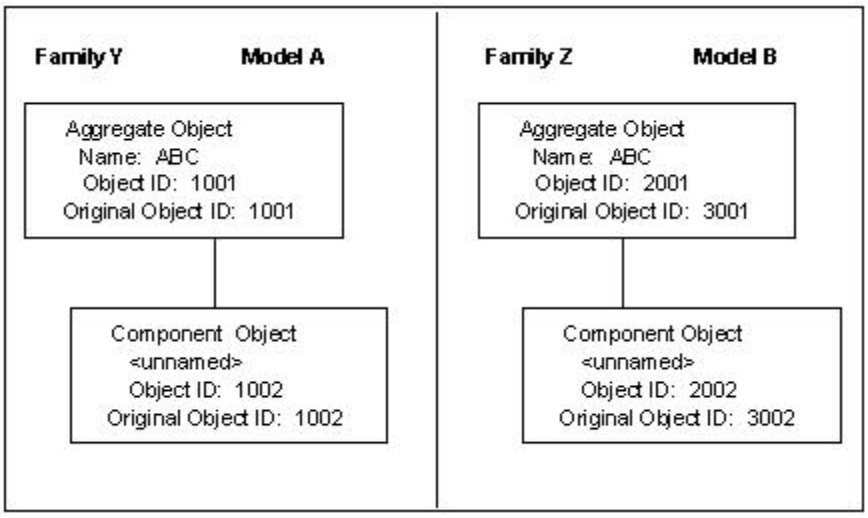
You can continue to migrate objects from the unadopted model without danger of replacing objects in the destination model.

Example

Before the unadoption, the aggregate object ABC and the unnamed component object have the same Original Object IDs in Model A and Model B.



When Model B is unadopted from Family Y, it is placed in a new model family, and the Original Object IDs of its aggregate object and unnamed component object are changed.



Perform Model Unadoption

Follow these steps:

1. Access the CA Gen Main menu.
2. Select options 1.2.5. Press Enter.

   The Unadopt Model panel appears.
3. Type the name of the model you want to unadopt.

   Alternatively, press Tab to go to the field and press the prompt key for a list of valid choices. Select a model from the list by typing any character except period (.) next to the model name and press Enter again.
4. Type the new model family name. The new model family name must be unique within the Host Encyclopedia.
5. Type **/** (slash) next to the execution mode you want.
   - Online
   - Batch
6. Press Enter.
   - If you selected the Online execution, this message appears: All processing is completed normally.
   - If you selected the Batch processing, the Batch Job card maintenance panel appears. Modify the Job card as necessary and SUBMIT the Job.

Chapter 5: Migrate Model from One Model to Another

Understanding Migration

Migration copies an aggregate object and its related components from one member of a model family to another model of the same family.

To do this, migration checks source and destination models to determine whether the aggregate object being migrated exists in both models. Objects that exist in both models either have common ancestry or conform to the special cases for that object type.

Note: For more information about expansions and special cases, see Aggregate Object Expansions in the Migration Rules (see page 68) appendix. For information about how to create common ancestry, see the Perform Trial Adoption and Adoption section in the Perform Adoption or Unadoption (see page 35) chapter. For information about removing common ancestry, see Perform Model Unadoption in the Perform Adoption or Unadoption (see page 35) chapter.

If the aggregate object being migrated exists in both source and destination models and has common ancestry, the migration process:

- Deletes the aggregate object and its components from the destination model
- Copies the aggregate object and its components from the source model into the destination model
- Validates the destination model to ensure that it is left in a valid state

Important! There is no command to back out a migration that has unforeseen consequences. Keep the backups of the destination models because migration changes cannot be easily undone.

Effect of Schema Level on Migration

You can migrate or trial migrate objects between models only if:

- Source and destination models have the same schema level or
- Source model schema level is less than destination model schema level.

Effect of Common Ancestry on Migration

When the destination model contains an object with the same name as an object being migrated but which does not share common ancestry with the like-named object, migration renames the object being migrated before adding it to the destination model. The destination model then contains the renamed object that shares common ancestry with the object in the source model plus the unrelated object that only happens to have the same name as the source model object.

The convention for renaming the object being migrated when a duplicate name is encountered is to add a dash and the object's Object ID (for example, Customer-12345). The name of the object already in the destination model is unchanged. For example:

Model Version	Before Migration	After Migration
Source Model	Customer Object ID: 12345 Original Object ID: 12345	Customer Object ID: 12345 Original Object ID: 12345
Destination Model	Customer Object ID: 67890 Original Object ID: 67890	Customer Object ID: 67890 Original Object ID: 67890 and Customer-12345 Object ID: 94034 Original Object ID: 12345

When aggregate objects in the source model and destination model have different names **and** common ancestry, migration replaces the name in the destination model with the name in the source model. For example:

Model Version	Before Migration	After Migration
Source Model	Client Object ID: 12345 Original Object ID: 12345	Client Object ID: 12345 Original Object ID: 12345
Destination Model	Customer Object ID: 56789 Original Object ID: 12345	Client Object ID: 56789 Original Object ID: 12345

Difference in the name but common ancestry can occur if an aggregate object has been previously migrated and then renamed in either the source or destination model. The subsequent migrations for such objects overlay the name of the object in the destination model.

Effect of Subsetting on Migration

Before initiating your first migration it is important to consider how version control and Subsetting interact. (Most the workstation activity involves using a subset of a model, not the entire model. Thus, the objects are sometimes checked out and unavailable for migration.)

Version control honors the same object-level protection rules that are used in subsetting to protect the validity of models. If some objects in a destination model are checked out with certain levels of protection, they cannot be altered by a migration.

Example of Object Level Protection

Object-level protection also includes objects that are related to the aggregate object if a migration would produce an invalid model. For example, if your migration would add a new relationship to the destination model but the new relationship's destination entity type is checked out with delete authority, the migration fail.

The migration failure occurs because the checked-out entity type is deleted and never return to the model. If the required entity type is deleted and the subset is checked in, the new relationship would be left without a target entity type, and the model would be in an invalid state. The migration software has no way of knowing the intentions of the developers when they check out an object so the software observes the subsetting object protection rules.

Effect of Subsetting Protection on Migrations

The effects of subsetting protection on migration follow:

- A migration fail if the aggregate object or any of its components are checked out with Modify or Delete authority in the destination model.
- A migration fail if the aggregate object's companion or enabling objects are checked out with Delete authority in the destination model.
- If the migration would cause the deletion of any object from the destination model, that object may not be checked out for delete, modify or access. The migration can proceed if the object is checked out with Read Only protection.

Note: The effect of subsetting protection on migration suggests guidelines: (1) Try to avoid using full expansion of subsetting scoping objects, and (2) Restrict delete authority on common occurrences of entity types such as Customer.

One way to determine if your migration can succeed is to ask: Could I create a subset from my destination model with my aggregate object as my scoping object and not have any of my access requests downgraded when I expand the subset? If the answer is yes, the migration succeed.

Aggregate Objects for Migration

The following table describes the aggregate objects that can be selected for migration:

Aggregate Object for Migration	Description
ACTION BLOCK	PAD, PrAD, Common, Default, and Derivation Action diagrams
ACTIVITY CLUSTER	Natural business system
ATTRIBUTE	Attribute of entity type or subtype
BATCH JOB	Batch job
BATCH JOB STEP	Batch job step
BUSINESS AREA	Business area
BUSINESS SYSTEM	Business system
BUSINESS SYS IMPL	Business system implementation
COMMAND	Command
COMPONENT IMPLEMENTATION	Component implementation
COMPONENT MODEL	Component model
COMPONENT SPECIFICATION	Component specification
CONFIGURATION INSTANCE	Configuration Instance
CRITICAL SUCCESS	Critical success factor
CURRENT DATA	Current database or data store
CURRENT INFO. SYSTEM	Current information system
CUSTOM PROXIES	Custom Proxies
CUSTOM VIDEO PROPERTY	Custom video property
DATA CLUSTER	Natural data store
DATA COLUMN	Data column definition
DATA TABLE	Data table definition
DATABASE	Database definition
DEFAULT DIALOG BOX	Default Dialog Box
DEFAULT EDIT PATTERN	System default edit pattern
DENORMALIZED COLUMN	Denormalized column
DFLT PRIMARY WINDOW	Default Primary Window

Aggregate Object for Migration	Description
DIALECT	Dialect
DIALOG FLOW	Dialog flow
ENTITY	Entity type
ENVIRONMENT	Environment
EVENT	External event
EXIT STATE	Exit state
EXTERNAL OBJECT	External object
FACILITY	Computing or communication facility
FOREIGN KEY COLUMN	Foreign key column definition
FUNCTION	Function definition
GOAL	Goal
INDEX	Index definition
INFORMATION NEED	Information need
INTERFACE TYPE	Interface type
LINK TABLE	Link table
LOCATION	Location of business assets
MATRIX	ISP matrix
NAVIGATION DIAGRAM	Navigation diagram
OBJECTIVE	Objective
ONLINE LOAD MODULE	Online load module
OPERATIONS LIBRARY	Operations library
ORGANIZATIONAL UNIT	Root organizational unit
PACKAGING	Packaging for procedure step
PERFORMANCE MEASURE	Performance measure
PROCEDURE	Procedure
PROCEDURE STEP	Procedure step
PROCESS	Process definition
RELATIONSHIP	Relationship membership
SCREEN	Screen
SCROLL AMOUNT VALUE	Scroll Amount Value

Aggregate Object for Migration	Description
SERVER MANAGER	Server manager
SPECIFICATION TYPE	Specification type
STORAGE GROUP	Storage Group
STRATEGY	Strategy
SUBJECT AREA	Subject area
SYSTEM-WIDE PF KEY	System program function key
TABLESPACE	Tablespace definition
TACTIC	Tactic
TECH DESIGN DEFAULT	Technical Design Default
TEMPLATE	System screen template
TRANS OPERATION	Transaction operation
TYPEMAP	Typemap
USER CLASS	User defined object class
USER OBJECT	User defined objects
WEB SERVICE DEFINITION	Web service definition
WORK ATTRIBUTE SET	Work attribute set
WINDOW LOAD MODULE	Window load module
WORK ATTRIBUTE	Attribute of work attribute set
z/OS LIBRARY	z/OS library

Migrating Non-Selectable Objects

Four objects on the Compare Aggregate Objects report cannot be migrated directly. They are not selectable for version control purposes. To migrate them, select their parent or grandparent for migration.

To Migrate This Object:	Select This Object for Migration
Local PF Keys	Parent Procedure Step
Permitted Values	Parent Attribute or Grandparent Entity Type
Command synonyms	Parent Command
Classifier	Parent Entity Type

Migrating the Deletion of Objects

Migration can delete objects in the destination model-whether you intend to delete them or not. For example, if Entity Type A exists in the source and destination models and its Attribute C is deleted from the source model, migration deletes Attribute C from the destination model.

If Attribute C is not referenced in the destination model-has no usages-the deletion of Attribute C causes no problem, and the migration can proceed. However, if Attribute C is referenced in the destination model (for example, in an attribute view), the migration terminates and all object actions are rolled back.

See details on the migration error message in the Object Cannot Be/Must Be Deleted section.

You want to generate a Trial Migrate to avoid unintended results although Trial Migrate requires more time to process.

If a deletion by migration creates a problem, try migrating the parent object of the deleted object. (Usually you migrate only the directly changed aggregate objects and not their parents.)

If migration of the parent object does not solve the problem, checkout an appropriate subset from the destination model and delete the object, or use the Host Encyclopedia's Delete Object function.

Perform Trial Migration and Migration

Version control lets you migrate aggregate objects to a new model or to an existing model.

Trial Migrate Aggregate Objects to an Existing Model

Trial Migrate Aggregate Objects to an Existing Model generates a report of anticipated migration results without actually applying updates to the destination model.

Note: Trial migration and trial adoption take longer than migration and adoption, respectively, because of rollbacks.

To perform trial migration, you must have at least Read Only access to both the Source Model and the Destination Model.

Steps to Trial Migrate

Follow these steps:

1. Access the CA Gen Main menu.
2. Select options 1.2.7. Press Enter.

   The Trial Migrate Source Model Name menu appears.
3. Specify the name of the source model. Press Enter.

   The Trial Migrate Destination Model Name menu appears.
4. Type the destination model name.
5. Type **/** (slash) next to the execution option you want.
   - Online
   - Batch
6. Press Enter.

   The Aggregate Set Retrieval panel appears.
7. To bypass set selection, press Enter.

   To retrieve a set, type **Y** and the set name. (A prompt is also available for set the selection.)
   - If you select a set, the contents of the aggregate set populate the Object Type Selection panels. The set that is shown becomes the starting point for building an aggregate object list for this execution.
   - When the Aggregate Object Type Selection panel appears, a message is displayed under the command line.
   - Number of aggregate objects retrieved: <x> out of <x>
   - If the numbers are not the same, some aggregate objects in the set are not present in the model being used or some of the aggregate objects in the set are not valid objects for the current function.
   - For example, screens can be migrated but not adopted. Therefore, a screen in an aggregate set is temporarily discarded when using the adoption function.
   - The Trial Migrate Aggregate Object Type Selection panel appears.
8. Type **/** (slash) by the aggregate object types to be expanded. To reset and begin the selection again, type **CANCEL**.
   a. Press Enter. A message appears.
   b. Press Enter. A selection list appears with the occurrences of the first object type selected.
   c. Type **S** next to the aggregate object occurrences you want to select. Type **E** to expand an occurrence. Press Enter.

d. Repeat until all object types have been processed. The Confirm Aggregate Objects for Trial Migration panel appears.

e. Type **/** (slash) next to any individual occurrences you want to remove.

f. Press Enter. The Confirm Selected Aggregate Objects panel appears.

9. Type ACCEPT, SAVE, END, or CANCEL.

   ACCEPT or SAVE stores the set and ends processing. ACCEPT processes but does not save. SAVE processes and saves the set.

10. Press Enter.

    - If you selected online execution, Migration processing begins.
    - The Trial Migrate Aggregate Object report appears in Browse mode for review.
    - If you selected batch execution, the Batch JOB card maintenance panel appears. Modify the JOB card as necessary and SUBMIT the Job.

Migrate Objects to a New Model

Migrate Aggregate Objects to a New Model creates a model within the model family. To populate a new model by migrating aggregate objects and their related components to the model, you specify a list of aggregate objects and the model from which they are copied. If you do not select any aggregate objects, this option can be used to create a model that contains only system-defined objects.

To migrate objects to a new model, you must have a minimum of Read Only authority on the Source Model and must Create Model authority.

Note: When you migrate an aggregate object to a new model, all external references to the new model reflect the name you select in the create procedure. However, the internal name, which is displayed in the workstation toolsets, is the name of the root subject area, not the long model name.

Steps to Migrate to a New Model

Follow these steps:

1. Access the CA Gen Main Menu.
2. Select options 1.2.2. Press Enter.

   The Migrate Source Model Name menu appears.

3. Type the name of the source model. Press Enter.

   Alternatively, tab to the field and press the prompt key for a list of valid choices. Select a model from the list by typing any character except period (.) next to one name and press Enter again.

   The Migrate to New Model menu appears.

4. Type the destination model name.
5. Type **/** (slash) next to the range of objects you want to migrate.
   - All objects
   - Selected objects
6. Type **/** (slash) next to the execution mode you want.
   - Online
   - Batch
7. Press Enter.
   - If you select All objects and Online execution, Migration processing begins.
   - The Migrate Aggregate Object report appears in Browse mode for review.
   - If you select All objects and Batch execution, the Batch JOB card maintenance panel appears. Modify the JOB card as necessary and SUBMIT the Job.
   - If you choose the Selected objects, the Aggregate Set Retrieval panel appears.
8. To use a set, type the name of an aggregate set, type Y in the Retrieve Aggregate Set field and press Enter. To continue without using a set, leave the set name blank, the Retrieve Aggregate Set field at N, and press Enter.

   The Select Aggregate Objects for Migration to New Model panel appears.
9. Type **/** (slash) by the aggregate object types to be expanded. To reset and begin the selection again, type **CANCEL** and press Enter.
   a. Press Enter. A message appears.
   b. Press Enter. A selection list appears with the occurrences of the first object type selected.
   c. Type **S** next to the aggregate object occurrences you want to select. Type **E** to expand an occurrence. Press Enter.

      Repeat until all object types have been processed. The Confirm Selection of Aggregate Objects for Migration panel appears.
   d. Type **/** (slash) next to any individual occurrences you want to remove.
   e. Press Enter.

      The Confirm Selected Aggregate Objects panel appears.
10. Type ACCEPT, SAVE, END, or CANCEL.

    ACCEPT or SAVE stores the set and ends processing. ACCEPT processes but does not save. SAVE processes and saves the set.

11. Press Enter.
    - If you select online execution, Migration processing begins.
    - The Migrate Aggregate Object report appears in Browse mode for review.
    - If you select batch execution, the Batch JOB card maintenance panel appears. Modify the JOB card as necessary and SUBMIT the Job.

Migrate Objects to an Existing Model

Migrate Aggregate Objects to an Existing Model is the basic version control function. Aggregate the objects from a source model are copied to a destination model, replacing the aggregate objects for which common ancestry can be established and creating aggregate objects that do not have common ancestry.

To migrate objects to an existing model, you must have at least Read Only authorization on the source model and Migrate To access on the destination model.

Steps to Migrate to an Existing Model

Follow these steps:

1. Access the CA Gen Main menu.
2. Select options 1.2.3. Press Enter.

   The Migrate Source Model Name menu appears.
3. Type the name of the source model. Press Enter.

   Alternatively, tab to the field and press the prompt key for a list of valid choices. Select a model from the list by typing any character except period (.) next to one name and press Enter again.

   The Migrate Destination Model Name menu appears.
4. Type the destination model name.
5. Type **/** (slash) next to the execution option you want.
   - Online
   - Batch
6. Press Enter and the Aggregate Set Retrieval panel appears.
7. To use a set, type the name of an aggregate set, type **Y** in the Retrieve Aggregate Set field and press Enter. To continue without using a set, leave the set name blank, the Retrieve Aggregate Set field at N, and press Enter.

   The Migrate Aggregate Object Type Selection panel appears.

8. Type **/** (slash) by the aggregate object types to be expanded. To reset and begin the selection again, type **CANCEL** and press Enter.
   a. Press Enter. A message appears.
   b. Press Enter. A selection list appears with the occurrences of the first object type selected.
   c. Type **S** next to the aggregate object occurrences you want to select. Type **E** to expand an occurrence. Press Enter.

      Repeat until all object types have been processed. The Confirm Aggregate Objects for Migration panel appears.
   d. Type / (slash) next to any individual occurrences you want to remove.
   e. Press Enter.

      The Confirm Selected Aggregate Objects panel appears.
9. Type ACCEPT, SAVE, END, or CANCEL.

   ACCEPT or SAVE stores the set and ends processing. ACCEPT processes but does not save. SAVE processes and saves the set.
10. Press Enter.
    - If you selected the Online execution, Migration processing begins.
    - The Migrate Aggregate Object report appears in Browse mode for review.
    - If you selected the Batch execution, the Batch JOB card maintenance panel appears. Modify the JOB card as necessary and SUBMIT the Job.

Evaluate Migration Error Messages

Migration produces the following error messages.

Object Requires Existence of Related Object

The related object in the message is actually an enabling object. Something that had to be present in the destination model for the migration to produce a valid model was missing. Usually this indicates that the parent object of the object in the message was missing. At other times, the aggregated object references the object that is listed.

For example, the migration of Entity Type Order fails if the Subject Area that contains Order does not exist in the destination model. If the Subject Area is migrated simultaneously as the Entity Type, the migration succeeds. The migration of a Navigation Diagram fails if all the Windows, Dialog Boxes, and Procedure Steps that are referenced by the Navigation Diagram do not exist in the destination model. The migration succeeds if all these referenced objects are present.

This message can also indicate that the enabling objects do not have common ancestry between the models.

Attempt to Modify/Delete Objects that are Checked Out

The objects listed were checked out in a subset when the migration was attempted. It is not necessary for all subsets to be checked to migrate but migration does not copy an object that is checked out with modify or delete authority. The message lists the object that is checked out, the subset name, the person who checked out the subset and what protection level the object had.

Object Cannot Be/Must Be Deleted

In general, this message indicates that migration tried to delete an object from the destination model that requires another object in that model.

For example, Entity Type A exists in the source and destination models. It has Attributes B and C. Attribute C is deleted from the source model. However, an attribute view in the destination model references Attribute C.

The migration of Entity Type A:

- Deletes Entity Type A and Attributes B and C from the destination model
- Copies Entity Type A and its component object Attribute B to the destination model
- Tries to re-establish the link between Attribute C and its attribute view

  When it cannot reestablish the link, it terminates the migration and rolls back all object actions.

Object Invalidates Object

This error message indicates that migration fails for one of the following reasons:

- To prevent the tablespace from being in a different database than the indexspace
- To prevent the indexspace for the index from being in a different database than the tablespace

To prevent this type of error, follow these guidelines:

- When you migrate a tablespace, migrate the parent data table also.
- When you migrate an index, migrate the related tablespace.

Chapter 6: Migration Rules

Usage and Applicability

When you request the migration of an aggregate object, it triggers the migration of its component objects, companion objects, and enabling objects too. The underlying design of the software determines the objects that must be migrated with a given aggregate object.

How Migration Rules Are Used

Two sets of rules are documented:

- Aggregate Object Expansions Tables-Aggregate object expansion is the process of determining which objects are migrated with the selected aggregate and whether proper conditions exist for the migration to succeed.

  The Aggregate Object Expansions Tables contain a list of all expandable aggregate object types for migration. For each type of aggregate object, the table lists:

  - Component objects-Objects that are always migrated with the aggregate object
  - Companion objects-Objects that are migrated with the aggregate object only if they do not already exist in the destination model.
  - Enabling objects-Objects that must exist in the destination model for the migration to succeed. If they are absent, migrate them with the aggregate object.
  - Special cases-Objects that are migrated with the aggregate object only if certain other objects exist in the destination. The migration can succeed even if these objects are absent.

- Special Situation Aggregate Actions-These actions relate to the special cases noted in the Aggregate Object Expansions Tables.

Aggregate Object Types with No Expansions

The following aggregate objects are not expandable and are not listed in the aggregate object expansions table:

- ACTIVITY CLUSTER
- BUSINESS AREA
- CRITICAL SUCCESS

- CURRENT DATA
- DATA CLUSTER
- DIALECT
- ENVIRONMENT
- EVENT
- EXTERNAL OBJECT
- FACILITY
- GOAL
- INFORMATION NEED
- LOCATION
- OBJECTIVE
- PERFORMANCE MEASURE
- USER CLASS

Aggregate Object Expansions

Action Block

The following table describes the action block expansions:

Component Objects	Companion Objects	Enabling Objects
Action statements, views, bind package defaults, implementation unit, DBRM	Commands and exit states, action blocks used by action block and their related components, operations library, z/OS library	Owning entity type or work attribute set. Relationships, entity types, subtypes, attributes, work attributes, and work attribute sets referenced by any migrated views and/or action statements. Business system, if migrated action block is implemented into a business system. Window controls referenced by window control usages. Windows and dialog boxes referenced by action statements.

Special Cases

The special cases for aggregate object expansions are as follows:

- A view match to a view in a USED action block is migrated only if the corresponding view in the USED action block already exists in the destination model. Conversely, a view match from a view in an action block that USES the migrated action block is migrated only if the corresponding view in the using action block already exists in the destination model.
- If the action block being migrated is an elementary process action block and the process exists in the destination model (but is not elementary), the elementary process automatically be migrated.
- If the action block being migrated is an elementary process action block and the process does NOT exist in the destination model, the elementary process automatically be migrated.

Special Case Examples: Data Views and View Matching When USING Action Block is migrated

In the following situations, Action Block A is an action block that is directly or indirectly selected to be migrated and Action Block B is another action block.

- View Match Copied

  Action Block A uses Action Block B in the source model.

  A View Match exists in the source model between a view in Action Block A and a view in Action Block B.

  Action Block B exists in the destination model.

  The View in Action Block B exists in the destination model.

  The View Match does not exist in the destination model.

  Result: The View Match is copied to the destination model.

- View Match Replaced

  Action Block A uses Action Block B in the source model.

  A View Match exists in the source model between a view in Action Block A and a view in Action Block B.

  Action Block B exists in the destination model.

  The View in Action Block B exists in the destination model.

  The View Match exists in the destination model.

  Result: The View Match is replaced in the destination model.

- View Match Not Copied

  Action Block A uses Action Block B in the source model.

  A View Match exists in the source model between a view in Action Block A and a view in Action Block B.

  The Action Block B exists in the destination model.

  The View in the Action Block B does not exist in the destination model.

  Result: The View Match is not copied to the destination model.

- View Match Deleted

  Action Block A uses Action Block B in the source model.

  A View Match exists in the destination model between a view in Action Block A and a view in Action Block B.

  The View in Action Block A does not exist in the source model

  OR the View Match does not exist in the source model.

  Result: The View Match is deleted from the destination model.

Attribute

The following table describes the attribute expansions:

Component Objects	Companion Objects	Enabling Objects
Permitted values and aliases	Dialect, default or derivation algorithm and its related components	Parent entity type or subtype

Special Cases

The special cases for attribute expansions are as follows:

- Prompts are created if they exist in the source model and do not already exist in the destination model. Prompts are replaced if they exist in both the source and destination model. Prompts are unchanged if they exist in the destination model but not in the source model.
- The prompt values are created if they exist in the source model and do not already exist in the destination model. The prompt values are replaced if they exist in both the source and destination models. The prompt values are unchanged if they exist in the destination model but not in the source model.

Special Case Examples: Attributes and Prompts

In the following situations, Attribute A is selected for migration.

- Prompt Created

  Prompt A exists for Attribute A in the source model.

  Attribute A exists in the destination model, but Prompt A is not in the destination model.

  Result: Prompt A is created in the destination model.

- Prompt Not Changed

  Attribute A does not have Prompt A in the source model.

  Attribute A has Prompt A in the destination model.

  Result: Prompt A is not changed in the destination model.

- Prompt Replaced

  Attribute A has Prompt A in the source model.

  Attribute A has Prompt A in the destination model.

  Result: Prompt A is replaced in the destination model.

Special Case Examples: Permitted Values and Prompt Values

In the following situation, Attribute A is selected for migration.

- Prompt Value Created

  Attribute A has Permitted Value A, which has Prompt Value A, in the source model.

  Attribute A and Permitted Value A exist in the destination model.

  Prompt Value A does not exist in the destination model.

  Result: Prompt Value A is copied to the destination model.

- Prompt Value Not Changed

  Attribute A has Permitted Value A in the source model.

  Attribute A and Permitted Value A exist in the destination model.

  Prompt Value A exists in the destination model for Permitted Value A.

  Prompt Value A does not exist in the source model.

  Result: Prompt Value A is not changed in the destination model.

- Prompt Value Replaced

  Attribute A has Permitted Value A, which has Prompt Value A, in the source model.

  Attribute A, Permitted Value A, and Prompt Value A exist in the destination model.

  Result: Prompt Value A is replaced in the destination model.

Batch Job

The following table describes the batch job expansions:

Component Objects	Companion Objects	Enabling Objects
Batch job steps and their related components	None	Associated procedure

Batch Job Step

The following table describes various batch job step expansions:

Component Objects	Companion Objects	Enabling Objects
Package list entries Packaging	None	Parent batch job, associated procedure step and generated action blocks.

Special Cases

To migrate a job step, select job step, or migrate the procedure or the batch job.

Business System Implementation

The following table describes the business system implementation expansions:

Component Objects	Companion Objects	Enabling Objects
Target environment parameters are properties of a BSI object. Bind package default and package list entry	Referenced libraries	Parent business system

Business System

The following table describes the business system expansions:

Component Objects	Companion Objects	Enabling Objects
Literal properties, error properties, prompt properties, field and special field properties, default GUI attributes, business system implementation and its related components.	Default edit patterns, system-wide program function keys, and custom video properties	None

Command

The following table describes the command expansions:

Component Objects	Companion Objects	Enabling Objects
Command synonyms	Dialect	Parent business system

Component Implementation

The following table describes various command implementation expansions:

Component Objects	Companion Objects	Enabling Objects
None	None	Parent subject area

Component Model

The following table describes the component model expansions:

Component Objects	Companion Objects	Enabling Objects
Art objects	None	Scoping subject area, attributes and action blocks

Component Specification

The following table describes the component specification expansions:

Component Objects	Companion Objects	Enabling Objects
Offerings	None	Parent subject area, interface type offered by component specification

Configuration Instance

The following table describes the configuration instance:

Component Objects	Companion Objects	Enabling Objects
Configuration Instance		Referenced business system, load modules, procedure steps, action blocks, windows, dialog boxes, databases, tablespaces, records, and storage groups

Current Information System

The following table describes the current information system expansions:

Component Objects	Companion Objects	Enabling Objects
Current Effect	None	Current database or stores

Special Cases

Current effect migrated only if current database or data store already exists.

Special Case Examples: Current Information Systems and Current Effects

In the following situations, Current Information System A is a Current Information System directly or indirectly selected for migration and Current Data B is Current Data.

- Current Effect Created

  A Current Info produces a Current Effect. System A in the source model.

  The Current Effect affects Current Data B in the source model.

  Current Data B exists in the destination model.

  The Current Effect does not exist in the destination model.

  Result: The Current Effect is copied to the destination model.

- Current Effect Replaced

  A Current Info produces a Current Effect. System A in the source model.

  The Current Effect affects Current Data B in the source model.

  Current Data B exists in the destination model.

  The Current Effect exists in the destination model.

  Result: The Current Effect is replaced in the destination model.

- Current Data Not Created

  A Current Info produces a Current Effect. System A in the source model.

  The Current Effect affects Current Data B in the source model.

  Current Data B does not exist in the destination model.

  Result: The Current Effect is not copied to the destination model.

- Current Effect Deleted

  A Current Info produces a Current Effect. System A in the destination model.

  The Current Effect affects Current Data B in the destination model.

  The Current Effect does not exist in the source model.

  Result: The Current Effect is deleted from the destination model.

Custom Proxies

The following table describes the custom proxies:

Component Objects	Companion Objects	Enabling Objects
Custom Proxies	None	Business System

Note: The association is recreated in the destination model between the Custom Proxy and the Interface if the Interface already exists in the destination model.

Custom Video Property

The following table describes the custom video property:

Component Objects	Companion Objects	Enabling Objects
None	None	Business System

Data Column

The following table describes the data column expansions:

Component Objects	Companion Objects	Enabling Objects
Extended columns	None	Parent data table, attribute being implemented by data column

Data Table

The following table describes the data table expansions:

Component Objects	Companion Objects	Enabling Objects
ndexes and their related components, constraints from data tables, related link tables, relationship triggers, columns (denormalized and foreign key), their related components, extended tables and extended constraints for migrated constraints	None	Parent table space, entity type being implemented, tablespaces referenced by extended tables

Database

The following table describes the database expansions:

Component Objects	Companion Objects	Enabling Objects
Extended databases and database files	Storage group, technical design	Tablespace referenced by extended databases

Default Dialog

The following table describes the default dialog expansions:

Component Objects	Companion Objects	Enabling Objects
Window Controls, GUI Events	Referenced prompts and Referenced commands	Parent procedure step, views, default edit patterns referenced by window controls, and GUI event handlers referenced by window controls

Window Controls are the set of objects contained in a window or dialog.

Default Edit Pattern

The following table describes the default edit pattern expansions:

Component Objects	Companion Objects	Enabling Objects
None	Dialect	Parent business system

Denormalized Column

The following table describes the denormalized column expansions:

Component Objects	Companion Objects	Enabling Objects
Extended columns	None	Parent table, attribute, relationship

Default Primary Window

The following table describes the default primary window expansions:

Component Objects	Companion Objects	Enabling Objects
Window Controls, GUI Events	Referenced prompts and referenced commands	Parent procedure step, views, default edit patterns referenced by window controls, GUI event handlers referenced by window controls

Window Controls are the set of objects contained in a window or dialog.

Dialog Flow

The following table describes the dialog flow expansions:

Component Objects	Companion Objects	Enabling Objects
None	Flows on, returns on exit states and commands	From and to procedure steps

Special Case Examples: Procedure Steps and Dialog Flows

In the following situations, Procedure Step A is a procedure step that is directly or indirectly selected to be migrated and Procedure Step B is another procedure step.

- Dialog Flow Created

  A Dialog Flow flows from/to Procedure Step A in the source model.

  The Dialog Flow flows to/from Procedure Step B in the source model.

  Procedure Step B exists in the destination model.

  The Dialog Flow does not exist in the destination model.

  Result: The Dialog Flow is copied to the destination model.

- Dialog Flow Replaced

  A Dialog Flow flows from/to Procedure Step A in the source model.

  The Dialog Flow flows to/from Procedure Step B in the source model.

  Procedure Step B exists in the destination model.

  The Dialog Flow exists in the destination model.

  Result: The Dialog Flow is replaced in the destination model.

- Dialog Flow Not Created

  A Dialog Flow flows from/to Procedure Step A in the source model.

  The Dialog Flow flows to/from Procedure Step B in the source model.

  Procedure Step B does not exist in the destination model.

  Result: The Dialog Flow is not copied to the destination model

- Dialog Flow Deleted

  A Dialog Flow flows from/to Procedure Step A in the destination model.

  The Dialog Flow does not exist in the source model.

  Result: The Dialog Flow is deleted from the destination model.

Special Case Examples: Data Views and View Matching

In the following situations, Procedure Step A is a Procedure Step that is selected directly or indirectly to be migrated and Procedure Step B is another Procedure Step.

- View Match Copied

  A Dialog Flow flows from/to Procedure Step A to/from Procedure Step B in the source model.

  A View Match exists in the source model between a view in Procedure Step A and a view in Procedure Step B.

  Procedure Step B exists in the destination model.

  The View in Procedure Step B exists in the destination model.

  The View Match does not exist in the destination model.

  Result: The View Match is copied to the destination model.

- View Match Replaced

  A Dialog Flow flows from/to Procedure Step A to/from Procedure Step B in the source model.

  A View Match exists in the source model between a view in Procedure Step A and a view in Procedure Step B.

  Procedure Step B exists in the destination model.

  The View in Procedure Step B exists in the destination model.

  The View Match exists in the destination model.

  Result: The View Match is replaced in the destination model.

- View Match Not Created

  A Dialog Flow flows from/to Procedure Step A to/from Procedure Step B in the source model.

  A View Match exists in the source model between a view in Procedure Step A and a view in Procedure Step B.

  The Procedure Step B exists in the destination model.

  The View in the Procedure Step B does not exist in the destination model.

  Result: The View Match is not copied to the destination model.

- View Match Deleted

  A Dialog Flow flows from/to Procedure Step A to/from Procedure Step B in the destination model.

  A View Match exists in the destination model between a view in Procedure Step A and a view in Procedure Step B.

  The Dialog Flow does not exist in the source model, the View in Procedure Step A does not exist in the source model, or the View Match does not exist in the source model.

  Result: The View Match is deleted from the destination model.

Entity

The following table describes the entity expansions:

Component Objects	Companion Objects	Enabling Objects
Aliases, attributes and their components, child subtypes and their components, classifiers, identifiers, partitionings, entity state changes and transitions, entity triggers, and contents.	None	Parent subject area, identifying relationships, entity types and work attribute sets used in component contents.

An identifying relationship is not a version control component of an identifier.

Special Cases

- To delete a relationship from an identifier, migrate the relationship with the entity type, and then delete the relationship from the identifier.
- Mutually exclusives are migrated if at least two of the relationships participating in the mutually exclusive exist in the destination model.

Special Case Examples: Entities and Mutually Exclusives

In the following situation, Entity A is an entity that is selected for migration.

- Mutually Exclusive Copied

  Entity A contains a Mutually Exclusive A that has Relationship A and Relationship B as participants.

  Entity A exists in the destination model.

  Relationship A and Relationship B exist in the destination model.

  Result: Mutually Exclusive A is copied to the destination model.

- Mutually Exclusive Not Copied

  Entity A contains a Mutually Exclusive A that has Relationship A and Relationship B as participants.

  Entity A exists in the destination model.

  Relationship A exists in the destination model.

  Relationship B does not exist in the destination model.

  Result: Mutually Exclusive A is not copied to the destination model.

Event

Component Objects	Companion Objects	Enabling Objects
None	None	None

Special Case

The migration of an event will not create the event in the destination model if there are no usages of the event.

Exit State

The following table describes the exit state expansions:

Component Objects	Companion Objects	Enabling Objects
None	Dialect	Parent business system unless this is a global exit state

Special Case

If migration of exit states changes their Code Generation (CG) value, regenerate all action blocks that use the migrated exit states.

Foreign Key Column

The following table describes the foreign key column expansions:

Component Objects	Companion Objects	Enabling Objects
Extended Columns	None	Parent table, constraint, attribute, source column (data or foreign key)

Function

The following table describes the function expansions:

Component Objects	Companion Objects	Enabling Objects
Views, expected effects	Dependent external objects, dependent events	Parent function unless aggregate function is a root function. Entity types, subtypes, attributes, work attributes and work attribute sets referenced by any migrated views. Entity types and subtypes referenced by migrated expected effects.

Special Cases

- An imported or exported dependency is migrated only if the other function importing or exporting the dependency exists in the destination model.
- A mutually exclusive or parallel dependency is migrated if the input function and at least two of the outputs functions participating in the dependency exist in the destination model.
- A closure dependency is migrated if the mutually exclusive dependency that is being closed and at least two of the input functions and the output function exist in the destination model.

Index

The following table describes the index expansions:

Component Objects	Companion Objects	Enabling Objects
Index spaces, extended indexes and extended index spaces	None	Identifier if identifier-based index, owning data table, referenced data columns. Relationships if identifier contains relationships. Tablespaces referenced by component extended indexspaces.

Interface Type

The following table describes various interface type expansions:

Component Objects	Companion Objects	Enabling Objects
Aliases, attributes and their components, child subtypes and their components, classifiers, identifiers, partitionings, entity state changes and transitions, entity triggers, contents, and interface type model. Art objects for interface type model.	None	Parent subject area, identifying relationships, entity types, and work attribute sets used in component contents. Entity types, entity subtypes, partitionings, and relationships referenced by interface type model.

Link Table

The following table describes the link table expansions:

Component Objects	Companion Objects	Enabling Objects
Indexes and components, constraints and components, columns and components, extended tables and extended constraints for migrated constraints	None	Relationship, source and destination table, tablespace, tablespaces referenced by extended tables

Matrix

The following table describes the matrix expansions:

Component Objects	Companion Objects	Enabling Objects
Cell values	User-defined class if the X and/or Y axis of the matrix is a user-defined class.	None

Special Cases

A row or column in the matrix is migrated only if the aggregate object the row or column references already exists in the destination model.

Navigation Diagram

The following table describes the navigation diagram expansions:

Component Objects	Companion Objects	Enabling Objects
Usages of windows, dialogs, and procedure steps	None	Referenced windows, dialogs, and procedure steps

Online Load Module

The following table describes the online load module expansions:

Component Objects	Companion Objects	Enabling Objects
Package list entries	None	None

Special Case

The migration of an online load module, or the migration of packaging for an online load module, can create an invalid situation in the destination model, where a load module would not contain packaging for any procedure steps. If this situation is detected, migration delete the load module to prevent the corruption.

Operations Library

The following table describes the operations library expansions:

Component Objects	Companion Objects	Enabling Objects
None	None	Technical System

Organizational Unit

The following table describes the organizational unit expansions:

Component Objects	Companion Objects	Enabling Objects
None	None	Parent organizational unit unless this is the root organizational unit

Packaging

The following table describes the packaging expansions:

Component Objects	Companion Objects	Enabling Objects
Transaction codes	Load module	Associated procedure steps

Special Cases

- The clear screen association is recreated in the destination model between the transaction code and packaging object if the transaction code is not already the clear screen transaction code for another procedure step in the destination model.
- If the migration of packaging causes an online load module to be unused in the destination model, the load module is deleted.

Procedure

The following table describes the procedure expansions:

Component Objects	Companion Objects	Enabling Objects
Procedure steps and their related components	None	Parent business system

Procedure Step

The following table describes the procedure step expansions:

Component Objects	Companion Objects	Enabling Objects
Procedure step action diagram and its related components, dialog boxes, primary window, screen and its related components, views, local program function keys, unformatted input information, packaging, and web operations	System-wide program function keys referenced by local program function keys, custom video properties	Parent procedure, entity types, subtypes, attributes, work attribute sets and work attributes referenced by migrated views

Packaging is the inclusion of a procedure step in a load module.

Note: The association is recreated in the destination model between the Web Operation and the Web Service if the Web Service already exists in the destination model.

Special Cases

- A dialog flow from the procedure step is migrated only if the procedure step at the other end of the dialog flow already exists in the destination model. A view match along the dialog flow is migrated only if an equivalent view exists in the destination model.

- The dialog flows that only exist in the destination model is deleted only if the source and target procedure steps exist in both models. The dialog flows are created or replaced if the source and target procedure steps exist in both models.

Special Case Examples: Procedure Steps and Dialog Flows

In situations 1 through 4, Procedure Step A is a procedure step that is directly or indirectly selected to be migrated and Procedure Step B is another procedure step.

- Dialog Flow Created

  A Dialog Flow flows from/to Procedure Step A in the source model.

  The Dialog Flow flows to/from Procedure Step B in the source model.

  Procedure Step B exists in the destination model.

  The Dialog Flow does not exist in the destination model.

  Result: The Dialog Flow is copied to the destination model.

- Dialog Flow Replaced

  A Dialog Flow flows from/to Procedure Step A in the source model.

  The Dialog Flow flows to/from Procedure Step B in the source model.

  Procedure Step B exists in the destination model.

  The Dialog Flow exists in the destination model.

  Result: The Dialog Flow is replaced in the destination model.

- Dialog Flow Not Created

  A Dialog Flow flows from/to Procedure Step A in the source model.

  The Dialog Flow flows to/from Procedure Step B in the source model.

  Procedure Step B does not exist in the destination model.

  Result: The Dialog Flow is not copied to the destination model.

- Dialog Flow Deleted

  A Dialog Flow flows from/to Procedure Step A in the destination model.

  The Dialog Flow does not exist in the source model.

  Result: The Dialog Flow is deleted from the destination model.

Special Case Examples: Data Views and View Matching between Procedure Steps

In the following situations, Procedure Step A is a Procedure Step that is directly or indirectly selected to be migrated and Procedure Step B is another Procedure Step.

- View Match Copied

  A Dialog Flow flows from/to Procedure Step A to/from Procedure Step B in the source model.

  A View Match exists in the source model between a view in Procedure Step A and a view in Procedure Step B.

  Procedure Step B exists in the destination model.

  The View in Procedure Step B exists in the destination model.

  The View Match does not exist in the destination.

  Result: The View Match is copied to the destination model.

- View Match Replaced

  A Dialog Flow flows from/to Procedure Step A to/from Procedure Step B in the source model.

  A View Match exists in the source model between a view in Procedure Step A and a view in Procedure Step B.

  Procedure Step B exists in the destination model.

  The View in Procedure Step B exists in the destination model.

  The View Match exists in the destination model.

  Result: The View Match is replaced in the destination model.

- View Match Not Copied

  A Dialog Flow flows from/to Procedure Step A to/from Procedure Step B in the source model.

  A View Match exists in the source model between a view in Procedure Step A and a view in Procedure Step B.

  The Procedure Step B exists in the destination model.

  The View in Procedure Step B does not exist in the destination model.

  Result: The View Match is not copied to the destination model.

- View Match Deleted

  A Dialog Flow flows from/to Procedure Step A to/from Procedure Step B in the destination model.

  A View Match exists in the destination model between a view in Procedure Step A and a view in Procedure Step B.

  One of the following:

- The Dialog Flow does not exist in the source model *OR*
- The View in Procedure Step A does not exist in the source model *OR*
- The View Match does not exist in the source model.

  Result: The View Match is deleted from the destination model.

Process

The following table describes the process expansions:

Component Objects	Companion Objects	Enabling Objects
Views, expected effects, its process action diagram if process is elementary; entity state transition usages	Dependent external objects, dependent events	Parent processes or functions, entity types, subtypes, attributes, work attributes, work attribute sets referenced by migrated views, entity type and subtypes referenced by migrated expected effects

Special Cases

- Parent processes or functions, entity types, subtypes, attributes, work attributes, work attribute sets referenced by migrated views, entity type and subtypes referenced by migrated expected effects.
- Imported or exported dependency is migrated only if the other processes importing or exporting the dependency exist in the destination model.
- A mutually exclusive or parallel dependency is migrated if the input process and at least two of the output processes that participate in the dependency exist in the destination model.
- A closure dependency is migrated if the mutually exclusive dependency that is being closed and at least two of the input processes and the output process exist in the destination model.

Special Case Examples: Data Views and View Matching between Processes

In the following situations, Process A is a process that is directly or indirectly selected to be migrated and External Object B is another external object.

- View Match Copied

  An Information Flow exists between Process A and External Object B in the source model.

  A View Match exists in the source model between a view in Process A and External Object B.

  The View Match does not exist in the destination model.

  Result: The View Match is copied to the destination model.

- View Match Replaced

  An Information Flow exists between Process A and External Object B in the source model.

  A View Match exists in the source model between a view in Process A and External Object B.

  The View Match exists in the destination model.

  Result: The View Match is replaced in the destination model.

- View Match Deleted

  An Information Flow exists between Process A and External Object B in the destination model.

  A View Match exists in the destination model between a view in Process A and External Object B.

  One of the following:

  - External Object does not exist in the source model
  - Information Flow does not exist in the source model
  - View in Process A does not exist in the source model
  - View Match does not exist in the source model.

    Result: The View Match is deleted from the destination model.

Special Case Examples: Processes and Dependencies

In the following situations, Process A is a process that is directly or indirectly selected to be migrated and Processes B and C are other processes.

- Dependency Copied

  A Dependency exists between Process A and Process B in the source model.

  Process B exists in the destination model.

  The Dependency does not exist in the destination model.

  Result: The Dependency is copied to the destination model.

- Dependency Replaced

  A Dependency exists between Process A and Process B in the source model.

  Process B exists in the destination model.

  The Dependency exists in the destination model.

  Result: The Dependency is replaced in the destination model.

- Mutually Exclusive or Parallel Dependency Copied

  A Mutually Exclusive or Parallel Dependency exists between Process A and Processes B and C in the source model.

  Processes B and C exist in the destination model.

  The Dependency does not exist in the destination model.

  Result: The Dependency is copied to the destination model.

- Mutually Exclusive or Parallel Dependency Replaced

  Parallel Dependency Replaced

  A Mutually Exclusive or Parallel Dependency exists between Process A and Processes B and C in the source model.

  Processes B and C exist in the destination model.

  The Dependency exists in the destination model.

  Result: The Dependency is replaced in the destination model.

- Mutually Exclusive or Parallel Dependency Not Copied

  A Mutually Exclusive or Parallel Dependency exists between Process A and Processes B and C in the source model.

  Process B does not exist in the destination model *OR*

  Process C does not exist in the destination model *OR*

  Processes B and C do not exist in the destination model.

  Result: The Dependency is not copied in the destination model.

- Mutually Exclusive or Parallel Dependency Deleted

  A Mutually Exclusive or Parallel Dependency exists between:

  Process A and Processes B and C in the destination model.

  Process B does not exist in the source model *OR*

  Process C does not exist in the source model *OR*

  Processes B and C do not exist in the source model *OR*

  The Dependency does not exist in the source model.

  Result: The Dependency is deleted in the destination model.

- Closure Dependency Copied

  A Closure Dependency exists between Processes A and B and Process C in the source model OR a Closure Dependency exists between Processes B and C and Process A in the source model.

  Processes B and C exist in the destination model.

  The Mutually Exclusive Dependency that is closed by the Closure Dependency exists in the destination model.

  The Closure Dependency does not exist in the destination model.

  Result: The Dependency is copied to the destination model.

- Closure Dependency Replaced

  A Closure Dependency exists between Processes A and B and Process C in the source model OR a Closure Dependency exists between Processes B and C and Process A in the source model.

  Processes B and C exist in the destination model.

  The Mutually Exclusive Dependency that is closed by the Closure Dependency exists in the destination model.

  The Closure Dependency exists in the destination model.

  Result: The Dependency is replaced in the destination model.

- Closure Dependency Not Copied

  A Closure Dependency exists between Processes A and B and Process C in the source model OR a Closure Dependency exists between Processes B and C and Process A in the source model.

  Process B does not exist in the destination model *OR*

  Process C does not exist in the destination model *OR*

  The Mutually Exclusive Dependency that is closed by the Closure Dependency does not exist in the destination model.

  Result: The Dependency is not copied to the destination model.

- Closure Dependency Deleted

  A Closure Dependency exists between Processes A and B and Process C in the destination model OR a Closure Dependency exists between Processes B and C and Process A in the destination model. The Closure does not exist in the source model.

  Result: The Dependency is deleted in the destination model.

Relationship

The following table describes the relationship expansions:

Component Objects	Companion Objects	Enabling Objects
None	None	Source and destination entity type or subtype

Special Cases

Mutually exclusives are migrated if at least two of the relationships participating in the mutually exclusive exist in the destination model.

Special Case Examples: Relationships and Mutually Exclusives

In the following situation, Relationship A is a relationship that is selected to be migrated.

- Mutually Exclusive Copied

  Relationship A participates in Mutually Exclusive A with Relationship B.

  Relationship A and Relationship B exist in the destination model.

  Result: Mutually Exclusive A is copied to the destination model.

- Mutually Exclusive Not Copied

  Relationship A participates in Mutually Exclusive A with Relationship B.

  Relationship A exists in the destination model.

  Relationship B does not exist in the destination model.

  Result: Mutually Exclusive A is not copied to the destination model.

Screen

The following table describes the screen expansions:

Component Objects	Companion Objects	Enabling Objects
Literals, fields, and special fields	Referenced templates and dialects	Parent procedure step, views and default edit patterns referenced by screen fields

For details on special situations, see Attributes and Prompts.

Scroll Amount Value

The following table describes the scroll amount value expansions:

Component Objects	Companion Objects	Enabling Objects
None	Dialect	None

Server Manager

The following table describes the server manager expansions:

Component Objects	Companion Objects	Enabling Objects
Package list entries	None	None

Specification Type

The following table describes the specification type expansions:

Component Objects	Companion Objects	Enabling Objects
Aliases, attributes and their components, child subtypes and their components, classifiers, identifiers, partitionings, entity state changes and transitions, entity triggers, and contents.	None	Parent subject area, identifying relationships, entity types and work attribute sets used in component contents.

Storage Group

The following table describes the storage group expansions:

Component Objects	Companion Objects	Enabling Objects
DASD volume	Technical design	None

Strategy

The following table describes the strategy expansions:

Component Objects	Companion Objects	Enabling Objects
Tactics	None	None

Subject Area

The following table describes the subject area expansions:

Component Objects	Companion Objects	Enabling Objects
None	None	Parent subject area unless aggregate object is root subject area

System-Wide PF Key

The following table describes the system-wide PF key expansions:

Component Objects	Companion Objects	Enabling Objects
None	Commands	Parent business system

Tablespace

The following table describes the tablespace expansions:

Component Objects	Companion Objects	Enabling Objects
Data sets, extended tablespaces and extended data sets	Storage groups	Parent database

Tactic

The following table describes the tactic expansions:

Component Objects	Companion Objects	Enabling Objects
None	None	Parent strategy

Technical Design Default

The following table describes the technical design default expansions:

Component Objects	Companion Objects	Enabling Objects
Extended technical design defaults, bind package defaults, and package lists	Default database, default storage groups	None

Template

The following table describes the template expansions:

Component Objects	Companion Objects	Enabling Objects
Literals and special fields	Referenced templates, dialects, and prompts	Parent business system, default edit patterns referenced by screen fields

Transaction Operation

The following table describes the transaction operation expansions:

Component Objects	Companion Objects	Enabling Objects
Constraints and external parameters	Referenced commands	Owning entity type or work attribute set, procedure step or delegated to transaction and views referenced by component constraints

Typemap

The following table describes the typemap expansions:

Component Objects	Companion Objects	Enabling Objects
Correspondences	None	Action block

User Object

The following table describes the user object expansions:

Component Objects	Companion Objects	Enabling Objects
None	User Class	None

Web Service Definition

The following table describes the web service definition:

Component Objects	Companion Objects	Enabling Objects
Web Service	None	Business System

Note: The association is recreated in the destination model between the Web Service and the Web Operation if the Web Operation already exists in the destination model.

Window Load Module

The following table describes the window load module expansions:

Component Objects	Companion Objects	Enabling Objects
Package list entries	None	None

Special Cases

The migration of a window load module or the migration of packaging for a window load module can create an invalid situation in the destination model, where a load module would not contain any packaging for procedure steps. If this situation is detected, migration delete the load module to prevent the corruption.

Work Attribute

The following table describes the work attribute expansions:

Component Objects	Companion Objects	Enabling Objects
Permitted values	Dialect	Parent work attribute set

For details on Special Situations, see Attributes and Prompts.

Special Cases

- Prompts are created if they exist in the source model and do not already exist in the destination model. Prompts are replaced if they exist in both the source and destination models. Prompts are unchanged if they exist in the destination model but not in the source model.
- The Prompt values are created if they exist in the source model and do not already exist in the destination model. The Prompt values are replaced if they exist in both the source and destination models. The Prompt values are unchanged if they exist in the destination model but not in the source model.

Work Attribute Set

The following table describes the work attribute set expansions:

Component Objects	Companion Objects	Enabling Objects
Work attributes and their related components	None	None

z/OS Library

The following table describes the z/OS library:

Component Objects	Companion Objects	Enabling Objects
None	None	Technical System

Modifying Referential Integrity Triggers for Migration

Job CEJOB05A modifies operational rules for the migration of referential integrity (RI) triggers with their parent objects: entity types and data records. The decision about whether to modify the default rules for migration must be made within your development group. Instructions for preparing the JCL to run CEJOB05A, and the effects of running it are described in the following sections.

Migration

When an entity type or data record is migrated from one model to another, component objects are also migrated. Referential integrity objects consist of an action block definition object (ACBLKTD) and an associated implementation logic object (IMPLGIC). These objects are components of an entity type for *delete* triggers and a data record (through the implemented linkage) for *relationship* triggers. The objects always migrate with the parent object.

Common Ancestry for RI Triggers

The RI triggers for entity types or data records typically have common ancestry if the entity types or data records to which they belong have common ancestry. However, when you remove the implementation for a relationship from technical design, the RI trigger objects are deleted. When you reimplement the relationship, new RI trigger objects are added which will not have common ancestry, even though their parent objects correspond.

If you remove a relationship from the Data Structure list, then reimplement, readopt the data record to which it belongs before attempting a migration. If you perform Transformation (rather than Retransformation), all relationship implementations are deleted and new ones are added. Adopt all data records before doing a migration.

RI Trigger Implementation Structure

When the RI triggers are generated on the Host Encyclopedia, an implementation structure is built that associates each trigger, through implementation usage (IMPUSE) objects, with the triggers that it calls or that call it.

```
TRIGGER 1 ...calls...IMPUSE...called by...TRIGGER 2
```

Maintaining Implementation Structure

With the default operational rules preloaded in the CA Gen schema tables, the implementation structure (IMPUSE objects, associations) from the SOURCE must be maintained in the DESTINATION or the migration fail.

- If the triggers have not been generated, this implementation structure does not exist, and migration completes successfully.
- If the triggers have been generated in the SOURCE, the IMPUSE objects and their associations are migrated as components. All triggers directly associated with the one being migrated must exist in the DESTINATION, or else migration fails.
- If the triggers have been generated in the DESTINATION, and they do *not* correspond to those from the SOURCE, they are flagged for the deletion during migration processing. Unless all associated triggers are deleted, the migration fails.
- Enforcing the SOURCE model implementation structure for RI triggers is appropriate when all models participating in migration of data objects have equivalent entity relationship diagrams and technical designs.

Relax the Rules

The schema update in CEJOB05A relaxes the rules for enforcing SOURCE model implementation structure for RI triggers. The IMPUSE objects and their associations from SOURCE are ignored for migration. RI trigger objects migrate with the parent entity type or data record.

- If the triggers correspond, those migrating from the SOURCE replace those in the DESTINATION. If the DESTINATION has had RI triggers generated, the implementation structure is retained.
- If the triggers do not correspond, any implementation structure in the DESTINATION is disrupted. The old triggers are deleted and those from SOURCE are added. Because the triggers are new objects for the DESTINATION, the old implementation structure does not apply.

Relaxing the rules for migrating RI triggers is appropriate when SOURCE and the DESTINATION models differ in the content of an entity relationship diagram or data structure. Typically, this is the case when development models contain only a portion of the full data model.

Edit the JCL

Edit CEJOB05A before running it. See the following sample and instructions to understand and plan your edits.

Sample CEJOB05A JCL

```
//*
//PS10          EXEC PGM=IKJEFT01,DYNAMNBR=30
//STEPLIB       DD DSN=XXXXXXXXXXXXXXXX,DISP=SHR
//SYSPROC       DD DSN=&TILCLIB,DISP=SHR
//** UNCOMMENT ONE OF THESE ENTRIES TO EXECUTE SQL **
//** SYSIN   DD DSN=&TILSQL(TRG54REV),DISP=SHR
//** SYSIN   DD DSN=&TILSQL(TRG54REV),DISP=SHR
//                    DD DSN=&TILSQL(COMMIT),DISP=SHR
//SYSPRINT      DD SYSOUT=&SOUT
//SYSPRINT      DD SYSOUT=&SOUT
```

- To modify the CA Gen Release 8.5 schema and relax the rules for migrating RI triggers, uncomment the line:

```
//**SYSIN DD DSN=&TILSQL(TRG54CHG),DISP=SHR
```

- To reset the CA Gen Release 8.5 schema values back to the original values and enforce source model duplication structure on RI triggers, uncomment the line:

```
//**SYSIN DD DSN=&TILSQL(TRG54REV),DISP=SHR
```

Index

N

O

P

R

S

T

U

V

W