CA Repository for z/OS

Product Guide

r7.2

CA Product References

This document references the following CA product:

- CA Repository for z/OS

Contact CA

Contact Technical Support

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

- 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

Provide Feedback

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

If you would like to provide feedback about CA product documentation, complete our short customer survey, which is also available on the CA Support website, found at http://ca.com/docs.

Contents

Chapter 8: Commands and Features 101

Chapter 1: Introduction

CA Repository for z/OS (the Repository) is a powerful data management tool that enables organizations to identify, understand, coordinate, and effectively use enterprise-wide information assets.

Based on open, non-proprietary architecture and popular relational databases, the Repository supports business and technical metadata for data warehousing, data administration, and application development efforts.

This section contains the following topics:

Benefits

The Repository provides valuable advantages at various stages of data integration, data warehousing, and application development projects, including design, development, deployment and management. It offers the following unique benefits:

- Enables a common, shared view of metadata that is automatically captured from virtually all aspects of your organization information environment.
- Improves access to information by providing users with a web-based directory of information in business terms. The Repository helps users understand existing information, its location, and its value to users.
- Enables clear understanding of the existing data environment by automatically documenting the location and nature of potential data sources. It ensures that appropriate data is used and enterprise-wide information assets from ERP and legacy systems are fully leveraged.
- Reduces development time and improves quality by providing definitions of available source and target data, and transformation rules and code values that can be used and reused in the data mapping process.
- Reduces maintenance time by enabling developers to easily identify the impact of changes in an application or a data structure. This significantly reduces time required for analysis in the early stages of a project.
- Improves support response time by facilitating rapid response to user questions about data. Support staff can quickly trace data origins to understand all factors that may lead to questionable data values.

- Streamlines application development by providing a single source for logical, physical and process aspects of the application environment and documenting all aspects of the application development life cycle. The Repository helps identify redundant processes, data, and applications to reduce duplicated efforts.
- Enforces naming standards on new assets while identifying and documenting redundancy and inconsistencies in existing assets. It enables consistent taxonomy across the enterprise and promotes effective communication.
- The advanced Web user interface ensures access to vital information asset descriptions and interrelationships, accelerating data warehouse development, reducing application development and maintenance costs, and enabling information sharing across heterogeneous environments.

Documentation

This guide assumes that the appropriate Repository components are installed at your site. For more information about installing CA Repository for z/OS, see the *CA Repository for z/OS Installation Guide.*

As you use this guide, you might find it helpful to have the *CA Repository for z/OS Administration Guide* available for reference.

Chapter 2: Understanding the Architecture

This chapter describes basic Repository concepts , its data models, dialogs, and user privileges.

This section contains the following topics:

What Is a Repository?

A data *repository* is a shareable collection of information supporting business data processing functions. It stores definitions of systems components that your organization can use in maintaining, running, and analyzing applications.

The Repository

The Repository is a centralized, DB2-based data repository. Unlike standard data dictionaries and other repository-like products, which support only a limited number of standard entity types and relationships, the Repository features sophisticated user interfaces that you use to define and maintain both standard and user-defined *entities*, *attributes*, and *relationships*.

This flexibility allows you to use the Repository to develop applications and tool sets that enhance your organization's application-development efficiency. By allowing you to coordinate application and data standards across development teams, the Repository also simplifies the evolution of applications.

The Repository Interface

The Repository interface is ISPF-based. It allows you to access and manipulate repository data from an MVS/ISPF terminal using ISPF window panels, programmable function (PFn) keys, function (Fn) keys, and text commands.

The Repository follows the IBM Systems Application Architecture (SAA) standard, including its Common User Access (CUA interface guidelines). The highly flexible, the SAA/CUA-compliant Repository interface provides you the means to customize the Repository data model to meet your organization's specific requirements. Depending on your level of authorization, you can use the Repository to select, insert, delete, update (modify), replace, or copy any entity, relationship, or association in the repository.

The Repository and SAA/CUA

Since the Repository follows the SAA/CUA interface guidelines, the Repository is relatively consistent with other SAA-compliant applications, especially in areas such as screen appearance, terminology, messages, and selection methods.

Because many other software firms also follow SAA standards, most users with prior computer experience find the Repository's general layout, functions, and commands familiar and easy to learn. If you are new to computers, the skills and knowledge you acquire using the Repository will make it easier for you to learn other SAA-compliant applications.

The Repository Data Models

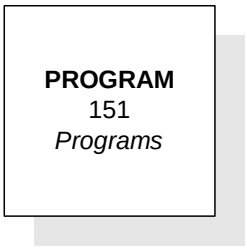
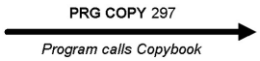
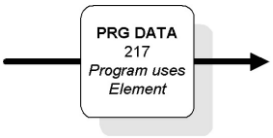
This section explains the Repository's Entity-Relationship model (also known as the *E/R model* or *data model*).

The Entity-Relationship (E/R) Model

Many different opinions exist as to which components constitute an entity-relationship model. The Repository E/R model has evolved over time to reflect both changes in industry standards and insights acquired through practical experience. In the simplest terms, it is an extension of the more common entity-attribute-relationship model.

The Repository uses the following basic types of components to store metadata: *entity types*, *relationship types*, and *association types*. Each component has its own type name and type number.

The following table shows how the Repository model diagrams represents these types.

This Diagram Object	Represents	Which Is
	An Entity Type	A stand-alone object, not dependent upon the existence of other objects. Examples: DATABASE, TABLE, PROGRAM, COPYBOOK.
	An Association Type	A link that ties two entities together. Example: The fact that PROGRAM X uses COPYBOOK Y is recorded by the existence of an association between the two called PRG COPY.
	A Relationship Type	A link that ties two entities together AND stores information about this link or tie. Example: PROGRAM X sets ELEMENT Y. Relationship PRG DATA records the tie from PROGRAM X to ELEMENT Y and stores specifics on how PROGRAM X uses or interacts with ELEMENT Y. A relationship can also be related or associated with other entities or relationships. Example: KEYCOL is a relationship between the entity type KEY and the relationship COLUMN.

Note: Entity, entity type, association, association type, relationship, and relationship type names are all UPPERCASE in text.

Entities and Entity Types

In the Repository, as in any relational database system, all data in the repository is displayed as a collection of tables. The Repository entity types are, for the most part, analogous to tables. Each entity type stores a number of rows of data, each of which represents an individual data item, or *entity*.

Entity Attributes and Attribute Types

Each entity is composed of a number of different values, each of which corresponds to one of the columns in the table. In the Repository, each of these values is known as an attribute, and the columns themselves are referred to as attribute types. The following table illustrates these concepts.

	Attribute Type 1	**Attribute Type 2**	**Attribute Type 3**
Entity A	attribute	attribute	attribute
Entity B	attribute	attribute	attribute
Entity C	attribute	attribute	attribute
Entity D	attribute	attribute	attribute

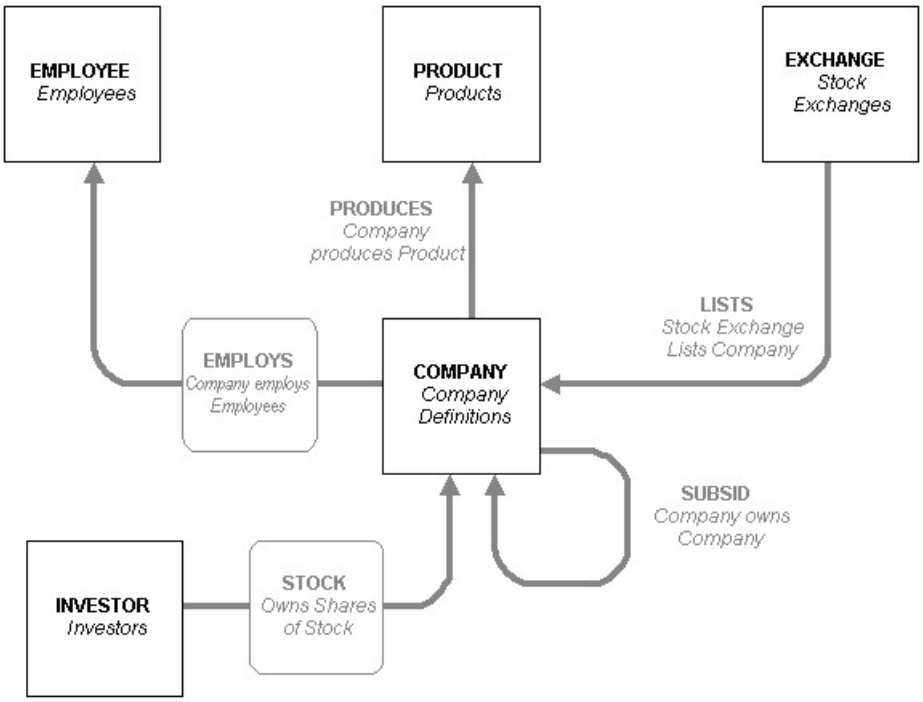
To further clarify these concepts, the following table is shown below with real-world values in place of the generic entity/attribute/relationship terms.

In this example, Entity A (case an individual) is defined in the repository as belonging to the entity type EMPLOYEE. All entities belonging to the entity type EMPLOYEE must have attributes that correspond to the attribute types Name, Social Security Number, and Date of Birth respectively. Entity A has the attributes Arthur Scribbler, 111-11-1111, and 09/25/61.

	Name	**Social Security #**	**Date of Birth**
Entity A	Arthur Scribbler	111-11-1111	09/25/61
Entity B	Lisa Stechmeister	222-22-2222	05/31/68
Entity C	Derwood Walters	333-33-3333	06/06/66
Entity D	Andy Suntoast	444-44-4444	09/13/72

Entity Types in the Repository Data Model

Entity types are represented in the Repository metadata models by rectangles similar to those pictured in the following illustration.



Entities and entity types are the fundamental components in the Repository. Association and relationship types (described in the following section) are used to make connections between entities. Their respective occurrences cannot exist in the repository independently of the entities they connect.

For example, in the repository, the TABLE entity type is used to store definitions of DB2 tables. The attribute types that make up the TABLE entity type all store data that is required to describe a DB2 table, such as Table name or Table creator.

Associations and Association Types

Associations are used to link together two entities. Like entities, associations are grouped together in a table, which, in the case of the associations, is known as an association type.

Note: While the component=table analogy helps to explain the meaning of the model diagrams, administrators should remember that the Repository components do not map directly to underlying DB2 storage tables. This is particularly true of association types that do not have their own storage tables.

Association types are depicted in model diagrams as a named arrow. The arrow always points from one entity type to another entity type.

Note: Association and relationship types can reference either entity types or relationship types for their source and target requirements.

Source Entity Type and Target Entity Type

The entity type at the base of the arrow is referred to as the source entity type, while the entity the arrow points at is referred to as the target entity type.

Each association in an association type links one entity from the source entity type to one entity in the target entity type. The significance of the directional aspect of an association (that is, source, target, and so on) is that a source entity is generally thought to **use** its respective target entity while a target entity is thought **to be used** by its source entity.

In the following example, the associations A-D belong to the association type EXAMPLE.

	Source: Entity A	**Target: Entity B**
Association A	entity	entity
Association B	entity	entity
Association C	entity	entity
Association D	entity	entity

It is important to note that while the entities listed in the source column are not necessary the same entity, they are all of the same entity type if the associations are of the same association type. This is also true of the target entities.

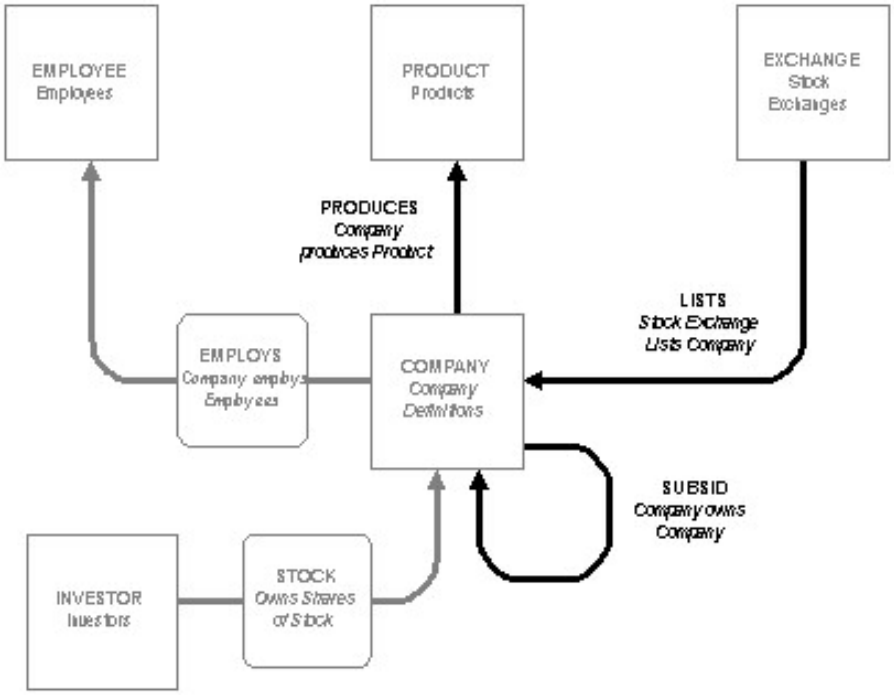
In simpler terms: associations point to entities and association types point to entity types.

The following table is the same table with real-world examples in place of the generic model terms.

	Source: COMPANY	**Target: PRODUCTS**
Association A	Standard Products	Doorknobs
Association B	Microsmall	Doors 1.0
Association C	QBM Software	BS/2
Association D	RPM	Toasters

Association types are represented in metadata models by named arrows. The arrow starts at the source entity type and points to the target entity type.

The following illustration shows association types in a sample model.



For example, in the Repository, the IN TS association type links the TABLE entity type to the TBSPACE entity type. The TABLE entity type stores definitions of DB2 tables, while the TBSPACE entity type stores definitions of DB2 table spaces. An IN TS association, which connects together occurrences of each of these types, indicates that a particular DB2 table is stored in a particular DB2 table space.

Relationships and Relationship Types

Like associations, relationships are used to link together two entities.

The primary difference between relationships and associations is that relationships also store additional information about the connection between the two entities they link. So, like entities, relationships also have attributes.

The basic table-like structure of a relationship type is shown in the following table.

Source: Entity Type A	Target: Entity Type B	Attribute Type 1

	Source: Entity	Type A	Target: Entity	Type B	Attribute Type 1
Attribute type 2	Attribute type 3	Relationship A	entity	entity	attribute
attribute	attribute	Relationship B	entity	entity	attribute
attribute	attribute	Relationship C	entity	entity	attribute
attribute	attribute	Relationship D	entity	entity	attribute
attribute	attribute				

As with association types, it is important to note that while the entities listed in the source column are not necessary the same entity, they will all be of the same entity type if the relationships are of the same relationship type. This is also true of the target entities.

In simpler terms:

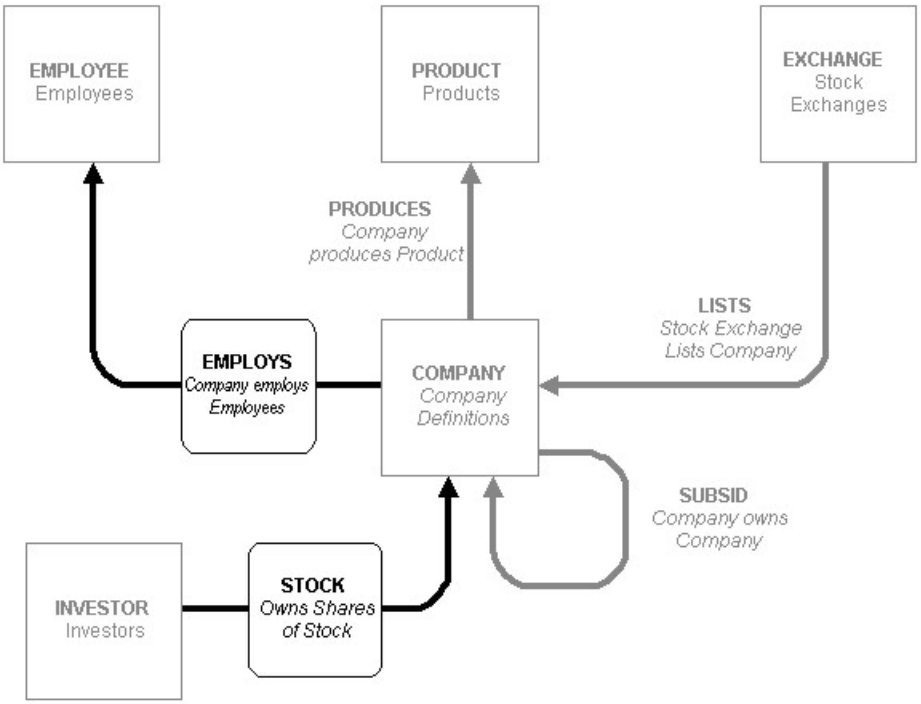
- Relationships point to entities
- Relationship types point to entity types.

The following is same table with real-world examples in place of the generic model terms.

	Source: Company	Target: Employee	Position	Date of Hire	Salary
Relationship A	Microsmall	Lisa Stechmeister	Art Director	05/13/92	$50k
Relationship B	QBM Software	Derwood Walters	Mail Handler	02/16/92	$75k
Relationship C	RPM	Andy Suntoast	Tester	12/25/92	$30k
Relationship D	Standard Products	Arthur Scribbler	Technical Writer	02/12/90	$40k

The basic purpose of a relationship is to link two entities. The data stored in a relationship will, in most cases, only be relevant with respect to the source and target of the relationship. In the example above, for instance, the meaning of the data stored under the attribute type Position becomes obvious only when retrieved with a source company and a target employee.

Relationship types are depicted in model diagrams as a combination of an arrow and a round-cornered box. As with association types, the arrow points from the source entity type to the target entity type.



For example, in the Repository, the COLUMN relationship type links the TABLE entity type to the ELEMENT entity type.

- The TABLE entity type stores definitions of DB2 tables, while the ELEMENT entity type stores definitions of data elements.
- A COLUMN relationship, which connects a TABLE entity to an ELEMENT entity, indicates that a particular DB2 table has a column that stores a particular type of data. The COLUMN relationship also stores information such the sequence number of the column, whether or not NULL values are allowed, and so on.

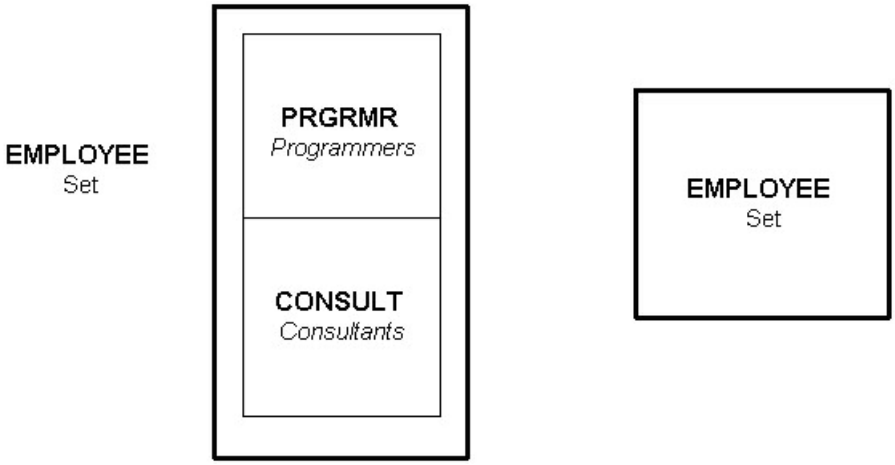
An Entity Set

An entity set is a variation of an entity type. It represents a group of entities that can, for most purposes, be used interchangeably.

In situations where several entity types are used to store similar data, entity sets can provide a more efficient method of designating the source and/or target entities for associations and relationships.

Using entity sets, you can select source and target entities from more than one entity type. Entity sets also force uniqueness among entities of all entity types belonging to the set. The entity type/entity set concept is similar to that of subtypes and supertypes.

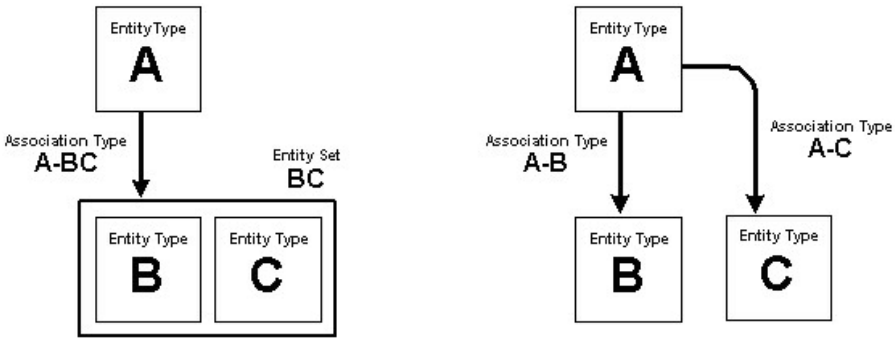
Entity sets are represented in data models by either of the heavy-bordered boxes pictured below. The box to the left depicts both the set and its constituent entity types.



By defining sets, several additional association and relationship types can be left out of the models.

If, for example, an entity set was the source entity type of a relationship type, entities from any of the entity types belonging to the set could be used as source entities. Without the set, a different relationship type would be needed to point to each entity type in the set.

This concept is illustrated in the following diagram.



On the left, the association type ABC links entity type A to the entity set BC. This arrangement allows entities of the A entity type to be linked to either B or C entities through ABC associations. Without the entity set two separate association types (right) would be required to achieve this same effect.

Entities and Models

Customizing data models, whether by adding or changing attribute types or by adding entirely new entity, relationship, or association types, is referred to in this product guide as extending. For more information, see the *Administration Guide*.

General Attributes: Name, Status, and Version

The Repository uses three attribute types to uniquely identify each entity in the repository. These are:

- Name
- Status
- Version

Every entity type in the Repository has attribute types for storing this information, and every entity has attributes for these types. Throughout the rest of this product guide, you will find references to these attributes when manipulating repository data.

Relationship types also have their own name, status, and version attribute types. In addition, they have attribute types for the name, status, and version of both their source and target entities. Though the total number of attribute types varies between relationship types, all relationship types have at least these nine attribute types.

Associations do not have their own name, status, and version. Instead, they consist only of the name, status, and version of their source and target entity. Every association type you work with in the repository has only these six attribute types.

The Cross Reference Table

The Cross Reference Table is an internal table that maps entities, relationships, and associations (otherwise known as objects).

- Each object type has a unique entity type
- Each object row has a unique entity ID

The Cross Reference Table maps each object and its sources and targets. The following tables describe how this is done.

DBXDB2.DBX_DB2_TABLES

ENTID	Name	Status	Version
1	CUSTOMER	DBXT	00
500	PRODUCT	DBXT	00
9000	ACCOUNT	DBXT	00

DBXDDL.DBX_DDL_ELEMENTS

ENTID	Name	Status	Version
6	CUST_ID	DBXT	00
9	CUST_ID	DBXT	00
4321	CUST_NAME	DBXT	00
9210	ACT_NO	DBXT	00

DBXDB2.DBX_DB2_COLUMN

ENTID	Name	Status	Version	Null	SEQ_ NUM
3	CUSTOMER.CUST_ID	DBXT	00	N	2
5	CUSTOMER.CUST_NAME	DBXT	00	Y	1

DBXREL30.DBX_XREF

ENT_TYPE	ENT_ID	SOURCE_ID	TARGET_ID
126	1		
126	500		
204	6		
204	4321		
102	3	1	6
102	5	1	4321

The Cross Reference Table contains one row for each row in every entity type, relationship type, and association type. The Repository uses SQL against the Cross Reference Table to retrieve mapping information. For example, if you want to show all element names within the table 'Customer' with a status of 'DBXT' and version '00.'

You can query the Repository tables as follows:

```
Select ELEMENT_NAME
FROM    DBXDDL.DBX_DDL_ELEMENT E,
        DBXDB2.DBX_DB2_TABLE T,
        DBXREL30.DBX_XREF X
where T.NAME='CUSTOMER' and
        T.STATUS='DBXT' and
        VERSION = '00' and
        T.ENT_ID=X.SOURCE_ID and
        X.TARGET_ID = E.ENT_ID and
        X.ENT_TYPE = 102
```

To list the ELEMENT names in column sequence, use the following query:

```
Select ELEMENT_NAME, C.SEQ_NUM
From DBXDDL.DBX_DDL_ELEMENT E,
        DBXDB2.DBX_DB2_TABLE T,
        DBXDB2.DBX_DB2_COLUMN C,
        DBXREL30.DBX_XREF X
where T.NAME = 'CUSTOMER' and
        T.STATUS = 'DBXT' and
        T.VERSION = '00' and
        T.ENT_ID = X.SOURCE_ID and
        X.TARGET_ID = E.ENT_ID and
        X.ENT_TYPE = 102 and
        X.ENT_ID = C.ENT_ID
ORDER BY C.SEQ_NUM
```

Dialogs and Data Models

One of the first things you do in Repository session is select a dialog. The dialog you select not only determines which entities you have access to, but also which attributes of those entities you are able to edit and how these attributes appear on your screen.

A dialog is a logical grouping of entity, relationship, and association types within a Repository data model.

In general, all of the components in a dialog are selected so as to support a particular functional area.

- For example, the DB2 dialog is a grouping of entity, relationship, and association types useful for storing information about DB2 objects like tables and columns.
- Similarly, the IMS dialog contains components for defining IMS segments, DBDs (Data Base Descriptions), and PSBs (Program Specification Blocks).

Dialogs Correspond to Metadata Models

Each of the dialogs you are able to access using the Repository roughly corresponds to one of the metadata models. It is always helpful to have a copy of the appropriate model handy when you are manipulating the Repository data, to help you keep track of the components the model uses and how they are related to one another.

For a complete listing of all current Repository models, see the *Metadata Models Reference Guide*.

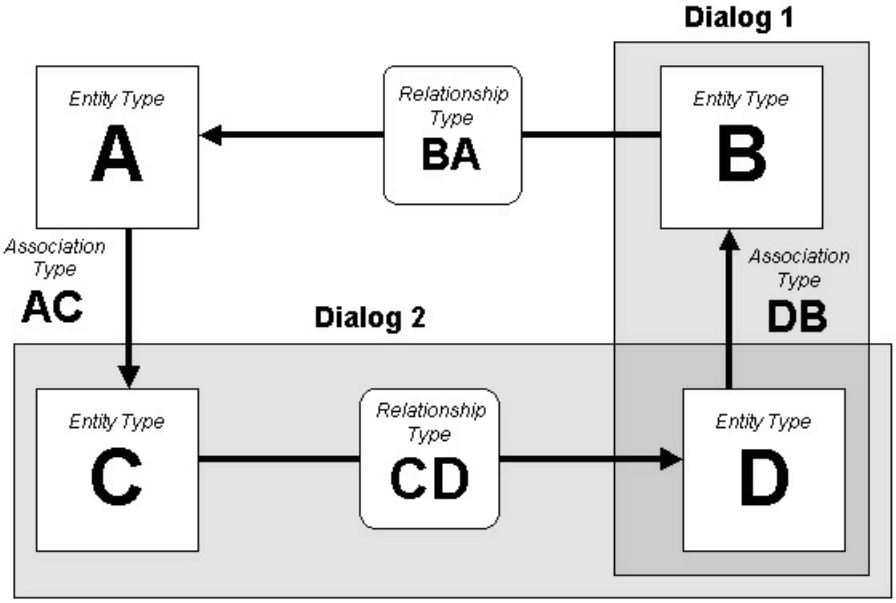
Note: Your Repository Administrator controls your access to dialogs and the entities they contain. The Administrator can also add dialogs and models, so you may find that you have access to dialogs not described in any CA Repository for z/OS guide.

Overlapping Dialogs

The following illustration represents a data model with two dialogs. Dialog 1 has access to entity type B, entity type D, and association type DB. Dialog 2 has access to entity type C, entity type D and relationship type CD.

Although simplified, this illustration shows two key points about dialogs:

- Sometimes dialogs overlap, meaning that they both have access to the entities in the overlapping section (in the illustration, this is entity type D). Many dialogs will have access to the same entity types, though how that entity type appears in each dialog is not necessarily be the same.
- In order for a dialog to have access to a relationship or association type, it must have access to the entity types at both ends of the relationship or association. In the illustration, Dialog 1 can access association type DB, but not relationship types CD or BA. Similarly, Dialog 2 can only access relationship type CD, not association types AC or DB.



Repository Privileges

The following sections explain the types of privileges you need to edit Repository information and how you can change your privileges.

Privilege Types

Repository privileges control your access to repository information by regulating the:

- Dialogs you can access
- Entity types you can process
- The commands you can execute
- The statuses against which you can execute these commands

The following sections summarize the effects each privilege type has on your use of the repository.

Dialog Privileges

Dialog privileges are the highest level of Repository data control; they regulate access to groups of entity types. If you do not have access to a particular dialog, the Repository does not display it in the Dialog List window. In most cases, you will be unaware of dialogs to which you do not have access privileges.

Entity Type Privileges

Entity type privileges control how you can manipulate the entity types within the dialogs to which you have access. Thus, while a dialog privilege can get you to the Edit window for a particular entity type, what you can do with the entity from that point on depends on the entity type privileges you have been assigned.

Command Privileges

Command privileges control the Repository commands you are able to execute in any dialog. In most cases, those commands for which you are not authorized are grayed out when displayed in a pull-down menu.

Status Privileges

Every Repository entity has a Status attribute indicating the entity application life-cycle phase (for example: TEST, DEV for Development, PROD for Production, and so on).

Depending on your status privileges you may not be able to perform certain commands against entities that have a particular status. In some cases, the list of commands available to you is determined by the statuses of the entities currently displayed on the screen.

This becomes more complicated in the Edit window horizontal mode, where multiple entities and their respective statuses can be displayed and manipulated simultaneously. For example, if you select a group of entities for processing with a particular command, but only some of those entities have statuses valid for that command, you will still be able to execute the command. However, the Repository processes only those entities having a status for which you are authorized to perform the command in question. The Repository ignores the remaining entities when you execute the command.

Change Your Privileges

Your Repository Administrator assigns all privileges. If, during your Repository for sessions, you find that you cannot perform your work properly using the actions described in this guide, check with your Repository Administrator to see if you have the necessary privileges.

For more information about privilege definition, see the *Administration Guide.*

Chapter 3: Using the Repository

This chapter describes the Repository user interface and how to use it.

This section contains the following topics:

Start the Repository

The Repository logon procedures vary from one organization to another, but most users log on to TSO and then use an ISPF menu system to access the Repository. See your Repository Administrator for instructions on how to log on to the Repository in your organization.

Once you have logged on to the Repository successfully, the Repository logo displays.

Note: Your TSO ID controls all your Repository privileges once you log on. If you have more than one TSO ID, use the ID authorized for CA Repository for z/OS access.

Select a Dialog

Once you log on to the Repository and display the Main Edit window, you select a dialog. You can select a dialog using the using the menus or command line.

To select a dialog using menus

1. Select VIEW. The VIEW pull-down menu is displayed.
2. Select DIALOG. A list of the available dialogs appears.
3. Type **S** in the Select byte (SEL column) next to the name of the dialog you want to open.

   Press Enter. The Main Edit window with the current dialog name in the center the top border of the window appears.

   Note: Once you become familiar with commands and dialog names, you will find it faster to open a dialog by typing VIEW.DIALOG followed by a space and the name of the dialog on the command line.

To select a dialog from the command line

1. Enter the command, VIEW.DIALOG, followed by a space and the name of the dialog.
2. Press Enter.

Change to Another Dialog

If your privileges allow you to access more than one dialog, you can change dialogs using either of the procedures in the preceding section. If the change is successful, the name of the new dialog appears in the center of the top border of the Main Edit window, in place of the previous dialog name.

If you have an entity selected at the time you change dialogs and you do not have sufficient privileges to that entity in the dialog to which you want to switch, you will not be able to change dialogs. Instead, the Repository the Main Edit window containing the CA Repository for z/OS logo appears. You must then either:

- Change to the new dialog without selecting an entity beforehand
- Before changing dialogs, select an entity type for which you are authorized in the dialog to which you want to change

Navigate Repository Windows

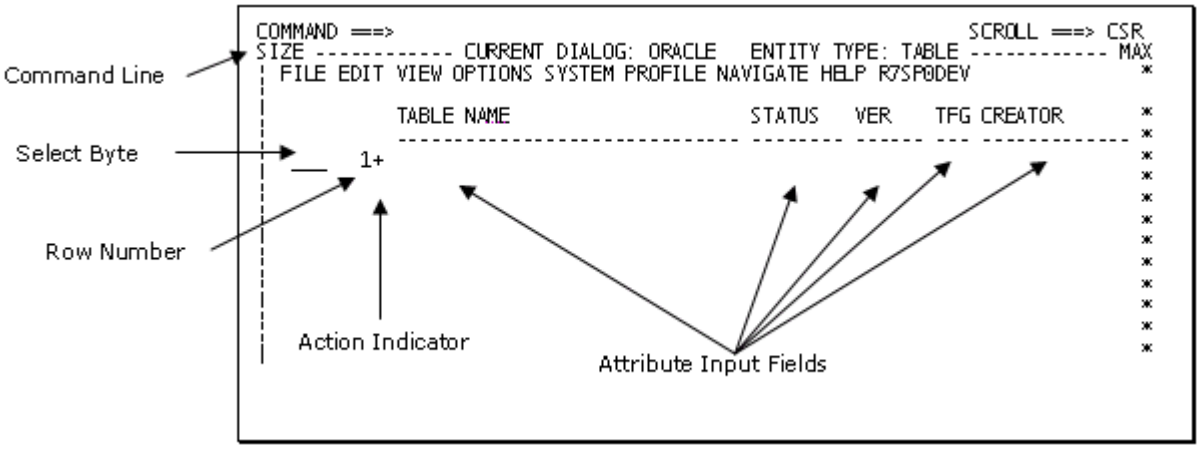
The Repository user interface follows the IBM Common User Access (CUA) guidelines. If you are familiar with other CUA-compliant applications, you should have no trouble using the Repository.

Repository Windows

The Repository different types of windows to present different types of information to you and allows you to add to, change or delete that information. The types of Repository windows include:

- Edit windows
- List windows
- Help windows

While window contents vary considerably, all of the windows have a similar structure, and you can manipulate them in the same ways regardless of type. The following illustration shows the components of a typical window.



The following table describes the components common to most Repository windows.

Component	Description
Command line	A standard TSO command line you can use to perform Repository tasks directly. To learn how to enter Repository commands, see the Line Commands in Horizontal Mode Edit Windows.
Title bar	Contains the windows sizing icon, (Size), the window title (Current Dialog and Entity Type), and the minimum/maximum icon (Max).
Size icon	Use to change the size or position of the window, or to lock one or more of its columns. To learn how to use this icon, see, Sizing Windows, Move Window, and Lock Windows.

Component	Description
Min/Max icon	Use to change the window to its maximum size (80 columns by 23 rows on standard terminals) or back to its normal size. To learn how to use this icon, see Maximize and Restore Windows.
u bar	A line of Repository options you can use to perform various functions, subfunctions, and commands. To learn how to use the menu bar, see Menus and Commands.
Column titles	A line of column labels identifying the kinds of data in the window. Column titles always remain at the top of the window.
Data lines	The information the window presents to you.
Slider box	An indicator showing the relative size and position of the data currently visible in the window versus the data the window could not show. A small slider box at the top of a vertical scroll bar indicates that only a small part of the beginning of the available data is visible in the window.
Scroll indicator	An indicator, SCROLL ==>, showing that there is more data to display than the window can show at present.
Horizontal scroll bar	You can display the extra data by scrolling the window to the left or right, or up or down.
Vertical scroll bar	To learn how to scroll windows, see Scrolling Windows.
Window border	A rectangle enclosing the window. Scroll bars and slider boxes replace parts of the window border if the window does not show all the data at one time. Depending on your monitor, the border may be drawn with solid or dashed lines.

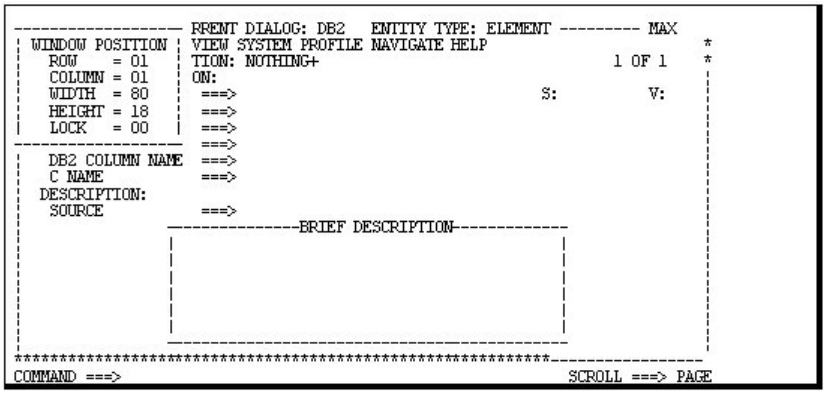
Move Windows

You can move the active window to a new location by specifying row and column coordinates for the upper left corner of the window.

To change the position of a window

1. Use the arrow keys to move the cursor over the Size icon in the upper left corner of the window.

   Press Enter. The Repository displays a Position window (labeled POSITION WINDOW) overlaying the upper left corner of the active window:



2. Specify the position of the window:

   Row

   The line on which you want to position the top border of the window. Row 01 is the line below the command line.

   Column

   The column at which you want to position the left border of the window. Column 01 is the left edge of the screen. The value in Column +, the Width value, cannot be greater than 80.

3. When you are finished, press Enter. The Repository repositions the upper left corner of the screen to match the coordinates you specified.

Size Windows

You can resize the active window by specifying its width in columns and height in rows. To ensure that at least one data line is always visible, the Repository imposes a minimum window height of four lines plus the number of title lines in the window.

To change the size of a window

1. Use the arrow keys to move the cursor over the Size icon in the upper left corner of the window.
2. Press Enter. The Repository displays a Position window (labeled WINDOW POSITION in the previous illustration) overlaying the upper left corner of the active window.
3. Complete the fields in the position window.

   Width

   The number of columns (left to right) you want the window to occupy. Make this value greater than 10 and less than 80.

   Height

   The number of lines (top to bottom) you want the window to occupy. Make this value less than 20, and at least four plus the number of title lines.

   Note: The value in Column + the value in Width cannot exceed 80 characters.
4. When you are finished, press Enter. The Repository resizes the window to the width and height values you specified.

Maximize and Restore Windows

Use the Max icon in the upper right corner of the window to expand a small window so that it occupies the whole screen. When you maximize a window in this way, the Max icon changes to Min. You can then restore the window to its normal size using the Min icon.

To maximize a window

1. Use the arrow keys to move the cursor over the Max icon in the upper right corner of the window.
2. Press Enter. The Repository displays the window at its maximum possible size (80 columns by 20 lines).

To restore a maximized window

1. Use the arrow keys to move the cursor over the Min icon in the upper right corner of the maximized window.
2. Press Enter. The Repository redisplays the window at the size and position you specified for it before it was maximized.

Lock and Unlock Window Columns

You can set the position window Lock field to prevent particular columns from moving when you scroll a window.

To lock columns

1. Use the arrow keys to move the cursor over the Size icon in the upper left corner of the window.
2. Press Enter. The Repository displays a Position window overlaying the upper left corner of the active window.
3. Do *one* of the following:
   - To lock the columns, enter the number of columns to remain stationary during a horizontal scroll. For example: To lock the first five columns of the screen, enter 05 in the Lock field.
   - To unlock the columns, enter 00 in the Lock field.
4. When you are finished, press Enter. The Repository locks or unlocks the columns you specified.

Scroll Windows

The Repository often needs to display more information than will fit on your screen at one time. When this is the case, it displays the information in a scrollable window. The Repository indicates that a window is scrollable by replacing part of the window border with horizontal or vertical scroll bars, which contain slider boxes, composed of asterisks (*). You must scroll the window to view the additional data.

Use the following table to identify the commands used to scroll information and the corresponding function keys.

Command	Key	Description
DOWN	F8	Display the next set of data lines.
UP	F7	Display the previous set of data lines.
RIGHT	F11	Display additional data for the current set of data lines.
LEFT	F10	Display the previous data for the current set of data lines.

Notes:

- If you use scroll commands, the Repository scrolls a *page* of information. A page is the total number of rows or columns visible in the window.
- If you use the function keys, the Repository scrolls the number of rows or columns determined by the current ISPF scroll-amount setting (such as PAGE, HALF, CSR). For example, if you press the key corresponding to the DOWN command when the scroll amount is CSR, the Repository scrolls the window until the row your cursor reaches the top border of the window.

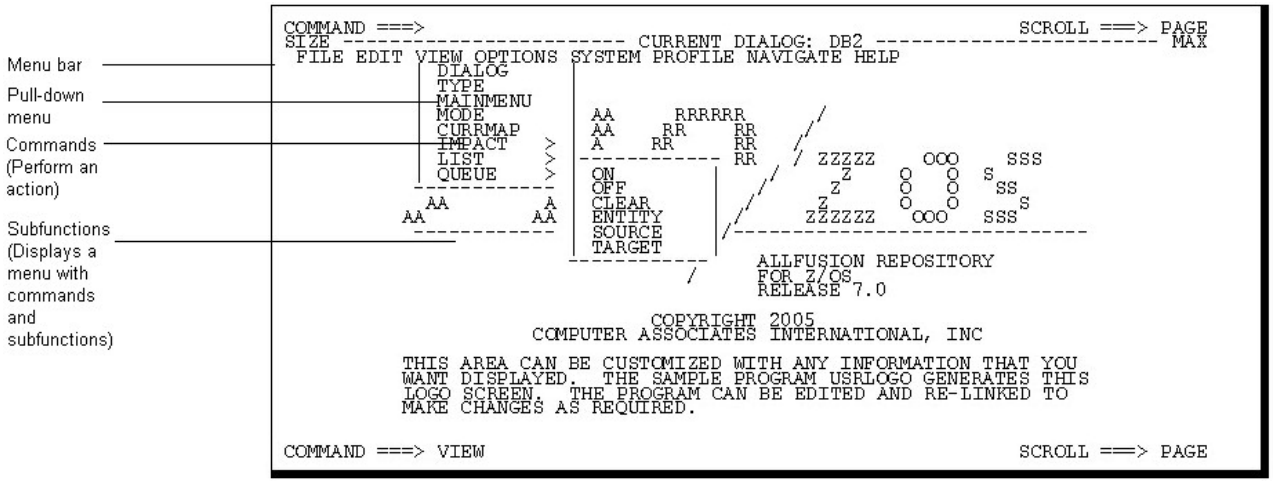
As you scroll the window over the available data, the slider boxes move proportionately along their respective scroll bars, indicating the position of the information you are viewing relative to the information not currently displayed.

Menus and Commands

This section describes the Repository's menus and commands.

Menus

The Repository windows use pull-down menu options to give you access to all of the most important actions you can perform using each window. The following illustration shows the Main Edit window with a typical menu structure.



The following table describes each component of a menu:

Component	Description
Menu bar	Displays the list of functions that you can select.

Component	Description
Pull-down menu	When selected, displays a pull-down menu or performs an action.
Command	When selected, always performs an action.
Subfunction	When selected, always displays a second (*cascading*) pull-down menu containing other commands or subfunctions.

The number and kinds of options (functions, commands, and subfunctions) you see on a menu varies with the currently active window, but you can select all of them in the same ways, using the following procedures.

To select a menu function using the cursor

1. Press the arrow keys or TAB to move the cursor to the function you want.
2. Press Enter.

To select a menu function using the function name

1. Type the first letter of the function name (for example, E for EDIT) on the command line.
2. Press Enter.

To select a menu command or subfunction

1. Press the arrow keys or TAB to move the cursor to the command or subfunction you want.
2. Press Enter.

To back up to a previous menu

Press F3 or enter the END command to move back to the previous pull-down menu, or to the menu bar.

The Command Line

To perform tasks more efficiently, use the command line (seen as COMMAND ===>). The Repository provides two kinds of commands you can enter on the command line:

- Standard commands you can enter at any time.
- Window-specific command strings you can enter only when the appropriate window is active.

The next two sections explain how to enter both kinds of commands. The procedures elsewhere in this guide give you the relevant command text for each task you want to perform.

You can make using the command line more efficient by defining your own list of fast commands, which are abbreviations of window-specific command strings like EDIT.COPY.TEXT. For details on fast commands, see "Defining User Profiles."

The next two sections do **not** explain how to enter and use the special line commands available only in List windows and Edit windows in horizontal mode.

Standard Commands

Standard commands are available in nearly all Repository windows and require no special authorization to use. While they usually have no equivalents on the menu bar, you can always execute them by entering them on the command line and pressing Enter.

Window-Specific Command Strings

When you enter a command string-the name of a menu function, subfunction, and command, separated by periods-on the command line, The Repository responds just as if you had selected these options from the menu bar and its pull-down menus. For example, instead of selecting EDIT from the menu bar, then subfunction COPY from the EDIT pull-down menu, and then command text from the cascading COPY pull-down menu, you can type **EDIT.COPY.TEXT** at the command line and then press Enter.

Most window-specific commands in the Repository are part of a command string. Because not all menus contain subfunctions, a command string may consist of only two words, but you must always enter them **as a complete string**.

Note: You cannot execute a command, which is part of a command string unless you specify all parts of the string.

The following illustration shows command strings specific to the Repository Edit windows, which are typical of other window-specific command strings.

In some cases, window-specific commands are not part of a command string, or contain only one word. You can enter them just as you do standard commands, by typing them on the command line and pressing Enter.

Note: Using window-specific commands requires that your Repository Administrator authorize you to access the window from which users execute those commands. If you do not have access to every Repository window, there may be some window-specific commands you cannot use.

Edit Windows

This section describes using Edit windows, switching between horizontal and vertical Edit window modes, Edit window field types, data entry restrictions, using horizontal mode Edit windows, using line commands in horizontal mode, tagging rows, copying rows, moving rows, marking rows for deletion, inserting blank rows, repeating rows, excluding rows, and using commands in vertical mode.

Edit Windows

Edit windows are the most frequently used Repository windows. You use them to perform all entity maintenance and to access the Repository add-on products. Each window has two modes:

- Horizontal (List) mode)
- Vertical (Detail) mode)

The mode you use depends on what kind of work you are doing and how many entity attributes you want to see on the screen at one time.

Mode	Description
Horizontal	Displays multiple entities. When you do not need to see all of their attributes on the screen at one time.
Vertical	Displays one entity at a time. When you want to see all or most of the entity attributes on the screen at one time.

Switch Between Horizontal and Vertical Edit Window Modes

To toggle or switch between horizontal and vertical Edit windows, enter VIEW.MODE on the command line or from the VIEW menu, choose MODE.

Edit Window Field Types and Data Entry Restrictions

When you use Edit windows to add or change entities, relationships, or associations, it is important to remember that you cannot edit all of their fields. Edit windows offer four types of fields, each with its own restrictions on data entry. The types of fields are:

Type of Field	Description
System-generated	The Repository completes these fields and you cannot place the cursor in or make changes in their contents. The number of system-generated fields depends on how your organization has customized the Repository, but examples might include Date Created or Date Modified.
Required	You must complete these fields in order to add or change an entity, relationship, or association. The number of required fields varies depending on how your organization has customized the Repository, but they could include: ■ For all entities: Name, Version, and Status fields ■ For all relationships and associations: The source entity and the target entity Name, Version, and Status fields
Optional	Use these fields to store additional information about an entity. You can add or change the entity even if you leave these fields blank.
Concatenated	These are completed using values in other fields. Although you cannot make direct entries in them, you can affect their values if they are concatenated using values in Required or Optional fields.

To ensure you get the data results you want, it is also important to observe the following rules:

- Never enter a question mark (?) as part of the data in any field unless you want to display a Help window containing Field or Valid Values help messages
- Never enter an equal sign (=) as the first character in an input field unless you want ISPF to issue a RETURN command

Horizontal Mode Edit Windows

The following illustration shows an Edit window in horizontal mode.

The following table describes each component of the Repository Edit window in horizontal mode.

Component	Description
Title bar	Lists the current dialog and entity, relationship, or association type.
Command line	Use to enter commands to process the entities represented by tagged rows in the window.
Menu bar	Lists the functions available in the window. You select menu commands to process entities represented by tagged rows in the window.
Select byte	■ Use to: ■ Enter commands to copy, move, delete, insert, repeat, or exclude rows in the Edit window ■ Tag rows for processing
Row number	The number of the row within the last screen. A highlighted row number without an Action indicator means that the row has changed and an update is pending.
Action indicator	Indicates that a row is marked for insertion or deletion. ■ A plus symbol (+) indicates a row marked for insertion ■ A minus symbol (-) indicates a row marked for deletion
Attribute input fields	Use to enter the values for each entity attribute. The title line above each field gives the attribute literal name.

Line Commands in Horizontal Mode Edit Windows

The horizontal mode Edit window provides a number of special line commands you can enter in the Select byte to copy, move, delete, insert, repeat, or exclude rows, or tag rows for processing.

Note: You cannot use line commands in an Edit window in vertical mode.

Rows in Edit Windows

Row selection, called *tagging*, is essential to doing useful work in any horizontal mode Edit window. For example, you cannot use the menu or command-line commands available in a horizontal mode Edit window to process any entity, relationship, or association until you have tagged the row representing that entity.

You can tag any number of rows, using either the S tag or the TAG standard command; you can also remove tags from all rows where you have inserted them using UNTAG.

To tag a row-Move the cursor to the Select byte next to the row you want to tag and type **S**.

To tag all rows-On the command line, type **TAG** and press Enter. the Repository enters an S next to every row, including those beyond the window border.

To untag a single row-Type a **space** over the S.

To untag all rows-On the command line, type **UNTAG** and press Enter. The Repository removes any S tags in the Select bytes next to all rows.

Copy Rows in Edit Windows

You can copy rows when you want to add a large number of similar entities to the repository, but do not want to enter repetitive information for each entity. Use the C, C*n,* and CC commands to copy rows in a horizontal mode Edit window.

Note: To create multiple copies of a single row, use the R*n* command.

To copy a row, series of rows, or block of rows

1. Specify the rows to copy:
   - To copy a single row: Move the cursor to the Select byte next to the row and type **C**.
   - To copy a series: Move the cursor to the Select byte next to the first row in the series and type **Cn**, where *n* is the number of succeeding rows to copy (for example, type **C3** in row 1 to copy rows 1 through 4).
   - To copy a block: Move the cursor to the Select byte next to the first row in the block and type *CC*. Then move the cursor to the last row in the block and type **CC** again.
2. If needed, specify where you want the copied rows located:
   - To copy the rows after a specific row: Move the cursor to the Select byte next to that row and type **A**
   - To copy the rows before a specific row: Move the cursor to the Select byte next to that row and type **B**
3. Press Enter. The Repository copies the rows you specified.

Move Rows in Edit Windows

Use the M, M*n*, and MM commands to create a block of rows for a special purpose.

Note: Rearranging rows in this way does not change how the repository stores them; any moves you execute in a horizontal mode Edit window are lost once you exit the window.

To move a row, series of rows, or block of rows

1. Specify the rows to be moved:
2. To move a single row: Move the cursor to the Select byte next to the row and type **M**.
   - To move a series: Move the cursor to the Select byte next to the first row of the series and type *Mn*, where *n* is the number of succeeding rows to move (for example, type M3 in row 1 to move rows 1 through 4).
   - To move a block: Move the cursor to the Select byte next to the first row of the block and type **MM**. Then move the cursor to the last row in the block and type **MM** again.
3. Specify where you want the rows moved:
   - To move the rows after a specific row: Move the cursor to the Select byte next to that row and type **A**
   - To move the rows before a specific row: Move the cursor to the Select byte next to that row and type **B**
4. Press Enter. The Repository moves the rows you specified.

Mark Rows for Deletion in Edit Windows

You can mark any row or group of rows for deletion in a horizontal mode Edit window using the D, D*n*, or DD commands.

Note: Deleting rows using these Select byte commands only marks them for deletion. To actually remove from the repository the entities, relationships or associations the rows represent, you must enter the EDIT.SYNC command on the command line or from the EDIT menu choose SYNC. For more information, see "Working with Entities."

To mark a row, series of rows, or block of rows for deletion

1. Specify the rows to be marked for deletion:
2. To mark a single row: Move the cursor to the Select byte next to the row and type **D**.
   - To mark a series: Move the cursor to the Select byte next to the first row of the series and type **Dn**, where *n* is the number of succeeding rows to mark for deletion (for example, type D3 in row 1 to delete rows 1 through 4).
   - To mark a block: Move the cursor to the Select byte next to the first row of the block and type **DD**. Then move the cursor to the last row of the block and type **DD** again.
3. Press Enter. The Repository marks the rows for deletion.

Insert Blank Rows in Edit Windows

Use the I and In commands to insert blank rows when all the rows in a horizontal mode Edit window are occupied and you want to add or retrieve a repository entity, relationship, or association type.

To insert blank rows

1. Move the cursor to the Select byte above the location where you want to insert a row.
2. Specify the number of blank rows to insert after this row:
   - To insert one blank row: Type **I**
   - To insert two or more blank rows: Type **In**, where *n* is the number of rows you want inserted
3. Press Enter. The Repository inserts the blank rows after the row where you entered the I or In command.

Repeat Rows in Edit Windows

You can use R, RR, and R*n* to copy rows and paste them immediately after the row you are copying.

To repeat a row or block of rows

1. Specify the rows to repeat:
   - To repeat a row once: Move the cursor to the Select byte next to the row and type **R**.
   - To repeat a row multiple times: Move the cursor to the Select byte next to the row and type **Rn**, where *n* is the number of times you want to repeat the row (for example, type **R25** in row 1 to repeat that row 25 times).
   - To repeat a block: Move the cursor to the Select byte next to the first row of the block and type **RR**. Then move the cursor to the last row of the block and type **RR** again.
2. Press Enter. The Repository repeats the rows you specified.

Exclude Rows in Edit Windows

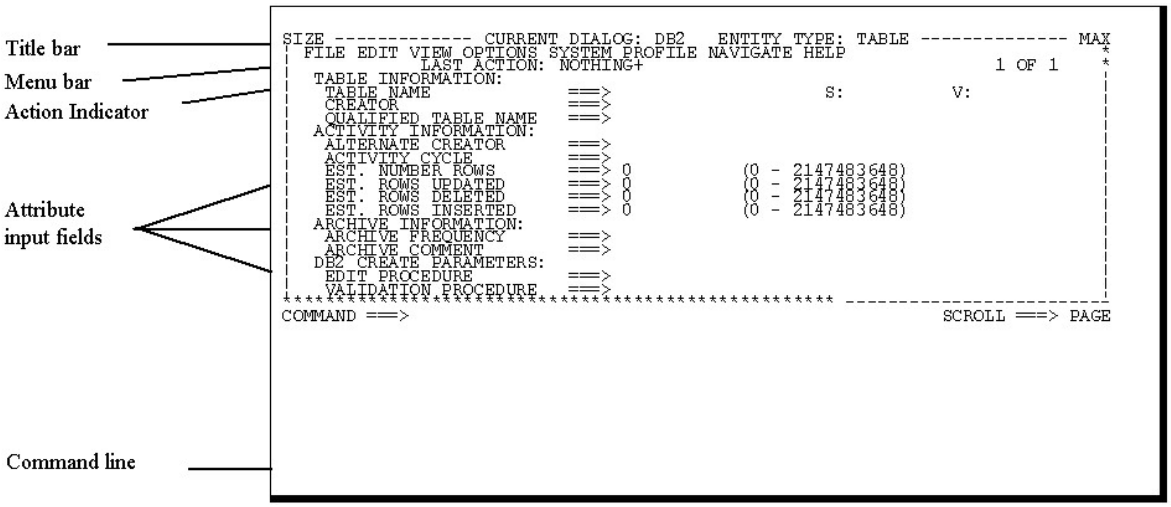
You can exclude any row or group of rows from processing in a horizontal mode Edit window using the X, X*n,* or XX commands. Use this command when you perform a series of edits on several entities, relationships, or associations and want to exclude some of the edits from processing with the EDIT.SYNC command. For details, see "Working with Entities."

To exclude a row, series of rows, or block of rows

1. Specify the rows to exclude:
   - To exclude a single row: Move the cursor to the Select byte next to the row and type **X**.
   - To exclude a series: Move the cursor to the Select byte next to the first row in the series and type **Xn**, where *n* is the number of succeeding rows to exclude (for example, type X3 in row 1 to exclude rows 1 through 4).
   - To exclude a block: Move the cursor to the Select byte next to the first row in the block and type **XX**. Then move the cursor to the last row in the block and type **XX** again.
2. Press Enter. The Repository excludes the rows you specified.

Vertical Mode Edit Windows

The following illustration shows an Edit window in vertical mode.



The vertical mode Edit window shows the attributes of a single entity.

The following table describes each component of the Edit window in vertical mode.

Component	Description
Title bar	Contains the Size icon, the title (current dialog and entity type) and the Min/Max icon.
Command line	Use to directly enter commands to process the currently displayed entity.

Component	Description
Menu bar	Lists the functions available in this window and gives you access to the commands to process the currently displayed entity.
Attribute input fields	Use to enter the values you want to add or change for each entity attribute. The titles next to each field give the attribute literal name.
Last Action field	Shows the last command executed against the currently displayed entity. If highlighted without an Action indicator (+/-), it indicates that the action is pending.
Action indicator	Shows that an addition or deletion has been performed against the currently displayed entity. A plus symbol (+) indicates an addition; a minus symbol (-) indicates a deletion.

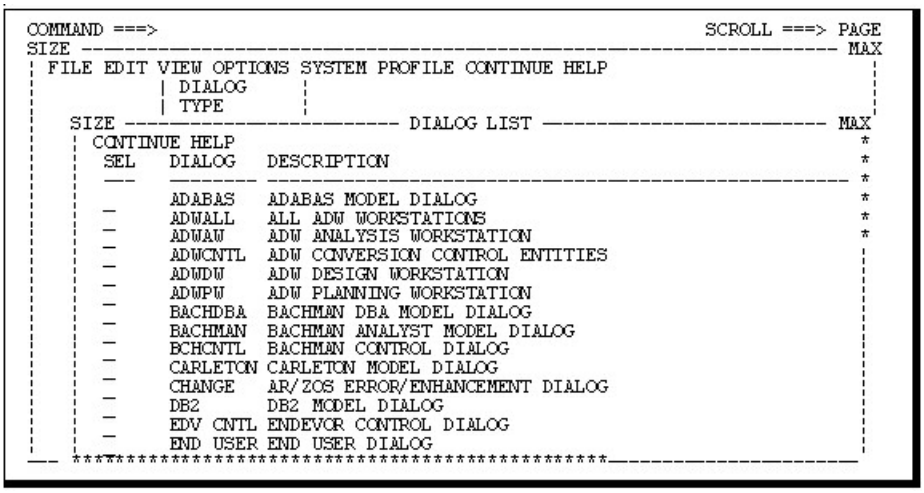
Commands in Vertical Mode

The line commands used in horizontal mode Edit windows are not available in vertical mode. However, when you retrieve multiple entities to an Edit window in vertical mode using a List window, you can use the following standard commands:

- NEXT (or NE) to view the entity succeeding the one you are currently viewing
- PREV (or PR) to view the entity preceding the one you are currently viewing

List Windows

The Repository provides a number of List windows for you to select items in a horizontal or vertical mode Edit window. The following is a typical List window.



You can tag any item on the list using the S tag. If the window permits you to tag multiple items, you can tag all the items in the list using the TAG standard command. You can also use UNTAG to remove tags from all list items.

To tag a list item-Move the cursor to the Select byte next to the item you want to tag and type **S**.

To tag all list items-On the command line, type **TAG** and press Enter.
The Repository enters an S next to every item, including those beyond the window border.

To untag all list items-On the command line, type **UNTAG** and press Enter.
The Repository removes any S tags in the Select bytes next to all list items, including those beyond the window border.

Limit List Windows

You use List windows to display lists of entities, relationships, and associations and to determine what items are in the repository.

Note that the effectiveness of a List window is reduced when it displays so many items that you must scroll through numerous pages.

You can easily limit the content of List windows by the entries you make in the corresponding Edit window Name, Status, and Version fields before you execute the command that generates the list. The procedure below summarizes the methods you can use in each field to limit the list.

Note: You can combine the methods presented below to greatly limit a list. For example, to list all elements starting with the letters COST in the PROD status, you would enter CUST in the Name field and Prod in the Status field.

You can also limit the contents of a List window using queries and search criteria. For details, see "Using Queries and Search Criteria."

Limit List Windows by Name, Status, and Version

To find an item whose name begins with certain characters-Type the characters. When you execute the list command, the Repository displays a list of items whose names begin with the characters you entered. For example, type **A** in the Name field and then execute VIEW.LIST.ENTITY to find an entity whose name begins with the letter A.

To find an item whose name contains certain characters-Type **%** (wildcard character) followed by the characters, and press Enter. When you execute the list command, the Repository displays a list of items whose names contain with the characters you entered. For example, type **%COB** in the Name field and then execute VIEW.LIST.ENTITY to find an entity whose name contains the letters COB.

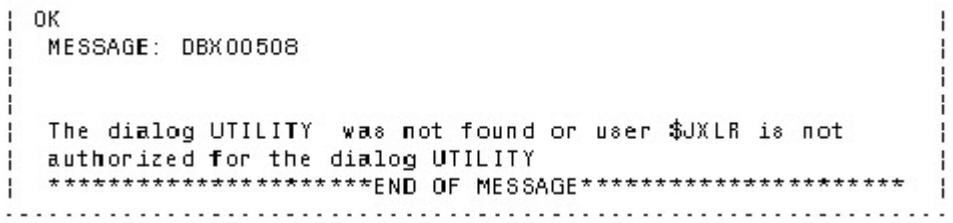
To find an item with a certain status or version-Type the complete status or version in their respective fields and press Enter.

Message Windows

The Repository displays three types of messages in Message windows:

Type of Message	Description
Progress	Displays the status of a process that is proceeding normally.
Information	Lets you know whether a process is proceeding normally and informs you when it has finished normally.
Warning	Informs you that an undesirable result has occurred, or that an undesirable result may occur if processing continues.

The following is a typical Message window displaying an Information message.



When the message requires a response, the Message window displays text icons (usually, the words YES, NO, OK, or CANCEL) in its upper left corner. You can respond either by entering the icon text on the command line, or selecting the icon itself.

- Use one of the following procedures to respond to information and warning messages:
- Press the arrow keys or tab to move the cursor to the text icon you want to select and then press Enter
- At the command line, type the text of the icon you want to select (for example, type OK) and then press Enter

Help Windows

You can display Help windows containing one of the following types of Repository-specific help messages:

Type of message	Description
Entity type	Describes *the currently selected entity type.*
Command	Explains menu functions, subfunctions, and commands.
Field	Gives the purpose and characteristics of data-entry fields.
Valid values	Lists valid input values for fields (available only when a code table has been created for the field).

Every Help window displays text icons (always OK, and sometimes NEXT, PREV) in its upper left corner.

Display Help

Use the following procedures to display a Help window, navigate through a series of help messages, and close the Help window.

Use the table below to display help.

For Help with:	Enter or Select
Entity types	The HELP option from the menu bar to get a description of the current entity type.
Commands	A question mark (?) over the function, sub-function, or command on the menu bar or pull-down and press Enter.
Fields	A question mark (?) in the input field and select HELP from the menu bar.
Valid values	A question mark (?) in the input field and press Enter.
ISPF	HELP on the command line and press Enter.

To see the:

- Next Help message in a series, select NEXT or type **NEXT** on the command line and press Enter.
- Previous Help message in a series, select PREV or type **PREV** on the command line and press Enter.

To close a Help window-Select OK or type **OK** on the command line and press Enter.

ISPF Messages and Tutorials

The Repository sometimes displays ISPF messages and tutorial panels. Once an ISPF message has been displayed, type the command **HELP** on the command line and press Enter to display a long message or invoke an ISPF tutorial on the subject.

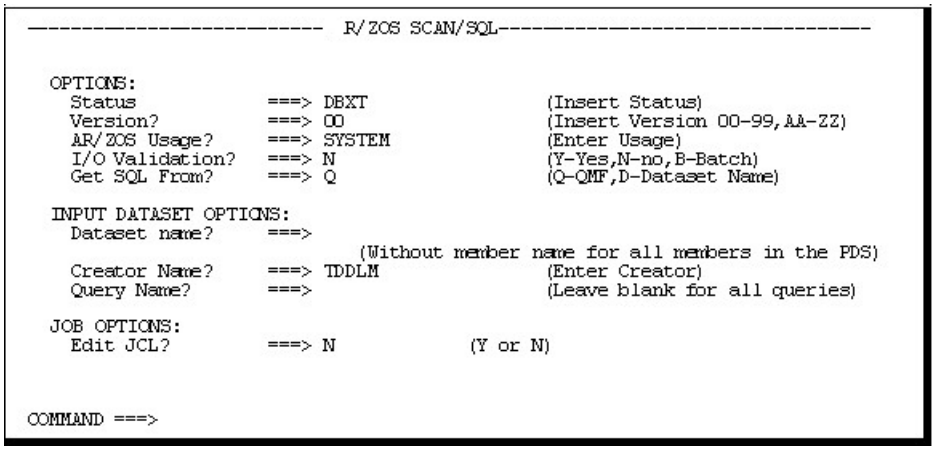
- If an ISPF message is defined to the Repository, the SETMSG command executed so that the next panel display shows the message.
- If more than one message is required before a panel is displayed, the Repository only the most recently set message.

The Repository Help messages can invoke ISPF tutorials if a panel ID for the tutorial is specified. In such instances, dialog control will be passed from Repository to ISPF.

Panels

A few Repository facilities and tools have command options and fields on panels instead of windows. For example, in the following illustration, the Scan/SQL Facility panel has no menu and all its options are contained in input fields.

You can enter data on panels just as you do in vertical mode Edit windows; both have the same kinds of fields and data entry restrictions. You can also use panels just as you use Edit windows, except that you cannot resize, move, lock columns, or display help on a panel.



Chapter 4: Working with Entities

This chapter describes how to add, change, and delete Repository entities using basic entity maintenance commands such as INSERT, SELECT, UPDATE, DELETE, and SYNC.

This section contains the following topics:

Before You Begin

Before you can add, change, or delete entities in the Repository, you need to be familiar with the following:

- Repository entities. For more information, see "Understanding the Architecture."
- How to logon to the Repository**.** For more information, see "Using the Repository."
- How to select a dialog. For more information, see "Using the Repository."
- How to select menu options. For more information, see œUsing the Repository.
- How to enter commands on the command line. For more information, see "Using the Repository."
- How to tag rows in a horizontal mode Edit window. For more information, see "Using the Repository."
- How to use the Repository line commands to copy, rearrange, or delete rows in a horizontal mode Edit window. For more information, see "Using the Repository."

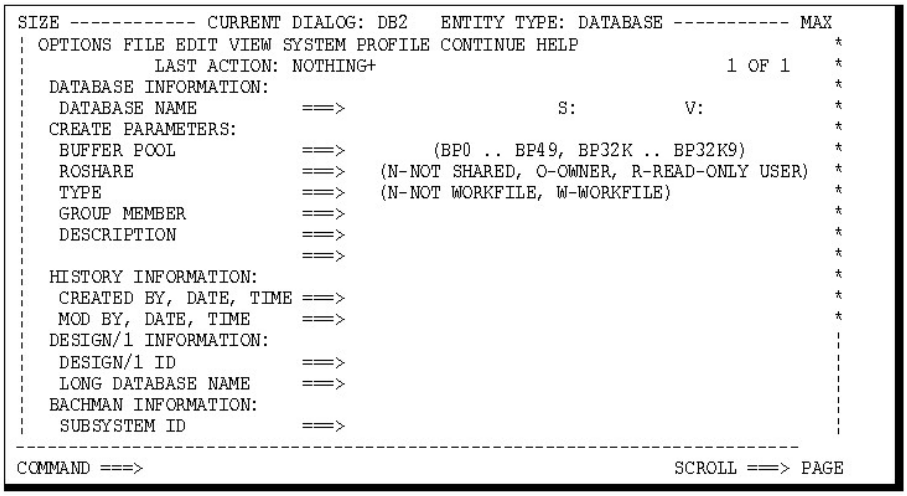
Note: In this chapter and the chapters that follow, the phrases Use COMMAND1.COMMAND2 and Select COMMAND1.COMMAND2 are equivalent to Enter **COMMAND1.COMMAND2** on the command line or from the COMMAND1 menu, choose COMMAND2.

View an Entity Type

To select the entity type for the entities you want to add, change, or delete

1. Log on to the Repository.
2. Use VIEW.DIALOG on the command line. A list of available dialogs appears.
3. Type **S** in the Select byte next to the dialog and press Enter.
4. Use **VIEW.TYPE** on the command line. A list of the available entity (ENT), relationship (REL), and association (ASN) types appears in a List window.

   Type **S** in the Select byte next to the entity type and press Enter. The Edit window displays the entity type you select.



Add Entities

The following sections explain how to add new entities to the repository using several different methods.

You can prepare entities for addition to repository one at a time and in groups, using either of the Edit window modes and (when in horizontal mode) a variety of line commands. Once you prepare the entities, you can add them to the repository by using **EDIT.INSERT** on the command line.

Notes:

- You can add entities at the same time you are changing or deleting others if you add them using EDIT.SYNC instead of EDIT.INSERT.
- When you add an entity to the repository, remember that the Name, Status, and Version attributes of the entity must be unique. If you add an entity with a name, status, and version that all match those of an existing entity, the insert fails and the Repository displays an error message.

Add Entities in Horizontal Mode

To create entities in the Repository using the Edit window horizontal mode

1. Select the entity type using the procedure in Viewing an Entity Type.

   If a vertical view displays, use VIEW.MODE on the command line.
   The Repository displays a blank row in the horizontal mode Edit window.

   Note: If you do not have a blank row, use the I line command to insert a new blank row.

2. Type information for the new entity in the columns, you must at least enter information in:

   - Name
   - Status
   - Version

   The Repository highlights the Row indicator (1) and displays an Action indicator (+) next to the row to show that the information in this row has not been inserted in the repository.

3. Repeat Steps 1 and 2 for each entity occurrence you want to add.
4. Type **S** on the Select byte next to each row you want to update in the repository and press Enter.
5. Use **EDIT.INSERT** on the command line. This action inserts the new entity in the repository, the Row indicator clears the selection, and the Action indicator (+) disappears.

Add Entities in Vertical Mode

To insert an entity into the repository from an Edit window in the vertical mode

1. Select the entity type for which you want to add a new entity instance using the proceed.
2. Switch to vertical mode.
3. Type information for the new entity in the columns, enter information in:
   - Name
   - Status
   - Version
4. Use EDIT.INSERT on the command line. This action inserts the new entity in the repository, the Row indicator clears the selection, and the action indicator (+) disappears.
5. Repeat Steps 1 through 4 for each entity occurrence you want to add. Use PREV and NEXT to scroll through your entities.

Copy an Entity

To add more than one entity at a time by copying a *template* entity, use the R*n* line command to *repeat* (or copy) one entity a fixed number of times.

Note: You can also use the C*n* line command in place of R*n*, with basically the same results.

Because you are working with copies of a template, this procedure is most useful when you want to add many entities with similar general attributes. For example, entities that all have a different name but the same status and version. Remember to edit the name, status, and version of all entities you copy, so that they are unique both among the entities you are adding and those already existing in the repository.

To copy an entity, follow these steps

1. Switch to horizontal mode.
2. Select the entity you want to copy or create a new entity.
3. Type **Rn** (*n* is the number of entities you want to add) on the Select byte of the row you want to copy and press Enter. Additional rows identical to the one you selected are displayed.
4. For each of the repeated rows, change the name, status, or version, as needed to ensure that each entity is unique.

5. Type an **S** in the Select byte next to each row you want to update in the repository and press Enter.
6. Use **EDIT.INSERT** on the command line. This action inserts a new entity in the repository, the Row indicator clears the selection, and the Action indicator (+) disappears.

Retrieve Entities

The following sections explain how to retrieve existing entities from the Repository. You must retrieve existing entities before you can change or delete them.

If you know the values for the general attributes (that is, its name, status, and version) of an entity, you can easily retrieve the entity by entering a combination of those attributes in Edit window fields, and then using the EDIT.SELECT command.

Even if you do not know or are not sure of the general attributes of an entity, you can still retrieve it in any Edit window using the **VIEW.LIST** command, then tagging the entity.

Retrieve a Single Entity

If the entity you want to retrieve has a unique name, status, or version, you can easily open an Edit window for that single entity. Use the following procedure to retrieve single entity in a horizontal mode Edit window.

To retrieve a single entity, follow these steps

1. Select the type of entity you want to retrieve, using the procedure in View an Entity Type. The Repository displays a blank row in a horizontal mode Edit window.
2. Use VIEW.MODE to switch to vertical mode.
3. Complete at least one of the following entity attribute fields:
   - Name
   - Status
   - Version
4. Use **EDIT.SELECT** on the command line. The Repository finds all the entities that match the information you specified in the previous step. Use PREV and NEXT to view other entities.

Retrieve Multiple Entities

Use the following procedure to retrieve single or multiple entities in a horizontal or vertical mode Edit window. If you choose to retrieve multiple entities in vertical mode, use the NEXT and PREV commands to move from one entity to another.

To retrieve entities in an edit window, follow these steps

1. Select the type of entity you want to retrieve, using the procedure in View an Entity Type. The Repository displays a blank row in the Edit window.
2. Use **VIEW.LIST.ENTITY** on the command line. The Repository displays a List window showing all entities for the current entity type.
3. Type **S** on the Select byte of each entity you want to retrieve.
4. Press Enter. The Edit window displays the entities you selected.

Change Entities

This procedure explains how to change the definition of existing entities you have retrieved from the repository.

You can change the attributes of any entity you retrieved into any Edit window by typing new information over the existing information and then entering **EDIT.UPDATE** on the command line.

The EDIT.UPDATE command changes entities without affecting the relationships or associations that tie these entities to others, even if part of the update includes altering the name, status, or version of the entity. The Repository can do this because it uses a special internal identifier (other than the combined name, status, and version) to select entities.

Note: You can change entities at the same time you are adding or deleting others if you update them using EDIT.SYNC instead of EDIT.UPDATE. For more information, see Perform Mixed Edits Using Edit.Sync.

Change Entities in Vertical Mode

To change existing entities using a vertical mode Edit window

1. Retrieve the entity you want to change in an Edit window. For details, see View an Entity Type earlier.
2. Change the contents of any field by typing over the existing information.
3. Use **EDIT.UPDATE** on the command line. The Repository updates the entity in the repository and displays UPDATE in the Last Action field to indicate that it successfully updated the entity.

Delete Entities

This section describes how to remove existing entities from the Repository.

You can delete any entity you retrieve into any Edit window by first tagging it and then using EDIT.DELETE on the command line.

Notes:

- You can delete entities at the same time you are adding or changing others so long as you first mark them for deletion using the D line command and then execute the deletions using EDIT.SYNC instead of EDIT.DELETE. For details, see View an Entity Type.
- When you delete entities from the Repository, the Repository also deletes any associations and relationships in which the deleted entities are either sources or targets.

Delete Entities in Horizontal Mode

To delete existing entities using a horizontal mode Edit window

1. Retrieve the entity you want to change in an Edit window. For details, see View an Entity Type.
2. Type **S** in the Select byte of the entity you want to delete.
3. Use **EDIT.DELETE** on the command line. The Repository deletes the tagged entities from the repository and removes the tagged rows from the window.

Delete Entities in Vertical Mode

To delete existing entities using a vertical mode Edit window

1. Retrieve the entity you want to delete in an Edit window. For details, see View an Entity Type.
2. Use **EDIT.DELETE** on the command line. Depending on the entity being deleted, the Repository displays either:
   - A Cross Reference List window showing entities related to the deleted entity by means of existing relationships or associations. Go to step 3.
   - A deletion confirmation window. Go to step 4.
3. If the Repository displays a Cross Reference List window, press F3 or enter END on the command line to continue with deletion. The Repository displays a deletion confirmation window.
4. When the confirmation window is displayed, select YES to confirm that you want to delete the entity. Depending on how many entities you retrieved in Step 1 previously, the Repository displays either a blank vertical mode Edit window or the next entity.

Perform Mixed Edits Using EDIT.SYNC

You can perform a mixed series of entity additions, changes, and deletions and, when you are finished, update them all by using EDIT.SYNC on the command line.

The EDIT.SYNC command makes it unnecessary for you to do all your additions, changes, and deletions in separate groups and execute individual EDIT.DELETE, EDIT.INSERT, and EDIT.UPDATE commands for each group.

Although you can execute EDIT.SYNC from vertical mode, you should use EDIT.SYNC in horizontal mode Edit windows. Since you can only process one entity at a time in vertical mode, EDIT.SYNC is no more useful in vertical mode than the other entity maintenance commands.

To perform mixed edits using EDIT.SYNC

1. Select the entity type for which you want to add, change, or delete entities, using the View an Entity Type procedure. The Repository displays a blank row in the Edit window.
2. Use one or more of the following procedures to add, edit, or delete entities without updating the changes in the repository:
   - Adding Entities in Horizontal Mode.
   - Changing Entities in Vertical Mode.
   - Marking Rows for Deletion in Edit Windows in œUsing the Repository.

Use **EDIT.SYNC** on the command line. The Repository processes your edits as specified.

Note: If EDIT.SYNC is unable to execute any add, change, or delete action successfully (for example, if an add fails because of an existing, duplicate entity), the Repository rolls all actions back to the point at which you executed the command. You can re-execute EDIT.SYNC again-once you have determined and resolved the cause of the error.

3. If you decide to cancel the edits before you update them into the repository, type **X** in the Select byte of the entity for which you want to cancel an update in the repository.

For more information, see "Using the Repository."

Chapter 5: Working with Relationships and Associations

This chapter describes how to add, change, and delete Repository relationships and associations.

Note: The phrases: Select COMMAND1.COMMAND2 and Use COMMAND1.COMMAND2 are equivalent to Enter COMMAND1.COMMAND2 on the command line or from the COMMAND1 menu, choose COMMAND2.

This section contains the following topics:

View a Relationship or Association Type

To view a relationship or association you want to add, change, or delete

1. Log on to the Repository.
2. Select VIEW.DIALOG. A list of available dialogs appears.
3. Type **S** in the Select byte next to the dialog you want to use and press Enter.
4. Type **S** next to the dialog you want to use and press Enter.
5. Select VIEW.TYPE. A list of available entity (ENT), relationship (REL), and association (ASN) types for the current dialog appear in a List window.
6. Type **S** next to the relationship or association type you want and press Enter. An Edit window for the selected relationship or association type, with the type name in the window title appears.

Add Relationships and Associations

The following sections explain how to add new relationships and associations to the Repository.

You can prepare relationships and associations for addition to the Repository one-at-a-time and in groups, using either the vertical mode Edit window or horizontal mode Edit window.

Notes:

- In horizontal mode, there are a variety of line commands. Once you prepare the entities, you can add them by using EDIT.INSERT on the command line.
- You can add relationships and associations while you are changing or deleting other relationships and associations if you use EDIT.SYNC instead of EDIT.INSERT.
- When adding a relationship to the Repository, remember that the name, status, and version attributes of the entity you are inserting must be unique. If you try to add a relationship with a name, status, and version that all match those of an existing relationship, the insert fails and a Repository error message appears.

Add Relationships and Associations in Vertical or Horizontal Mode

To add a relationship or association

1. Select the relationship or association type for which you want to add a new relationship or association, using the procedure in View Relationships and Associations in the Relationship or Association Type. The Repository displays a blank row in the horizontal mode Edit window.

   Note: If you do not have a blank row, use the I line command to insert a new blank row.

2. Select VIEW.LIST.TARGET. A list of available target entities appears.
3. Type an **S** in the Select byte of the target you want to use and press Enter. The name, status, and version number for that instance appears in the Edit window for your new relationship or association.
4. Select VIEW.LIST.SOURCE. A list of available target entities appears.
5. Type an B in the Select byte of the source you want to use and press Enter. The name, status, and version number for that instance appears in the Edit window for your new relationship or association.
6. You can edit the contents of any of the relationship fields. Repeat Step 1 and Step 2 for each relationship or association occurrence you want to add.

   Note: Associations do not have attributes.

7. Type **S** in the Select byte of each row you want to update in the repository and press Enter.
8. Select EDIT.INSERT. This action inserts the new entity into the repository, the Row indicator clears, and the Action indicator (+) disappears.

Retrieve Relationships and Associations

This section describes how to retrieve existing relationships and associations from the Repository. You need to retrieve existing relationships and associations before you can change or delete them.

If you know the values for the general attributes (that is, its name, status, and version) of an entity, you can easily retrieve the entity by entering a combination of those attributes in Edit window fields, and then using the EDIT.SELECT command.

Note: Associations do not have their own name, status, or versions. You can enter the source and target name, status, and version and then use EDIT.SELECT.

Even if you do not know or are not sure of the general attributes of a relationship, you can still retrieve it in any Edit window by using the VIEW.LIST.ENTITY command and then tagging the relationship you want.

View Relationships and Associations in the Relationship or Association Type

To view single or multiple relationships and associations in a horizontal mode Edit window

1. Select the type of relationship or association you want to retrieve. The Repository displays a blank row in a horizontal mode Edit window.
2. Select VIEW.LIST.ENTITY. A list of all the relationships and associations appears.
3. Type **S** in the Select byte of the relationship or association you want to view. An Edit window appears.

Retrieve Relationships and Associations in Horizontal Mode

To retrieve single or multiple relationships and associations in a horizontal mode Edit window

1. Select the entity type for which you want to view occurrences. The Repository displays a blank row in the horizontal mode Edit window.

   Note: If you do not have a blank row, use the I line command to insert a new blank row.

2. Complete enough information in the fields that enable the Repository to find your relationship or association. For example, if the relationship has a unique name, type the name in the name column.

3. Type **S** in the Select byte of the row in which you just entered information.

4. Use EDIT.SELECT on the command line. The Repository fills in the other attribute fields for the retrieved relationship or association, and removes the Row indicator highlight and Action indicator to show that it successfully retrieved the entity.

5. If needed, use the LEFT or RIGHT standard commands to see the additional completed fields.

Retrieve Relationships and Associations in Vertical Mode

To retrieve relationships and associations in a vertical mode Edit window

1. Select the type of relationship or association you want to retrieve. The Repository displays a blank detail screen.

2. Switch to vertical mode.

3. Complete enough information in the fields that enable the Repository to find your relationship or association. For example, if the relationship has a unique name, type the name in the name column.

4. Use EDIT.SELECT on the command line. The Repository fills in the other attribute fields for the retrieved entity, removes the Action indicator from and displays SELECT in the Last Action field to show that it successfully retrieved the entity.

5. If more than one relationship or association is retrieved, use PREV and NEXT to view them.

Change Relationships

This section describes how to change the definition of existing entities you retrieved from the Repository.

You can change the attributes of any entity you retrieve into any Edit window by typing new information over the existing information for the entity and then using EDIT.UPDATE on the command line.

The EDIT.UPDATE command changes entities without affecting the relationships or associations that ties them to other entities or associations, even if part of the update includes altering the name, status, or version of the entity. The Repository can do this because it uses a special internal identifier (other than the combined name, status, and version) to select entities.

Note: You can change entities at the same time you are adding or deleting others if you update them using EDIT.SYNC instead of EDIT.UPDATE.

Change Relationships in Horizontal Mode

To change existing entities using a horizontal mode Edit window

1. Open an Edit window for the relationship or association you want to change.
2. Change the contents of any field by typing over the existing information. If needed, scroll the window to locate the field you want to change.
3. Press Enter. The Repository highlights the Row indicator for each row on which you changed information.
4. Type **S** next to the Select byte of the row you want to update in the repository.
5. Use EDIT.UPDATE on the command line. The Repository removes the Row indicator highlight for each row you tagged, indicating that it successfully updated these entities in the repository.

Change Relationships in Vertical Mode

To change existing entities using a vertical mode Edit window

1. Open an Edit window for the relationship or association you want to change.
2. Change the contents of any field by typing over the existing information. If needed, scroll the window to locate the field you want to change.
3. Use EDIT.UPDATE on the command line. The Repository updates the relationship in the repository and displays UPDATE in the Last Action field indicating that it successfully updated the relationship.

Delete Relationships and Associations

This section describes how to remove existing entities from the Repository.

You can delete any relationship you retrieve into any Edit window by first tagging it and then using EDIT.DELETE on the command line.

Notes:

- You can delete relationships while you are adding or changing other relationships so long as you first mark them for deletion using the D line command and then execute the deletions using EDIT.SYNC instead of EDIT.DELETE.
- When you delete relationships from the Repository, the Repository also deletes any associations and relationships in which the deleted relationships are either sources or targets.

Delete Relationships and Associations in Horizontal Mode

To delete existing relationships using a horizontal mode Edit window

1. Open an Edit window for the relationship or association you want to delete.
2. Type **D** on the Select byte next to the row you want to delete.
3. Use EDIT.DELETE on the command line. The Repository deletes the tagged entities from the Repository and removes the tagged rows from the window.

Delete Relationships and Associations in Vertical Mode

To delete existing entities using a vertical mode Edit window

1. Open an Edit window for the relationship or association you want to delete.
2. Use EDIT.DELETE on the command line. Depending on the entity being deleted, the Repository displays either:
   - A Cross Reference List window showing relationships related to the deleted relationship through existing relationships. Skip to Step 3.
   - A deletion confirmation window. Skip to Step 4.
3. Press F3 or enter **END** on the command line to continue with deletion. The Repository displays a deletion confirmation window.
4. Select YES to confirm that you want to delete the relationship. Depending on how many relationships you retrieved in Step 1, the Repository displays either a blank vertical mode Edit window or the next relationship.

Perform Mixed Edits Using EDIT.SYNC

You can perform a mixed series of entity additions, changes, and deletions and, when you are finished, update them all by using EDIT.SYNC on the command line. The EDIT.SYNC command makes it unnecessary for you to do all your additions, changes, and deletions in separate groups and execute individual EDIT.DELETE, EDIT.INSERT, and EDIT.UPDATE commands for each group.

Note: Although you can execute EDIT.SYNC from vertical mode, you should use EDIT.SYNC in horizontal mode Edit windows only. Since you can only process one entity at a time in vertical mode, EDIT.SYNC is no more useful in vertical mode than the other entity maintenance commands.

To perform mixed edits using EDIT.SYNC

1. Select the relationship or association type for which you want to add, change, or delete entities. The Repository displays a blank row in the Edit window.
2. Use the following table to determine which procedure to use.

If you want to	Use this procedure:
Add relationships	Add Relationships and Associations in Vertical or Horizontal Mode.
Change relationships	Change Relationships in Horizontal Mode or Changing Relationships in Vertical Mode.
Delete relationships or associations	Delete Relationships and Associations in Horizontal Mode or Delete Relationships and Associations in Vertical Mode.

Note: To cancel an edit, type **X** on the Select byte of the row you want to do **not** want to update. You can also change to another entity type or dialog to remove all changes. Use EDIT.SYNC on the command line. Your edits are updated in the repository.

If EDIT.SYNC is unable to execute any add, change, or delete action successfully (for example, if an add fails because of an existing, duplicate entity), the Repository rolls all actions back to the point at which you executed the command. You can re-execute EDIT.SYNC once you determine and resolve the cause of the error.

Chapter 6: Performing Impact Analysis

This chapter describes each of the Impact Analysis window types. In order to understand the differences in the window types, the same sample model and the same selected entity within that model is used for all examples.

This section contains the following topics:

What is Impact Analysis?

Every so often changes must be made to entities in the . Whether these changes are in the form of minor updates of attributes or deletions of entire entities, they can have an effect on many related associations and relationships.

To aid you in understanding the various functions of individual entities in the repository and the possible consequences of making changes to them, CA Repository for z/OS provides Impact Analysis windows.

Impact Analysis windows contain lists of entities that are related or associated to a specific entity. The following types of Impact Analysis windows are available:

- Where-used windows
- Uses windows
- Cross Reference windows

The direction of the relationship or association determines which relationships and associations appear in the Where and Uses windows. The Cross Reference window, on the other hand, displays **all** relationships and associations that use the specified entity.

Where-Used Windows

All CA Repository for z/OS meta-models consist of a number of different entities linked to one another through relationships or associations.

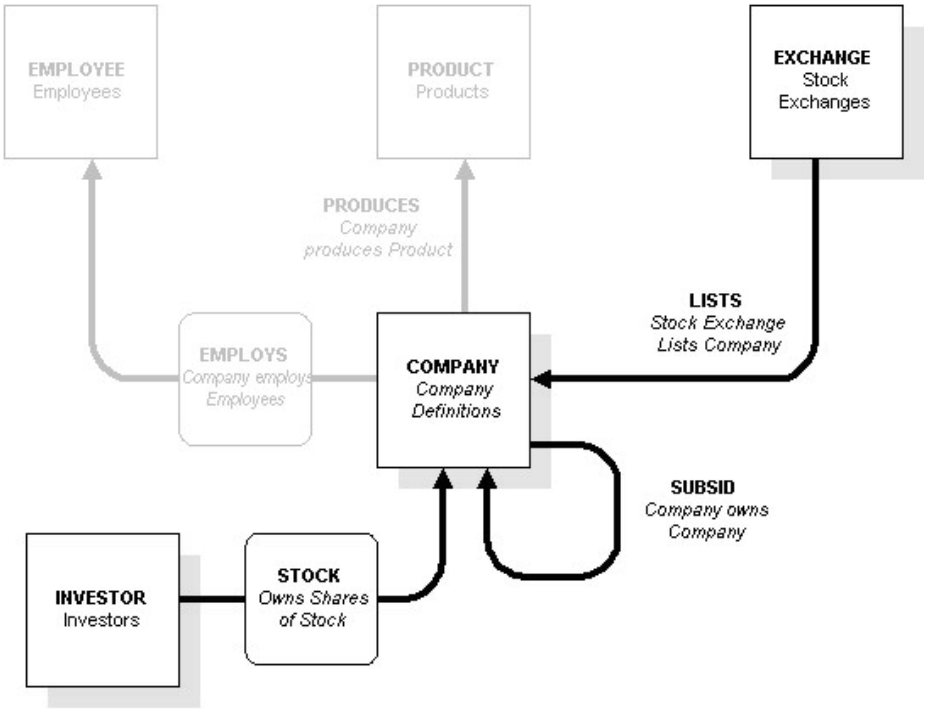
- An entity that is the source of a relationship or association is considered to be referencing or using the entity that is the target.
- Where-used windows show where an entity is used in the repository, that is, where it is the target of a relationship or association.

You use the Where-used window when you make changes to entities in the repository. If you are planning to delete a particular entity, generate a Where-used window in advance to show all the entities, relationships, and associations that would be affected by that action.

Once you generate a list of entities and relationships that are used by the entity you selected, you can select the related entity or the relationship using the MINIEDIT or TEXT commands for further processing.

Where-Used Window Example

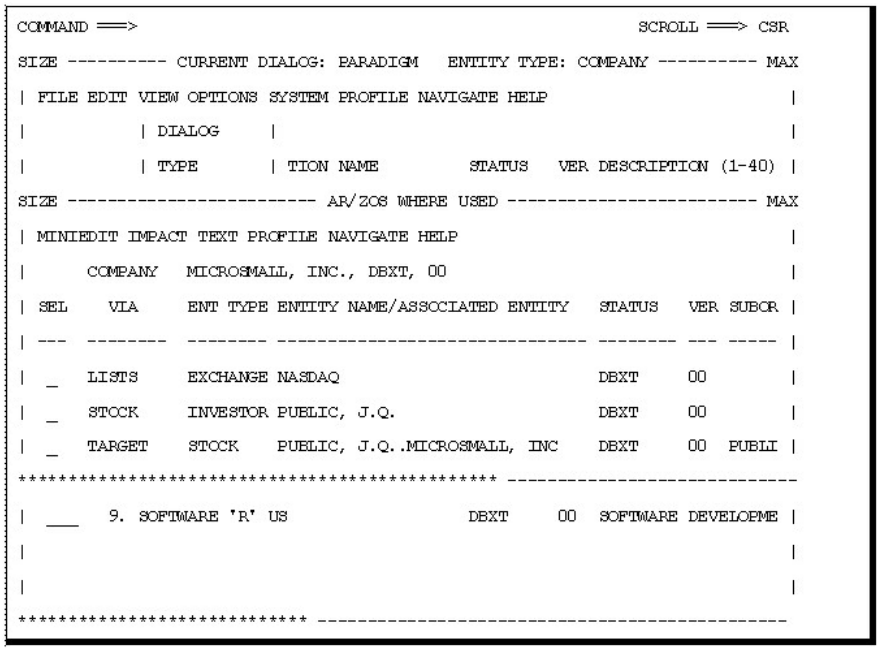
The following sample data model shows an entity type, COMPANY, whose occurrences both use and are used by other entities.



If you generate a Where-used window for a COMPANY entity, it lists any entities that used the selected entity. These could include:

- EXCHANGE entities linked to the selected entity through a LISTS association
- COMPANY entities linked to the selected entity through a SUBSID association
- INVESTOR entities linked to the selected entity through a STOCK relationship

The following is an example of a Where-used window.



This window was generated for a COMPANY entity as indicated in the line just below the Where-used menu bar. The first two data rows in this window represent entities that are linked to the selected COMPANY entity.

- The first row shows an EXCHANGE entity linked to the COMPANY entity through a LISTS association
- The second row shows an INVESTOR entity linked to the COMPANY entity through a STOCK relationship

Subordinate Entity

The third row demonstrates a subordinate entity.

- The word TARGET in the Via column indicates that the identified entity is the target of the relationship in this row. It contains the STOCK relationship linking the INVESTOR entity to the COMPANY entity.
- The INVESTOR entity itself appears in the Subordinate Entity column. You can see it by scrolling the Where-used window to the right.

Generate a Where-Used Window

To generate a Where-used window

1. Select the entity you want to perform an impact analysis on.
2. Select VIEW.IMPACT.WHERE. The Impact Analysis window for the entity is displayed.

Uses Windows

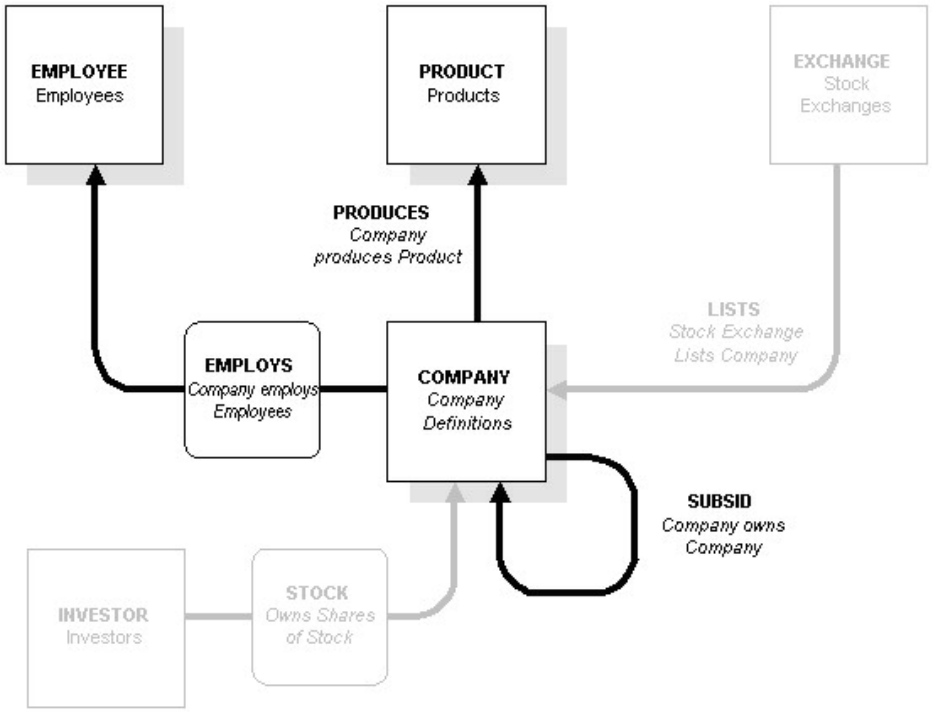
Uses windows are essentially the exact opposite of the Where-used windows. Instead of showing which entities use a specific entity, Uses windows show which entities a specific entity use, that is, where it is the source of a relationship or association.

Use the Uses window when you make changes to entities in the repository. If you are planning to delete a particular entity, generate a Uses window in advance to show all the entities, relationships, and associations that would be affected by that action.

Once you generate a list of entities and relationships that the current entity uses, you can select these entities and relationships using the MINIEDIT or TEXT commands for further processing.

Uses Window Example

The following sample data model shows an entity type, COMPANY, whose occurrences use and are used by other entities.



If you generate a Uses window for a COMPANY entity, the window lists any entities that the selected entity uses. These could include:

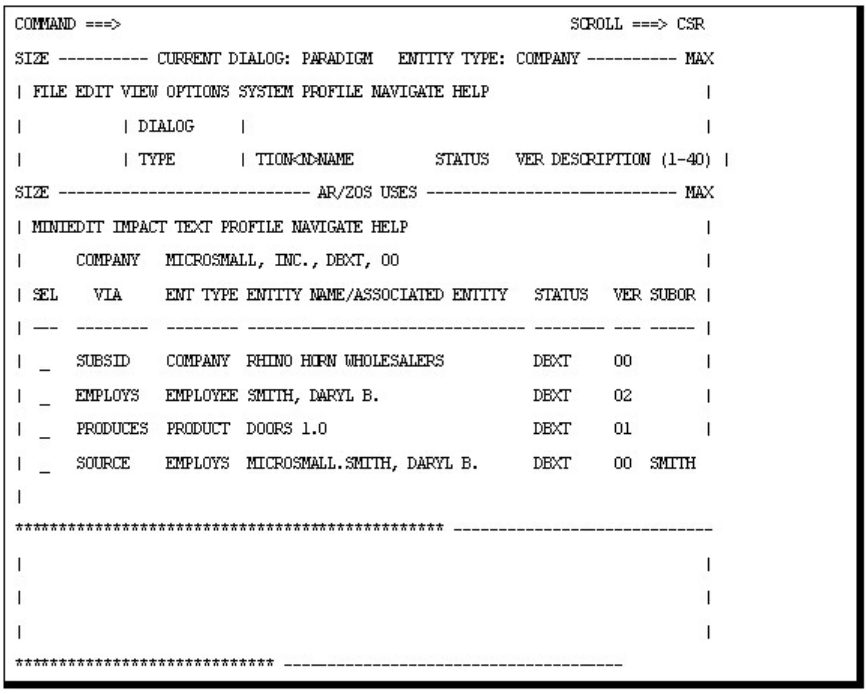
- PRODUCT entities linked to the selected entity through a PRODUCES association
- COMPANY entities linked to the selected entity through a SUBSID association
- EMPLOYEE entities linked to the selected entity through a EMPLOYS relationship

The following window was generated for a COMPANY entity as indicated in the line just below the Uses menu bar. The first three data rows in this window represent entities that are linked to the selected COMPANY entity.

- The first row shows another COMPANY entity linked to the selected COMPANY entity through a SUBSID association
- The second row shows an EMPLOYEE entity linked to the COMPANY entity through a EMPLOYS relationship

- The third row shows a PRODUCT entity linked to the COMPANY entity through a PRODUCES association
- The fourth row contains the actual EMPLOYS relationship linking the EMPLOYEE entity to the COMPANY entity
  - The EMPLOYEE entity itself appears in the Subordinate Entity column-you can see it by scrolling the Uses window to the right
  - The word *SOURCE* in the Via column indicates that the identified COMPANY entity is the source of the relationship in this row

The following illustration is an example of the Uses window.



Generate a Uses Window

To generate a Uses window

1. Select the entity for which you want to generate and display a Uses window.
2. Select VIEW.IMPACT.USES. The Uses window appears.

Cross Reference Windows

Cross Reference (XREF) windows are a combination of both the Where-used and Uses windows. These windows show both the entities that use and are used by a selected entity.

Like all Impact Analysis windows, you use the Cross Reference window when you make changes to entities in the repository. For example, if you are planning to use the DELETE, UPDATE, or REPLACE command on a particular entity, you generate a Cross Reference window in advance to show all the entities, relationships, and associations that would be affected that action.

Relationships are always represented in impact analysis windows from both the related entity and relationship perspective. You can select either the related entity or the relationship using the MINIEDIT or TEXT commands for further processing directly from the Cross Reference window.

Generate a Cross Reference Window

To generate a Cross Reference window

1. Select the entity for which you want to generate a Cross Reference window.
2. Select VIEW.IMPACT.XREF. The Cross Reference Impact Analysis window appears.

Secondary Impact Analysis Windows

To gather information about subordinate entities, you display a Secondary Impact Analysis window.

Generate a Secondary Impact Analysis Window

To generate a Secondary Impact Analysis window

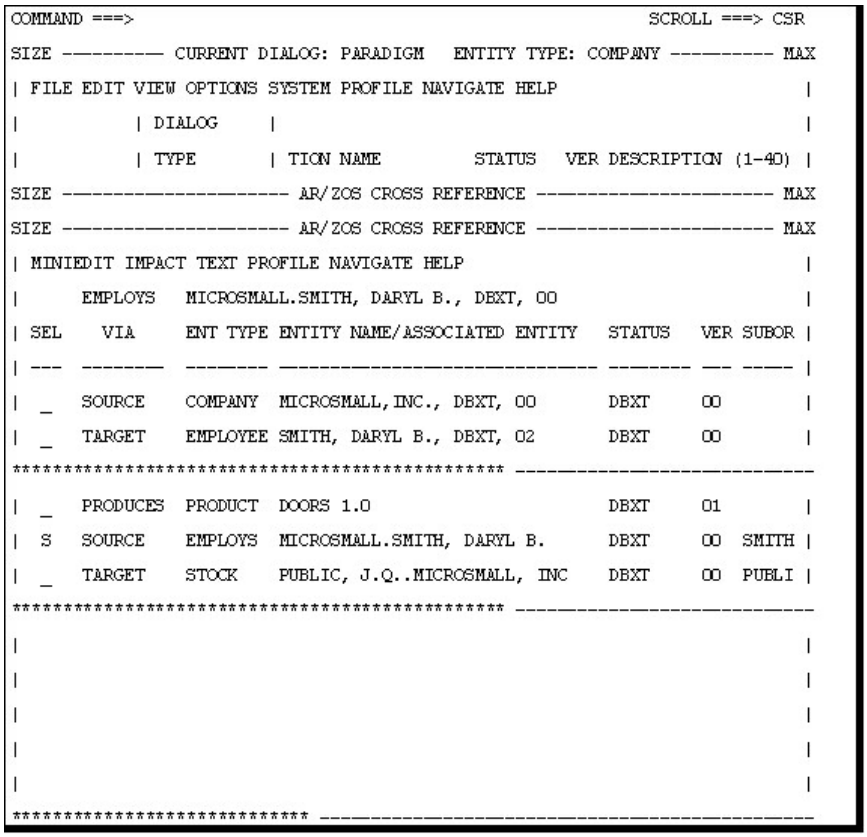
1. Select the subordinate relationship you want to use.
2. Select IMPACT, then select the desired secondary analysis type desired: XREF, WHERE or USES. A new Cross Reference Impact Analysis window for the relationship appears.

Note: You can open up to ten new Cross Reference Impact Analysis windows.

Secondary Impact Analysis Window Example

The following illustration is an example of a Secondary Impact Analysis window. The window resulted from tagging EMPLOYS relationship in the previous example and then generating a Cross Reference window.

Note: The COMPANY entity used to generate the initial Cross Reference window appears in this Secondary Cross Reference window. You can stack up to ten Impact Analysis windows in this manner.



Chapter 7: Using Queries and Search Criteria

This chapter discusses using List windows to provide lists of existing entities, relationships, and associations.

This section contains the following topics:

List Windows

You use List windows to retrieve entities from the Repository. There are different types of List windows:

- List of Entities
- List of Source
- List of Target

Use List of Entities windows to generate a list of all the entities for the entity type that is currently selected. This window is useful for retrieving entities to the Edit window.

Use the List of Source and List of Target windows when you generate a list of possible source and target entities when creating a new relationship or association. These window commands are only available from a relationship or association window.

Generate a List Window

You can retrieve all the entities for a particular entity type from the repository.

Note: Use this method when you do not know the exact name, version, or status of the entity you want to find.

To view a list of all the entities for an entity type

1. Open an Edit window for the entity type you want to use.
2. Do one of the following:
   - To list of all the entities, select VIEW.LIST.ENTITY .
   - To list of all the entities that you can use as the source of a relationship or association, select VIEW.LIST.SOURCE.
   - To list of all the entities that you can use as the target of a relationship or association, select VIEW.LIST.TARGET
   - A List window displays a list of all the entities for the currently selected entity type. If there are more entities than the maximum list length allows, a message to that effect is displayed. If you selected Source or Target and no sources or target entities exist in the entity type you are using, the List window does not appear.
3. Type **S** in the Select bytes of the entities you want to retrieve and press Enter. The Edit window appears with the entities you selected.

Limit Lists by Name, Status, and Version

Although List windows are very helpful in determining what entities, relationships, and associations exist in the repository, they can lose some of their effectiveness once the number of entities displayed in any one window becomes so great that you need to scroll through numerous pages.

The simplest way to limit the entities in a List window is by entering data into one or more of the following fields:

- Name
- Status
- Version

Before you generate a List window, you can enter information in the Name, Status, and Version fields about the entities you want to find.

- If you enter a status or a version number, the Repository finds entities containing the version number or status that you entered and displays them in the List window. For example, if you enter 00 in the Version field, only entities with the version attribute equal to 00 appear in the List window.
- If you enter a letter or string of characters in the Name field, the Repository finds entities whose name contains the letter or string and displays them in the List window. For example, if you enter the letter A in the Name field of a tagged Edit window row and then generated a List window, the resulting list only contains entities whose names start with the letter A.

Note: The Repository automatically appends a % (wildcard) to any characters entered in the Name field. If you want a list of entities whose names simply contain an A (as opposed to starting with an A), enter %A prior to generating the list.

Entering values into all three fields causes the list to be further reduced. For example, if you enter 01 in the Version field, DEV in the Status field and AB in the Name field, all the entities containing the letters AB in its name and belonging to the DEV status and version 01 appear in the List window.

To limit the number of entities in a List window

1. Open an Edit window for the entity type you want to use.
2. Do one or more of the following:
   - If you know the name of the entity you want to find or a few characters of the name, type the name in the Name field
   - If you know the status number of the entity you want to find, enter the status in the **S** field (Status field)
   - If you know the version number of the entity you want to find, enter the version number in the V field (Version field)
3. Do one of the following:
   - To list of all entities that match the name, version, or status you specified, select VIEW.LIST.ENTITY
   - To list of all entities that match the name, version, or status you specified that are possible source entities for the relationship or association you are using, select VIEW.LIST.SOURCE
   - To list of all entities that match the name, version, or status you specified that are possible target entities for the relationship or association you are using, select VIEW.LIST.TARGET

   The entities that match the name, version, or status specified are displayed.

Work with Queries

This section describes information and procedures for creating, modifying, and using queries.

Understand Queries

Finding a particular entity in a large entity list can be a time consuming task, especially if you were looking for a group of entities with a particular version and the entities were listed alphabetically by name. Rather than scroll through the entire list, tagging the appropriate entities as you find them, you can use the Search Criteria window to build a query and subsequently narrow the list.

You use a query to specify certain attribute value requirements, known collectively as *search criteria*. The Repository uses the query to control the entities that appear in a List window. The Repository displays only those entities whose attributes meet the specified criteria. You can also use queries to determine which attributes of entities appear and in what order.

The Repository deliverable contains a few sample queries that were created for you. Or you can create your own and save frequently used or complex queries for future use.

You can use search criteria regardless of where you started generating the List window (Entity, Source, or Target).

Specify Search Criteria

To specify search criteria for a query in the Search Criteria window

1. Open an Edit window for the entity type you want to use.
2. Select SYSTEM.CRITERIA. The Search Criteria window appears.

Search Criteria Parameters

To perform searches in the repository, you use the Search Criteria window to specify the parameters that tell the Repository how to perform the search.

You use the Search Criteria window fields to specify the fields you want to use in your query. These fields are described next.

Field	Description
Qualifier	Enter your TSO logon ID. The qualifier identifies the creator of the

Field	Description
	query.
Query Name	Specify the name of the query (up to 8 characters). The name you specify can include blank spaces, but not decimal points or periods (“.”). You use the query name to retrieve the query in the future.
Description	An optional field. Enter a text description (up to 50 characters) of the query.
Columns	■ Use to list the names of the entities attributes you want to display in the List window. ■ Enter a question mark (?) in the first column field to generate a list of all the columns. ■ Select the columns you want to see in the new List window. The order in which you select these columns determines the order in which the attributes are displayed.
Where	Use to create SQL search statements. You can use the following components to create a search statement: ■ Columns ■ Operators ■ Functions ■ Values You access windows for selecting columns, operators, functions, or values for use in the Where section through the SQL function on the menu bar.
Order By	Specify the order in which the entities are to be displayed in the List window. (The Search Criteria window on the previous page does not show this field.)

Select the Columns on Which to Search

The Columns section of the Search Criteria window displays a list of DB2 columns for the currently selected entity type. These columns correspond to the attribute types of the current entity type, though the actual names may vary slightly from the literals displayed in the Edit window. Each of the column names are prefixed by one of three letters.

- Columns that refer to the attributes of source entities are prefixed with an S
- Columns that refer to the attributes of target entities are prefixed with a T
- All other columns are prefixed with an E

The Name, Status, and Version attributes always appear in the List window even if they are not specified in the Columns section. They appear in the List window to the right of any columns that are specified in the Columns section.

Use the columns as parameters in the Columns, Where, and Order By sections of the Search Criteria window.

Notes:

- Only the attributes corresponding to the columns specified in the Columns section of the Search Criteria window appear in the window
- Status and Version, along with Name are always included in List windows and are to the right of the selected columns.
- The entities are listed alphabetically by name as specified in the section

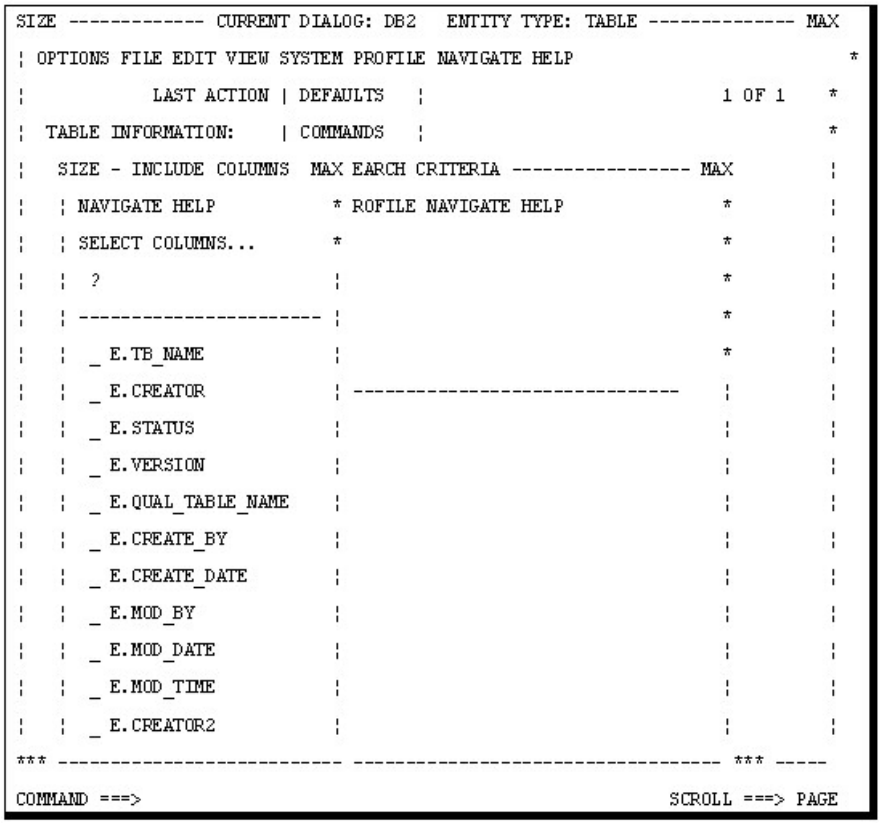
Determine the Sort Order of the Columns

You can also use the numbers 1-9 as tags (instead of the *S*) to specify the order of the columns. If you use both S and the numbers 1-9, the numbered columns appear in numerical order before the columns tagged with an S.

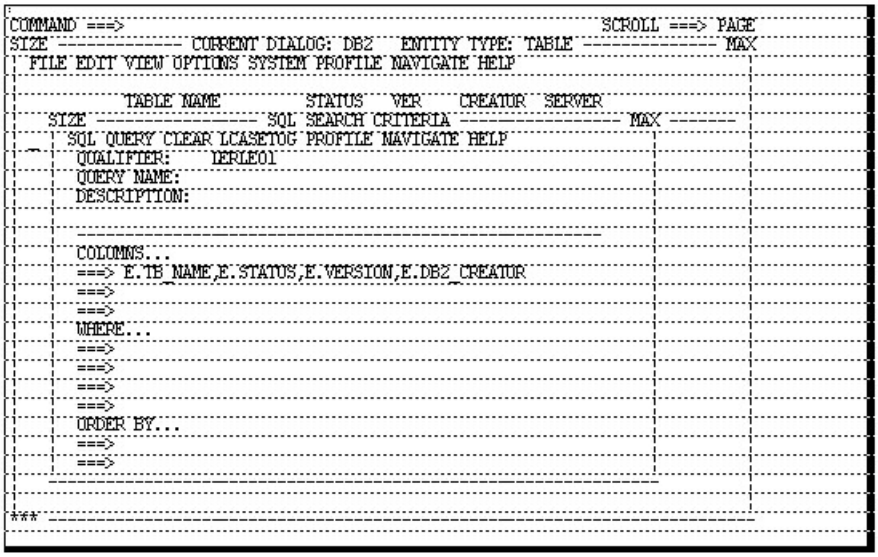
To select the columns on which to search

1. Open the Search Criteria window.
2. Use the arrow keys to move the cursor to the first blank line in the Columns section.

3. Type a question mark (**?**), and press Enter. A list of columns for the entity type is displayed in a Column window.



4. Type **S** (or numbers) in the Select byte fields next to the columns you want to display on the list window and press Enter. The names of the columns you selected appear in the Columns section of the Search Criteria window.



Note: You can type the names of the columns directly in the Columns section. Separate column names with commas.

Create the SQL Statement

You can create SQL statements in the Where section of the Search Criteria window. You can either type the search strings directly into the Where section of the Search Criteria window or you can use one or more of the four windows that are available to help you create the SQL Search Statement.

Before you can create an SQL statement, you need to understand its components:

SQL Statement Components	Description
Column	Columns are entity attributes that you want to reference during the search. The columns for this section are the same as those used in the COLUMNS section. At least one column is required for every search statement.
Operator	Operators are an essential component of all search statements as they are used for making the actual comparison conditions that run a search statement. For a list of the operators available in the Search Criteria window, see Selecting the Operator to Use in the SQL Statement.
Function	Functions are advanced search parameters They can be found by executing the SQL.FUNCTION command while in the Search Criteria window.
Value	Values are user-specified numeric or character groups used to complete a search string. This is the text or the number values you want to use during the search.

Two examples of SQL statements are:

```
VERSION > '04' (CHARACTER_COLUMN)
DECIMAL > 53 numeric column
```

If you are specifying more than one search criteria, separate the statements with the words *AND* or *OR*.

Note: If you use OR, place parenthesis around both the individual phrases and the entire statement to ensure proper interpretation by DB2.

Select the Operator to Use in the SQL Statement

The Operator window contains a list of SQL operators that you use to specify the actual comparison conditions that run the search criteria.

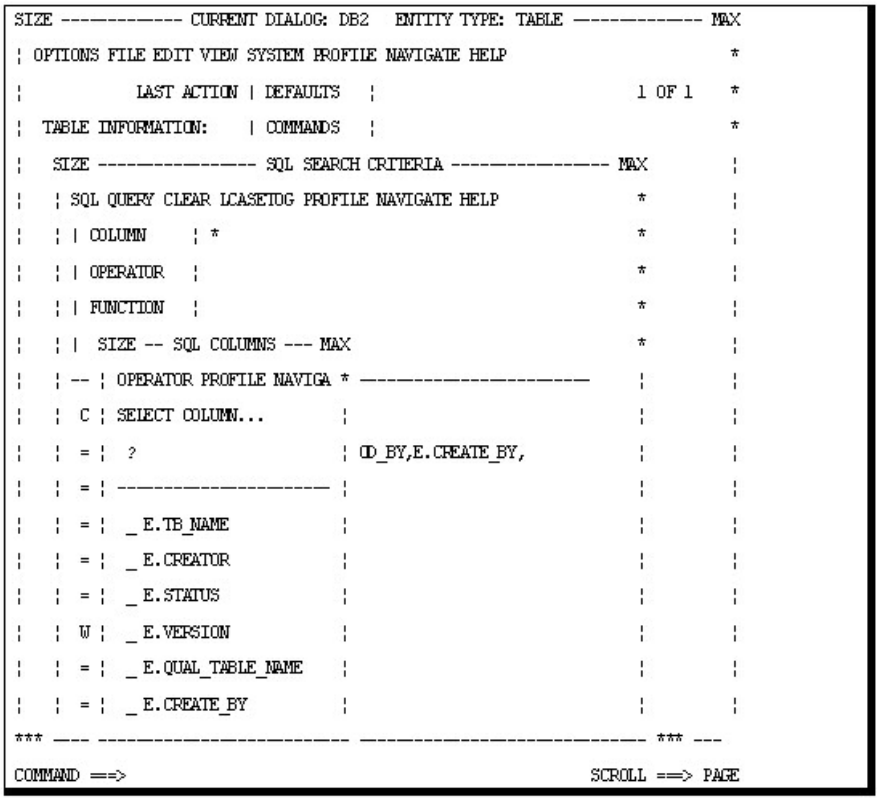
The following operators are available in the Repository.

=	-=>	<>
>	->	>=
<=	LIKE	NOT LIKE
IN	NOT IN	

To create a SQL statement

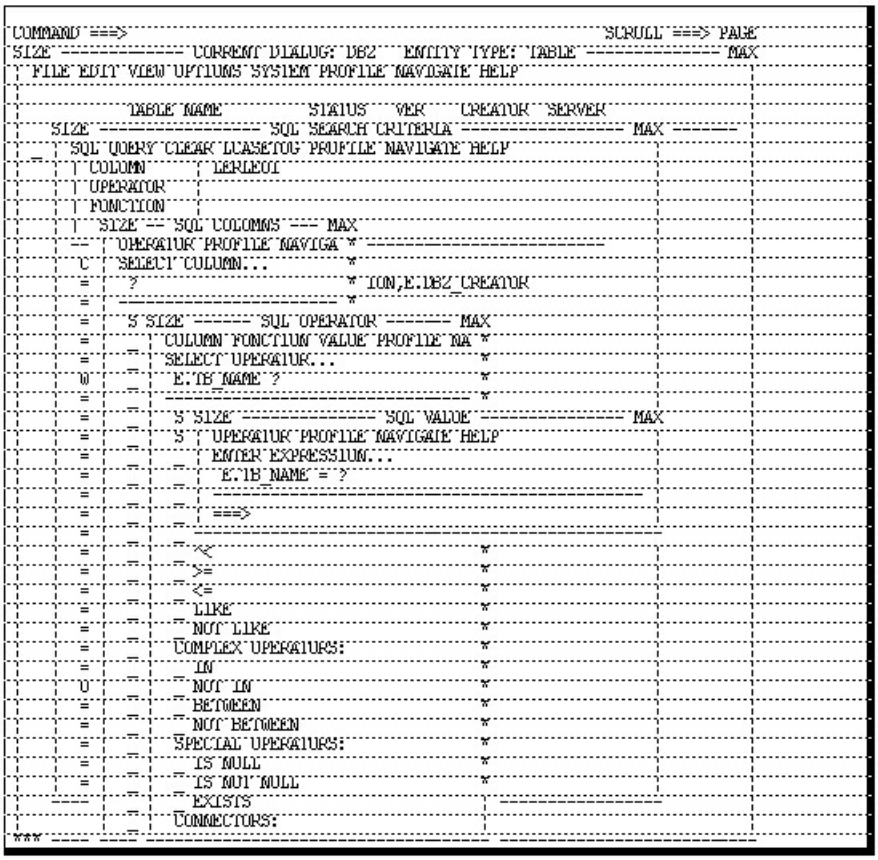
1. Open the Search Criteria window.
2. Use the arrow keys to move the cursor to the first input field of the Where section.

3. Type **?** and press Enter. This marks the position in the Where section where you are going to insert the SQL statement you are creating.
4. Select SQL.COLUMNS. A list of columns (attribute types) for the current entity type appears.



5. Type **S** in the Select byte field next to the column you want to use. Do **not** press Enter.
6. Select OPERATOR. The Operator window listing the available operators appears.
7. Type **S** in the Select byte field of the operator you want.

8. Select VALUE. The Value window appears.



9. Enter the value you want to use in the SQL statement in the Value field and press Enter. For example, if you want to search for Sample type **('SAMPLE%')** in the Value field. The stacked SQL windows disappear and the SQL and the string is added to the Search Criteria window.

Sort Entities in the List Window

The last section of the Search Criteria window is the Order By section and is located at the bottom of the Search Criteria window (you may have to scroll the window down to see this section).

The Repository uses the information stored in the Order By section of the Search Criteria window to determine the order in which entities are presented in the List window.

Select Ascending or Descending

Entities are normally sorted in ascending (lowest to highest) order. You can specify to sort the list in descending order by typing DESC. The method in which you enter information in the Order By section is very similar to that used in the Columns section.

For example, if you enter E.NAME in the Order By section, the Repository lists the entities alphabetically by their Name attributes.

- If you list more than one column in this section, the column listed first is given priority
- If two or more entities have identical attributes for the first column, the second column is then used to determine their order

You can specify any number of columns in this section so long as the total number of characters does not exceed 250. If you specify too many columns, the list is truncated after 250 characters.

The information you enter in the Order By section remains until you clear it or exit the current List window.

You can use this simple query sort the entities in the current List window and discard it or you can save it for future use.

To sort entities in the List window

1. Open the Search Criteria window.
2. Using the arrow keys move the cursor to the first line in the Order By section of the Search Criteria window.
3. Select SQL.COLUMNS. A list of columns for the entity appears.
4. Do **one** of the following:
   - To sort on only one column, type **S** in the Select byte field of the column you want
   - To sort on more than one column, type **S** in the Select byte field of the column you want followed by a number. Repeat this step for additional columns, you can specify up to nine columns.
5. Press Enter. The column you selected appears in the Order By section of the Search Criteria window.
6. Press F3 or enter END on the command line. the Repository sorts the entities and displays them in the List window.

Create a Query

The following example shows how to access the Search Criteria window to create SQL statements that narrow a list of entities displayed in the List window.

The following is a summary of what you do to create a query:

- Open a List window for the entity type.
- Open the Search Criteria window.
- Specify parameters in one or more of the following sections:

  Columns

  Where

  Order By
- Save the query.

To create a query

1. Open a List window for the entity type.

   Select CRITERIA from the menu bar of the List window. The Search Criteria window appears.

   Note: You can also access the Search Criteria window directly from the Edit window by entering the appropriate LIST command at the command line followed by a Q switch **(VIEW.LIST.ENTITY Q**). Bypassing the initial, undelimited List window in this manner speeds up processing in instances where the list of entities would otherwise be very large.
2. Complete the following fields:

   Qualifier
   Your TSO logon ID

   Query name
   The name you want to give the query.

   Description
   A description of the query.
3. To select the columns in which the Repository conducts its search, enter the names of the columns in the Column section.
4. To create an SQL statement, follow the procedure Creating the SQL Statement.

5. To specify the order in which the columns appear in the List window, follow the procedure Sorting Entities in the List Window.
6. Select one of the following:
   - QUERY.SAVE to use this query for only this entity type
   - QUERY.SAVEALL to make this query available for all entity types

   Either of these commands saves the query with your TSO logon ID.

Save a Query

Once you have created a query, you can generate a List window. You can retrieve the entities in the List window to the Edit window. To do this, tag the Select byte of the entities you want to retrieve and press Enter. You return to an Edit window containing the selected entities.

You have several options at this point.

- You can use the displayed query immediately *as is* by exiting the Search Criteria window
- You can make changes to any of the search criteria fields and then exit the window. These changes are saved as the current query but will not affect the selected query
- Finally, you can create a new query using the original or modified search criteria by typing a new name over the previous name and then using the QUERY.SAVE or QUERY.SAVEALL commands
  - The SAVE command saves the query for use at the current entity type
  - The SAVEALL command, on the other hand, saves the query so that it will be available for all entity types
  - Because most entity types have unique attribute types, you should use the SAVE command in most instances

If you execute the VIEW.LIST.ENTITY command, the Repository returns a List window containing the entities that match the information specified in the Search Criteria window. The information remains on the Search Criteria window until you remove it.

To save a query

1. Create a query following the steps in the To create a query section.
2. Do one of the following:
   - To use the query for the current entity type only, select QUERY.SAVE
   - To make the query available for all entity types, select QUERY.SAVEALL

   Either of these commands saves the query. The query appears in the list on the Where Criteria window.

Delete a Query

You can delete any query that you created and saved under your TSO logon ID.

To delete a query

1. Select LIST.CRITERIA. The Search Criteria window appears.
2. Select QUERY.DELETE. The Where Criteria window appears with a list of queries that you are authorized to delete.
3. Type **S** in the Select byte next to the query you want to delete.
4. Press F3 or enter END on the command line to delete the query.

Use an Existing Query

Selecting and using an existing query is a far simpler process than creating a new one.

You can create queries to use only for specific entity types. If you are not using the entity type to which the query is specific, it does not appear on the query list.

Queries are saved with the ID of the person that created it. Any query that you create is only visible when you are logged on to the Repository with your ID.

The Repository Administrator generally creates queries that are visible to all users.

To use an existing query

1. Open an Edit window for the entity type you want.
2. Select LIST.CRITERIA. The Search Criteria window appears.

3. Select QUERY.SELECT. The Where Criteria window appears displaying a list of available queries that you can use. If you decide you do not want to use an existing query, select CURRENT. The Search Criteria window reappears with no changes.
4. Type **S** in the Select byte next to the query you want to use and press Enter. The Search Criteria window appears containing the name, description, and search criteria of the query you selected.
5. If necessary, make changes to the information in the Search Criteria window.
6. Press F3 or enter END on the command line to exit the Search Criteria window. The List window appears containing the entities that match the search criteria in the query you selected.

Clear the Selection Criteria Window

The search criteria you specify in the Search Criteria window remains in effect until you:

- Generate a list without using these search criteria; do one of the following:

  End the edit session.

  Change entity types

  Clear the criteria window using the CLEAR option

- Use the following procedure to clear the Search Criteria window.

To clear the Search Criteria window

1. Open the Search Criteria window.
2. Select CLEAR. The query information in the Search Criteria window disappears.
3. Press F3 or enter END on the command line to exit the Search Criteria window. The List window appears containing the all the entities for the current type.

Shortcuts for Fast Query Selection

If you know the name of a query, you can bypass the QUERY pull-down menu and subsequent Where Criteria window.

1. From the Search Criteria window, type the name of the query and the user ID in the appropriate fields of the Search Criteria window (over any existing name or ID, if necessary).
2. Use the QUERY.SELECT command. The components of the new query displace the components of any query currently displayed in the Search Criteria window.
3. From the command line simply use the following syntax:

   ```
   VIEW.LIST.(option) Q=creator.query-name
   ```

   ```
   The creator. parameter defaults to your TSO logon ID if it is not specified.
   ```

Example Queries

The following example queries can be used to limit the entities appearing in a List window.

To find all entities you created on a particular day:

```
E.CREATE_DATE = '2002-XX-XX' AND
E.CREATE_BY = 'your USER ID'
```

To find orphan queries *(*all entities not used anywhere):

```
NOT EXISTS (SELECT TARGET_ID FROM DBXREL30.DBX_XREF
WHERE TARGET_ID=X.ENT_ID)
```

To find widow queries (all entities that do not use anything):

```
NOT EXISTS (SELECT SOURCE_ID FROM DBXREL30.DBX_XREF
WHERE SOURCE_ID=X.ENT_ID)
```

To find tables not in the DB2 catalog:

```
E.TB_NAME NOT IN (SELECT NAME FROM SYSIBM.SYSTABLES
WHERE TYPE = 'T')
```

To search for occurrences using the data type of the currently selected entity:

```
E.DATA_TYPE = 'CURRVAL(E.DATA_TYPE)'
```

Note: CURRVAL is a special value used by the Repository to retrieve occurrences using values currently listed in the Edit window.

- If you are using the vertical mode, the Repository uses the attribute of the currently selected entity
- If you are using the horizontal mode, the Repository uses the value from the tagged entity

Chapter 8: Commands and Features

In addition to the basic entity maintenance commands such as INSERT, SELECT, UPDATE, DELETE, and SYNC, the Repository provides several special commands and features to simplify or augment the entity definition process. This chapter describes these features.

This section contains the following topics:

Change an Attribute in Several Entities

Use the CHANGE command to change a specified attribute of several entities from one value to another. You can make the same change many entities.

Important! Since the CHANGE command is designed to modify multiple entities simultaneously, it can only be used in the horizontal mode.

The CHANGE command does not require you to tag the entities you want to process. The attribute value itself is used as a qualifier for determining which entities are affected by the command. Consequently, any entities that you do not want to change should first be excluded from the Edit window.

To change an attribute in several entities

1. Open an Edit window containing the entities you want to change in horizontal mode.
2. Select EDIT.CHGSCR.CHANGE. The Change window appears. Enter **one** of the following values for the column containing the attribute type you want to change:
   - The DB2 name corresponding to the column. For example: VERSION.
   - The literal in the Edit window representing the column name. For example: VER.
   - The column number. For example: 3.

   Note: If you enter *, the change is made in every column. To see a list of all the literals and column names, type **?** in this field. You can select the column you want to use from this list.
3. Complete the To and From fields in the Change window.

   From

   The value you want to change. For example, if you want to change the version number from AA to 00, enter AA.

   To

   The new value you want to use. For example, if you want to change the version number from AA to 00, enter 00.
4. Press Enter. The changes are made and appear in the Edit window.
5. Select EDIT.UPDATE to save the changes in the repository.

Use CHANGE from the Command Line

You can execute the CHANGE command directly from the command line, bypassing the Change window. The syntax is:

```
EDIT.CHGSCR.CHANGE literal, from value, to value
```

Move Entities from One Type to Another Within an Entity Set

Use the CHGTYPE command to move entities from one entity type to another. You can only use this command for entity types that are part of entity sets, and it can only move entities between the entity types that are part of that set.

Before You Change an Entity Type

Before you change an entity from one type to another, make sure that:

- Relationships or associations that use the entity you want to change will still be valid as the new entity.
- Relationships or associations that become necessary because of the change are created (for example, if an ELEMENT entity is changed to an ALIAS entity, a PHY AL association will have to be defined in order for the ALIAS entity to be valid).

To change an entity's entity type

1. Open an Edit window that contains the entity you want to change.
2. Enter **S** in the Select byte of the entity you want to change.
3. Select EDIT.SPECIAL.CHGTYPE. The Entity Type List window displays a list of entity types to which the entity selected could be changed. The types listed in the window occupy the same entity set as the entity you selected.
4. Enter **S** in the Select byte of the entity type you want to use and press Enter. The Edit window appears. Because the selected entity is now part of a new entity type, it is no longer displayed in the Edit window.
5. Select EDIT.UPDATE to save the change in the repository.

The COPY Subfunction

To simplify processing by eliminating the need to re-enter similar or redundant data, you can use any of the following commands that are part of the COPY subfunction (under the EDIT function):

EDIT.COPY.ALL	Copies the relationships, associations, and text of an existing entity to another entity of the same type.
EDIT.COPY.OTHER	Copies attributes from an entity of a different type.
EDIT.COPY.RELATED	Copies the relationships linked to one entity to another entity of the same type.
EDIT.COPY.TEXT	Copies an entity along with its extended text.

Important! You can only use the COPY.ALL and COPY.RELATED subfunctions to copy relationships from one entity to another. If the entities are in different statuses the relationship name, status, and version is not altered.

The COPY subfunction does not copy text to multiple entities simultaneously. If more than one entity was tagged prior to executing these commands, a separate Copy window appears, sequentially, for each.

Use EDIT.COPY.ALL

You can use the EDIT.COPY.ALL to copy the relationships, associations, and text of an existing entity to another entity of the same entity type. It is a combination of the functions performed by RELATED and TEXT commands.

EDIT.COPY.OTHER

You can use the EDIT.COPY.OTHER command to copy attributes from an entity of a different type. For example, if you were defining record layouts for a particular table, you would use EDIT.COPY.OTHER to copy the COLUMN relationship data of a specified table to the FIELD relationships of a specified record.

All attributes with the same name are copied and any information on the tagged row or rows (horizontal mode) is repeated for each retrieved row. You can enter the source name, status, and version, tag the row, execute the OTHER command, and the source name, status, and version is copied on each of the new rows.

Before you use EDIT.COPY.OTHER, make sure the appropriate source and target entities already exist before you copy a relationship. For example, the entities RECORD and ELEMENT must exist before you can insert the FIELD relationship.

To copy attributes from an entity of a different type

1. Open an Edit window for the entity in which you want to copy attributes.
2. Select EDIT.COPY.OTHER. The Entity Type List window appears.
3. Enter **S** in the Select byte of the entity type from which you can copy information. The Copy Other Criteria window appears.
4. Use the following information to complete the fields in the Copy Other Criteria window.

   Source/Target

   Specify one of the following:

   - S—To see a list of source entities.
   - T—To see a list of target entities.

   Leave this field blank if you want to see both source and target entities.

   Name

   Specifies the name of the entity from which you want to copy information. You can use % wildcards to find entities.

Status

Specifies the status of the entity from which you want to copy information.

Version

Specifies the version of the entity from which you want to copy information.

Note: Use the Copy Other Criteria window to specify search criteria for the entity you want to copy information from. Use this window to limit the size of the selection list.

5. Press Enter. A list of entities that match the criteria you specified in the Copy Other Criteria window appears.
6. Enter **S** in the Select byte field of the entity type you want to use and press Enter. A list of the entity instances you can copy appears.
7. Enter **S** in the Select byte field of the entity instance you want to copy and press Enter. The information from the entity instance you copied is displayed in the Edit window.

Note: If the Name, Status, or Version fields contain any data, the COPY.OTHER command adds to the existing data; it is not overwritten.

EDIT.COPY.RELATED

Use the EDIT.COPY.RELATED command to copy relationships that are linked to one entity to another entity of the same type.

When you use the EDIT.COPY.RELATED command, relationships are copied using the name of the relationship being copied, and the status and version of the original entity. If a relationship already exists with that combination of name, status, and version, a warning message appears and the copy fails.

In some cases, the name of a relationship is a system-generated concatenation of the names of its source and target entities. If this is the case with the relationships you are copying, you will need to access that relationship type and select any newly created relationships. Once these relationships are displayed, pressing Enter correctly adjusts the relationship names. They must be updated in the repository in order for the changes to become permanent.

To copy relationships linked to one entity to another

1. Open an Edit window for the entity to which you want to copy relationships. Select VIEW.MODE to switch to horizontal mode.
2. Enter **S** in the Select byte next to the entity **to** which you want to copy a relationship.

3. Select EDIT.COPY.RELATED. A list of all existing entities for the current entity type is displayed.
4. Enter **S** in the Select byte next to the entity type **from** which you want to copy relationships and press Enter. A list of relationship types that are linked to the entity type is displayed appears.
5. Enter **S** in the Select byte next to the relationship type containing the relationships that you want to copy and press Enter. The Edit window appears.
6. Select EDIT.UPDATE to save the changes in the repository.
7. Verify the changes were made by opening a Cross Reference window.

EDIT.COPY.TEXT

Though most entities can be easily duplicated using the horizontal mode of the Edit window and the R line command, this procedure will not copy any existing extended text. Use the EDIT.COPY.TEXT to duplicate an entity along with its extended text.

To copy extended text

1. Open an Edit window for the entity to which you want to copy another entity and its extended text. Use VIEW.MODE to switch to horizontal mode.
2. Enter **S** in the Select byte next to the entity to which you want to copy another entity and its extended text.
3. Select EDIT.COPY.TEXT. The list of entity types from which you can copy an entity is displayed.
4. Enter **S** in the Select byte of the entity you want to copy and press Enter.
5. Select EDIT.UPDATE to save the changes in the repository.

Domains

In the Repository, a domain is a group of values for a specific group of attribute types within a given entity type. These values can automatically be inserted into the appropriate Edit window fields by the specification of a predetermined key value in a particular attribute field. Domains are very useful in instances where certain attribute values within a given entity type frequently occur together.

Entity types that use domains have one or more domain key attributes. Specifying a particular key value in this attribute field automatically adds predetermined domain values to one or more other attribute fields. Which attribute type, if any, of a particular entity type is the key domain attribute is site-specific and is determined by your product administrator. Often the attribute type literal for these key attributes contain the word DOMAIN.

Access Domain Values

There are two methods of accessing domain values:

- Typing a domain key directly into the key attribute field. This method completes only those domain attribute values that are currently blank or zero.
- Use the EDIT.DOMAIN command (by means of the menu bar or command line). This method inserts all existing domain values, overwriting existing values as needed.

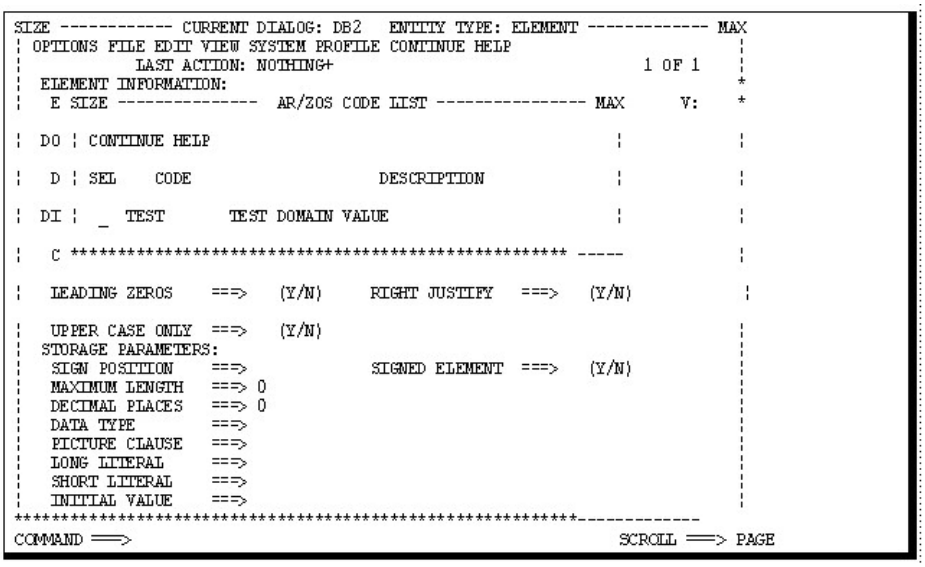
  Entering a question mark (?) in the key domain attribute displays a list of valid keys. You can then directly select a key from this list. The EDIT.DOMAIN command generates a similar list.

The following example shows the use of a domain key. The Edit window for the ELEMENT entity type (part of the Repository DDL Manager) is shown in the following illustration. It was scrolled to reveal the attribute DOMAIN KEY. This literal identifies this attribute as the domain key for the ELEMENT entity type.

Selecting a domain key from a code list only fills in blank domain values. Executing the EDIT.DOMAIN command, on the other hand, completes all the domain fields, overwriting any data already in the fields.

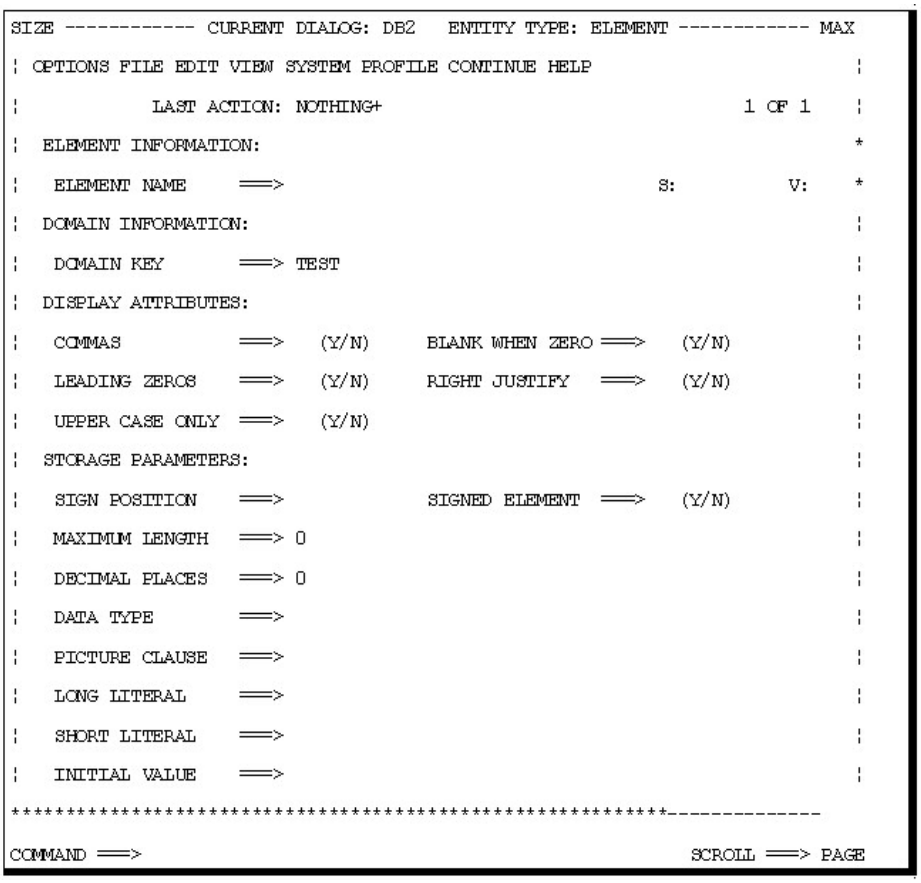
To view a list of domains

1. Open an Edit window for the entity type you want to use.
2. Enter **?** in the attribute input field of the domain key and press Enter. A list of available domain sets for the entity type you are viewing appears.



Note: To see a detailed description of a domain, type **S** in the Select byte next to the domain, move the cursor to HELP, and press Enter.

3. Enter **S** in the Select byte next to the domain key you want to use and press Enter. The Repository displays the values defined for the domain attributes are entered by the system.



4. Alter domain values.
5. Select EDIT.UPDATE to save the changes to the repository.

Reset Domain Values

If you make changes to values in the entity type you are working with and you want to reset them to the original domain values

1. Make sure the name of the domain is in the Domain Key field.
2. Select EDIT.DOMAIN. The domain values are reset.

Extended Text

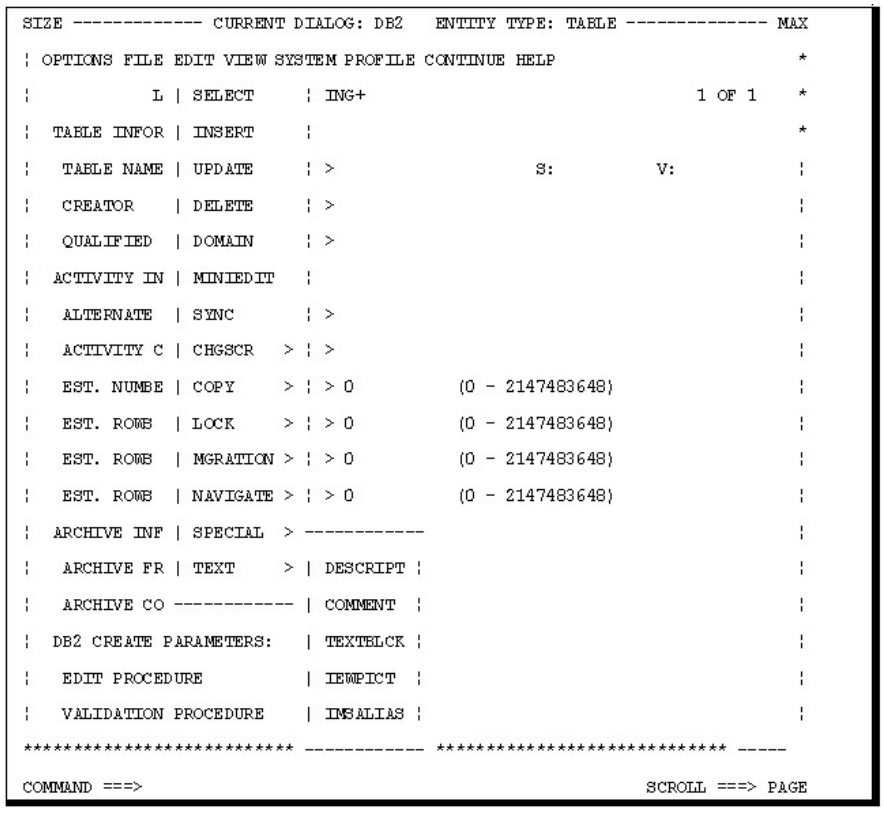
You can use the Extended Text feature to add lengthy free-form textual information to entities and relationships. Extended text is essentially an attribute type, so it may not be a part of every entity type. Your Repository Administrator determines which entity types can have extended text.

Add Extended Text

You can add extended text to an entity from either the horizontal or vertical Edit window mode. When using the horizontal mode, however, you must tag the entity to which the text is added before accessing the TEXT function. If multiple entities are tagged, a separate Text panel is displayed for each. You cannot add the same block of text to multiple entities simultaneously.

To add extended text to an entity

1. Open an Edit window for the entity type you want to use and tag the entity instance to which you want to add text.



2. Select EDIT.TEXT**.** A cascading pull-down menu displays the available extended text types.

   Note: The options in the TEXT pull-down menu corresponds to each of the available text types, if any, for the currently selected entity type. An entity type can have up to five types, each of which allows the user to add a different set of text. The names of each of these types are site-specific and may vary considerably from shown in the example.

3. Move your cursor to the text type you want to use and press Enter. An ISPF Edit panel appears.

   The Name, Status and Version attributes are displayed in a message line in the editor. If text already exists then the user ID and timestamp of the last update also appears in a message line.

4. Enter the text to add. The standard and line commands are available while in the Text panel to aid in the manipulation of text. The amount of text you can add to the entity through each text type (the file size) is determined by the limitations of the ISPF text editor.

5. Do one of the following:

   - Press F3 to exit the text panel and save the extended text.
   - Press F4 to exit the text panel without saving the extended text.

Copy Extended Text

The extended text attribute is not automatically copied when new entities are created from existing entities by means of the horizontal edit window line commands.

The GOTO and JUMP Commands

Use the GOTO and JUMP commands to quickly move between related entity types. These commands are very useful when creating or updating relationships because you can use them to access and view subordinate entities.

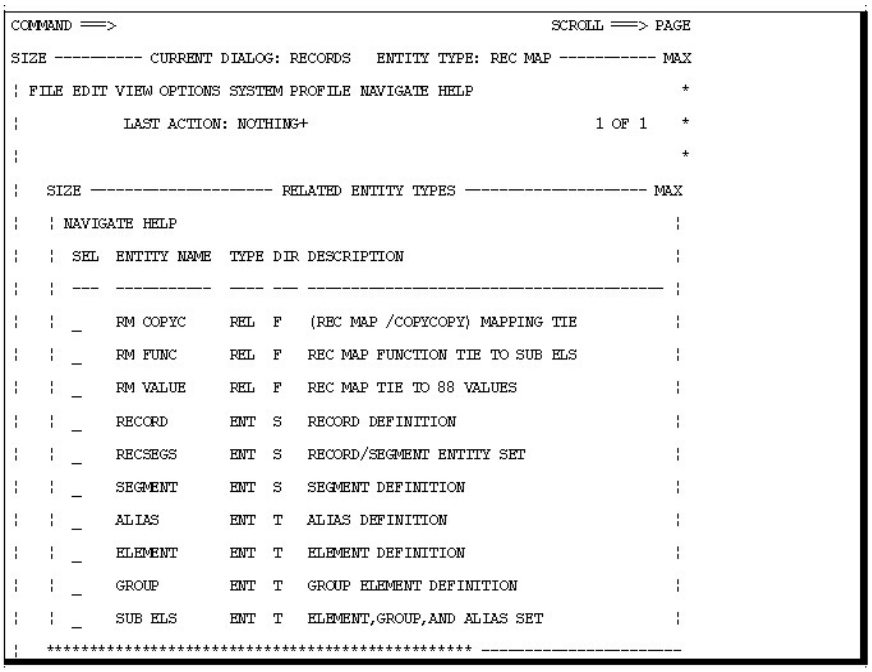
The GOTO Command

The GOTO command displays only those entity types that are related or associated to the current entity type. This greatly reduces the number of entity types that you see when you use the VIEW.TYPE command.

You can use the GOTO command to automatically select and display any existing entities of the new entity type that are related to entities of the initial type. In the illustration in the following section, the REC MAP entity type is displayed in the Edit window. The GOTO command opens the Related Entity Types window.

The Related Entity Types Window

The Related Entity Types window contains a list of all the entity types that use or are used by the current entity type. The following illustration shows the Related Entity Type window.



The Related Entity Types window contains four columns of information that is used to identify related entities. Column descriptions are as follows:

Column	Description
Entity Name	The entity, relationship, or association type that is linked to the currently selected type.
Type	How the component is linked to the current entity type: ■ ENT indicates entity ■ ASN indicates it is related by an association ■ REL indicates it is related by a relationship
Dir	The direction in which the relationship or association links the current entity type. When the current entity type is a:

Column	Description
	Standard entity type or a relationship type: ■ B means the component uses the entity as its target ■ F means the current entity type is the source Relationship or association type: ■ T means the component uses the entity as its target ■ S means the current entity type is the source An entity can be used as both a source and a target. If this is the case, the entity is listed twice, once each with a B and F (or S and T) flag.
Description	A brief description of the entity.

GOTO in the Horizontal Mode

When you change entity types using the GOTO command, the horizontal mode of the Edit window for association and relationship types may appear somewhat different than the standard Edit window.

This occurs when there are no instances within the entity type you selected that are related to instance of the initial entity type. The title lines include the name, status, and version of the selected COMPANY entity from the previous Edit window. This indicates that *this* association Edit window is used exclusively for adding associations, which use the specified entity as the source. Accordingly, input fields are provided only for the target entity.

GOTO in the Vertical Mode

If you are using an Edit window in vertical mode and GOTO finds more than one instance that is linked to the current entity, use the PREV and NEXT commands to view the other instances.

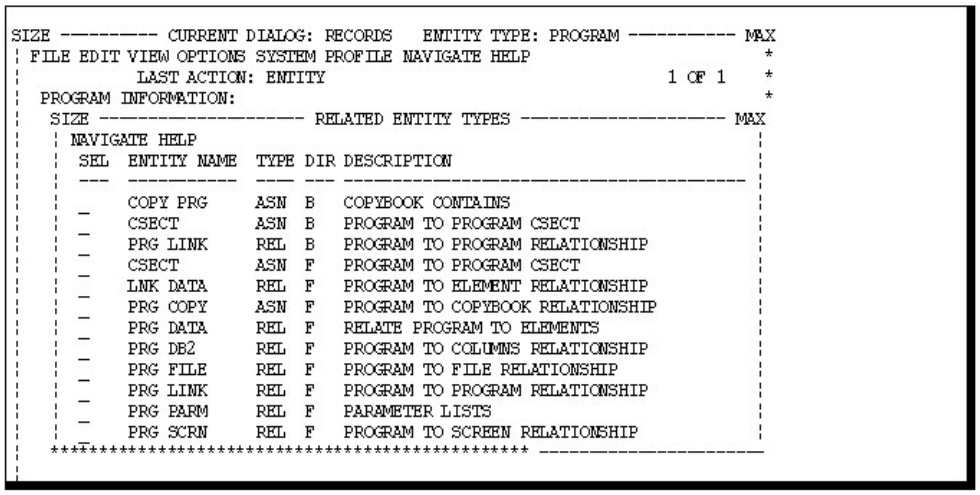
GOTO with Entity Sets

If you are using the GOTO command from an entity set, it is possible to tag entities from several different entity types. Some of these entity types may not be related to the entity type you are *going to*. Entities that belong to entity types that are unrelated to the new type are ignored during processing.

Forward and Backward Linkages

The vertical Edit window the Dir column shows you whether the entity is a source or a target of the selected entity.

For example, in the following illustration, the RELATED ENTITY TYPES box shows all of the entity types that could potentially be associated with the program that is currently selected.



A **B** in the Dir column indicates a program, copybook, or CSECT that calls the current program.

An **F** in the Dir column indicates an object that is called by the current program.

The GOTO Command

You can use the GOTO command in either vertical or horizontal mode. Use the following procedures to find related entities using the GOTO command.

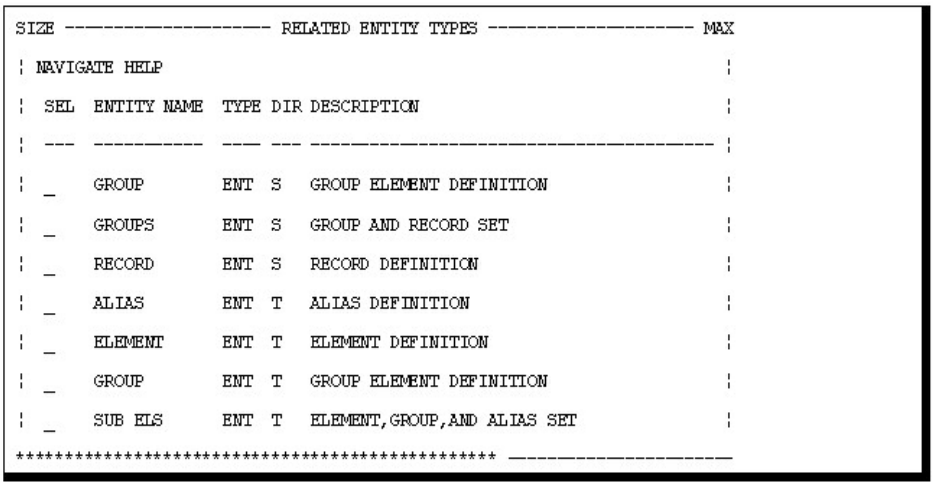
When you use the GOTO command in horizontal mode, you can select specific entities for processing by typing an **S** in the Select byte. If no entities are tagged prior to executing the GOTO command, the first entity listed (row 1) is used by default.

Note: Having a copy of the meta-model handy makes navigating with the GOTO command significantly easier.

To use GOTO to see related entities

1. Open the Edit window for the entity type for which you want to find related instances.
2. Do one of the following:
   - If you are using horizontal mode, enter an **S** on the Select byte for the entity

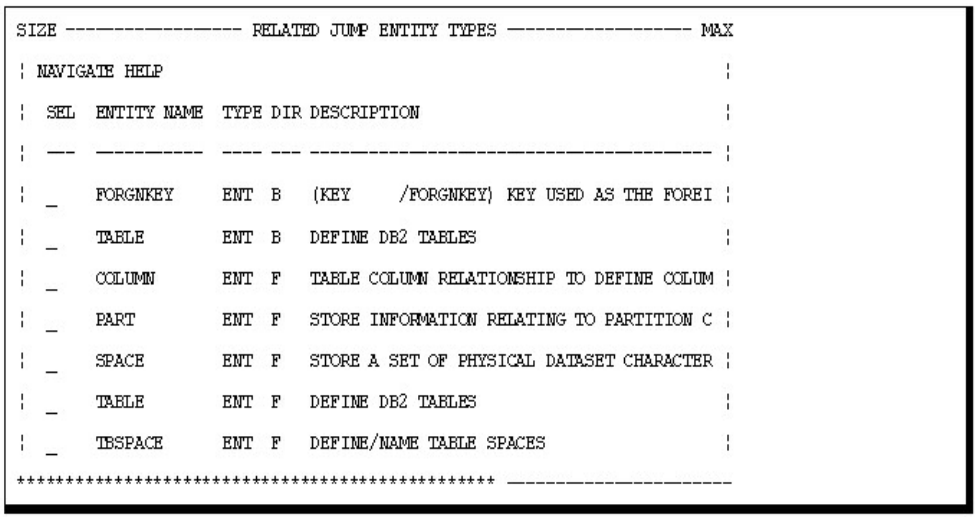
     Note: If no entities were tagged prior to executing the GOTO command, the new Edit window contains entities related to the entity in the first row of the Edit window.
   - If you are using vertical mode, go onto the next step
3. Select EDIT.NAVIGATE.GOTO. The Related Entity Types window appears.



4. Enter **S** in the Select byte of the related entity you want to see and press Enter. An Edit window appears with the entity you selected.

The JUMP Command

The JUMP command is very similar to the GOTO command. However, rather than switching to an entity type that is directly linked to your current entity type, you access entity types linked to your current entity type through a relationship or association type. This difference is shown in the following illustration.



Like the GOTO command, the JUMP command enables you to switch to a specific group of entities.

If any entities are selected (tagged) prior to performing the jump, the new Edit window contains any entities of the new type that are related to the tagged entities through intervening relationships and associations. For example, if you were jumping from the TABLE entity type to the ELEMENT entity type and you tagged one or more TABLE entities before executing the JUMP command, the Edit window for the ELEMENT entity type would contain any ELEMENT entities that were related to the tagged TABLE entities through COLUMN relationships.

Use the JUMP Command

To move to entity types that are related to the entity you have currently selected

1. Open the Edit window for the entity type for which you want to find related entities.
2. Do one of the following:
   - If you are using horizontal mode, enter an **S** on the Select byte for the entity. If no entities are tagged prior to executing the GOTO command, the new Edit window contains entities related to the entity in the first row of the Edit window
   - If you are using vertical mode, go to the next step.
3. Select EDIT.NAVIGATE.JUMP. The Related Jump Entity Types window appears.

The MERGE Command

You use the MERGE (EDIT.SPECIAL.MERGE) command to combine two entities of the same entity type into a single entity. The entity that results from such a merger are used for the source or target requirements of any relationships or associations that referenced either of the initial two entities.

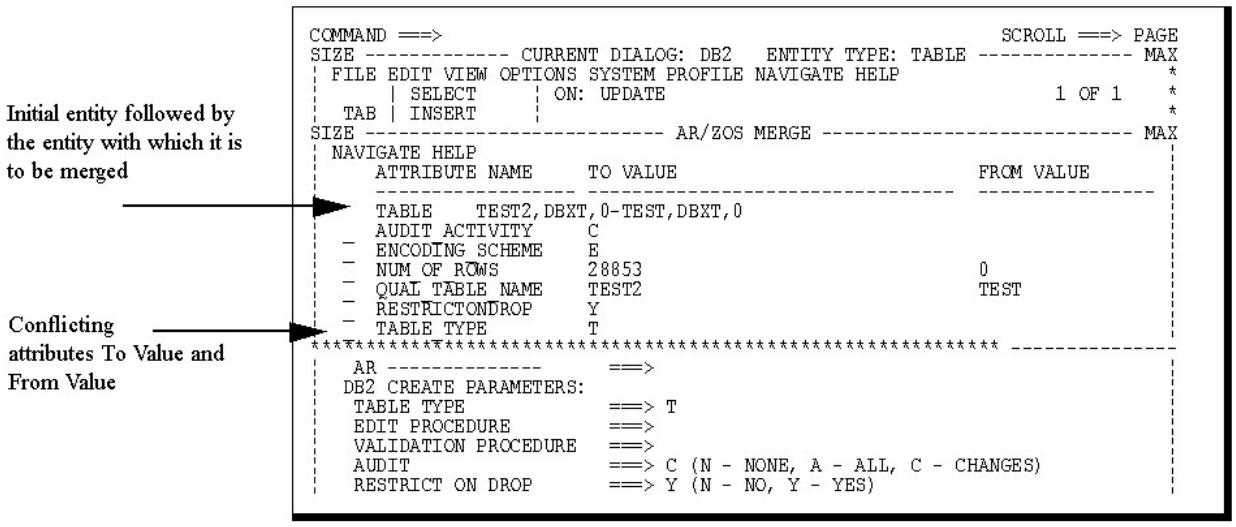
Typically, you use the MERGE command when you have two entities that are identical and you want to eliminate redundancies. Even in these situations, there may be some differences in attribute values of the two entities being merged. In these cases, you can specify which of the attributes of original two entities you want to retain in the new entity.

The Merge Window

The Merge window contains several rows corresponding to the attribute differences between the two merged entities. The left-most column contains the name of the attribute type, the second and third columns contain the actual attribute values for each of the merged entities.

Unless otherwise specified, the attributes of the entity you selected from the Edit window prior to executing the MERGE command is retained in the new entity. This entity and its attributes are found to the left under the To Value column.

The following is a sample Merge window.



To merge two entities

1. Open an Edit window for one of the two entities you want to merge.
2. Select EDIT.SPECIAL.MERGE. A list of the all entities of the current entity type appears.
3. Enter **S** in the Select byte of the entity with which you want to merge and press Enter. The Merge window appears.
4. Verify the changes in the Merge window. If you want the attributes of the second entity kept, enter **S** in the Select byte of the appropriate attribute type and press Enter. The Edit window appears.
5. Reselect the new entity to see the changes resulting from the merge.

The MINIEDIT Command

You use the MINIEDIT command to access another edit session for a different entity type, without exiting your current session. This command is available in the following places:

- Edit windows
- List windows
- Impact Analysis windows

You can use the Miniedit window to select, insert, update, and delete entities in the same manner as the Edit window.

To access the Miniedit window

1. Select EDIT.MINIEDIT. A list of available entities appears.
2. Enter **S** in the Select byte of the entity you want to use and press Enter. The Miniedit window appears.
3. Type over the information you want to add or modify.
4. Press F3 to close the Miniedit window. The window from which you executed the MINIEDIT command re-appears displaying any changes you made.

The following sections describe how the Miniedit window works with Edit, List, and Impact Analysis windows.

Use MINIEDIT from the Edit Window

When you are executing the MINIEDIT command from the Edit window, a list of entity types for the current dialog appears. Once you select the entity type you want to edit, a Miniedit window for that type is displayed. When you exit the Miniedit window the original Edit window appears.

Use MINIEDIT from List Windows

The MINIEDIT command in a List window is intended to be used only with entities listed in the List window. You need to tag specific entities in the List window before you execute the MINIEDIT command. The selected entities are displayed in a Miniedit window where they can be edited. When you exit the MINIEDIT window the original List window is re-appears.

Note: If multiple entities are selected from the List window for MINIEDIT processing, the END command takes you from one entity to the next. When the last entity is processed, control is returned to the List window.

Use MINIEDIT from Impact Analysis Windows

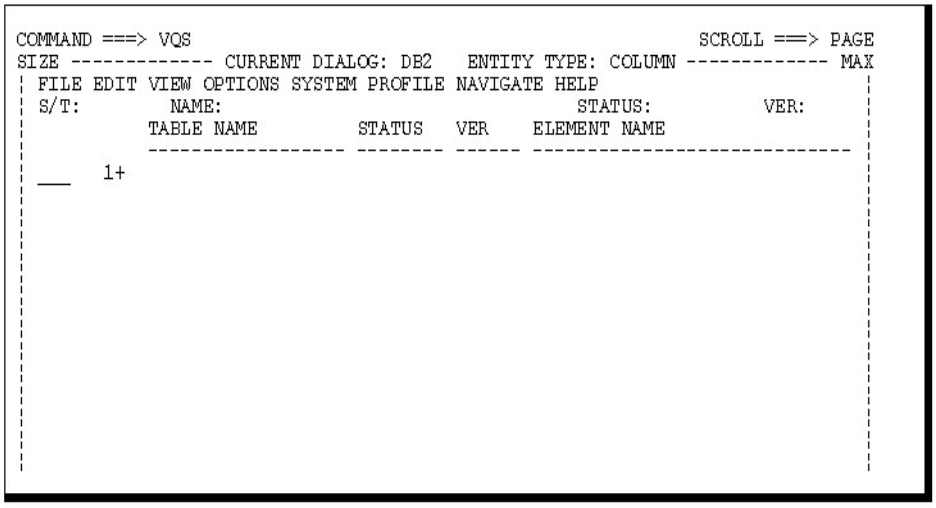
Impact Analysis windows, like List windows, contain lists of entities. You can edit these entities using the MINIEDIT command in the same manner as List window entities.

The QUEUE Command

The Repository entity queue is a temporary storage location that allows you memorize individual entities while navigating through the repository. Once added to the queue, an entity can be retrieved during the course of an edit session. The entity queue is maintained and manipulated through the commands found under VIEW.QUEUE. These commands include:

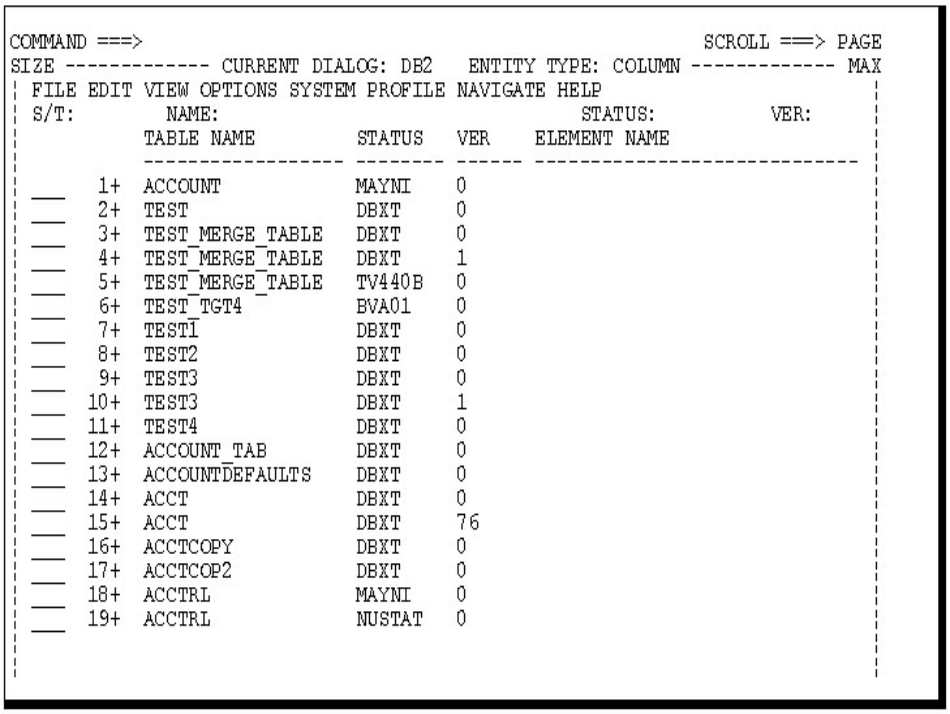
Command	Description
VIEW.QUEUE.ON	Starts the queue. When the queue is turned on, any entities, relationships, or associations that are displayed on the screen are added to the queue (up to 10,000). ■ The Repository continues to add subsequently displayed entities until the queue is turned off ■ Once added to the queue, entities can be recalled as needed using the ENTITY, SOURCE, and TARGET commands See the following commands.
VIEW.QUEUE.OFF	Stops the queue. The Repository no longer saves entities that appear on the screen.
VIEW.QUEUE.CLEAR	Removes all entities previously added to the queue.
VIEW.QUEUE.ENTITY	Clears the current screen and retrieves all entities from the queue that belong to the currently selected entity type (or entity set).
VIEW.QUEUE.SOURCE	Used on relationship or association type Edit windows. Clears all data from the Edit window and retrieves any entities from the queue that belong to the source entity type for the currently selected relationship or association type.
VIEW.QUEUE.TARGET	Used on relationship or association type Edit windows. Clears all data from the Edit window and retrieves any entities from the queue that belong the target entity type for the currently selected relationship or association type.

The following windows show how the entity queue can be used to retrieve source entities. This one shows an empty relationship window.



Executing VIEW.QUEUE.SOURCE from this window retrieves from the queue any entities belonging to the SOURCE entity type as shown in the following window.

Rows are added as needed to accommodate all of the matching entities in the queue.



The REPLACE Command

Use the REPLACE command to replace one entity with another. This command deletes the replaced entity. Any relationships and associations that used the replaced entity as a target are updated so that they reference the replacement entity.

To use the REPLACE command

1. Open an Edit window for the entity you want to replace.
2. Select EDIT.SPECIAL.REPLACE. A list of entities that can be used to replace the entity you selected appears.
3. Enter an **S** next to the Select byte of the entity you want to use and press Enter. The Edit window re-appears with the new entity marked as a pending insert.
4. Enter **EDIT.UPDATE** to save the changes in the repository. The new entity is updated in the repository and any relationships and associations that used the replaced entity as a source are deleted.

Name Generation

The Repository uses the ELEMENT entity type to define data elements, the basic unit of data in the repository. The ELEMENT entity type has attribute types that allow you to specify different names for the element, such as a COBOL name, an Assembler name, and a DB2 column name. These different names are used to identify the data element when you are generating DDL (Data Description Language) or scanning code to populate the repository.

To help create and enforce site-wide naming standards, The Repository provides a Name Generation Utility. Using an element Name attribute and a site-defined list of standard abbreviations, the Name Generation Utility creates Assembler, COBOL, and DB2 column names for a data element.

How It Works

When you use the Name Generation Utility to generate a Business name for the element, the Repository populates the Business name attribute with the Glossary Item Names of occurrences having abbreviations that match the tokens of the data element COBOL name field.

A token is defined as any substring of a data element name that is delimited by a space, hyphen (-) or an underscore (_). When using the Name Generation Utility to generate COBOL, Assembler, or DB2 names of a data element, the Repository searches the Business name attribute for matches in the Glossary Item Name attribute of the GLOSSARY entity type.

Alternatively, the Name Generation Utility can display a list of elements where the name contains the tokens from the glossary that are based on the Business name.

Note: The performance of the Name Generation Utility is dependent on the size of the Elements Table.

Installation Parameters

The Repository Administrator determines the parameters for the Name Generation Utility during the installation of the Repository. These parameters include:

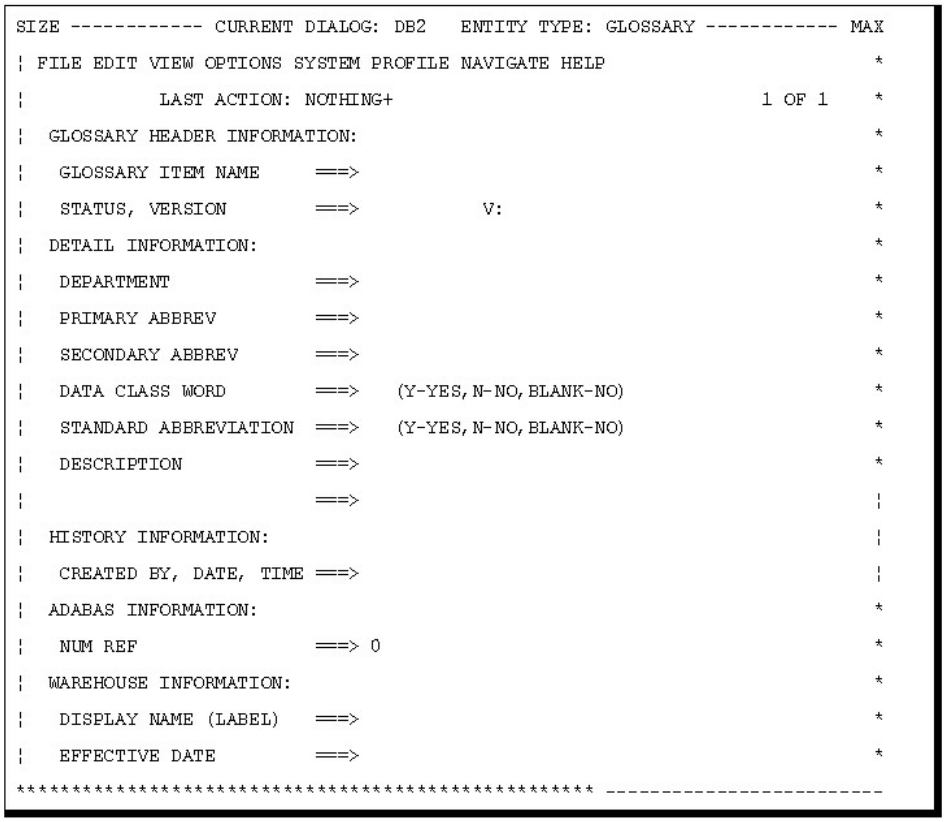
- Whether to generate a DB2 column name
- Whether to generate an Assembler name
- Whether class words are positioned as prefixes or suffixes within the generated names
- Default delimiters for each generated name
- Whether to display a list of matching elements or to generate a new name

The Relational Translator

If you are using the Repository Relational Translator, you can use the Name Generation Utility to transform the logical attribute name field, on the ATTR TP entity type, into data element names. The Relational Translator transforms logical models into first-cut DB2 models.

Standard Abbreviations

The repository stores standard abbreviations as occurrences of the GLOSSARY entity type. This arrangement gives you the capability of using the Repository to add to and maintain the list of standard abbreviations for your site.



Each GLOSSARY entity has the following attributes:

Attribute	Description
Glossary Item Name	The Name attribute of the GLOSSARY entity type. Also specifies the word which is assigned as a standard abbreviation.
Status, Version	Use to specify a status and version for each GLOSSARY entity. This information is used only for identification purposes in the Repository and does not pertain to the abbreviation itself.
Department	Use to specify the department within your organization that is most closely associated with the abbreviation. This is an optional field and will not affect the name generation.
Primary Abbreviation	Stores the abbreviation of the word stored in the Glossary Item Name attribute. If a word in Glossary Item Name field is found in the data element Business name attribute during name

Attribute	Description
	generation, the abbreviation in this field is used for the COBOL name, DB2 column name, and the Assembler name.
Secondary Abbreviation	Specifies the abbreviation to be used when the Glossary Item Name is deemed a data class word. Whether the abbreviation is used as a prefix or a suffix is determined by input to the parameters specified during installation.
Data Class Word	Specifies how you want to deal with the Glossary Item. Valid options are: ■ Y-Specify the glossary item as a data class word ■ N-Do not specify the Glossary Item as a data class word During name generation, the Glossary Item is placed as a prefix or a suffix, depending on parameters set during the Repository installation.
Standard Abbreviation	Optional attribute field that specifies whether to designate the abbreviation as a standard abbreviation.
Description	Stores a description of the Glossary Item.

Sample Glossary Data

The SAMPGLOS member of the Repository loads sample glossary data into the GLOSSARY entity type. You can customize this syntax to loading your own standard business abbreviations

Use Business Name to Generate Other Field Names

The Repository data elements are defined using four classifications of names:

- A 32-byte COBOL name
- A 128-byte Business name
- An 8-byte Assembler name
- A 30-byte DB2 column name

Note: The COBOL name is also known as the ELEMENT name, ALIAS name, and GROUP name for each of the different data ELEMENT entity types.

When you generate Assembler, DB2, and COBOL names from a Business name, the total length of the generated name often exceeds the total length of the name field. The Name Generation Utility automatically removes characters from the Assembler, DB2, and COBOL names in order for the generated name to fit within the length of the appropriate name field. This situation occurs most frequently when generating Assembler names.

If the Business name is specified and the COBOL, DB2, and the Assembler names are blank, then the Business name is used to generate a COBOL name, a DB2 column name, and an Assembler name.

Search for String Matches

When you generate a COBOL name from the Business name, the Name Generation Utility first searches for string matches on multiple word phrases, then for matches of individual words.

For example, if the Business Name of a data element is: Health Maintenance Organization, the Name Generation Utility returns an abbreviation of *HMO* even though each of the words are defined in the glossary with an abbreviation. The abbreviations stored in the glossary for this example are:

```
HEALTH = HLTH
MAINTENANCE = MAINT
ORGANIZATION = ORG
HEALTH MAINTENANCE ORGANIZATION = HMO
```

Because it searches for multiple word phrase matches before single word matches, the Name Generation Utility generates the appropriate abbreviation: HMO.

Use COBOL Name to Generate Field Names

If you specify the COBOL name of a data element and the Business name is blank, then the Name Generation Utility generates a Business Name. The Repository also uses the COBOL name as the key attribute of the data element entity types: ELEMENT, ALIAS, and GROUP.

If you enter a COBOL name and you want to generate Assembler and DB2 column names, you need run the Name Generation Utility twice. Once you generate the Business name, you can use it to generate Assembler and DB2 column names.

Data Class Words

The GLOSSARY entity type is designed to let you choose certain words as data class words. Parameters are used to control the way in which the Name Generation Utility handles glossary abbreviations when a Glossary Item match is found. The Name Generation Utility provides the Repository Administrator with two options for the sequence of a class word within the COBOL, DB2 and Assembler name attributes: as a prefix or as a suffix. As mentioned earlier, the specifications in the Repository installation panel determine whether the Prefix is used before or Suffix attribute is used after other abbreviations.

If, for example, you were to generate names from the data element Business name:

```
SAMPLE CUSTOMER NUMBER
```

The data element name generated from this might be:

```
SAMP-CUST-NUM
```

If, during installation, you specified that data class words were to be used as prefixes, and the Number Glossary Item had an abbreviation specified in the Prefix or Suffix attribute and was flagged a data class word by the Data Class Word attribute, then generated COBOL name would be:

```
NUM-CUST-SAMP
```

In this way, all data elements are defined as numbers (that is, numeric) could be grouped together logically by their Data Element names.

Note: If two or more class words are found in a Business name, only the first class word encountered is treated by the Name Generation Utility as a class word.

Customize Name Generation

Because of the differences in naming standards among sites, the Name Generation algorithm for building COBOL, DB2, and Assembler names may not suit the needs of a company. Therefore, the source code for the Name Generation Utility is provided at installation time in the SAMPPGM data set of the Repository libraries. The program is written in COBOL and consists of a main program, DBXNAME, with two subroutines, COBBTOE and COBETOB. Contact your Repository Administrator if you feel that the Name Generation Utility needs to be customized for your site's naming standards.

Note: The Name Generation Utility is an excellent example of a Repository command that calls an exit program to process repository data. It is recommended that the Repository Administrator use the Name Generation Utility when designing program exits for the Repository. A detailed discussion of extending the repository to add commands and program exits is located in the *Administration Guide*.

The Name Generation Utility

If both the Business name and the COBOL name exist, then executing the Name Generation Utility has no effect.

To start the Name Generation Utility

1. Open a blank Edit window for one of the following entity types:
   - ELEMENT
   - ALIAS
   - GROUP
2. Do one of the following:
   - Enter a name in the Business name field
   - Enter a name in the COBOL name field.
3. Enter OPTIONS.NAME. New names are generated by the system.

If you entered:	Names are generated for:
Business name	COBOL name
Assembler name	DB2 column name
COBOL name	Business name

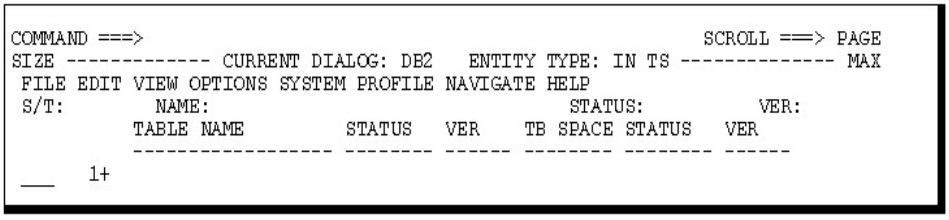
Note: The Name Generation Utility only inserts names into blank name fields. Any names already in the names fields are left intact when the Name Generation Utility is executed.

4. If you entered a COBOL name and you want to generate Assembler and DB2 column names, enter OPTIONS.NAME again.

On-Screen Ties

The Repository supports a special type of map that enables users to create relationships and associations between entities without opening a relationship or association Edit window. These maps contain input fields corresponding to two or more entity types. When a user inserts or updates an occurrence using this type of map, the Repository automatically creates the relationships or associations between the entities identified by these fields.

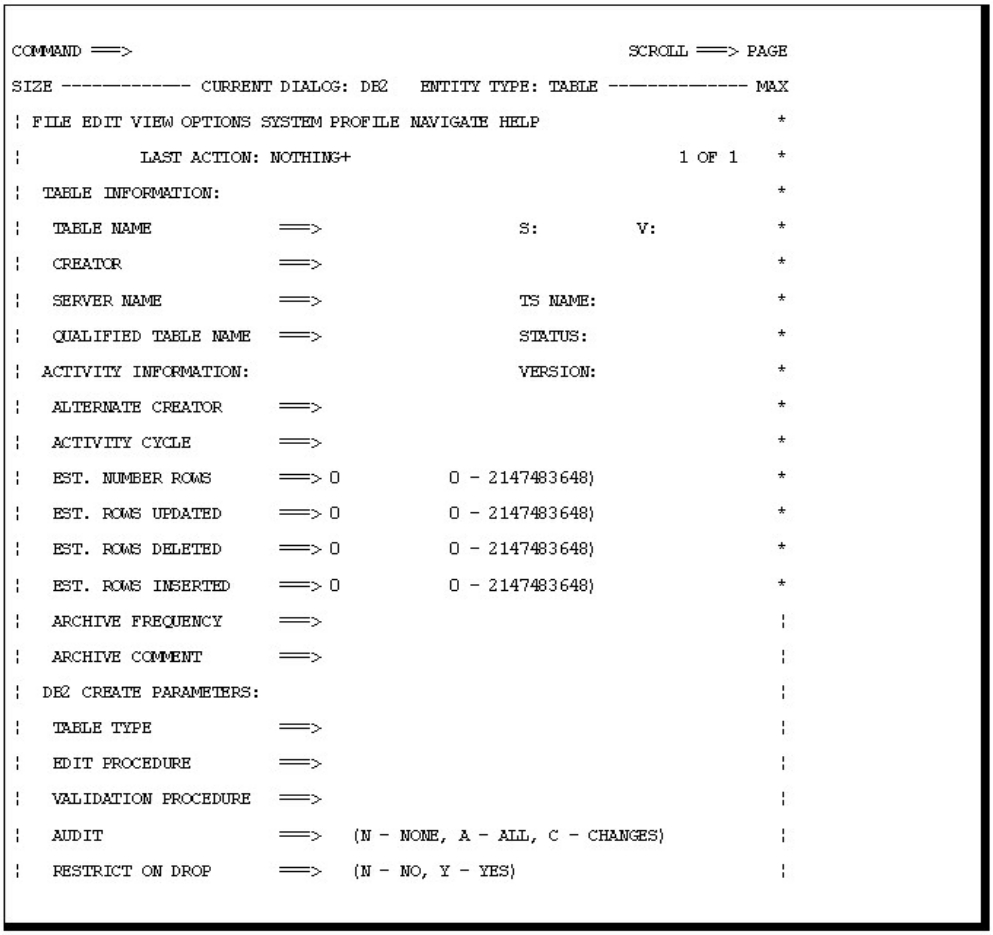
The following illustration shows two entity types from the DB2 model. In this example, the TABLE entity type is connected to the TB SPACE entity type through the IN TS association type. Using standard repository editing methods described in "Working with Entities," you would have to access the IN TS association type (map 110) to make connections between occurrences of these two entity types.



With on-screen ties capability, it is now possible to have an Edit window (map) for the TABLE entity type that has input fields for the name, status, and version of a TBSPACE entity. When you insert or update a TABLE entity with this map, the Repository automatically creates an association to the TBSPACE occurrence.

An example of how such an Edit window might appear for the TABLE entity type is shown in the following screen. This sample screen is for demonstration purposes only.

Input fields for referencing a TBSPACE entity.



Note: None of the *default* entity types included with the Repository uses on-screen ties. This feature must be set up and maintained by your Repository Administrator. The screen represented in the previous illustration is for demonstration purposes only.

Although the Repository creates associations and relationships between occurrences, it does not insert occurrences of the related types. The sample map described above, for example, could insert IN TS associations, but it could not insert TBSPACE entities.

If the name, status, and version specified for this type does not correspond to an existing occurrence, an error message appears when you try to insert or update the TABLE entity.

To make selecting an existing occurrence simpler, on-screen ties fields are supported by online help. Simply enter a question mark into the appropriate Name field, and a list of existing entities is displayed. This list can be qualified by entering a partial name, a status, or a version.

Because of the conflicts that would result if an entity were tied to multiple occurrences of the same entity type, on-screen ties are only possible between entity types that are limited to *many-to-one* ties between their respective occurrences. That is, occurrences of an entity type that has on-screen ties to another entity type cannot be related to more than one occurrence of that particular entity type.

Update Associations and Relationships with On-Screen Ties

When you retrieve an occurrence to an Edit window that contains on-screen ties, the attributes of the related entity automatically appear (if the selected entity is related to another entity of that type). If you want to tie the current entity to a different occurrence, change the Name, Status, and Version in the on-screen ties fields. The name you specify must be an existing occurrence. When the current entity is subsequently updated, the Repository deletes the association to the original occurrence and creates a new association to the occurrence you specified.

Path Delete

The Repository paths are conceptual collections of steps or navigational movements that *progress* from one meta-entity type to another across the associations or relationships that connect the types. Paths are defined by the Repository Administrator.

You can use the Path Delete Facility to remove a root occurrence and all the occurrences in its *path*. Before occurrences are removed from the repository, they are analyzed to make sure they meet the qualifications specified in the path definition.

For example, an entire DB2 database definition can be removed from the repository by a path that runs from the following entity types:

DATABASE - TBSPACE - TABLE - KEY - COLUMN

To delete a root and its path using the Path Delete Facility

1. Open an Edit window for the root occurrence you want to delete.
2. Select EDIT.SPECIAL.PATHDEL. A list of path names for the root occurrence appears.
3. Enter **S** in the Select byte of the path you want to delete and press Enter. The JCL is generated.
4. Submit the job.

Chapter 9: Defining User Profiles

This chapter explains how you can use the Repository User Profiles feature to customize commands, keyboard layouts, and settings for your own use.

This section contains the following topics:

Customize Commands

Though most users new to the Repository use the menu bar and related pull-down menus to process entities, you may find it faster and easier to use the command line.

For example, inserting an entity into the repository by means of the menu bar requires a user to first select EDIT from the menu bar and then to select INSERT from the EDIT pull-down menu. These two steps can be combined into one by entering the command string EDIT.INSERT at the command line. This command can be further shortened by creating a fast command, such as EI, that will perform the same actions. The Repository gives you the ability to customize fast commands using the Repository User Profile feature.

The User Profile Window

The User Profile feature is tied to your TSO logon ID, so these commands are unique to you. You can create your own commands, as you will be the only one using them.

Because each window has its own set of commands, each also has it own command profile. PROFILE is always one of the menu bar option, regardless of which window you open.

To open the User Profile

1. Open the window whose profile you want to edit.
2. Select PROFILE.COMMANDS. The User Commands window appears.

The User Commands window consists of six columns and a row for each command. The following provides a description of each column.

Column	Description
User Command	Defines the user-specific fast commands.
Global Command	Contains the system-wide fast commands. The Repository Administrator creates these commands.
Window	The name of the window on which the command is displayed.
Function	The name of the menu item on the window under which the command is displayed.
Subfunction	The name on the drop-down menu on which the command is displayed.
Command	The name of the command.

Fast Command Restrictions

The following restrictions apply when creating your own fast commands:

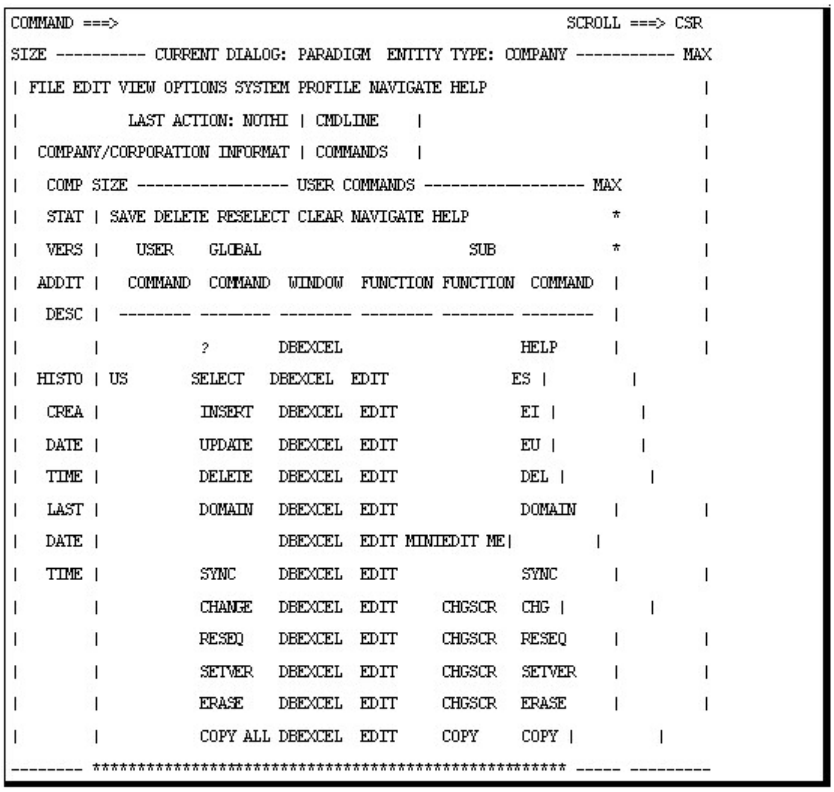
- You cannot create a user command that has the same name as a standard commands.
- You cannot use the same fast command for two or more commands.
- You cannot create a fast command that is the same as a global fast command. See the Global Commands column of the User Commands window for a list of Global Commands.

Create Customized Commands

To add or edit your own fast command

1. Open the User Profile window for the command for which you want to add a fast command.
2. Select PROFILE.COMMANDS. A User Commands window appears.
3. Move the cursor down the User Command column until it is on the row for which you want to add or edit a fast command. For example, if you want to add a fast command for EDIT.SELECT, you would move the cursor to the row in which EDIT is the function and the COMMAND is SELECT.

4. Enter the letters you want to use for your new fast command in the User Command column. For example: US



5. Repeat Steps 2 - 4 for other fast commands you want to create or edit.
6. Select SAVE. A message window appears with stating the commands were successfully saved. Click OK to close the window.
7. Exit the User Commands window. Your new fast commands are available for you to use.

Note: The menu bar and standard commands are not changed and are still usable after you create a fast command.

Customize Function Keys

Another way to facilitate use of the command line is through the function keys. Through ISPF, you can assign certain function keys to perform specific commands. These can be standard commands, The Repository commands, or your User Profile fast commands.

Note: Because the Repository is started as an ISPF *NEWAPPL*, key definitions created while in the Repository are in effect when using the Repository.

To customize your function key definitions

1. Enter **KEYS** on the command line. The ISPF Key Definition screen appears.
2. Move your cursor to the key you want to change and type over the existing command.
3. Exit the Key Definition screen. Your new key definitions are saved.

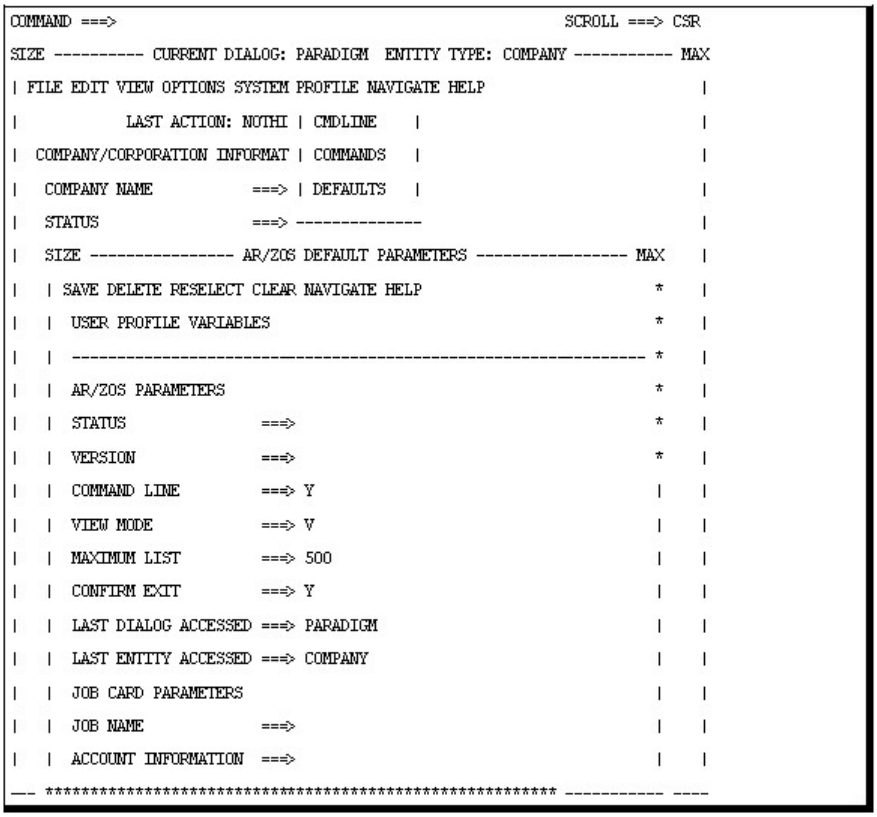
Use the ISPF command **PFSHOW ON** to turn on the key descriptions at the bottom of your screen. The PFSHOW OFF command turns the descriptions off. Before you can turn these descriptions on, you must enter values into one or more of the LABEL fields. These fields are also found on the Key Description screen.

See the IBM's *ISPF Dialog Management Services* for additional information. Be careful not to remove KEY definitions that are used frequently in the Repository (such as F7: scroll up and F8: scroll down).

Customize Default Settings

Another convenient function of the User Profile feature is default setting customization. A default setting is an option that the system uses when the user fails to specify otherwise. You can customize the default values in the Default Profile window.

This example shows the use of the Default Profile window. You can access the Default Profile window from any window in the Repository. The following illustration shows a sample Default Profile window.



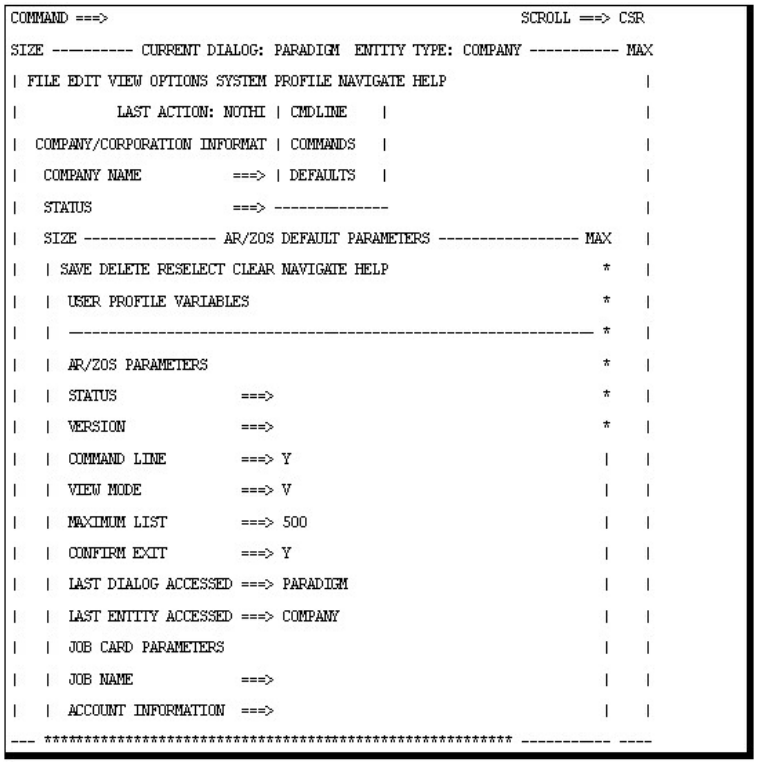
The following fields are found on the Default Profile window.

Field	Description
Status	The default value entered in the Status field of the Edit window.
Version	The default entered the default value in the Version field of the Edit window.
Command line	Specifies whether to display the command line. Valid values are: ■ **Y**-to display the command line. ■ **N**-to not display the command line.
View mode	This field is related to batch job processing. It is analogous to a field in the Print window, which is discussed in "Printing Reports."
Maximum list	Specifies the maximum number of entities that can be

Field	Description
	displayed in a List window or in the horizontal mode of the Edit window. For example, if the maximum list length is set to 200, you will see the first 200 entities within the list. If you scroll to the bottom and want to see more, enter NEXT at the command line. Any rows that you selected from the first list are ignored. Only rows from the current active list are acted upon. If you change the maximum list size, you must exit and re-enter the Repository before the new list size takes effect.
Confirm list	Specifies whether to display a confirmation message each time the user exits the Repository. Valid options are: ■ **Y**-to display the confirmation window before exiting the Repository. ■ **N**-to exit without a confirmation window.
Last dialog accessed	The most recently selected dialog. ■ When you exit the Repository the name of the last dialog you opened is saved. ■ When you re-enter the Repository the last dialog is automatically selected. These are system-generated fields and the user cannot alter the data they contain. This function does not become active until you have saved your user profile at least once.
Last entity accessed	The most recently selected entity. ■ When you exit the Repository the name of the last entity you opened is saved. ■ When you re-enter the Repository the last entity is automatically selected. These are system-generated fields and the user cannot alter the data they contain. This function does not become active until you have saved your user profile at least once.
Job name	The name you assign to the print job.
Account information	The account number you to which you want to charge the job.

To customize default settings

1. Select PROFILE.DEFAULT. The Default Profile window appears.



2. Move the cursor to the field you want to edit.
3. Enter your changes over the current information.
4. Select SAVE. The changes are saved. If you made a change to the Maximum list field, you must exit and restart the Repository before the change takes affect.

Chapter 10: Printing Reports

The Repository Reports Facility produces a number of standard reports. You need to be familiar with the Repository interface, navigation, and terminology to understand the material presented in this chapter.

This section contains the following topics:

Standard Reports

Several reports are provided to extract information on all Repository entity types. These include the following:

- Detail Report
- List Report
- Cross Reference Report
- Name Token Report
- Path Report

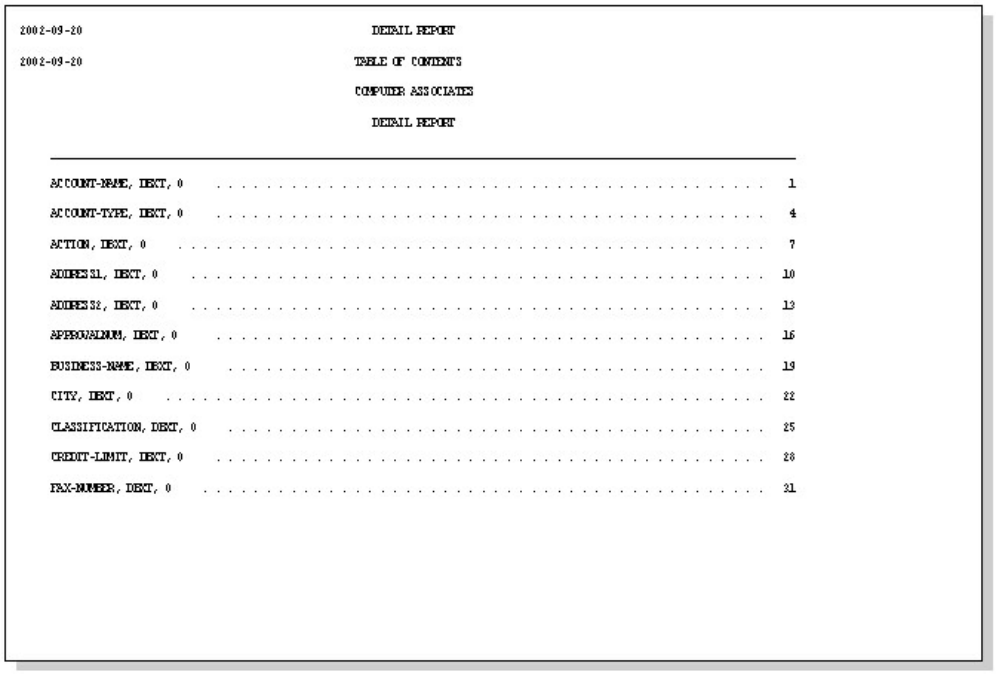
Each standard Repository report begins with a single title page and one or more table of contents pages. You can see a sample of each on the following pages.

Reports that are wider than 133 columns print in sections: the first printed page contains the first 133 columns of the first 60 rows; the second page contains any remaining columns of the first 60 rows; the third page contains the first 133 columns of the second 60 rows, and so on.

The following sections describe the content and purpose of each of these reports.

The information presented in these Repository reports is only representative of the entities, as they exist at the moment, the report was generated.

The following is a sample table of contents.



2002-09-20 DETAIL REPORT

2002-09-20 TABLE OF CONTENTS

COMPUTER ASSOCIATES

DETAIL REPORT

Cross Reference Reports

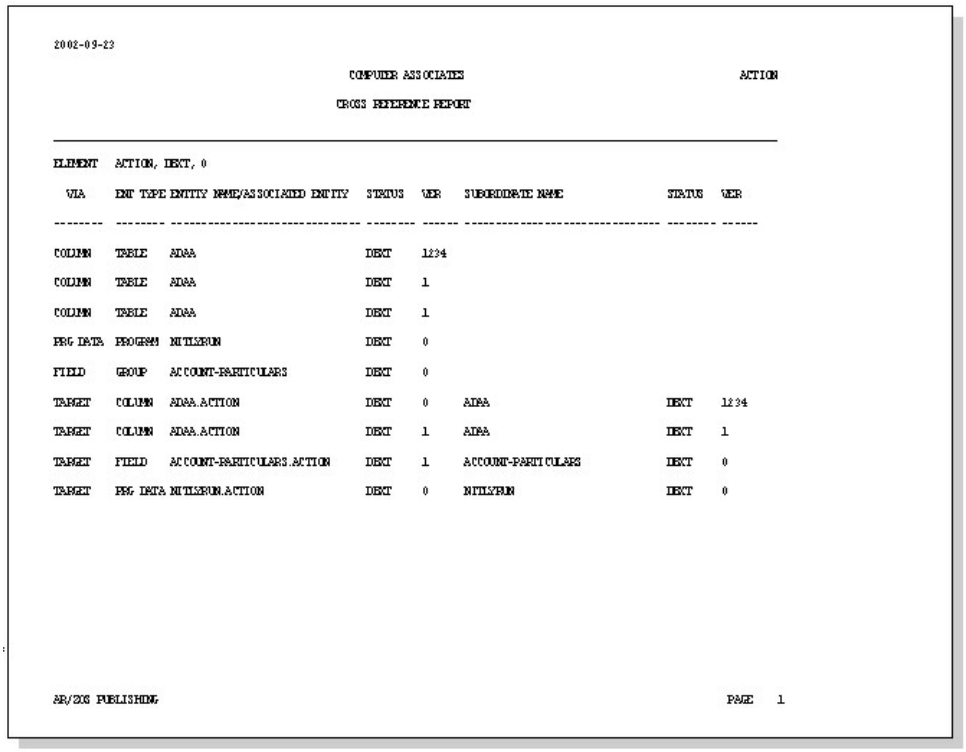
The Cross Reference Report produces a list of entities that use or are used by the entities tagged in the Edit window.

The report is a hardcopy of the information that would be presented in Cross Reference windows for each of the tagged entities. The data provided by the Cross Reference window is helpful when performing data analysis. The report format provides you with enough information to determine the impact entities in the repository have on one another.

Like a Cross Reference window, the Cross Reference Report contains a list of entities, which are related to an entity tagged at the Edit window.

- Each page of the report refers to a single entity, the name of which is displayed in the upper right corner of the page. If a selected entity is related to more entities than can be printed on a single page, the information continues on subsequent pages.
- The columns of the Cross Reference Report are similar to those of the Impact Analysis windows. The exact meaning of each column varies depending on whether a row represents an association, entity, or relationship and whether the selected entity *is* an association, entity, or relationship.

The following is a sample Cross Reference Report.



2002-09-23

COMPUTER ASSOCIATES ACTION

CROSS REFERENCE REPORT

ELEMENT ACTION, DEXT, 0

VIA	ENT TYPE	ENTITY NAME/ASSOCIATED ENTITY	STATUS	VER	SUBORDINATE NAME	STATUS	VER
COLUMN	TABLE	ADAA	DEXT	1234			
COLUMN	TABLE	ADAA	DEXT	1			
COLUMN	TABLE	ADAA	DEXT	1			
PRG DATA	PROGRAM	NITLYRUN	DEXT	0			
FIELD	GROUP	ACCOUNT-PARTICULARS	DEXT	0			
TARGET	COLUMN	ADAA.ACTION	DEXT	0	ADAA	DEXT	1234
TARGET	COLUMN	ADAA.ACTION	DEXT	1	ADAA	DEXT	1
TARGET	FIELD	ACCOUNT-PARTICULARS.ACTION	DEXT	1	ACCOUNT-PARTICULARS	DEXT	0
TARGET	PRG DATA	NITLYRUN.ACTION	DEXT	0	NITLYRUN	DEXT	0

AR/2GS PUBLISHING PAGE 1

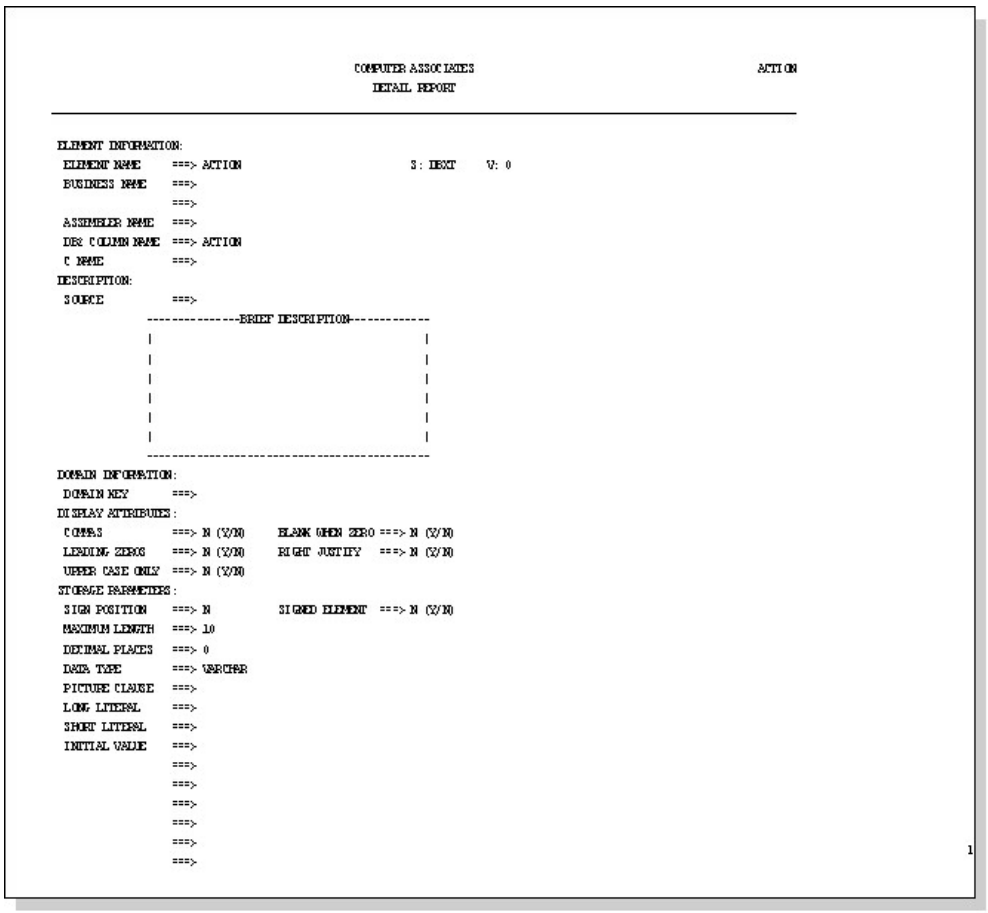
Detail Reports

The Detail Report produces a single page for each of the entities you select in the Edit window. The Detail Report:

- Lists all the attributes for each entity as well as any extended text
- Provides a *hardcopy* of the information displayed in the vertical mode of the Edit window.

In the Print window, you have the option of submitting a batch job to print the report, or displaying the report data on your screen. If you are submitting a report to your screen, you scroll to view all the information.

Each Repository report is 133 columns wide, whereas standard terminals display only 80 columns at a time. The following illustration shows a portion of a sample detail report.



```
                                    COMPUTER ASSOCIATES                                    ACTION
                                      DETAIL REPORT
_____________________________________________________________________________________________

ELEMENT INFORMATION:
 ELEMENT NAME      ===> ACTION                        S: TEXT     V: 0
 BUSINESS NAME     ===>
                   ===>
 ASSEMBLER NAME    ===>
 DB2 COLUMN NAME   ===> ACTION
 C NAME            ===>
DESCRIPTION:
 SOURCE            ===>
             ---------------BRIEF DESCRIPTION------------
             |                                          |
             |                                          |
             |                                          |
             |                                          |
             |                                          |
             |                                          |
             --------------------------------------------
DOMAIN INFORMATION:
 DOMAIN KEY        ===>
DISPLAY ATTRIBUTES:
 COMMAS            ===> N (Y/N)     BLANK WHEN ZERO ===> N (Y/N)
 LEADING ZEROS     ===> N (Y/N)     RIGHT JUSTIFY   ===> N (Y/N)
 UPPER CASE ONLY   ===> N (Y/N)
STORAGE PARAMETERS:
 SIGN POSITION     ===> N           SIGNED ELEMENT  ===> N (Y/N)
 MAXIMUM LENGTH    ===> 10
 DECIMAL PLACES    ===> 0
 DATA TYPE         ===> VARCHAR
 PICTURE CLAUSE    ===>
 LONG LITERAL      ===>
 SHORT LITERAL     ===>
 INITIAL VALUE     ===>
                   ===>
                   ===>
                   ===>
                   ===>
                   ===>
                   ===>                                                                    1
```

Each page of the report represents a single entity that is identified by name in the upper right corner of the page (in this case the entity name is ACTION). Those entities with many attributes may require more than one page.

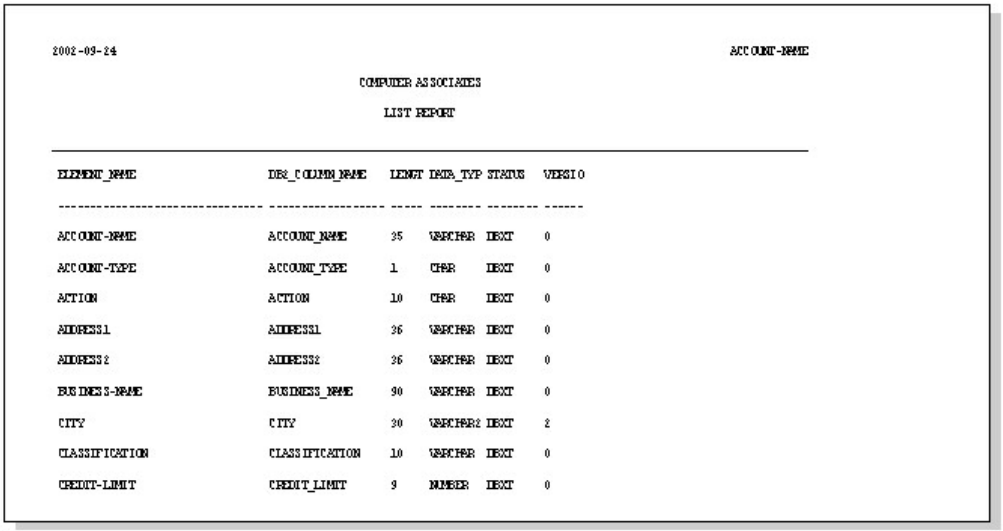
Additional pages are provided for entities that have extended text.

Although the Detail Report only provides a hardcopy of what the Edit and Browse windows already show, it is very helpful when reviewing more than one entity, especially since it does not require scrolling. In addition, by producing a hardcopy, analysts can quickly monitor entire sets of data definitions. The entity Detail Report is essential when creating system design documentation.

List Reports

The List Report is essentially a hardcopy of the List window. It produces a single line for each tagged entity within a specified entity type. The columns currently specified in the Search Criteria window determine which attributes of the selected entities are included on the report.

You can easily scan the List Report for desired information. The List Report provides only the entity-attribute information that the user determined to be relevant, omitting unimportant attributes and greatly reducing the length of the report. The following is a sample List Report.



2002-09-24 ACCOUNT-NAME

COMPUTER ASSOCIATES

LIST REPORT

ELEMENT_NAME	DB2_COLUMN_NAME	LENGT	DATA_TYP	STATUS	VERSIO
ACCOUNT-NAME	ACCOUNT_NAME	35	VARCHAR	DEXT	0
ACCOUNT-TYPE	ACCOUNT_TYPE	1	CHAR	DEXT	0
ACTION	ACTION	10	CHAR	DEXT	0
ADDRESS1	ADDRESS1	36	VARCHAR	DEXT	0
ADDRESS2	ADDRESS2	36	VARCHAR	DEXT	0
BUSINESS-NAME	BUSINESS_NAME	90	VARCHAR	DEXT	0
CITY	CITY	30	VARCHAR2	DEXT	2
CLASSIFICATION	CLASSIFICATION	10	VARCHAR	DEXT	0
CREDIT-LIMIT	CREDIT_LIMIT	9	NUMBER	DEXT	0

The columns on the List Report correspond to the columns of the DB2 tables in which the entities are stored.

- You specify which columns to display by entering these columns in the Search Criteria window prior to generating the report.
- If the Search Criteria window is blank, only the name, status, and version of the selected entities are printed on the report.
- If you want to edit the Search Criteria window follow these steps:

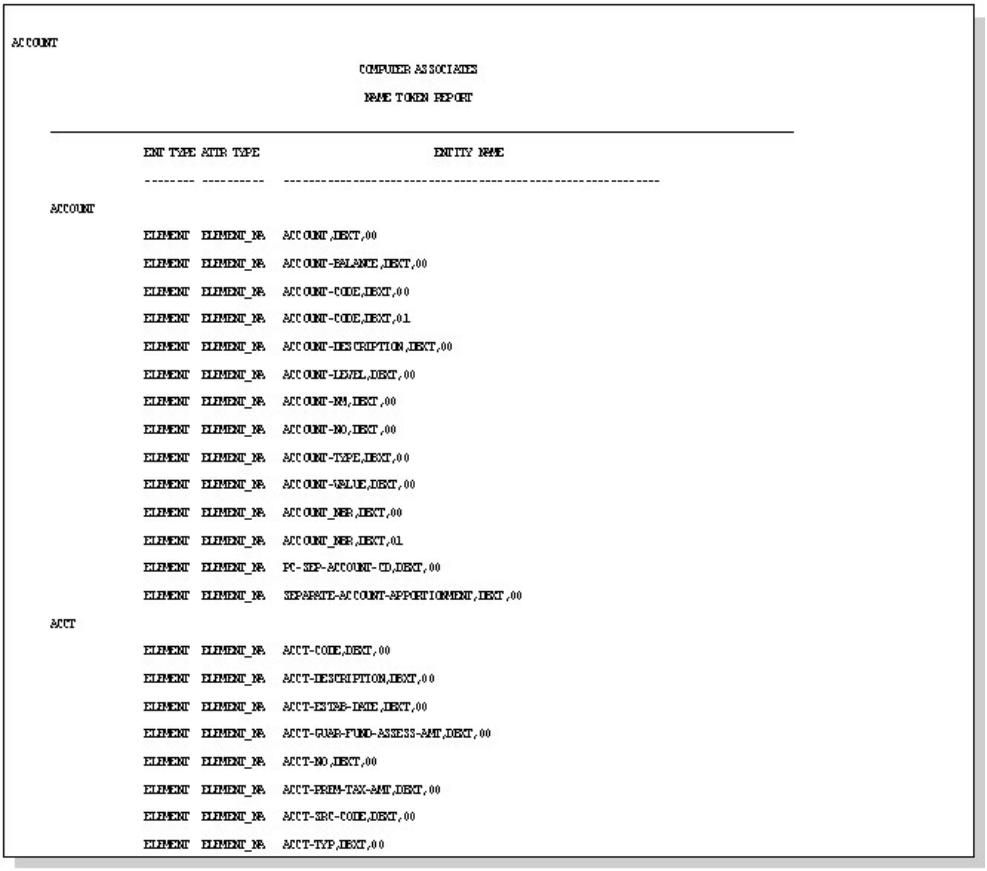
  1. Execute the View.List.Entity Q Command.
  2. Place a ? in the Columns section, and select the columns you would like to see from the resulting list.
  3. Press PF 3 to execute the query.
  4. Select the rows you would like to see in the report and execute Options.Reports.List to run the report.

For further details on Search Criteria, see the Work with Queries section in "Using Queries and Search Criteria."

Name Token Reports

Many of the names used for entity attributes are often composed of separate parts that together describe the attribute. CUSTOMER_NAME, for instance, is made up of the parts CUSTOMER and NAME. These parts are referred to as tokens.

You can use the Name Token Report to view attributes of existing repository entities grouped by their tokens. This report can help you determine if the same information is being stored in more than one location. The following is a sample Name Token Report.



```
ACCOUNT                                  COMPUTER ASSOCIATES
                                         NAME TOKEN REPORT

                ENT TYPE ATTR TYPE                    ENTITY NAME
                -------- ---------  ------------------------------------------------------------
    ACCOUNT
                ELEMENT  ELEMENT_NA  ACCOUNT,DEXT,00
                ELEMENT  ELEMENT_NA  ACCOUNT-BALANCE,DEXT,00
                ELEMENT  ELEMENT_NA  ACCOUNT-CODE,DEXT,00
                ELEMENT  ELEMENT_NA  ACCOUNT-CODE,DEXT,01
                ELEMENT  ELEMENT_NA  ACCOUNT-DESCRIPTION,DEXT,00
                ELEMENT  ELEMENT_NA  ACCOUNT-LEVEL,DEXT,00
                ELEMENT  ELEMENT_NA  ACCOUNT-NM,DEXT,00
                ELEMENT  ELEMENT_NA  ACCOUNT-NO,DEXT,00
                ELEMENT  ELEMENT_NA  ACCOUNT-TYPE,DEXT,00
                ELEMENT  ELEMENT_NA  ACCOUNT-VALUE,DEXT,00
                ELEMENT  ELEMENT_NA  ACCOUNT_NBR,DEXT,00
                ELEMENT  ELEMENT_NA  ACCOUNT_NBR,DEXT,01
                ELEMENT  ELEMENT_NA  PC-SEP-ACCOUNT-CD,DEXT,00
                ELEMENT  ELEMENT_NA  SEPARATE-ACCOUNT-APPORTIONMENT,DEXT,00
    ACCT
                ELEMENT  ELEMENT_NA  ACCT-CODE,DEXT,00
                ELEMENT  ELEMENT_NA  ACCT-DESCRIPTION,DEXT,00
                ELEMENT  ELEMENT_NA  ACCT-ESTAB-DATE,DEXT,00
                ELEMENT  ELEMENT_NA  ACCT-GUAR-FUND-ASSESS-AMT,DEXT,00
                ELEMENT  ELEMENT_NA  ACCT-NO,DEXT,00
                ELEMENT  ELEMENT_NA  ACCT-PREM-TAX-AMT,DEXT,00
                ELEMENT  ELEMENT_NA  ACCT-SRC-CODE,DEXT,00
                ELEMENT  ELEMENT_NA  ACCT-TYP,DEXT,00
```

The Repository looks for underscores (_), dashes (-), and blank spaces to determine the beginning and end of name tokens. The tokens found by the Report Facility are in the left-most column (AL, ALIAS, ANOTHER, ASEM, and so on).

Each of the lines following a token represents an attribute of a repository entity that uses that particular token. The information for each attribute includes: the entity type, the attribute type, the attribute itself, and the status, and version of the entity to which the attribute belongs.

Two additional columns, titled Data type and Length (not shown in the previous Name Token Report), are included only when you run a report for the ELEMENT entity. When you run a report for any other entity, these columns are omitted.

Edit the Name Token JCL

The Name Token Report is generated as a batch job. This job contains the following basic steps:

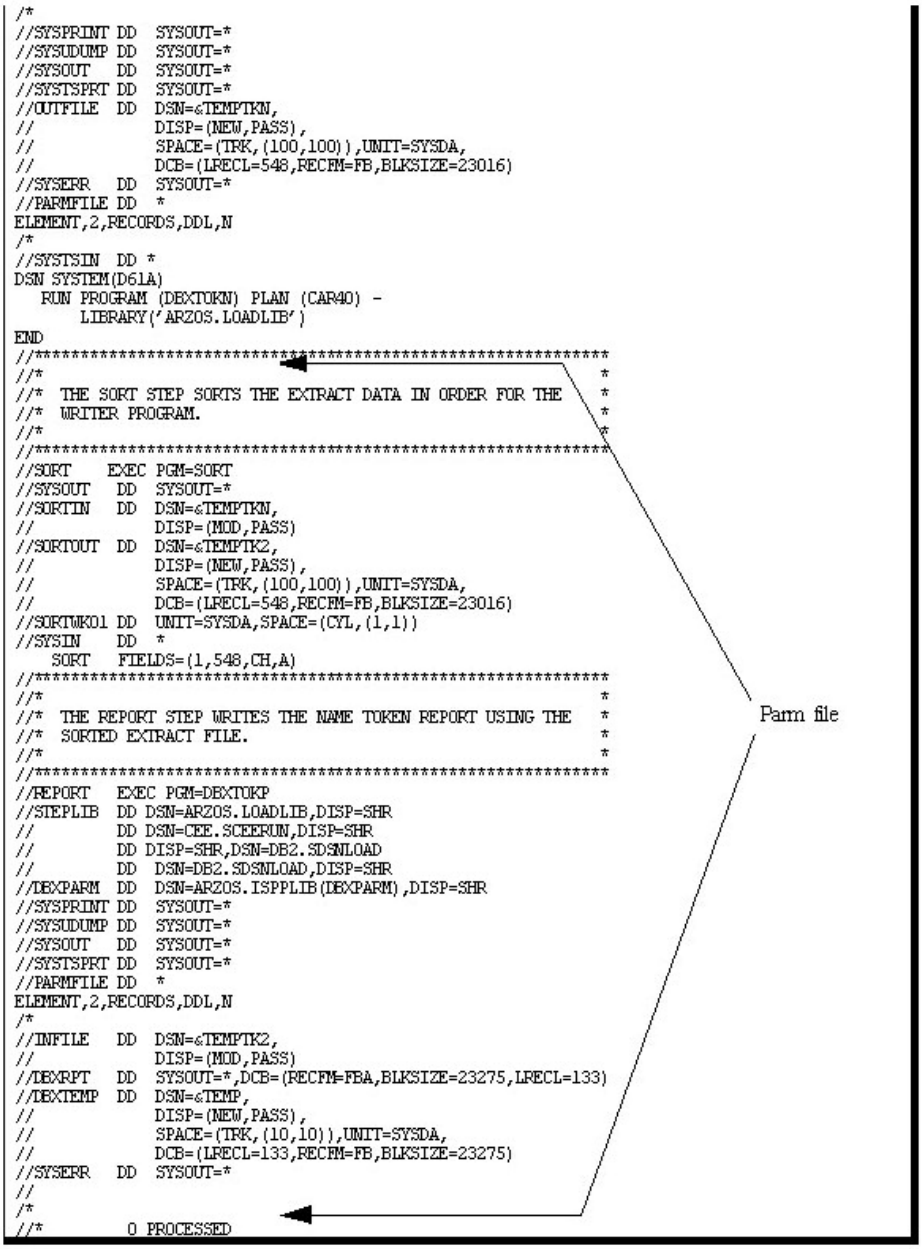
1. Extracting the actual data from the repository tables.
2. Sorting the data.
3. Writing the report using the sorted extract file.

Because of the scope of the job performed by the Name Token Report, you can only generate the report in batch mode. If you select Screen mode on the Print window, it is ignored.

The JCL used to run the batch job follows.

```
//*************************************************************
//*                                                           *
//*              NAME TOKEN REPORT JCL                        *
//*  SEE COMMENTS LATER IN THE JCL FOR OPTIONAL PROCESSING    *
//*                                                           *
//*************************************************************
//EXTRACT  EXEC PGM=IKJEFT01
//STEPLIB  DD DSN=ARZOS.LOADLIB,DISP=SHR
//         DD DSN=CEE.SCEERUN,DISP=SHR
//         DD DISP=SHR,DSN=DB2.SDSNLOAD
//         DD  DSN=DB2.SDSNLOAD,DISP=SHR
//DBXPARM  DD  DSN=ARZOS.ISPPLIB(DBXPARM),DISP=SHR
//*************************************************************
//*  ENTER IN THIS FILE THE NAMES OF THE ATTRIBUTES THAT      *
//*  ARE TO BE SEARCHED FOR TOKENS. AT LEAST ONE MUST BE      *
//*  ENTERED (FIVE MAX) AND THEY MUST BEGIN ON COLUMN 1.      *
//*  FOR EXAMPLE, ENTER ELEMENT_NAME, DB2_COLUMN_NAME, AND    *
//*  BUSINESS_NAME FOR THE ELEMENT ENTITY TYPE.               *
//*  THE NAME COLUMN IS ENTERED AS A DEFAULT.                 *
//*************************************************************
//NAMEFILE DD  *                                    <-- Name file
ELEMENT_NAME
/*
//*************************************************************
//*  ENTER IN THIS FILE A LIST OF TOKENS IF YOU ONLY WANT     *
//*  TO RUN THE REPORT FOR A SUBSET. MUST BEGIN IN COLUMN 1   *
//*  AND UP TO 300 CAN BE SPECIFIED.                          *
//*************************************************************
//INCFILE  DD  *                                    <-- Include
ACCOUNT                                             <-- Include file
ACCT
/*
//*************************************************************
//*  ENTER IN THIS FILE A LIST OF TOKENS IF YOU WANT TO       *
//*  EXCLUDE SOME TOKENS. TOKENS MUST BEGIN IN COLUMN 1       *
//*  AND UP TO 300 CAN BE SPECIFIED.                          *
//*************************************************************
//EXCFILE  DD  *                                    <-- Exclude file
/*
//*************************************************************
//*  ENTER IN THIS FILE ANY WHERE CRITERIA YOU WOULD LIKE     *
//*  USED WHEN SELECTING THE ENTITIES                         *
//*************************************************************

//WHERFILE DD  *
E.STATUS='DEXT'
```


/*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//OUTFILE DD DSN=&TEMPTKN,
//   DISP=(NEW,PASS),
//   SPACE=(TRK,(100,100)),UNIT=SYSDA,
//   DCB=(LRECL=548,RECFM=FB,BLKSIZE=23016)
//SYSERR   DD SYSOUT=*
//PARMFILE DD *
ELEMENT,2,RECORDS,DDL,N
/*
//SYSTSIN DD *
DSN SYSTEM(D61A)
   RUN PROGRAM (DEXTOKN) PLAN (CAR40) -
       LIBRARY('ARZOS.LOADLIB')
END
//***
//*   *
//* THE SORT STEP SORTS THE EXTRACT DATA IN ORDER FOR THE   *
//* WRITER PROGRAM.   *
//*   *
//***
//SORT   EXEC PGM=SORT
//SYSOUT   DD SYSOUT=*
//SORTIN   DD DSN=&TEMPTKN,
//   DISP=(MOD,PASS)
//SORTOUT DD DSN=&TEMPTK2,
//   DISP=(NEW,PASS),
//   SPACE=(TRK,(100,100)),UNIT=SYSDA,
//   DCB=(LRECL=548,RECFM=FB,BLKSIZE=23016)
//SORTWK01 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSIN   DD *
    SORT   FIELDS=(1,548,CH,A)
//***
//*   *
//* THE REPORT STEP WRITES THE NAME TOKEN REPORT USING THE   *
//* SORTED EXTRACT FILE.   *
//*   *
//***
//REPORT   EXEC PGM=DEXTOKP
//STEPLIB DD DSN=ARZOS.LOADLIB,DISP=SHR
//   DD DSN=CEE.SCEERUN,DISP=SHR
//   DD DISP=SHR,DSN=DB2.SDSNLOAD
//   DD DSN=DB2.SDSNLOAD,DISP=SHR
//DEXPARM DD DSN=ARZOS.ISPPLIB(DEXPARM),DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//PARMFILE DD *
ELEMENT,2,RECORDS,DDL,N
/*
//INFILE   DD DSN=&TEMPTK2,
//   DISP=(MOD,PASS)
//DEXRPT   DD SYSOUT=*,DCB=(RECFM=FBA,BLKSIZE=23275,LRECL=133)
//DEXTEMP DD DSN=&TEMP,
//   DISP=(NEW,PASS),
//   SPACE=(TRK,(10,10)),UNIT=SYSDA,
//   DCB=(LRECL=133,RECFM=FB,BLKSIZE=23275)
//SYSERR   DD SYSOUT=*
//
/*
//*   0 PROCESSED
Parm file

To see the JCL for the Name Token Report in an edit session, set the EDIT switch to E (edit).

This enables you to alter one or more of the report parameters to further control which particular entities are processed.

The following table describes the areas in the JCL that you can change.

File name	Description
Namefile (Name file)	The Name file lists the attribute types you want to process. The values in this file are created from the Columns section of the current SQL Search Criteria. The Name attribute is the default value for this field. You must specify at least one column, but no more than five. If you specified more than five in the SQL Search Criteria, only the first five are used.
Incfile (Include file)	Controls the specific tokens that are processed by the Report Facility. You use the Include file (Incfile) to limit processing to a few specific tokens. You can specify up to 300 tokens. There is no default value for this field.
Excfile (Exclude file)	Controls the specific tokens that are processed by the Report Facility. You use the Exclude file (Excfile) to specify the tokens you do not want to include in the report. You can specify up to 300 tokens. There is no default value for this field.
Wherfile (Where file)	Contains the Where criteria for limiting the entities that are processed for the Name Token Report. The Report Facility extracts this information from the current SQL Search Criteria. You can enter up to ten rows or 750 characters of Where information.
Parm (Parm files)	Tells the Repository which entity type and dialog to process, what type of report to create, and whether to process in debug mode.

Specify the Parameter File

The Parm file appears in two places in the JCL. This file contains the following parameters for the Name Token Report job, including:

- What entity type is being processed
- What type of Token Report to create
- Which dialog to use
- Which Repository usage (map) to use
- Whether to run in DEBUG mode

The format of the of the Parm file is:

Entity type,Report type,Current dialog,Usage,Debug switch

For example, in the previous illustration, the Parm file is:

ELEMENT,2,RECORDS,DDL,N

The following table describes each of these Parm file parameters.

Parameter	Description
Entity type	Specify the name of the entity type you want to process. If you select an entity set for processing, all of its constituent entity types will be processed (subject to the Where criteria).
Report type	Specify which Token Reports to create. Valid options are: ■ Reports for the Elements set. This report includes the Data type and Length information. ■ Reports for any non-ELEMENT entity types.
Current dialog	Specify the name of the Repository dialog that contains the entity type you want to process.
Usage	Specify the map to use for processing if you are creating a report for the Elements set (report type 1).
Debug switch	Specify whether to run the report in debug mode. Valid options are: ■ Y—To run the report in debug mode. ■ N—To not run in debug mode.

Path Reports

Repository Paths are conceptual collections of steps or navigational movements that *progress* from one meta-entity type to another across the associations or relationships that connect the types. The Repository Administrator defines these paths.

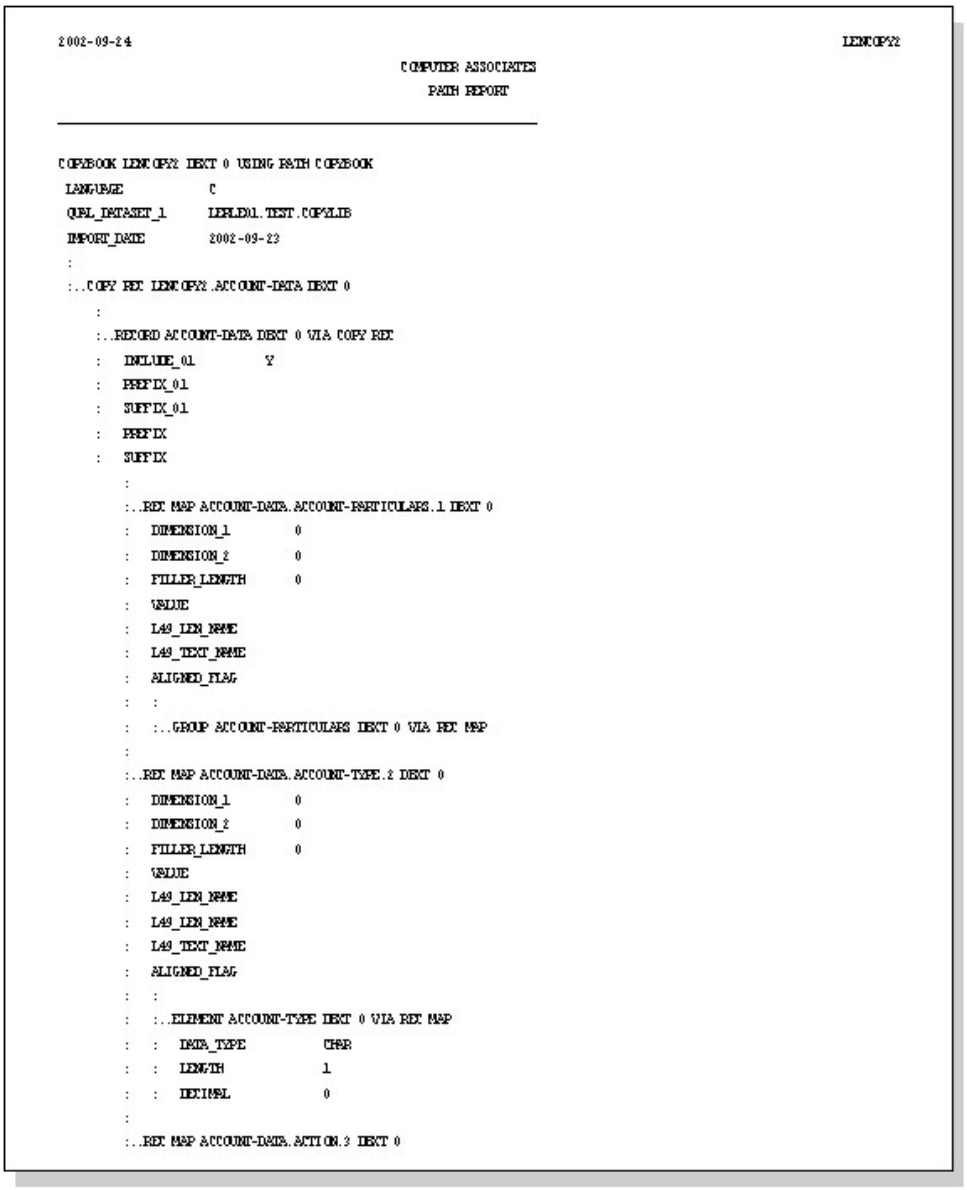
When you select an entity instance and generate a Path Report, the instance is identified as the root of the report and all the occurrences connected to the root occurrence along the specified path are included in the Path Report.

The Path Report Facility was designed to provide Impact Analysis Reports that reach beyond the standard Cross Reference (XREF) Report. Whereas the XREF Report only lists definitions that are linked by *one* association/relationship to the occurrence about which the report is being generated, the Path Report uses paths that string together multiple associations and relationships. Every entity type (and association/relationship, if desired) encountered along the path is included within the report. This greatly simplifies the process of determining which definitions are linked together when there is no *direct* connection between them.

While the paths used by Path Reports direct the facility from one definition to another along established routes, they also specify which attributes of those definitions should appear in the reports. If you do not explicitly include attributes in the report, then only the key attributes are reported.

You can generate the Path Reports either online or in a batch mode. If you choose batch mode, the JCL that executes the Report Facility is generated by using the OPTIONS.REPORTS.PATH command. Submitting this command displays a list of path names that the Repository processes for the chosen root occurrence before the JCL is generated.

The following is an example of a Path Report.



```
2002-09-24                                                              LENCOPY2

                                        COMPUTER ASSOCIATES
                                           PATH REPORT

_____________________________________________________________

COPYBOOK LENCOPY2 NEXT 0 USING PATH COPYBOOK
 LANGUAGE            C
 QUAL_DATASET_1      LERLD01.TEST.COPYLIB
 IMPORT_DATE         2002-09-23
 :
 :...COPY REC LENCOPY2.ACCOUNT-DATA NEXT 0
     :
     :...RECORD ACCOUNT-DATA NEXT 0 VIA COPY REC
     :   INCLUDE_01          Y
     :   PREFIX_01
     :   SUFFIX_01
     :   PREFIX
     :   SUFFIX
         :
         :...REC MAP ACCOUNT-DATA.ACCOUNT-PARTICULARS.1 NEXT 0
         :   DIMENSION_1         0
         :   DIMENSION_2         0
         :   FILLER_LENGTH       0
         :   VALUE
         :   L49_LEN_NAME
         :   L49_TEXT_NAME
         :   ALIGNED_FLAG
         :   :
         :   :...GROUP ACCOUNT-PARTICULARS NEXT 0 VIA REC MAP
         :
         :...REC MAP ACCOUNT-DATA.ACCOUNT-TYPE.2 NEXT 0
         :   DIMENSION_1         0
         :   DIMENSION_2         0
         :   FILLER_LENGTH       0
         :   VALUE
         :   L49_LEN_NAME
         :   L49_LEN_NAME
         :   L49_TEXT_NAME
         :   ALIGNED_FLAG
         :   :
         :   :...ELEMENT ACCOUNT-TYPE NEXT 0 VIA REC MAP
         :   :   DATA_TYPE           CHAR
         :   :   LENGTH              1
         :   :   DECIMAL             0
         :
         :...REC MAP ACCOUNT-DATA.ACTION.3 NEXT 0
```

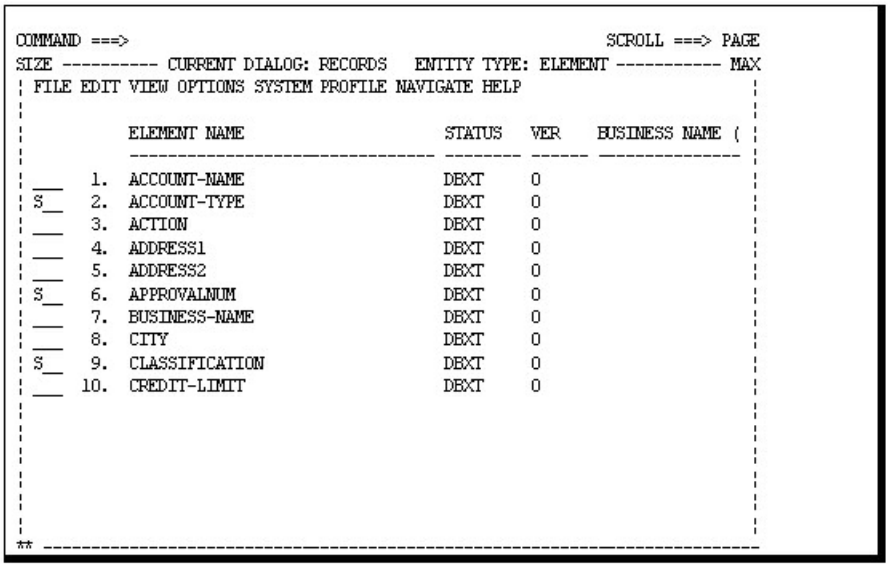
Generate Reports

The Repository includes several types of standard reports to provide information on all entities, relationships, and associations defined to the repository. Reports are based on the entity type and can be generated for any entity type, including user-created types added through the Extend feature.

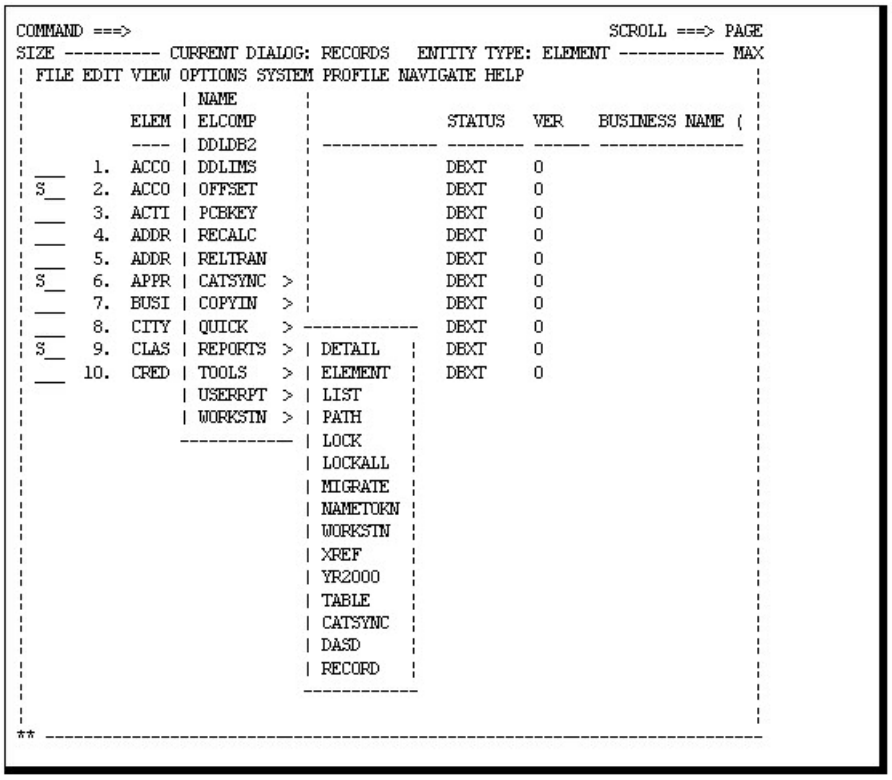
- For more information about the types of report you can create, see Understanding Standard Reports.
- For more information on the Extend feature, see the *Administration Guide*

To generate a report

1. Open an Edit window for the entity type on which you want to base the report.
2. Select the entities on which you want to base the report. The following is a sample Edit window containing entities tagged for processing.

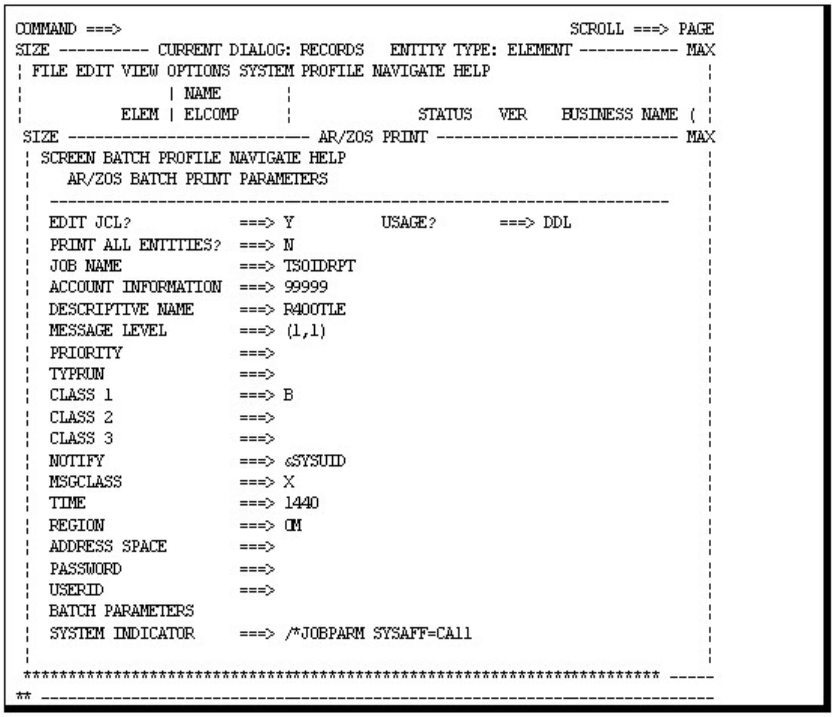


3. Select OPTIONS.REPORTS. A list of available reports appears.



Note: Some of these reports are for specific entity types, so not all are available for every entity type

4. Select the report and press Enter. The Print window appears.



5. Complete the fields in the Print window:

Edit JCL

Whether to display the JCL for the print job. Valid options are:

- Y—Change the parameters before they are submitted.
- N—Do not display the JCL for the print job.

Usage

The Repository uses the product usage to determine which attributes of each entity type to include in the report. Specify DDL. This field does not affect standard reports. For information on usage, see the Administration Guide.

Print All Entities

Generally reports are limited to those entities selected in the Edit window prior to accessing the Print window. You can use this field to override that feature and print a report containing every entity of the current entity type. Specify:

- Y—to generate the report for all entities. Any search criteria currently in affect limits the report to those entities matching the criteria.
- N—to print only the entities you specified in the Edit window.

Job Name

A name for this particular print job.

Account Information

The account number to which the job is charged.

Descriptive Name

A descriptive name for the JCL Job card.

Message Level

Whether or not to list the JCL statements or print allocation and deallocation messages.

Priority

The priority of the job while in the input queue.

TYPRUN

The type of execution. Class 1, 3, or 3. Only one of these fields is used.

Notify

To see a message at your terminal when the job is complete, enter your TSO logon ID in this field.

MSGCLASS

Specify the output print class.

Time

A time limit for the job. Jobs which exceed their specified time limit are canceled.

Region

The region size allocated to the job.

Address Space

Whether virtual (VIRT) or real (REAL) memory is to be used for each step.

Password

A RACF (ACF2) password.

USERID

The RACF (ACF2) user ID.

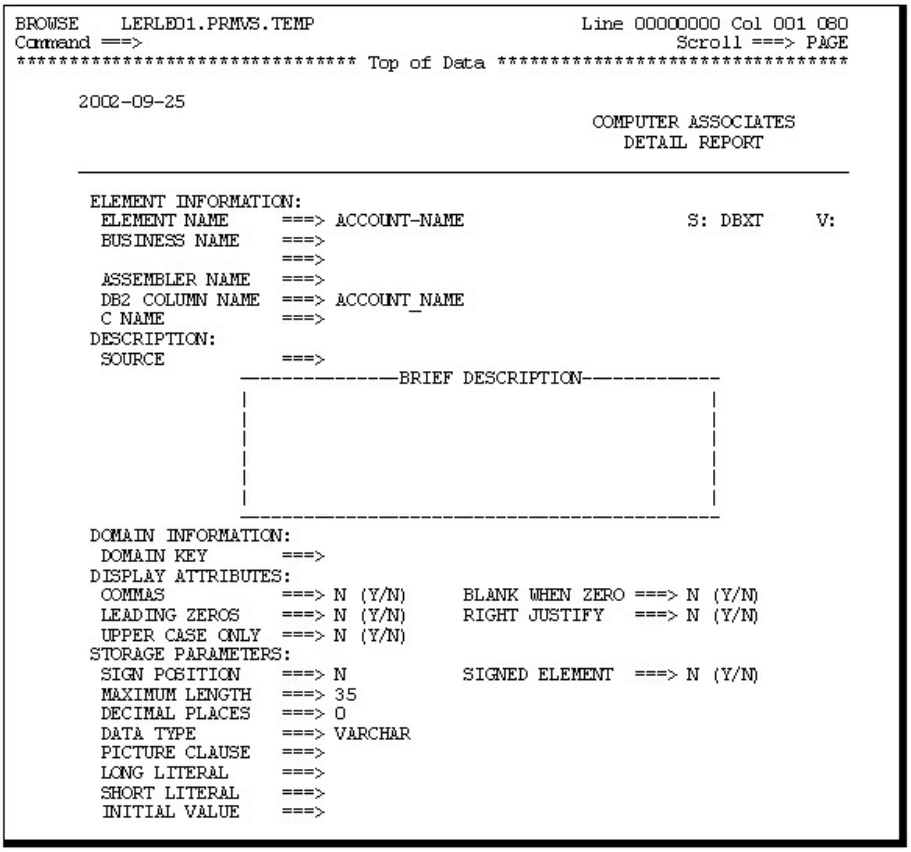
6. In the Print window menu bar, select one of the following:
   - If you do not want to view the report online before it is printed, select BATCH. The report is generated in the Print window.
   - If you want to generate the report online and review it before it is printed, select SCREEN. The report is generated and displayed in an Edit window.
7. If you selected BATCH in the previous step, go to Step 8.
   If you selected SCREEN in the previous step, review the report and make changes if necessary.
8. Press F3 or enter END on the command line to return to the Print window.

View Reports Online

In the Repository, you can generate the report to your online session and view the report in an ISPF browse window before it is printed.

- Use the standard browse commands to view the report while in the screen mode
- Press F3 or enter the END command to return to the Print window.

The following illustration shows a section of a report.



To view your report online

1. Complete Steps 1-4 described in the Generate Reports section.
2. Enter a Y in the PRINT ALL ENTITIES field in the Print window.
3. Select SCREEN. The report is generated and displayed in an Edit window.
4. Review the report and make changes if needed. Press PF3 or enter the END command to return to the Print window.

Note: If you specified any search criteria prior to accessing the Print window, the entities are limited to those that match the search criteria.

Print Reports in Batch Mode

In the Repository, you can also generate the report in batch mode. If you choose to generate the report in Batch mode, the Repository creates a batch job to print the report to the output device that is specified in the Print ISPF skeleton DBXPRNT for the ddname SYSOUT.

Use the Job card parameters displayed in the Print window to customize the print JCL.

- These parameters are derived from the user profile and global profile
- Depending on the values selected for each parameter, portions of the JCL may or may not be included for report submission

The standard parameters are listed in the Generate Reports section in this chapter. Ask your Repository Administrator if there are site-specific batch parameters.

Product-Specific Reports

In addition to the standard reports included as part of the Repository, each Repository product comes with its own set of reports. Once the individual products are installed, these product specific reports can be accessed in the same manner as the standard reports.

The following table contains some of the Repository product-specific reports.

Report Title	Description
Table Definition Report	Provides a consolidated view of the data and relationships for a DB2 table defined in the repository.
Record Offset Report	Provides detailed information regarding the physical layout of a flat file/program record defined in the repository.
DASD Estimation Report	Provides estimated space requirements for DB2 objects based on the attributes and relationships defined in the repository.

Chapter 11: Using Workstations

A Repository workstation is a group of logically associated entity occurrences from one or more entity types used to simplify tasks involving groups of entity occurrences.

This section contains the following topics:

Workstations

Workstation functionality provides a powerful analysis tool for the Repository users. You can examine and process the contents of workstation, allowing you to group occurrences by their use.

In the Repository environment, workstations are used to migrate entities from one status to another. By migrating a workstation instead of individual entities or a group of tagged entities, you can migrate entities from multiple entity types simultaneously. The alternative would require migrating entities according to entity type.

You can also use workstations as logical units of work for the various Repository CASE (Computer Aided Software Engineering) tool interfaces. The Repository uses workstations when importing and exporting CASE tool data.

The following Repository Load Facilities automatically populate workstations:

- Copybook In
- DBMS Catalog Synchronization
- Path Workstation Add
- Segment In
- PSB In
- DBD In
- Program Scans
- Job and Proc Scans

The user batch load mechanism, called the Repository Batch Load, can also be configured to load directly into a workstation.

Entity, relationship, or association instances can belong to more than one workstation. You can apply workstations to a variety of situations, as they are integral to successful metadata management.

About Workstation Entity Types

The following Repository entity types are used with workstations:

- WORKPACK
- WORKPROJ
- WORKSTN
- WORKSUBJ
- PROJSTN (technically an association)

The Repository provides different workstation classifications. Each workstation entity type uses the same set of workstation commands and is identically processed. The classification allows Repository administration to more clearly distinguish workstations and how they are applied.

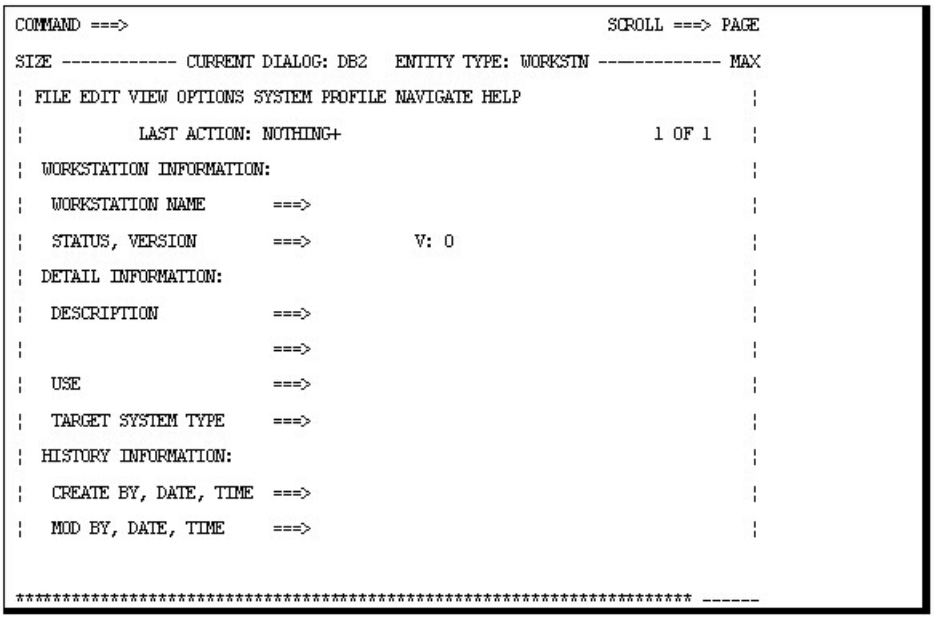
The following table describes the workstation classifications.

Workstation Classifications	Descriptions
WORKPACK	Use to group packages of metadata that are to be processed in the same manner, such as migration, reporting, or DDL delivery. The attributes of WORKPACK include a Requester and Request date. For example, you could put an IMS Database occurrence into a WORKPACK to signal that it should be migrated to production and then generated into the production DBD source library.
WORKPROJ	Use to group the metadata for an entire CASE project. WORKPROJ is the recommended workstation type for grouping metadata loaded with the Repository Load Facilities, such as DB2 Catalog Synchronization, PSB In, Copybook In, and so on.
WORKSTN	A generic entity type. Use as the anchor point for logical ties to the entity occurrences that make up the workstation. Use to group the metadata for individual CASE models. Each

Workstation Classifications	Descriptions
	CASE model belongs to a CASE project. WORKSTNs group individual models within a CASE project.
WORKSUBJ	Use to group occurrences that belong to a work subject area. Data administration can use WORKSUBJ when they want to group occurrences with the same data subject, such as all occurrences related to the customer.
PROJSTN	Use to associate work projects to workstations (WORKPROJ to WORKSTN). This configuration indicates that a CASE project is composed of many CASE project models. The metadata for the entire project is associated with the WORKPROJ workstation, while each individual CASE project model is associated with its own WORKSTN workstation. To use the PROJSTN association, simply insert PROJSTNs associations linking the WORKPROJ occurrence to its WORKSTN occurrences.

The Workstation Edit Window

The following illustration shows the Workstation Edit window for the WORKSTN entity type.



The following fields appear in the WORKSTN Edit window:

Workstation name

A workstation name. Identifies the WORKSTN entity to the Repository and to the workstation to the user. This is the name the user sees when working on the workstation.

Status, Version

The status and version of the workstation.

Description

A description of the information stored in the workstation. This information displays when you generate a list of workstations.

Use

The purpose of the workstation. Generally refers to one of the CASE tools that the Repository supports.

Create by, Date, Time

The User ID of the person who created the workstation, and the date and time it was created.

Mod by, Date, Time

The User ID of the person who last edited the workstation and the date and time it was edited.

The Workstation Display Window

You can use the following commands in the Workstation Display window:

Command	Description
DELETE	Removes one or more entities from the workstation. The entities are not removed from the repository, nor is the workstation deleted.
MINIEDIT	Use to review and modify the attributes of one or more of the entities in the Workstation Display window. To edit an entity, type an S in its Select byte field and select MINIEDIT from the menu bar.
IMPACT	Use to generate Impact Analysis windows for the entities in the Workstation Display window.
TEXT	Use to browse the extended text (if any) of the entities displayed in a Workstation Display window. To view the extended text of an entity, type an S in its Select byte field and then select TEXT from the menu bar.

Command	Description
WORKSTN	Use to execute ADD, ADDONLY, DELETE, and DELONLY commands on tagged occurrences in the Workstation Display window. You can execute these commands against the current workstation or other workstations.
LOCK	■ Use to add or delete a workstation lock. ■ You can execute these commands against the current workstation or additional workstations. ■ Select and update locks are available.
NEXT	Use to view successive sets of entities when the number of entities in a workstation exceeds the maximum that can be displayed in a single list. The number of entities that can be displayed in the Workstation Display window is five times the Max list value in your user profile.

Create a Workstation

Before you are able to use the workstation-related commands to add entities to a workstation, you need to access one of the workstation entity types and define (INSERT) a workstation. For convenience, the workstation entity types are available in all the Repository dialogs. For more information about the workstation entity types, see Workstation Entity Types.

To create a new workstation

1. Select OPTIONS.REPORT.WORKSTN. An AR/ZOS Print Dialog appears.
2. Determine if you want to generate the report to the screen or generate it in batch mode. To:
   - Generate the report to the screen, select SCREEN
   - Generate the report in batch, complete the parameters on the Print dialog, and select BATCH
3. Press Enter and a WORKSTATIONS dialog appears.
4. Select a workstation and press Enter.
   - If you selected SCREEN, the report is generated and appears on the specified workstation's screen
   - If you selected BATCH, the JCL is generated and displayed-you must submit the job to generate the report

Maintain Workstations

The following commands are available for creating and maintaining the Repository workstations. All of these commands are subfunctions of OPTIONS.WORKSTN. The workstation commands are:

Command	Description
ADD	Adds tagged entity occurrences and related entity occurrences that are one level away to a user-specified workstation.
ADDONLY	Adds only tagged entities to a user-specified workstation, does not add related entities.
CLONE	Copies the occurrences from one workstation to another. The target workstation of the CLONE command must exist (have been previously inserted) for the CLONE command to be successful.
DELETE	Deletes any tagged entity occurrences and entity occurrences that are one level away form from a user-specified workstation. The DELETE command does not delete occurrences from the repository. The occurrences are deleted only as members of the specified workstation.
DELONLY	Deletes tagged entities from a user-specified workstation. Unlike the DELETE command, DELONLY does not delete related entities from the workstation. The DELETE command does not delete occurrences from the repository. The occurrences are deleted only as members of the specified workstation.
DISPLAY	Use to display the contents of a workstation using the Workstation Display Facility.
LOCK	Places a select or update lock on the entity occurrences in a workstation. See Locking a Workstation.
REMOVE	Removes all occurrences from a workstation, leaving the workstation completely empty. The occurrences are removed as members of the specified workstation, not from the repository.
RENAME	Use to change the name of a workstation or to merge two existing workstations. RENAME essentially moves the contents of one workstation to another.
SHOW	Use to display the names of workstations in which an occurrence is contained.

View the Contents of a Workstation

To view the contents of a workstation

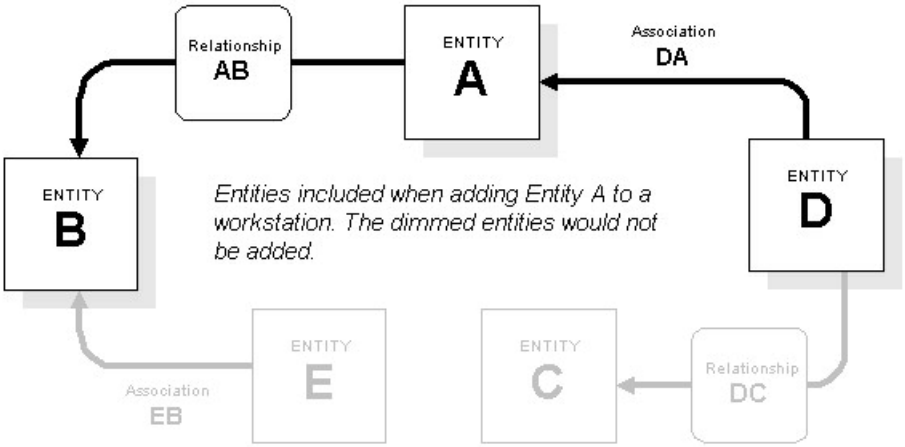
1. Open an Edit window for the workstation you want to use.
2. Select DISPLAY. The Workstation Display window appears.

Add Entities to a Workstation

Once you create a workstation, you use the ADD command to populate the workstation. Using the ADD command also adds any entities that are connected to the tagged entity through a relationship or association.

The ADD command does not add entities that are linked to those entities. This concept is perhaps more clearly illustrated by the following figure. Adding Entity A to a workstation would also add entities B and D, relationship AB, and association DA. It would not add entities C or E, association EB, or relationship DC.

If you do not want any of the related or associated entities to be added to the workstation, you can use the ADDONLY command to add only the selected entity.



Once you populate a workstation, you can use it to migrate its constituent entities, or to download the repository data as part of one of the Repository CASE tool interfaces.

Note: Workstations can be populated automatically when uploading CASE interface data, as well as by several of the Repository Load Facilities: DB2 Catalog Synchronization, Copybook In, Segment In, DBD In, PSB In, and user-customized Batch Load Syntax.

To add entities to a workstation

1. Select (and tag) the entities you want to add to the workstation.
2. Do one of the following:
   - If you want to add only the entity to the workstation, select OPTIONS.WORKSTATION.ADDONLY
   - If you want to include the relationships, associations, and entities associated with the entity you are adding, select OPTIONS.WORKSTATION.ADD
   - The Workstation Entry for Add window appears
3. Enter the name of the workstation in which you want to add the entities you selected above and press Enter. The workstation you specify must have been defined using one of the following four workstation entity types; WORKPACK, WORKPROJ, WORKSTN, or WORKSUBJ. For more information, see Workstation Entity Types.
4. Enter a question mark in this field to display a list of existing workstations.

The Path Workstation Add Facility

The Repository paths are collections of steps or navigational movements that *progress* from one meta-entity type to another across the associations or relationships that connect the types. The Repository Administrator defines these Paths.

When a user retrieves a root occurrence from the repository and runs the Path Workstation Add Facility, all occurrences that are connected to the root occurrence along the specified Path are added to the specified Repository workstation. Although it can be empty, the workstation must exist before executing this facility.

Whereas the standard workstation ADD commands, such as OPTIONS.WORKSTN.ADD, add only definitions that are linked by *one* association or relationship to the root occurrence, the Path Workstation Add Facility uses paths that string together multiple associations and relationships. Every entity type (and association or relationship, if desired) encountered along the path is included within the specified workstation.

Workstations are groupings themselves, used in locking, migrations, and with the CASE interfaces. Workstations differ from paths in that workstations contain occurrences of any and all meta-entities, while paths are collections of navigational movements from one meta-entity to another.

To run the Path Workstation Add Facility

1. Open an Edit window for the occurrence you want to use.
2. Select EDIT.SPECIAL.PATHADD. A list of path names that can process the chosen root occurrence appears.
3. Select the workstation you want to use as the subject of the Path Workstation Add. The JCL appears.
4. Review the JCL and make changes if needed.
5. Submit the job.

Rename and Merge Workstations

You can use the RENAME command to rename an existing workstation or to merge it with another workstation.

To rename a workstation or to merge two workstations

1. Select OPTIONS.WRKSTN.RENAME. The Workstation Entry for Rename window appears.
2. Complete the fields:

   Old Workstation

   The name of the workstation you want to rename or from which you want to copy information.

   New Workstation

   The name of the workstation into which you want to copy the information from the workstation you specified in the Old workstation field.
3. Press F3 or enter END on the command line to close the window. All occurrences are deleted from the old workstation and copied into the new workstation.

Delete Specific Entities from a Workstation

Use the DELETE and DELONLY commands to tag specific entities you want to remove from a workstation. These commands do not delete the entities from the repository. They only delete the tie between the occurrences and the workstation.

- The DELETE command removes the entity and any related or associated entities from the workstation
- The DELONLY command removes only the entities and leaves the related or associated entities in the workstation

To delete an entity from the workstation

1. Open the workstation that contains the entities that you want to delete.
2. Type **S** in the Select byte fields next to the entities you want to delete
3. Do one of the following:
   - If you want to delete the entity and the relationships, associations, and entities associated with the entity, select OPTIONS.WORKSTATION.DELETE
   - If you want to delete only the entity from the workstation, select OPTIONS.WORKSTATION.DELONLY.

   The Workstation Entry for Delete window appears.
4. Type the name of the workstation from which you want to delete the entities and press Enter.

Delete a Workstation

Deleting a workstation consists of the following steps:

1. Remove all entities from the workstation
2. Delete the workstation entity type occurrence defining the workstation.

   Use the REMOVE command (OPTIONS.WORKSTN.REMOVE) to remove all entities from the repository workstation. This command does not affect the physical repository definition of the workstation entity or any of its constituent entities. It only eliminates the affiliation of the entities with the workstation.

To remove all entities from a workstation

1. Select the REMOVE option from the WORKSTN menu to display a Workstation Remove window.
2. Type the name of the workstation to remove in the Remove Workstation field.
3. Press Enter to remove the workstation. The REMOVE command does not delete the WORKSTN entity that defines the workstation. To delete this entity, you delete the appropriate occurrence from the appropriate workstation entity type using the EDIT.DELETE command.

Note: Do not delete a workstation occurrence until you have used the REMOVE command to sever the affiliation of the member entities from the workstation.

Lock a Workstation

Workstations are often used to export data for manipulation outside of the Repository. However, once the repository data is exported, that data is maintained in two locations. To overcome problems associated with the maintenance of such redundant data, you can place locks on the Repository versions of exported entities. You can use these locks as a means of tracking the actions performed against entities (namely modifications) or to prevent such actions altogether.

Any time after you have locked a workstation, you can remove the locks from the workstation.

The Repository provides two types of locks for workstations:

- **Update lock**: Prevents modifications to any of its constituent entities. The presence of an Update lock on a workstation indicates that the entities in the locked workstation are presently being maintained outside the Repository. Only one Update lock can be placed on any individual workstation.
- **Select lock**: Does not prevent the manipulation of the workstation entities. It does permit the activity performed against an entity to be tracked. Actions performed against entities with a Select lock are recorded and can be listed in a report.

To lock or unlock a workstation

1. Select OPTIONS.WORKSTN.LOCK. The Workstation Entry for Lock window is displayed.
2. Complete the fields:

   Workstation

   The name of the workstation for which you want to place or remove a lock.

   Lock type

   The type of lock you want to use:

   - **S** to track activity performed on the entity in a report
   - **U** to prevent modifications and to indicate that the entities are being maintained outside the Repository
   - Blank to remove the lock
3. Press F3 or enter END on the command line to exit the Workstation Entry for Lock window. The lock is placed on (or removed from) the workstation.

Delete Occurrences Using Workstations

If you have the proper the Repository authority, you can use workstations to delete occurrences from the repository.

Almost all the Repository Load Facilities load directly into workstations, making Workstation Delete a powerful roll back tool.

- Entire workstations can be deleted from the repository in batch. If an entity occurrence is being used in other workstations it is not deleted.
- You can find sample Workstation Delete JCL in the Repository CCP$JCL data set DELWKSN member.
- The Workstation Delete JCL permanently deletes from the repository the occurrences contained in the specified workstation.

Workstations and SQL Search Criteria

Although workstations are used to group *logically* related data, it is possible that, even within a single entity type, the contents of a workstation may be *physically* diverse (that is, the metadata occurrences may have few common attributes). This diversity makes it difficult to build effective lists of workstation contents for processing within the Repository Edit window.

One way to get around this lack of physical similarity is to query the Repository control tables, selecting occurrences based on their ties to a workstation definition. The following query is an example how this can be done.

```
E.ENT_ID IN (SELECT W1.ENT_ID
FROM DBXREL30.DBX_WKSN_XREF W1
, DBXREL30.DBX_WORKSTATION_D W2
WHERE W2.ENT_ID = W1.WKSN_ID
AND W2.NAME = 'CURRVAL(E.LONG_DESCRIPTION)')
```

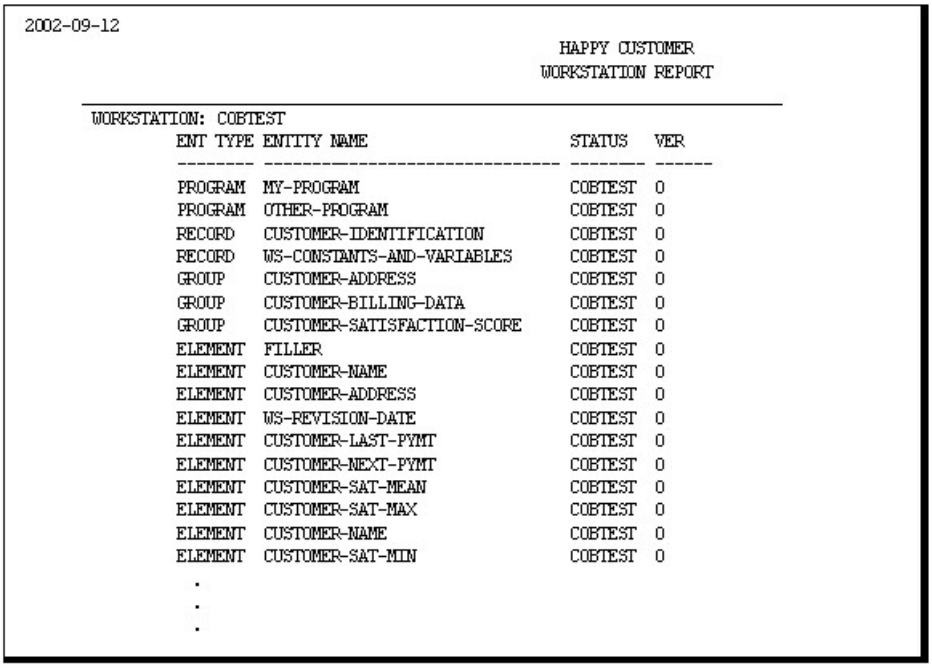
When used with the SQL Search Criteria window, you can use this SQL statement to assist in the retrieval of workstation specific occurrences into the Repository Edit window. This query returns a list of occurrences belonging to the selected entity type contained in the workstation having a name that matches any text in the first line of the Long description attribute (most Repository entity types have this attribute). You can also use this SQL statement in user-defined Query Management Facility (QMF) queries.

Create a Workstation Report

You can print the contents of a workstation as a report using the Print Facility. The Workstation Report is different from the other reports in that you do not have to tag specific occurrences in the Edit window before you generate the report.

The Repository automatically opens a window containing all the workstations defined at your site and you choose which workstations you want to use to generate the report.

The following is a sample report.



2002-09-12

HAPPY CUSTOMER
WORKSTATION REPORT

WORKSTATION: COBTEST

ENT TYPE	ENTITY NAME	STATUS	VER
PROGRAM	MY-PROGRAM	COBTEST	0
PROGRAM	OTHER-PROGRAM	COBTEST	0
RECORD	CUSTOMER-IDENTIFICATION	COBTEST	0
RECORD	WS-CONSTANTS-AND-VARIABLES	COBTEST	0
GROUP	CUSTOMER-ADDRESS	COBTEST	0
GROUP	CUSTOMER-BILLING-DATA	COBTEST	0
GROUP	CUSTOMER-SATISFACTION-SCORE	COBTEST	0
ELEMENT	FILLER	COBTEST	0
ELEMENT	CUSTOMER-NAME	COBTEST	0
ELEMENT	CUSTOMER-ADDRESS	COBTEST	0
ELEMENT	WS-REVISION-DATE	COBTEST	0
ELEMENT	CUSTOMER-LAST-PYMT	COBTEST	0
ELEMENT	CUSTOMER-NEXT-PYMT	COBTEST	0
ELEMENT	CUSTOMER-SAT-MEAN	COBTEST	0
ELEMENT	CUSTOMER-SAT-MAX	COBTEST	0
ELEMENT	CUSTOMER-NAME	COBTEST	0
ELEMENT	CUSTOMER-SAT-MIN	COBTEST	0

To create the Workstation Report

1. Select OPTIONS.REPORTS.WORKSTN. A list of available workstations appears.
2. Type **S** in the Select byte of the workstation you want to use and press Enter. The Workstation Report appears.

Chapter 12: Working with Navigations

Navigation is a custom made, semi-automatic process that is used to complete a specific repository task. The Repository Navigator is a facility you use to automate certain repository tasks.

This section contains the following topics:

Navigations

You can create your own navigation or you can use one that is shipped with CA Repository for z/OS.

The DB2 Table Navigation is an example of a typical navigation. This navigation takes you through the steps you need to take to define a DB2 table.

- When you start the DB2 Table Navigation, the DB2 dialog appears and leads you through each of the entity types you need to use to define a DB2 table (first ELEMENT, then TABLE, and then COLUMN).
- It then moves you to the Quick DB2 Facility. This process is automatic; you enter the entities you want to use for each step of the process.

The navigation selects the appropriate command and tags entities as needed to move data from one Repository Edit window to the next.

Start a Navigation

To start navigation, select NAVIGATE from the main menu bar. The Navigation List window appears.

The Navigation List window displays the ID of the person who created the navigation, the name of the navigation, and a brief description of what the navigation does.

- When the Repository Administrator creates navigation, the Creator column contains an asterisk (*).
- When individual users create navigations, the Creator column contains their TSO Logon ID. Usually, a navigation created by an individual user is available only to that user.

The rows on the Navigation List window show the navigations available for the entity type you selected.

Parts of a Navigation

Navigation is made up of the following parts:

Part	Description
Branches	Use to select from two or more options.
Pauses	Use to add information to input fields.
Help messages	Aids you in understanding either the task as a whole or a particular step in the process.

For the sake of an example, we will execute the DB2 Table Navigation. This navigation is included with CA Repository for z/OS as a globally-available navigation, so you should be able to select it if you want to follow along with the example.

Branch

At certain points in some navigations, selecting the CONTINUE option displays a pull-down menu rather than automatically continuing to the next step. The options on this menu represent a separate processing path, or a branch.

To see information about the branch, type a question mark in the Select byte of the branch and press enter. Select the branch you want to use by typing **S** in the Select byte.

Help Messages

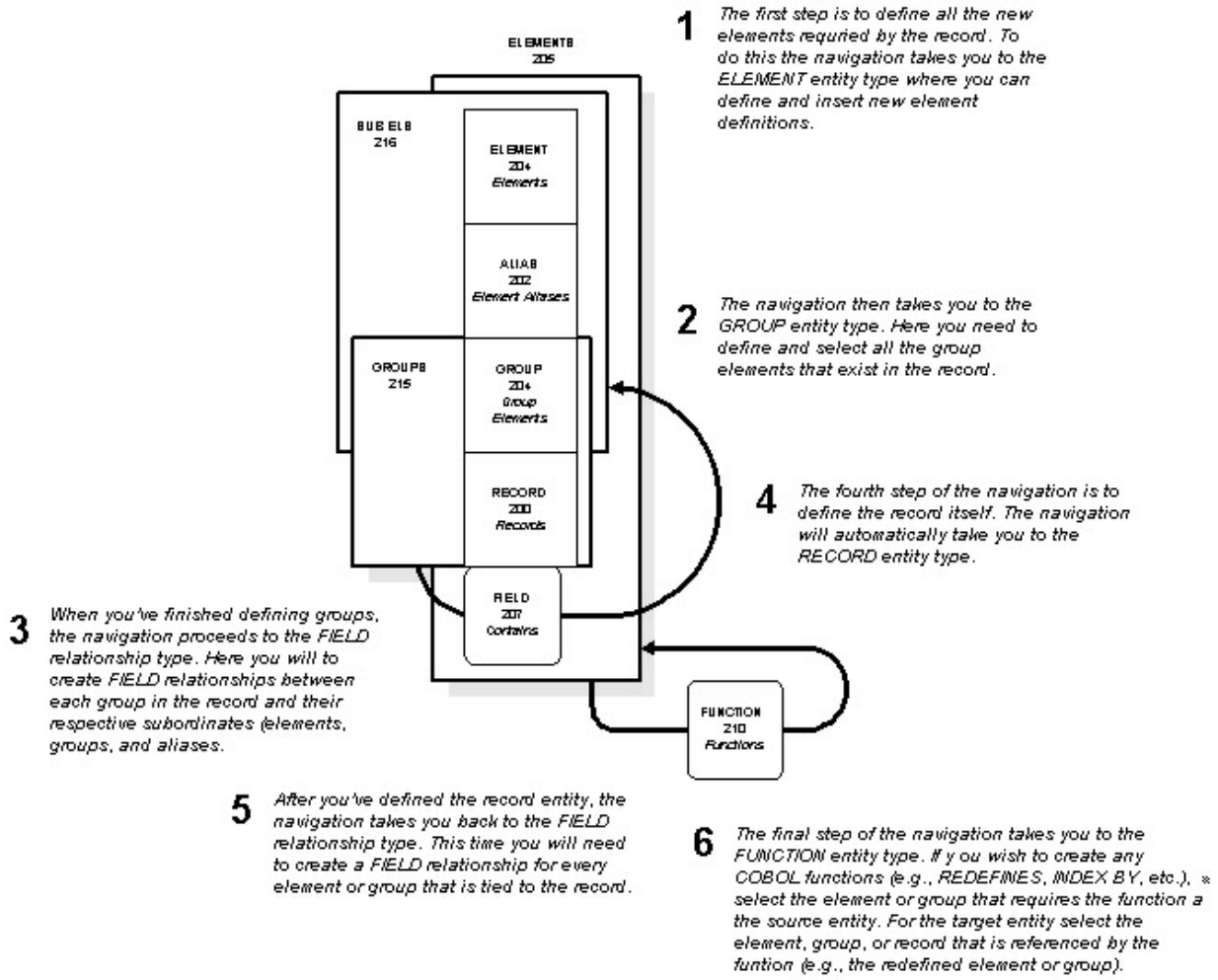
You can get information for the options that appear in the CONTINUE pull-down menu. To see a Help message for a branch, type **?** in the Select byte of the option in the pull-down and press Enter.

Sample Navigations

This section describes three sample navigations.

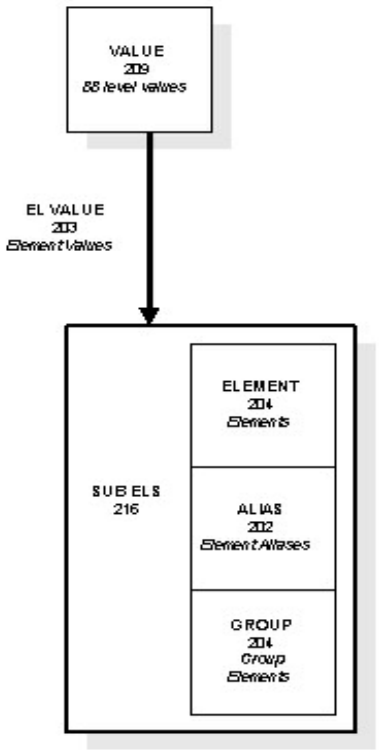
Sample COBOL Record Definition Navigation

Title	COBOL Record
Description	Leads the user through the steps required to define a COBOL record in the Repository.
Typical user	DB2 Database Administrator.
Meta-model	Subset of the Record model.



Sample Element Values Navigation

Title	Element Values
Description	A tutorial navigation that leads the user through the process of adding code table support to a particular element.
Typical user	Basic or first-time users with little or no Repository experience.
Meta-model	Record model.



1 *This navigation starts by taking you to the Records dialog and then to the VALUE entity type. It then pauses to allow you to add attributes for a VALUE entity definition. When the CONTINUE option is selected, the VALUE will be inserted into the Repository automatically by the navigation.*

2 *Once the VALUE entity has been inserted, the navigation uses the GOTO command to take you to the EL VALUE association type. This association type links value definitions to element definitions. The name, status, and version of the VALUE entity you just inserted will be added by the system to the appropriate fields of the EL VALUE Edit window.*

3 *After displaying a message describing this part of the process, the navigation will generate a list of possible source entities (existing ELEMENT definitions). Simply tag those elements to which you wish to attach your value and select the CONTINUE option.*

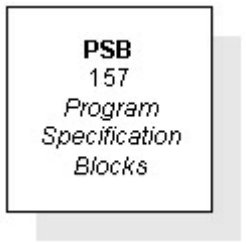
4 *At this point the navigation will switch to the horizontal mode of the Edit window so it can insert multiple entities. All the rows will automatically be tagged for processing. When the navigation executes the INSERT command, the tagged associations will be added to the Repository and the navigation will end.*

NOTE: This navigation was designed as a tutorial for new F users. Accordingly, it includes abundant help messages to describe each phase of the process. After you have used the navigation a few times and have become familiar with the actions it employs, you may find it faster to create EL VALUE entities without the navigation and its associated help messages.

Sample IMS PSB Definition Navigation

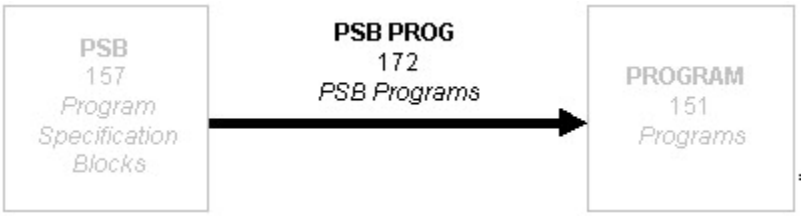
Title	IMS PSB
Description	Leads the user through the process of defining an IMS Program Specification Block (PSB).
Typical user	IMS Database Administrator.
Meta-model	A subset of the components in the IMS dialog.

1. The first step in defining a PSB to the repository is to create a PSB entity. Accordingly, the navigation starts by opening the Edit window for this entity type. To define multiple PSBs, insert multiple PSB entities during this step. After you insert one or more PSB entities, select CONTINUE.

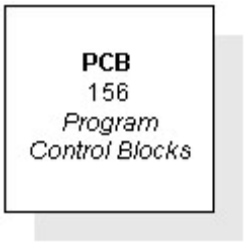


2. The navigation now proceeds to the PROGRAM entity type. Here, you define a program for each of the PSBs. Once you insert the programs for the PBSs, select CONTINUE.

3. The next step is to tie the PSBs to their respective programs. This step uses the PSB PROG association type, which is automatically selected by the navigation. Insert an association linking your PSB definitions to their respective programs. When finished, select continue.



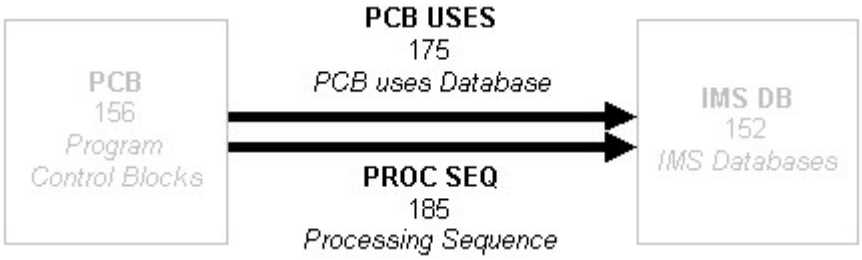
4. The navigation now selects the PCB entity type. Here, you define the program control blocks for the PSB. When finished, select CONTINUE.



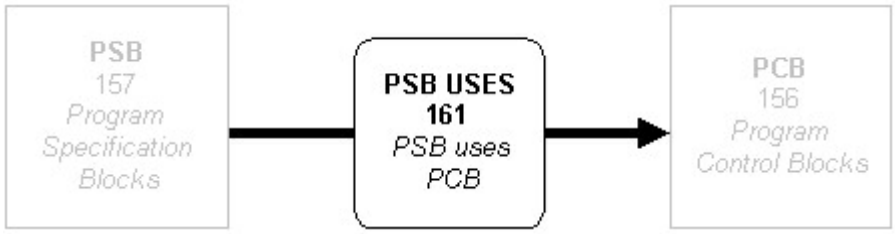
5. At this point, you tie the PCBs to their respective databases. The previously defined PCBs are automatically selected as source entities when the navigation switches to the PCB USES association type.

   Use VIEW.LIST.TARGET to select the appropriate source database entities. Insert the associations and select CONTINUE.

6. The navigation takes you to the PROC SEQ association type. Here you can insert associations to tie any PCBs that have processing sequences to the appropriate databases. When finished, select CONTINUE.



7. The next navigation step uses the PSB USES relationship type where you define relationships linking each of the PSBs to their respective PCBs. When finished, select CONTINUE.



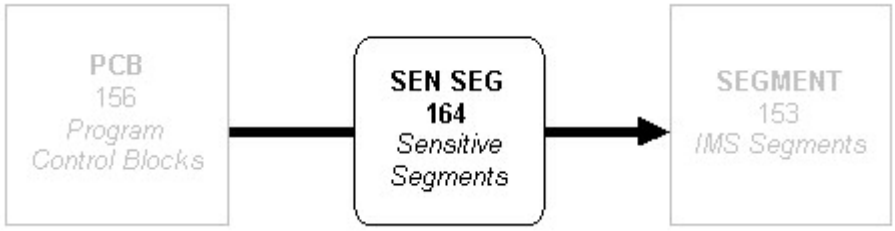
   Optional Navigation Branch: Sensitive Segments

   At this point the navigation branches, so that you can define sensitive segments for your PSB.

   If you do not want to include sensitive segments, select the NO SENSE option from the CONTINUE menu. This takes you to the next branch, bypassing Steps 8 and 9.

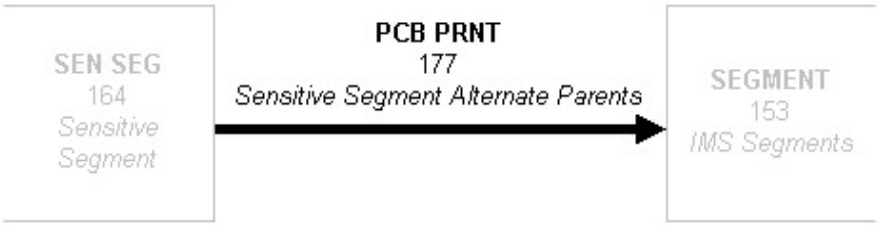
8. If you select the SENSEG option from the CONTINUE menu, the navigation takes you to the SEN SEG relationship type. These relationships are used to link PCB definitions to SEGMENT definitions.

   Your PCB is preselected by the navigation, so you can use VIEW.LIST.TARGET to select the appropriate segment definitions.



9. The navigation now selects the PCB PRNT association type. These associations link SENSEG relationships to SEGMENT entities and are used to define an alternate parent when a database is being access using the secondary index.

   If you do not need to specify alternate parents, select CONTINUE.



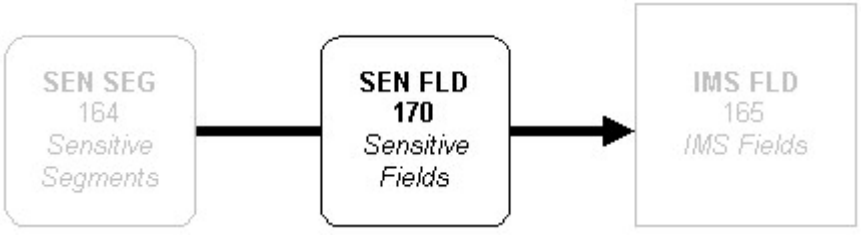
   Optional Navigation Branch: Sensitive Fields

   At this point the navigation branches so that you can define sensitive fields for the PSB.

   If you do not want to include sensitive fields, select the NO SENFLD option from the CONTINUE menu. This takes you back to the PSB entity type and end the navigation.

10. If you select the SENFLD option from the CONTINUE menu, the navigation takes you to the SEN FLD relationship type. These relationships are used to link sensitive segment definitions to IMS field definitions in order to define sensitive fields.

    After you insert the necessary relationships, select the CONTINUE option to return to the PSB Edit window and end the navigation.



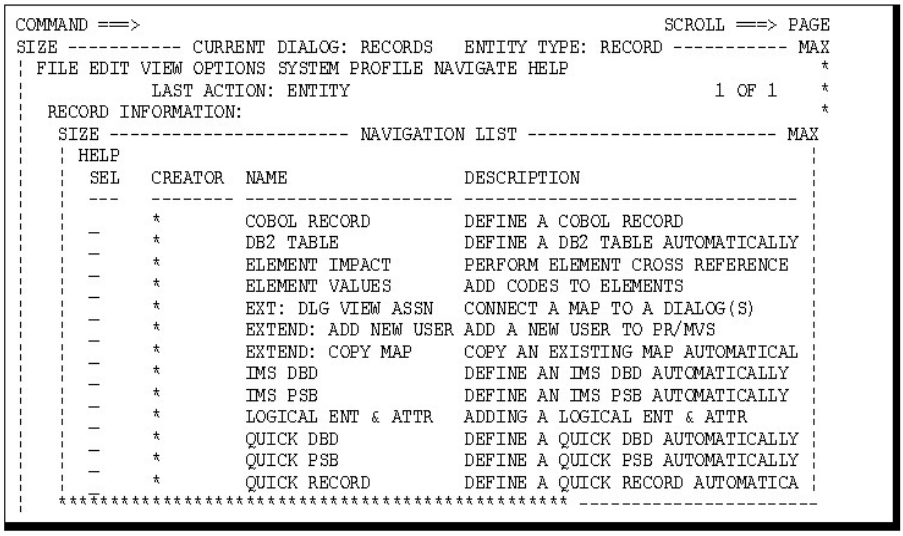
Stop a Navigation

If during the course of a navigation, you need to stop processing, enter the CONTINUE.STOP on the command line. This terminates the navigation at the current point (the NAVIGATE option returns to the menu bar). When you end a navigation in this manner, you can not restart the navigation at the point you left off.

Use a Navigation

To use a navigation

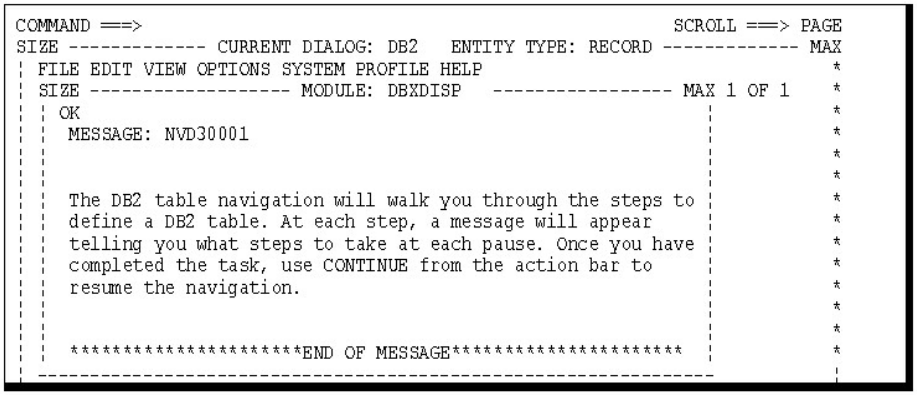
1. Move the cursor over the NAVIGATE option and press Enter or enter N on the command line. The Navigation List window appears containing a list of existing navigations, as shown in the following sample window.



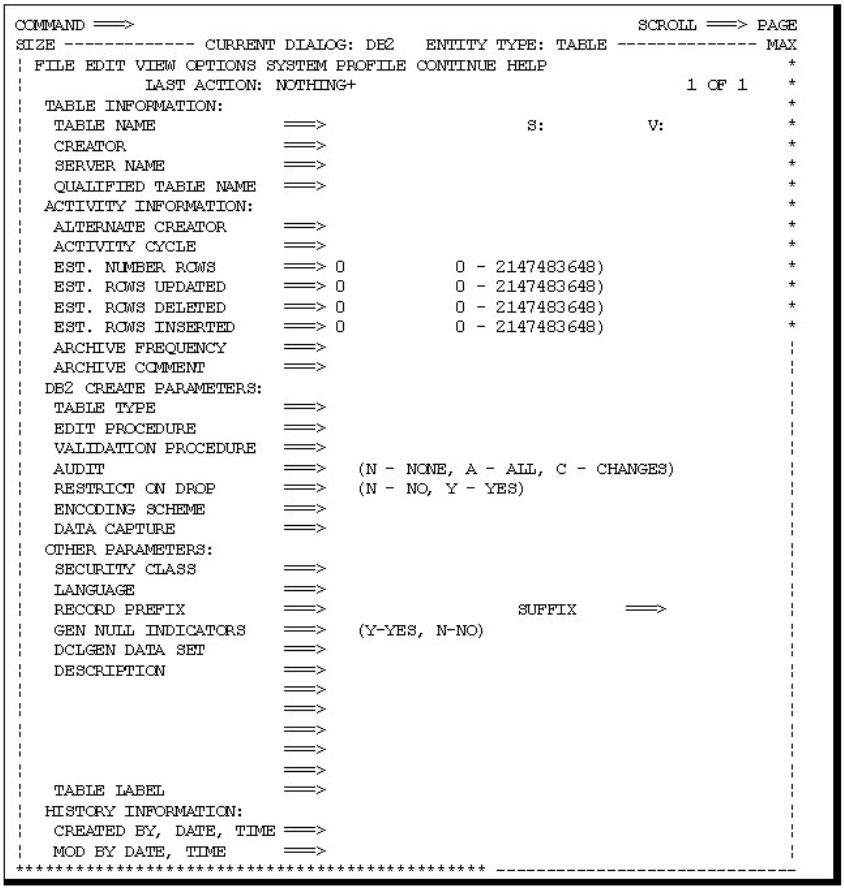
2. Select the navigation by typing an **S** in its Select byte field and press Enter.

   What happens after you select a navigation from the Navigation List window is contingent on the navigation itself.

In the case of the DB2 Table Navigation, the Repository immediately accesses the DB2 dialog, selects the TABLE entity type, and displays a message describing the navigation, as shown in the following sample window.



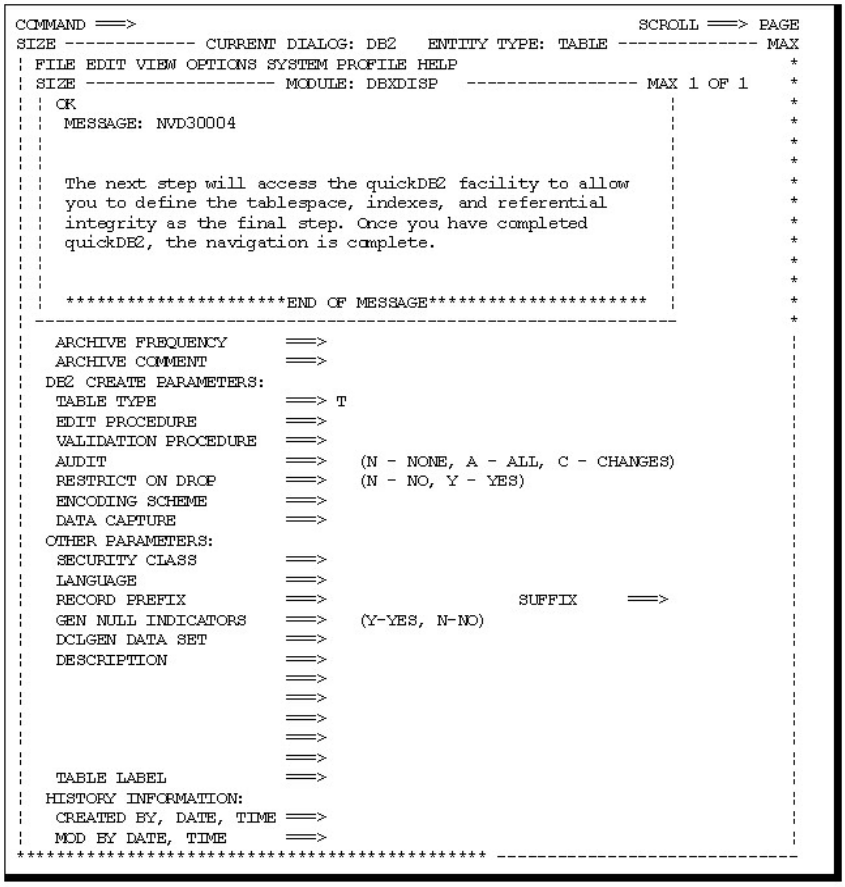
When you acknowledge the message by typing **OK**, the navigation pauses so you can define and insert a TABLE definition, as shown in the following sample window.



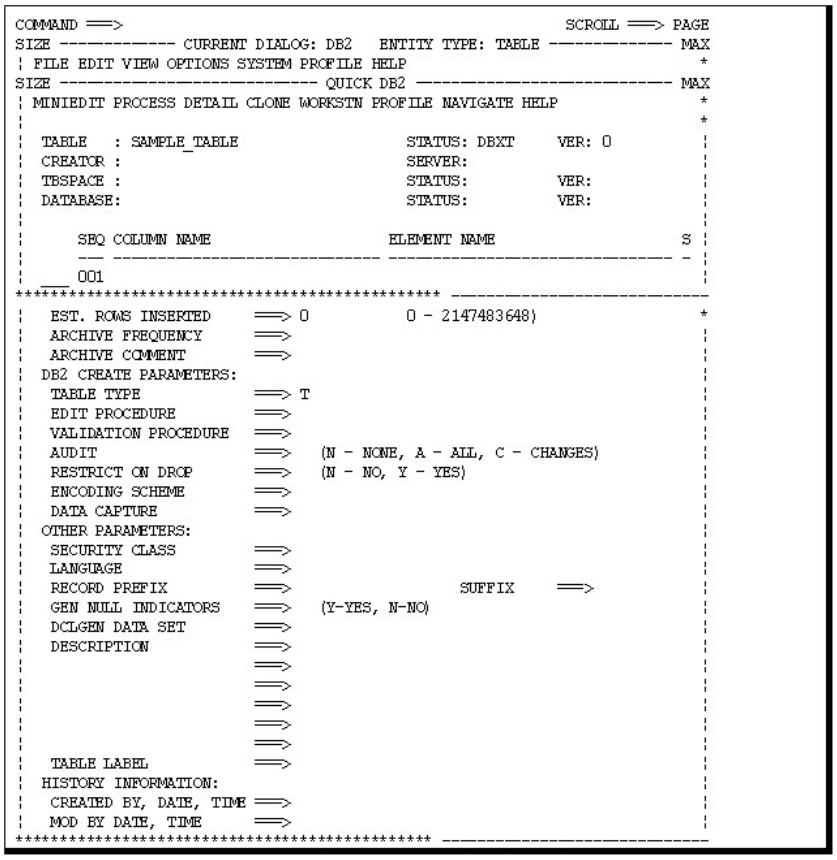
3. Notice that the NAVIGATE option on the menu bar changes to CONTINUE. When you are finished inserting TABLE entities, select this option to proceed with the navigation.

   If this were the end of the navigation sequence, the NAVIGATE option would again be on the menu bar.

4. When you finish defining tables, select the CONTINUE option to proceed with the navigation. A message appears describing the next step of the navigation, accessing the Quick DB2 Facility. See the following.



When you respond to the message, the navigation automatically accesses Quick DB2 using the selected table. At this point, the navigation ends and you process the table using Quick DB2.



Create a Navigation

You can use the Repository Navigator to create your own navigation to perform a specific Repository tasks.

To create an additional navigation, edit the NAVIGATE member in the DATA data set. The layout for this file is:

Column	Position	Len	Description
SEQ_NUM	2	3	Specifies the sequence in which the command will be executed. The numbers must start at 1 and increase in increments of 1. **Note:** You must specify SEQ_NUM.

Column	Position	Len	Description
CREATOR	9	8	Specify an asterisk (*) to designate that this navigation will be accessible to all users.
NAME	18	20	Specify a name for the navigation. This name must be unique (regardless of the creator).
MSG_QUAL	43	3	Specify the qualifier of the message to be displayed before the command is executed or to provide help at a branch.
MSG_ID	47	5	The message ID number.
COMMAND_ TYPE	53	1	Controls the flow of the navigation. Valid values are: **B (branch):** Branches cannot be embedded within one another. The first eight characters of each branch entry is displayed in a pull-down menu that extends from the CONTINUE option. The Help message for a branch displays when you enter a question mark (?) in the Select byte of the option in the pull-down. **E (branch end)**.
COMMAND_ STRING	55	50	Any valid Repository command or the PAUSE option (allowing the user to maintain entities).
DESCRIPTION	107	50	A description of the entire navigation. **Note:** This field must be the same for each entry in the navigation-it is not a description of each step.

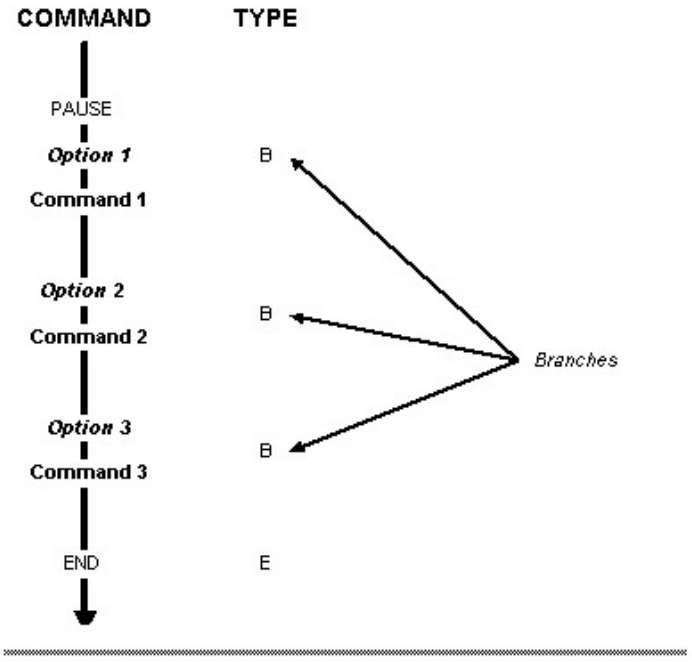
Define Branches

At certain points in some navigations, selecting the CONTINUE option displays a pull-down menu rather than automatically continuing to the next step.

The options on this menu represent a separate processing path, or a branch. The following rules apply:

- A pause must precede each set of branches
- Each set of branches can contain up to 17 options
- After the last branch, you must define a branch end
- Branches cannot be contained within other branches

The following diagram shows the definition of a branch with these options.



Load a New Navigation

To load new navigation

1. Access the Repository Installation Facility and select option 6, Load Data.
2. Set the Navigate switch of the resulting panel to Y (yes) and the Currents Contents switch to R (replace).
3. Submit the job.
4. Verify that it has completed successfully.

Important! The load replaces all existing navigations with the definitions currently in the file. Do not change any information in this file that is not a part of your navigation.

Chapter 13: Migrating Entities

This chapter describes the Repository Migration Facility concepts and procedures as related to the migration of entities.

This section contains the following topics:

Migration

In the Repository, migration or occurrence migration refers to the movement of an occurrence of metadata from one level of the application life cycle to another level.

- Essentially, this *movement* entails changing the status attribute of each occurrence from one value to another. Although such changes could be accomplished using the UPDATE command, the migration process is far more powerful and more efficient because it changes the status of any and all related occurrences in subordinate meta-occurrence types as well as the occurrence being migrated.
- Most importantly, it also enables you to evaluate and reconcile (down to the attribute level) any metadata collisions with existing occurrences that may result from the status changes.

Standard Migration (Migration, No Copy)

A standard migration entails a *move* from one status to another. It does not *copy* definitions to the target status. Therefore, after metadata is migrated from one status to another, the metadata resides only in the target status. The repository automatically removes the metadata definitions from the source status.

Migration-Copy (Migrate-Copy)

The Repository Migration Facility can also *migrate-copy* definitions from one status to another. Using this form of migration, the definitions exist in **both** the source and target status after the processing completes.

Both standard migration and migration-copy handle collision management in the same manner.

Test Migrations

In addition to providing users with a means to move groups of occurrences from one level of the application life cycle to another, the Migration Facility also provides the ability to perform test migrations. A test migration provides information on the effects of a migration without altering the Repository's metadata.

Note: Both real and test migrations can be performed either online or in batch.

The test migration and real migration processes were designed to be thoroughly integrated with each other. You can execute a test migration, evaluate all collisions, choose *responses* to them, and then print and analyze the results of the test migration.

You can then execute a real migration in batch, indicating to the Migration Facility that the batch migration should enforce all the decisions made during the test migration without forcing you to choose the *responses* again. If necessary, you can make modifications to some or all of the responses after the completion of the test migration but before the beginning of the real batch migration.

Authorities to Migrate

To use the Migration Facility, you must have:

- Select, Update, and Delete authority on the source or FROM status.
- Insert and Update authority on the target or TO status.

What Is Migrated?

You can run the Migration Facility against individual metadata occurrences or entire collections of metadata grouped in the Repository workstations.

If individual metadata occurrences are selected and migrated, then the following types of occurrences are evaluated for collisions and migrated. They are:

- The selected entity type occurrence.
- Any entity type occurrences directly linked to the selected occurrence through associations or relationships. These types of occurrences are not restricted by the direction of the connective associations or relationships. That is, whether the dependent occurrence is the source or target of a link to the selected metadata is unimportant.
- The associations and relationships that connect the selected occurrence to the related entity occurrences.
- Any associations or relationships that depend upon the related entity occurrences but do not necessarily depend on the selected metadata. Whether the related entity occurrences act as source or target is not important.
- Any additional associations and relationships that depend on any of the associations and relationships discussed previously. Whether the association and relationship occurrences discussed previously act as source or target is not important.

Migration Order

Note: The first two types described previously are entity type occurrences only. These form the *base objects* and are migrated first.

The remaining types are either associations or relationships, and they are processed after the base objects are migrated.

Dependent Occurrences

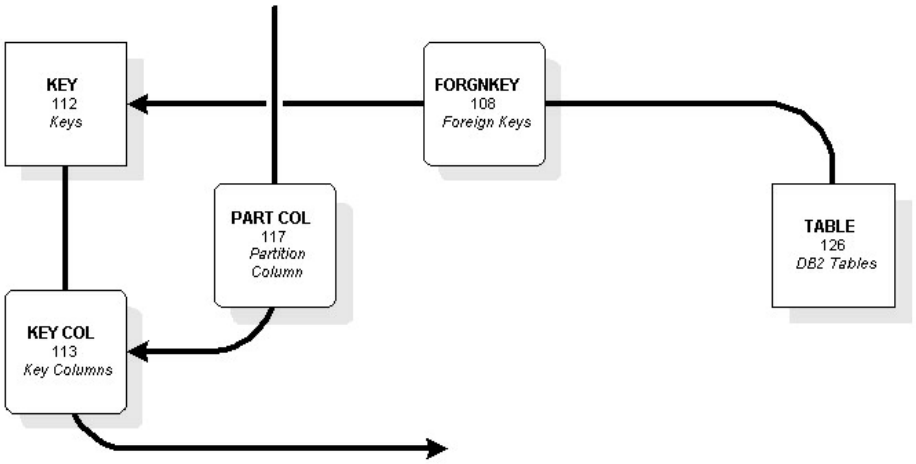
The last four types, discussed previously, are collectively referred to as dependent occurrences because their presence in a migration process is dependent on whether they are related to the selected metadata occurrence.

Therefore, dependent occurrences are any occurrences involved in the migration that were not explicitly selected by the user.

Sample Migration of a TABLE Occurrence

As an example, a small part of an existing dialog (some the meta-entities surrounding the TABLE entity type) are analyzed and presented in this discussion. This example does not attempt to explain the full scope of a migration of a TABLE occurrence.

Within the DB2 meta-model, the TABLE entity type is connected to the KEY entity type through (among others) the FORGNKEY relationship type. From KEY, there is a relationship type called KEY COL, which in turn has another relationship type originating from it called PART COL.



If an individual occurrence of TABLE is selected for migration, then the Migration Facility processes the selected occurrence. Next the Migration Facility processes KEY (#2) and FORGNKEY (#3), then KEY COL (#4), and finally, PART COL (#5).

Occurrences Are Not Directly Related

The entity or relationship occurrence connected to KEY by means of KEY COL, or the occurrence linked to KEY COL by means of PART COL, is not necessarily migrated because those occurrences are not directly related to the selected occurrence of TABLE.

Because this is the case, it is possible that certain associations and relationships could have target occurrences that are lower in the status hierarchy than the source occurrence (because the source was migrated while the target was not). This is technically illegal, you should take great care when migrating individually selected metadata.

Note: The meta-entity connected to KEY by means of KEY COL (a relationship type called COLUMN) is migrated not because it is connected to KEY, but because it is a relationship that involves TABLE. Therefore, it is an example of type #3 described previously.

Migrate Using Workstations

The alternative is to migrate collections of metadata using Repository workstations.

- If you use this method, then every metadata occurrence contained in the workstation is migrated.

  In addition, every association and relationship occurrence involving the workstation occurrences (source or target) is processed, whether or not they are contained in the workstation.

- If the entity or relationship occurrences that are connected to the workstation occurrences by means of the associations and relationships are not in the workstation being migrated, they are not processed by the Migration Facility.

  This may result in target occurrences in a lower status than the source occurrences, which is illegal. Therefore, you should take great care to include all logically linked metadata definitions in the workstation being migrated.

Manage Collisions

When migrating occurrences from one status to another, a primary concerns is the possibility of collision. This occurs when changing an occurrence's status through migration would make that occurrence of metadata a duplicate of an existing occurrence.

If the two occurrences were identical prior to the migration, the collision would not be an issue: you would not need to migrate the occurrence to a status because it already exists there.

When Collisions Become an Issue

Collisions become an issue that you must resolve, when there are differences between analogous occurrences, either at the attribute level, the dependent occurrence level, or both. In these cases, you must decide which version of the occurrence to retain.

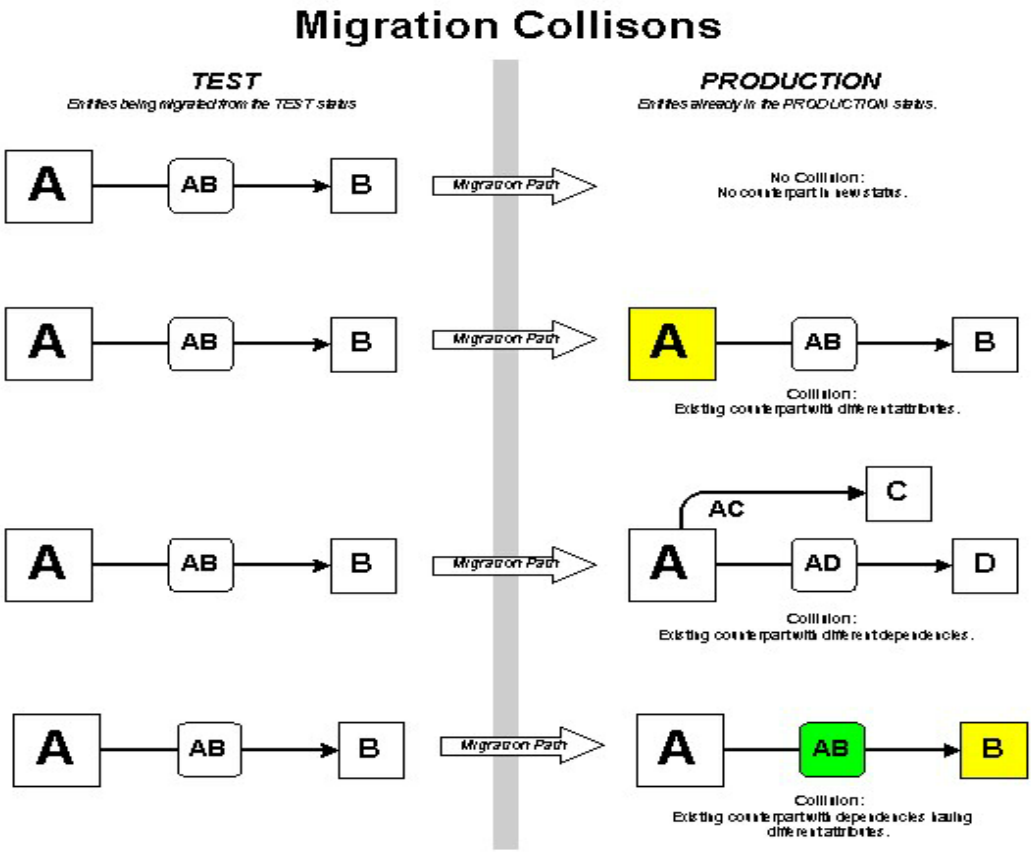
When you use the Migration Facility, it compares the occurrences being migrated against any analogous occurrences that already exist in the target status. If differences are found in *significant* attributes (significance determined by the Repository Administrator), the Migration Facility prompts the user for the appropriate actions or responses to resolve the collision.

Note: If the values found in a given attribute of source and target occurrences differ, those differences will not cause a collision if that attribute has not been identified as significant by the Repository Administrator.

For more information, see the discussion about the Repository maps and the Compare on Migrate field in the *Administration Guide.*

Note: The terms FROM and TO are used throughout this chapter to describe the occurrences being processed. The FROM occurrences are those that are being migrated, while the TO occurrences are those that already exist in the target status.

Some of the possible reasons for an occurrence collision during migration are shown in the following diagram.



The diagram shows an occurrence **A** migrating from a TEST status to a PRODUCTION status. In the first example, A has no counterpart in the destination status, so there is no collision.

The remaining examples caused a collision for the following reasons:

- **Existing Counterpart has Different Attributes**. In the second example, there is already an occurrence **A** with a PRODUCTION level status. Though this occurrence has the same dependencies as the TEST version, some of the attributes of the PRODUCTION version are different from those of the TEST version.

  In this case, the user decides which attribute values should be retained in the PRODUCTION status after the migration.

  Existing Counterpart has Different Dependencies. The third example represents another type of collision. An identical occurrence **A** exists in the PRODUCTION status, but it is linked to occurrences from different meta-entities than those of the TEST version.

  In these situations, the user decides which relationships and associations should be retained in the PRODUCTION status.

- **Existing Counterpart has Dependencies that have Different Attributes**. The final example represents a third type of collision. The occurrence **A** in the PRODUCTION status is identical to its test counterpart, and it is linked to an analogous set of dependent occurrences. However, the attributes of the dependent occurrences vary between the TEST and PRODUCTION versions.

  In these situations, the user decides which version of the attribute values should be retained in the dependent occurrences once in the PRODUCTION status.

The Migration Facility

Using the Migration Facility, you can migrate occurrences in two ways:

- Individually
- By workstation

The first action in the migration process depends on which of these methods you use.

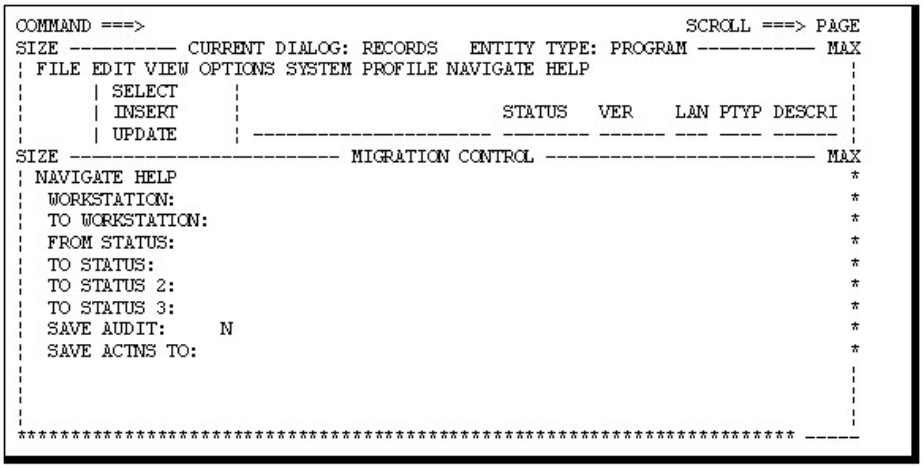
- If you migrate individual occurrences, you need to select (tag) them prior to starting the Migration Facility. Executing the Migration Facility on these occurrences migrates not only the tagged occurrences, but also all the dependent occurrences that are linked to the tagged occurrence.
- If you migrate a workstation, you can omit the tagging step and proceed directly to the Migration Facility.

  During a workstation migration, the Repository migrates all occurrences found in the FROM workstation and all associations and relationships connecting them to dependent occurrences, whether or not the associations and relationships or the dependent occurrences are in the FROM workstation.

  The dependent occurrences themselves are not migrated.

The Online Migration Control Window

You use the Migration Control window to start the migration process. To open the Migration Control window, select EDIT.MGRATION.MIGRATE from the menu bar. The following is a sample Migration Control window.



The following sections describe the fields in the Migration Control window.

Note: The field names appear in all uppercase in the Repository window (for example, TO WORKSTATION) and appear in mixed case in text (For example, To Workstation field).

The Workstation Field

Use the Workstation field to collectively process a group of occurrences that are defined as a workstation.

- Enter the name of the workstation to be migrated, or leave it blank to process the occurrences tagged at the Edit window
- The workstation you specify in the Workstation field is referred to as the FROM workstation

The To Workstation Field

Use the To Workstation field to specify a destination workstation for the migration.

- The workstation you specify in the To Workstation field is referred to as the TO workstation.
- Specifying a TO workstation reduces the scope of the search for potential migration collisions, and therefore helps reduce the number of potential collisions.

When you specify both a FROM and a TO workstation, the last step of the migration process renames the FROM workstation to the TO workstation. This effectively makes the TO workstation a union of both of the workstations involved in this migration.

Specify a To Workstation Field Recommendation

Specifying the To Workstation field is recommended only when the FROM and TO workstations are different incarnations of the same logical grouping.

- For instance, if you have an ALPHA and a BETA workstation, they should never be migrated into one another, no matter how similar they are to one another. Whichever workstation acted as the FROM workstation would cease to exist because it merged with the other workstation.
- However, if one was migrating a recently modified version of the ALPHA workstation into the original version of ALPHA, it would not only be acceptable but preferable to specify a TO workstation.

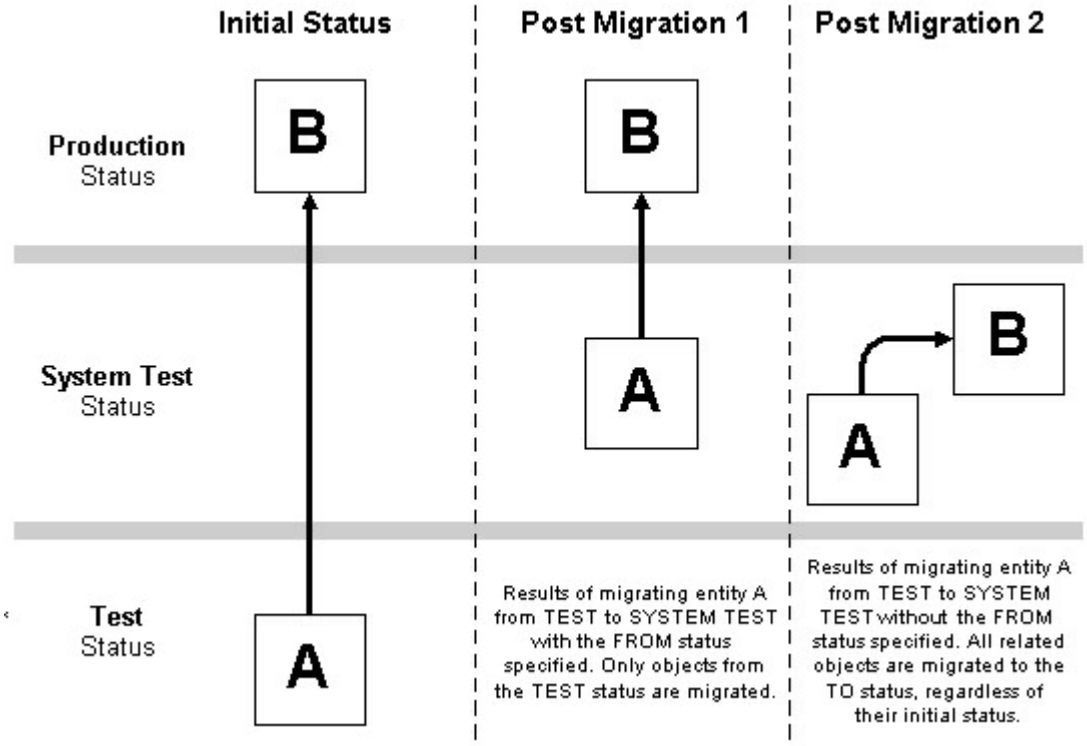
Leave the To Workstation Field Blank

If you leave the To Workstation field blank, then all the definitions with the target status are evaluated for collisions, not only those that are found in a given workstation. Additionally, the post-migration definitions will be found in the FROM workstation.

The From Status Field

Specify the From Status field to prevent the migration of dependent occurrences to a status that is at a lower level of the application life cycle.

The following diagram shows this concept.



In this case, an occurrence **A** migrating from a TEST status to a SYSTEM TEST status. Occurrence A is linked to occurrence B which is in a PRODUCTION level status. Because the PRODUCTION status is a more advanced stage of the application life cycle than the SYSTEM TEST status, one may not want occurrence *B* to be migrated to SYSTEM TEST.

Specifying a From Status of TEST during the migration of occurrence A would ensure that any dependencies in a status other than TEST would remain where they are and that their respective ties to occurrence A would remain intact.

The From Status Field When Migrating Workstations

You can also use the From Status field when migrating workstations; only those workstation occurrences having a status that matches that specified in the field migrate. Each of the other occurrences in the workstation keeps its respective status.

Leave the From Status Field Blank

If you leave the From Status field blank, all dependent occurrences migrate to the TO status, regardless of their initial status. As shown in the previous figure, this can result in an occurrence being migrated *backwards*. That is, to a lower level in the application development life cycle.

Again, this field also applies to workstations. If you leave the From Status field blank, all the occurrences in the workstation migrate to the TO status, regardless of their respective initial statuses.

The To Status Field

Use the To Status field to specify the status to which the selected occurrences (or specified workstation) are to migrate. The migration cannot start until you enter a valid status in this field.

Note: Before you can run a migration, the administrator must create a migration path between the FROM and TO statuses. For information on adding a migration path to control tables, see the *Administration Guide*.

The To Status 2 and To Status 3 Fields

Use the two optional fields, To Status 2 and To Status 3, to provide alternate status types in which to search for occurrence collisions during a migration process.

- If the Migration Facility does not find a match at the To Status level, it checks these two status types for matching occurrences.
- If no collisions are found in any of the three status types, the selected occurrences migrate to the TO status.

When the Migration Facility finds collisions at the To Status 2 or To Status 3 levels, the migrated occurrence is processed against the occurrence found in the alternate status.

- The resulting occurrence status is not changed to the primary target status (TO status).
- This effectively provides the ability to re-use occurrences that already exist at levels of the application life cycle different from those specified in the From Status and To Status fields, which in turn aids in the reduction of redundant data and the enforcement of standards.

Note: It should be noted that specification of these alternate TO statuses affects the performance of the migration.

The Save Audit Field

Use the Save Audit field to specify that the audit information (MOD_BY, and so on) of the FROM occurrence should be moved to the TO definition for any UPDATEs. This means that the audit information imported from CASE tools is maintained after the migration. Specifying a **Y** in this field moves the audit values into the TO status.

The Save Actns To Field

Use the Save Actns To field to specify the data set used to store the responses to the collisions supplied by the user during a test migration. You can then use these responses as input to subsequent test migrations, eventually determining which actions will be taken during the real migration of the metadata in question.

Note: The Migration Facility uses this data set as input for that real migration (in batch) as well. You must allocate this sequential or partitioned data set before the test migration. It should be fixed block and have a LRECL = 255.

Perform Online Migrations

To begin an online migration

1. Select EDIT.MGRATION.MIGRATE. The Online Migration Control window appears.
2. Complete the fields in the Migration Control window.

   Workstation

   Specify the name of the workstation from which you want to migrate data.

   To Workstation

   Specify the name of the workstation to which you want to migrate data.

   From Status

   Specify the status from which you want to migrate data.

   To Status

   Specify the status to which you want to migrate data.

   To Status 2 and To Status 3

   Specify alternate statuses in which you want to search for possible collisions during migration.

   Save Audit

   Enter Y if you want to migrate audit information of the FROM occurrence to the TO definition. Enter N if you do not want to migrate audit information.

   Save Actns To

   Specify the data set in which you want to store responses to collisions during a test migration. You can include this information as input during subsequent test migrations and the real migration.
3. Once you enter the appropriate values into the Migration Control window fields, press Enter to start the migration process.

Match Occurrences

The Migration Facility begins by searching for occurrences in each of the three To Status fields that have names that match those of the definitions being migrated.

- If no match is found for a particular occurrence, the occurrence migrates to the status specified in the first To Status field.
- If a match is found, the matching occurrences are compared to determine if they are the same, based upon the attributes flagged as significant on the map.

  For information about map attribute definitions and the compare on Migrate field, see the *Administration Guide.*

Match Attributes and Dependencies

The following points describe the actions taken when certain conditions are met:

- If the attributes of an occurrence are found to be identical to its match, the dependencies of both the FROM and TO occurrences are compared.
- If the dependencies match as well, no action is taken against the version in the TO status and the version in the FROM status is deleted.
- If the attributes or dependencies of the FROM and TO occurrences are not identical, they (and a list of their differences) appear in one of two Migration Collision windows.

Comparing Versions

If the compare_migrate flag for column version is set to 0 for compare version only, then a collision is performed against the corresponding object with the matched version in the target status.

If the compare_migrate flag for column version is not 0, then comparison is performed against all versions for an exact match.

- If no match is found, the occurrence with the matching version number is involved in the collision with the FROM occurrence.
- If no matching version is found or if that version was previously used in the migration, the occurrence with the last version found is involved in the collision with the FROM occurrence.

- If there are no available versions to be collided against a new version is created in the target status.
- If there is no matching version in the target status, the FROM occurrence is moved.

For more information on MAT ATTR definitions and the Compare on Migrate Flag, see the *Administration Guide*.

The Entity Collision Window

The appearance of the Entity Collision window indicates a difference at the attribute level between the FROM and TO versions of the *base objects* being migrated. That is, this window displays collisions for not only the metadata occurrences that you select and are currently being migrated, but also any dependent entity type occurrences directly linked to the migrated occurrences.

The fields in the Entity Collision window are described in the following sections.

Row Types on the Entity Collision Window

The Entity Collision window contains:

- On the occurrence row, the name and entity type of each of the base object entity type occurrences being migrated and for which the process has found a collision.
- Beneath each occurrence row are the *attribute rows*, listing every attribute type with different values between the source and target definitions. The varying values themselves appear on these attribute row.

Collisions Based on Attributes Flagged as Significant

When the Migration Facility attempts to identify differences between a source and target definition to determine whether a collision should occur, it only looks at the attributes that the Repository Administrator has flagged as significant.

Map Attributes

Once the Migration Facility finds a difference in value between the source and target definitions for one significant attribute type, then the values for every attribute on that map with differing values appears in the Entity Collision window (whether or not the other attributes are deemed significant by the Repository Administrator).

If values differ for certain attributes that have not been deemed significant, then no collisions occur.

Extended Text Attribute Row

Note the attribute row containing the label *DB EXCEL TEXT*. This row represents the extended text for the colliding occurrences. Though the extended text attribute is not used when determining if two occurrences are identical and will not by itself cause a collision, this DB EXCEL TEXT attribute row always appears for any colliding occurrences. It is treated like any other attribute.

Responses to the Entity Collision Window

In the Entity Collision window, you specify options for occurrence rows, tag attribute rows, and select the PROCESS option.

Specifying Occurrence Rows

For each of the occurrence rows, specify one of the following options in the Select byte:

Option	Description
S	The FROM and TO versions should be considered the same, which will cause the Migration Facility to default to the TO version, do not move the FROM definition, and do not overwrite the TO definition. Later in the migration process, the dependent relationships of the FROM definition are still evaluated for collisions against the dependent relationships of the TO definitions.
M	Move the FROM values of the conflicting attributes to the TO version of the occurrence. When using this option, you need to tag (with an **S** on the Select byte) each row representing attributes that should be moved from the FROM occurrence to the TO occurrence. The current TO value is retained for any attribute rows not tagged.
I	Ignore the FROM definition and do not consider it the same as the TO definition. For the remainder of the migration process, the Migration Facility acts as if the FROM occurrence and its dependencies did not exist. At the conclusion of the process, the Migration Facility deletes the FROM occurrence and its dependent relationships.
N	Migrate the FROM occurrence as a *new* version of the TO definition. The dependent occurrences will not be compared for collisions; they will be moved over automatically (and provided new versions if necessary).

Note: Entering a ? in the Select byte of an occurrence row displays a message listing and explaining the responses described in the preceding table.

Tagging Attribute Rows

For attribute rows, you tag the rows (by specifying an **S** on the Select byte) to determine whether or not the values in those attributes will be moved from the source definition and overwrite the value in the target definition.

Processing Entity Collisions

To process these types of collisions and move on to the next step in the migration

1. Type one of the letters listed above in the Select byte of the occurrence rows.
2. Tag the desired attribute rows, if appropriate.
3. Select the PROCESS option from the Migration Collision window menu bar. Any occurrence rows with no responses or attribute rows that are not tagged when the PROCESS command is activated will not be migrated (occurrence rows will default to S; attribute rows will be left blank by default).

Note: The MINIEDIT, IMPACT, and TEXT functions are available in the Entity Collision windows in order to help determine how to process the displayed occurrences. The FROM and TO designations in these functions apply as above.

Entity Names Longer than 32 Characters

The Entity Collision window cannot display entity names longer than 32 characters. If entities with longer names are involved in a migration collision, the Collision window displays the first 16 characters, a | symbol, and then last 15 characters. For example, an entity named:

```
SAMPLE_ENTITY_WITH_A_RIDICULOUSLY_LONG_NAME,DBXT,00
```

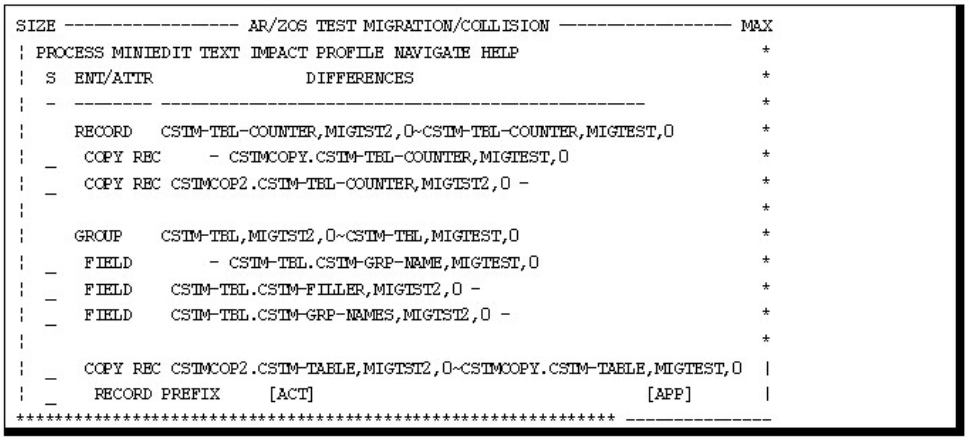
displays as:

```
SAMPLE_ENTITY_WI|OUSLY_LONG_NAME,DBXT,00
```

The Association/Relationship Collision Window

On exiting the Entity Collision window, the following occurs:

- If neither the FROM occurrences nor the TO occurrences have dependent relationships or associations, then the migration process completes and control returns to the Edit window.
- If dependent associations or relationships do exist in each status and they do not perfectly match, or they exist in one status and not the other, then one or more successive Association/Relationship Collision windows could display, as shown in the following sample window.



Occurrence row

Base object entities with varying dependencies

Dependency rows

Dependency with attributes that vary between TO and FROM status

TO dependency not present in the FROM status

Use one of the following commands to respond to collisions in the Association/Relationship Collision window:

Command	Description
M	Move the FROM dependency to the TO status.
I	Ignore (and delete) the FROM dependency.
K	Keep the dependent definition and attach it to the migrated occurrence.
D	Delete the TO dependency.
S	Consider the dependencies as the same. The TO definitions are not overwritten by the TO definitions.
N	Migrate the FROM dependency as a new version of the existing TO dependency.

Row Types in the Association/Relationship Collision Window

The Association/Relationship Collision windows contain:

- On the occurrence row, the name and meta-entity type of each of the *base object* entity type occurrences being migrated for which the process has found a dependency association or relationship collision
- Beneath each occurrence is a list of the association and relationship types and names of each of the colliding occurrences (the dependency rows)
- Underneath certain dependency rows are attribute rows, listing the differences between two dependencies at the attribute level

Occurrence Rows

On the occurrence rows in the Association/Relationship Collision window (the rows without Select bytes), the names of the two colliding entity type occurrences are separated by a dash (-). The FROM occurrence is to the left of the dash, the TO occurrence is to the right of the dash.

Dependency Rows

The same format holds for dependency rows displayed below the occurrence row.

- If the dependency association or relationship collision is caused by the existence of a FROM dependency with no analogous TO dependency, no occurrence is listed to the right of the dash
- If the collision is caused by the existence of a TO dependency where no analogous FROM dependency exists, no occurrence is listed to the left of the dash

Responses to the Association/Relationship Collision Window

This section addresses dependency row specifies key attributes to left, dependency row specifies key attributes to right, dependent association or relationship on the dependency row, and entity names longer than 32 characters.

Dependency Row Specifies Key Attributes to Left

If a dependency row specifies key attributes to the left of the dash, but not to the right (that is, a FROM dependency exists without a corresponding TO dependency), then specify one of the following options on the Select byte of the dependency row:

- Move (M)
- Ignore (I)

Note: In these situations, you must specify an M or an I on the dependency row. There is no default a value and the Migration Facility will be unable to continue processing until you specify a valid option.

Dependency Row Specifies Key Attributes to the Right

If a dependency row specifies key attributes to the right of the dash but not to the left (that is, a TO dependency exists without a corresponding FROM dependency), specify either of the following options:

- Keep (K)
- Delete (D)

Note: In these situations, you must specify either a K or a D on the dependency row. There is no default value and the Migration Facility will be unable to continue processing until you specify an option.

Dependent Association or Relationship on the Dependency Row

If both a FROM and TO dependent association or relationship are displayed on the dependency row, the collision is caused by a difference in their respective attributes.

The differing attributes are listed by attribute type beneath the dependent occurrences in the attribute rows.

The same options that are available on the Entity Collision window are also used in the Association/Relationship Collision window to resolve these types of collisions. The only difference is that now dependent associations and relationships are the concern, not the entity occurrences that have already been migrated.

Any dependency row Select bytes of this nature that are blank, when the PROCESS command is activated, defaults to S (same).

The following figure shows two examples of dependency differences.

- The first group of rows concerns the migration of an ELEMENT occurrence, FILLER. It was being migrated from the MIGTST2 status to the MIGTEST status, but collided with its TO version.

  The reason for the collision was a difference in the attributes of a FIELD relationship, specifically CSTM_TBL.FILLER.

  Though both the FROM and TO versions of the ELEMENT had this relationship, the FROM and TO versions of this relationship had different values for the attribute SEQ_NUM. The value for the FROM version is 10 while the value for the TO version is 6.

- The second group of rows concerns the migration of a GROUP occurrence, CSTM-TBL. It was being migrated from the MIGTST2 status to the MIGTEST status when it collided with its TO version due to two types of dependency differences.

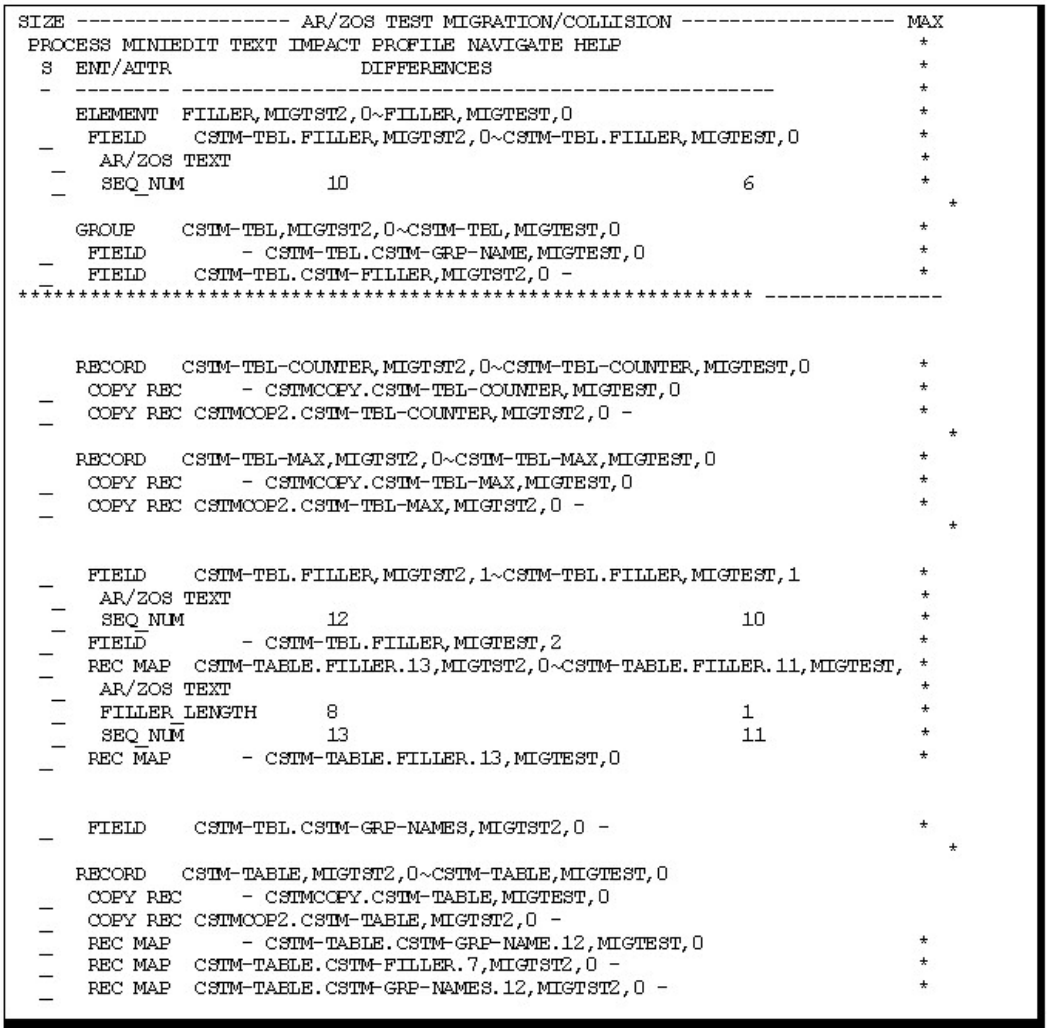
- The first collision is a TO with no FROM collision: the TO version of CSTM-TBL had a dependent occurrence of the FIELD association type called CSTM-TBL.CSTM-GRP-NAME which the FROM version of CSTM-TBL did not have. The user must choose to keep or delete this dependent occurrence.

- The second collision is a FROM with no TO collision: the FROM version of CSTM-TBL had a dependent occurrence of the FIELD association type called CSTM-TBL.CSTM-FILLER which the TO version of CSTM-TBL did not have. The user must choose to move or ignore this dependent occurrence.

As discussed previously, the attribute AR/ZOS TEXT always appears in collision windows as one of the *colliding* attributes when attribute differences are discovered, whether or not the extended text is actually different. Note that if the only difference between a FROM and a TO occurrence is in the extended text, then no attribute collisions occurs.

Note: As in the Entity Collision window, the MINIEDIT, IMPACT, and TEXT functions are made available in the Association/Relationship Collision window in order to help determine how to process the displayed occurrences. The FROM and TO designations in these functions apply as above.

Any attribute row Select bytes that are blank default to blank.



Entity Names Longer than 32 Characters

Like the Entity Collision window, the Association/Relationship Collision window does display entity names longer than 32 characters. If entities with longer names are involved in a migration collision, the Association/Relationship window displays the first 16 characters, a | symbol (vertical bar), and then last 15 characters.

Respond to Multiple Collisions with One Command

Because there can be many collisions displayed in either the Entity Collision window, Association/Relationship Collision window, or both (especially when migrating workstations), you may be forced to scroll through the windows to provide a response to every collision.

In an attempt to accelerate the process, the Migration Facility has a special *tag* command that provides a user-specified response to every instance of a certain collision type included in the window. In this manner, you can provide a response to multiple collisions with only one command.

The possible formats for this TAG command are described in the following table. For an online description of this command, choose the HELP function on the menu bar of the Migration Facility Collision windows.

Command	Description
TAG M *x*	Places the user-specified value (**x**) in the Select bytes of all the *matching* collisions (caused by different values in corresponding attributes). The valid values of *x* are: S (Same), M (Move), I (Ignore), or N (New).
TAG F *x*	Places the user-specified value in the Select bytes of all the *FROM with no TO* dependency collisions. The valid values of *x* are: M (Move) or I (Ignore).
TAG T *x*	Places the user-specified value in the Select bytes of all the *TO with no FROM* dependency collisions. The valid values of x are: K (Keep) or D (Delete).

Once you perform the *tag* operation, you can change individual responses to suit the needs of the migration.

Batch Migrations

You can perform migrations either online or in batch.

- Online migrations provide a greater level of control in managing collisions; however, they monopolize the user logon session.
- By contrast, batch migrations allow the user to work on other tasks in their logon session. The user loses some control over how collisions are resolved during the migration.

Specify Collision Resolution in Advance

While all the options available for online migrations are available to batch migrations, collision resolution is determined by the type of collision rather than by the individual occurrences. The user specifies in advance how different types of collisions should be resolved.

Batch Migrations Require a FROM Workstation

Batch migrations also require the use of a FROM workstation as it is the only means of indicating to the Migration Facility which occurrences of metadata should be migrated.

Fields That Specify Actions

Match Action, To Only Action, and From Only Action fields are required. You use them to specify the actions that should be performed to resolve collisions resulting from the migration.

Specifying How Classes of Collisions are Resolved Before Processing

When performing a migration online, you can respond to each occurrence collision individually, specifying which attributes and dependencies should be retained or deleted on a collision-by-collision basis. Because such interaction is not possible in batch, you must decide how each of the three general classes of collisions will be resolved before initiating the processing.

You use the Saved Actn File, Save New Actn To, Lock To Workstn, and Copy Rels For Tgt fields to control how test and batch migrations interact with each other. Leave these fields blank if the batch migration is not to use previously-made responses to collisions as input.

Start a Batch Migration

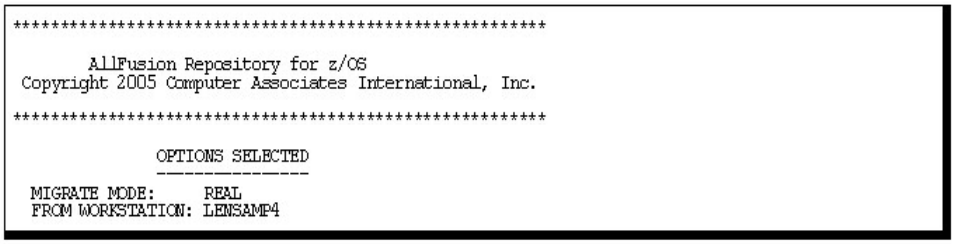
To start a batch migration

1. Select EDIT.MGRATION.MIGBATCH. The Batch Migration Control window appears.
2. Complete the fields in the Batch Migration Control window. The first seven fields of the Batch Migration Control window are analogous to those in the Online Migration Control window. The remaining fields are unique to the Batch Migration Control window. For help with an entry field, place a question mark (**?**) in the field; then select HELP from the menu selection. Once you specify the appropriate options, press Enter. The migration JCL displays in an Edit session for review. Use the **SUB** command to submit the job.

Note: You cannot execute both an online migration and a batch migration job simultaneously.

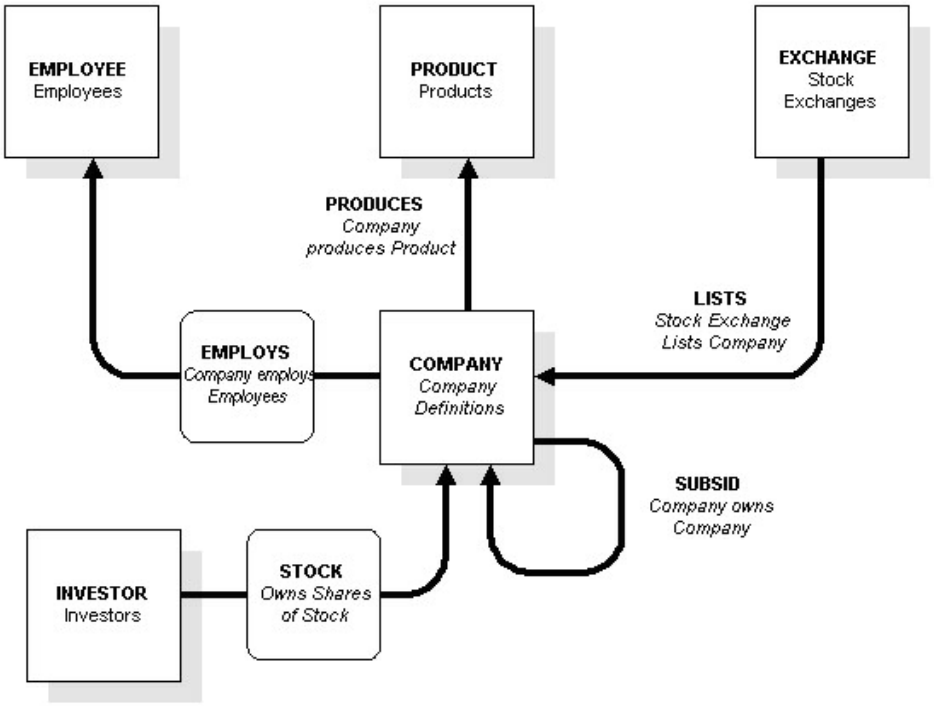
Batch Migration Output

The following is sample output from a batch migration. The migration options are listed at the top and a message indicating the success or failure of the migration is displayed at the bottom. A message "processed ??? entities" appears after every 100 occurrences are processed.



Cross Reference Window Example

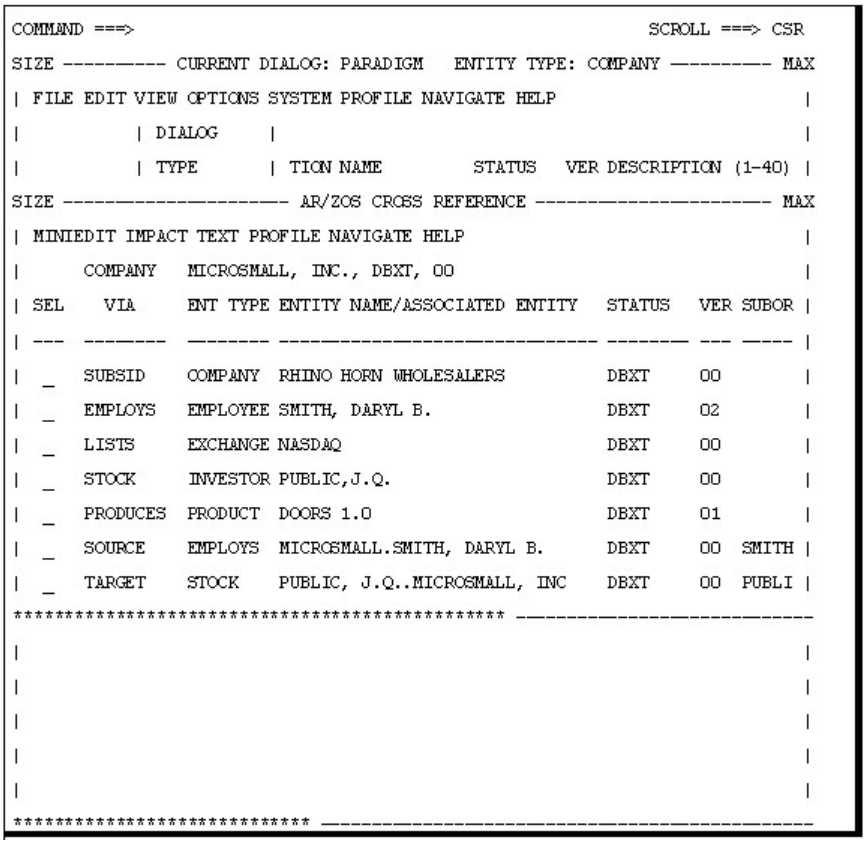
The following sample data model shows an entity type, COMPANY, whose occurrences both use and are used by other entities.



When you generate a Cross Reference window for a COMPANY entity, the window lists any entities that the selected entity uses and entities that are used by the selected entity. These can include:

- COMPANY entities linked to the selected entity through a SUBSID association (either as the source or target)
- EXCHANGE entities linked to the selected entity through a LISTS association
- EMPLOYEE entities linked to the selected entity through an EMPLOYS relationship
- INVESTOR entities linked to the selected entity through a STOCK relationship
- PRODUCT entities linked to the selected entity through a PRODUCES association

The following illustration shows an example Cross Reference window. This window was generated for a COMPANY entity as indicated in the line just below the Cross Reference menu bar.



The first five data rows in this window represent entities that are linked to the selected COMPANY entity.

- The first row shows another COMPANY entity linked to the selected COMPANY entity through a SUBSID association
- The second row shows an EMPLOYEE entity linked to the COMPANY entity through a EMPLOYS relationship
- The third row shows a EXCHANGE entity linked to the COMPANY entity through a LISTS association
- The fourth row shows an INVESTOR entity linked to the COMPANY entity through a STOCK relationship

- The fifth row shows a PRODUCT entity linked to the COMPANY entity through a PRODUCES association
- The sixth and seventh rows contain the actual relationships linking the EMPLOYEE and INVESTOR entities to the COMPANY entity
  - The EMPLOYEE and INVESTOR entities appear in the Subordinate Entity column and can be observed by scrolling the Cross Reference window to the right
  - The word SOURCE and TARGET in the Via column indicates whether the selected COMPANY entity is either the source or target of the relationship in each particular row

Perform a Migration-Copy

Executing a migration-copy (also referred to as migrate-copy) against a set of definitions will result in those definitions existing in both the source and target status. Collision management is handled as with standard migrations.

To begin an online migration-copy, select EDIT.MGRATION.MIGCOPY from the menu. The resulting Migration-Copy Control window is identical to the Online Migration Control window.

To begin a batch migration-copy

1. Select EDIT.MGRATION.MIGBATCH from the menu.
2. Complete the fields on the Batch Migration Control screen. The first seven fields on this window are analogous to those on the Batch Migration Control screen.
3. Make sure to specify **C** in the Test, Real, or Copy field of the resulting Batch Migration Control screen.

Executing one of these commands (EDIT.MGRATION.MIGCOPY or EDIT.MGRATION.MIGBATCH) displays Migration Collision windows identical to those that display during a standard migration. This feature enables you to create duplications of definitions in the target status without losing the original definitions.

Copy Relationships for Target

The Copy Rels For Tgt field is unique to a migration-copy. Copy Rels For Tgt (Copy relationships for target) allows relationships to be copied for the target of relationships during a migration-copy.

- Setting this parameter to N for relationships to be copied for the target of relationships during a migration-copy if both the source and the target are in the item (status or workstation) being copied.
- Setting this parameter to Y causes relationships that reference an item being copied as the target (and the source is not part of the item being copied) to be copied as well. This causes the source entity to point to both the old and the new item created from the copy.

Note: Use Copy Rels For Tgt with migration-copy only.

Migration-Copy with Workstations

Using migration-copy with workstations differs subtly from that of standard migrations.

- In a standard migration, specifying only a FROM workstation in the Migration Control window results in the post-migration definitions being placed in that same workstation.
- Specifying both a FROM and a TO workstation results in the post-migration occurrences being placed in a workstation that is the concatenation of both the FROM and TO workstation.

Executing a migrate-copy while specifying only a FROM workstation results in the new definitions being placed outside any workstation. The occurrences are not placed in the FROM workstation because workstations containing both versions of the same definition (only the status is different) serve little or no purpose. However, when one specifies both the FROM and TO workstation, the new definitions are placed in the TO workstation.

Check-Out Using Migration-Copy

The migration-copy mode of the Migration Facility has proven to be very convenient when one needs to check-*out* a production definition and modify that definition in the test status. Obviously, you would not want to remove the definition from the production status in the repository, because the repository must reflect the fact that the analogous production object is still being used in the site production environment.

The migration-copy is ideal in this situation because the definition will *exist* in two statuses, allowing the user to make all the necessary changes in the test status while leaving the production definition untouched. When the test object is fully tested and moved into the site production environment, the user only has to migrate the test definition to the production status.

The Migration Control window, Entity Collision window, and Association/Relationship Collision window in the online migrate-copy mode all appear and behave in the same manner as the collision windows of a standard online migration.

Simulated (Test) Migrations

Prior to migrating an occurrence or group of occurrences, it is strongly suggested that you *test* the migration.

To begin an online test migration

1. Select EDIT.MGRATION.MIGTEST.
2. The resulting Test Migration Control window is identical to the Online Migration Control window.

To begin a batch test migration

1. Select EDIT.MGRATION.MIGBATCH. the resultant Test Migration Control window is identical to the Batch Migration Control window.
2. Specify T in the Test, Real, or Copy field.

Executing one of these commands (EDIT.MGRATION.MIGTEST or EDIT.MGRATION.MIGBATCH) takes you through a *simulated* migration of any selected occurrences, displaying migration and collision windows identical to those that appear during a standard migration, but without making the corresponding changes to repository data.

Preview Possible Effects

This feature enables you to see, both online and in report form, the possible effects of migrating an occurrence, group of occurrences, or workstation prior to performing the actual migration.

Focus on Potential Collisions

A test migration concentrates on the potential collisions that would result from a migration, and not on the actual definition of the occurrence being migrated. That is, test migrations concern themselves with the FROM occurrence of metadata in its relation to its TO version, and are not worried about the values of the FROM or TO occurrence's own attributes.

In contrast, real migrations not only perform the migrations but validate the values of the occurrences as well. Therefore, there may be some messages alerting you to a problem displayed during the real migration that were not seen during the previous test migration.

Test Migration Windows

The Migration Control window, Entity Collision window, and Association/Relationship Collision window in the online test mode all appear and behave in the same manner as the collision windows of a real online migration.

Migration Action Files

The following migration situations indicate a need to save the data you enter for the migration:

- An online, real migration can be executed, using the MIGRATE command, but it is not sensitive to decisions regarding collisions made during a previously executed test migration, forcing you to respecify the responses to the encountered collisions.
- A standard batch migration can be submitted, but this method also ignores any previously specified responses.
- In addition, using the standard batch migration process results in the loss of some control because it is necessary to decide how each of the three types of collisions will be resolved prior to initiating the migration, meaning that every instance of a collision type (TO with no FROM, and so on) is treated with the same response as those given to every other instance of that collision type.

Note: There can be no exceptions, no special circumstances.

In recognizing the need to eliminate the requirement that users repeat interactive responses to collisions, while still allowing for special circumstances, the Migration Facility provides the user with the means to save and edit the responses chosen during a test migration, either online or batch. You can use these choices, saved in an *Action* file or *Action data set*, as migration input for either a real batch migration or another test batch migration.

Create an Action File

The steps involved in executing a batch migration using previous test migration responses are as follows:

1. To begin, allocate the Action data set, either partitioned or sequential, to be used to store the responses specified during a test migration. Make this data set fixed block and set LRECL = 255.
2. If you are executing an online test migration, issue the **MIGTEST** command.
   - The resulting window has a Save Actn To field, where you supply the name of the Action data set.
   - Continue as you would with a normal online test migration.
3. If the test migration is to be in batch, issue the **MIGBATCH** command.
   - The resulting window includes (among others) a Test or Real field (to be set to T), a Saved Actn File field, and a Save New Actn To field, where you specify the name of the Action data set.
   - Do not specify the name of the Action data set in the Saved Actn File field in the Batch Migration window. This field should be left blank.
   - Continue as you would with a normal batch test migration.

Use an Action Data Set

During the execution of the test migration, the actions to be taken in response to the encountered attribute and dependency collisions are recorded in the Action data set, in a format identical to the format used in the Entity Collision and Association/Relationship Collision windows.

- The Action data set is similar to a collection of screen prints of the various collision windows displayed during the test migration.
- To the right of these screen prints are some control characters with which you do not need to be concerned.

Note: You should **never** edit any characters in an Action data set (except for the *Select bytes* found in columns #2 and #3) or the Action data set will be invalidated.

Develop a Final Action Data Set

After successfully completing a test migration, you should analyze the Test Migration Report and confirm that the responses to the various collisions resulted in expected and acceptable changes to the metadata being migrated.

- If you are satisfied with the report, move on to the real migration (discussed in the following text).
- If, on examining the Migration Report, you decide to make alterations to the responses specified in the test migration, the suggested method is to directly edit the Action data set and modify the responses recorded there.

  You must then execute another test migration (this time, it **must** be in batch) to investigate the effects of the new responses.

  - Specify the name of the original, recently-edited Action data set in the Saved Actn File field of the Batch Migration window.
  - Specify the name of another data set, the *New Action* data set, in the Save New Actn To field.

This New Action data set has the same physical structure as the original Action data set, which is used to record the results of this new test migration, should at the conclusion of the new test migration, include most of the collisions and their responses that were recorded in the *edited* Action data set, and the new collisions and their responses that resulted from your modifications to the edited Action data set.

The New Action data set should now be considered the authoritative version of the input data set to the current migration. This process should be repeated until you are satisfied with the responses specified in the last, *final* Action data set. At this point, you are ready to begin the real migration that uses the responses stored in this final Action data set as input to the migration process.

Protect the Integrity of an Action Data Set

You can apply a Repository Update lock to all the occurrences found in a TO workstation after a test migration. This action is valuable if the responses to collisions are stored in an Action data set and a significant period of time separates the execution of a test migration from its real migration.

The Update lock prevents modification of any occurrences found in the TO workstation. Any changes to the TO versions of occurrences involved in the migration could significantly alter the collisions that occur during the subsequent migration process, thereby invalidating the information stored in the Action data set.

The Update lock is automatically removed by the execution of the real migration against the TO workstation.

Special Migration Considerations

Special migration considerations include concurrent migrations and locks.

Concurrent Migrations

The Repository does allow multiple users to perform separate migrations simultaneously. Problems can occur when two or more users are attempting to migrate the same occurrences at the same time. You should avoid this situation.

The Migration Facility does not allow multiple users to migrate the same workstation at the same time as different workstations may *overlap* and contain one or more of the same occurrences. Attempting to migrate two overlapping workstations at the same time causes problems.

In the case of test migrations, there should be no problems with overlapping workstations. However, performing a real migration with one workstation and a test migration with another workstation that overlaps the first workstation will confuse the test migration and provide you with inaccurate results.

Migration Administrators

In the interest of avoiding problems such as these, it is recommended that the authority to perform real migrations be limited to a person or small group of people who are knowledgeable about the operation of the Migration Facility in particular and the Repository in general. These *migration administrators* should also remain thoroughly familiar with the Repository metadata and with the recent and planned migration activity in the Repository.

Sites that are interested in implementing some form of migration administration may want to consider reviewing the following scenario: a site may want to design their repository security levels so that authorized users of the Repository could repeatedly perform the test migrations until these users are satisfied with their migration results. These users would then supply the name of their Action data sets to the migration administrators, who would analyze the results, make modifications to the responses, and retest them if necessary, and then perform the real migration.

Locks and Migration

Locks do not need to be removed for test migrations. However, a test migration will reveal which *locked* occurrences would have been modified if it had been a real migration.

The Migration Facility automatically removes any Select and Update locks placed on occurrences by the TO workstation.

Note: The Migration Facility will not remove locks placed on those occurrences by other workstations. If any such *unremoved* locks are Update locks, they must be manually removed prior to the real migration.

Recover from Migration Failures

Migration failures are categorized into the following topics:

- Common processing failures
- Online migration failures
- Batch migration failures
- Cancellations
- Recovery from a failed migration of a workstation
- Cleaning out of a workstation
- Backing up a workstation

Common Migration Processing Failures

Common migration processing failures include validation failures, cardinality failures, duplicate associations/relationships failures, and DB2 errors.

Validation Failures

The failures that occur most often during a migration of data from one status to another are validation failures. These failures occur when the migration process attempts to place an illegal value into an attribute.

These errors are easily resolved by either legalizing the value (by having the Repository Administrator modify the appropriate code table) or by simply changing the value.

Cardinality Failures

Migrations may also run into cardinality failures. This occurs when you attempt to add an association or relationship that exceeds the legal number of associations/relationships involving a given entity type definition.

For example, if a DB2 table is limited to 250 columns, then the migration process displays an error message when you, through responses to collisions, attempt to add the 301st COLUMN definition to a given occurrence of the TABLE entity type.

To fix this error, you must:

- Change the response to the collision
- Delete an existing COLUMN relationship from the target status
- Modify the definition of the relationship type and increase its cardinality

Be careful with this last option: the Repository default cardinalities are not random, but reflect certain restrictions placed on the connection by the DBMS, CASE tool, or programming language being described. For more information about cardinality, see the discussion about the Max Num Source and Max Num Target fields of the META ENT entity type in the *Administration Guide*.

Duplicate Associations/Relationships Are Not Allowed

If the *Duplicate Associations/Relationships are Not Allowed* message appears during the migration process, it means that the migration process is trying to create an association/relationship in the target status where an association/relationship already exists. If you know the link already exists in the target status, you might ask why was not it involved in a collision?

- The answer lies in the contents of the workstation being migrated.
- The workstation most likely contained a base object definition, but did not contain the associations or relationships that involved the base object. Because they were not contained in the workstation being migrated, these associations and relationships were not considered for any collisions.

However, when the migration process attempts to move the base object definition to the target status, it has to *move* or create the dependent associations/relationships in the target status; it cannot ignore them.

Therefore, it attempts to create a new definition instead of colliding with the existing association or relationship because the original link was not considered when collisions were being evaluated.

DB2 Errors

Finally, DB2 errors, such as unavailable resources, lack of DB2 authority, and so on, can also occur. Database Administrators or someone with similar authority and responsibilities can easily resolve these errors.

Online Migration Failures

This section describes fixing and restarting online migrations, failing during a workstation migration, and errors from automatic relationship movements with the use of workstations.

Fix and Restart an Online Migration

In the event that an online migration fails before the first collision window appears, the process ends after displaying an issue-specific message.

If the failure happens after the first collision window displays, the migration returns to the most recent commit point-after the last successful issuance of the PROCESS command.

Both of these situations enable you to fix the problem before restarting the migration. You can resolve these issues by:

- Specifying different responses
- Using the MINIEDIT command available in the collision window to modify the offending occurrences
- Accessing another user's Repository session to modify or legalize the cause of the error in the repository

Failure During a Workstation Migration

If you are migrating a workstation, you need to remove from that workstation any occurrences that have the TO /target status before restarting the migration.

Errors from Automatic Relationship Movements

During an online migration, if an error occurred because of automatic relationship movements, you may be returned to an empty screen (an Entity Collision window with no collisions). In this case, you must correct the error from another user ID (or use MINIEDIT) and then continue the migration by entering the **PROCESS** command.

Errors from Automatic Relationship Movements During a Workstation Migration

If a workstation is being migrated, you must remove from that workstation any occurrences that have the TO/target status before restarting the migration.

Batch Migration Failures

A failure during a batch migration can be tracked by inspecting the SYSPRINT of the migration job. You can find messages in this file indicating that the migration failed while it was attempting to migrate one of the following types of metadata:

- Base object entity type occurrences
- Level 1 associations and relationships, which connect base object entity type occurrences to one another (example: FORGNKEY)

- Level 2 associations and relationships, which connect Level 1 relationship type occurrences to other definitions (example: KEY COL)
- Level 3 associations and relationships, which connect Level 2 relationship type occurrences to other definitions (example: PART COL)

Additional messages indicate how many of the above types of metadata were processed by the batch migration program.

Cancel a Migration

There is no suitable method for canceling a migration. Ending the migration process before completion corrupts the metadata in unpredictable ways.

Therefore, it is strongly suggested that:

- Execute test migrations, carefully analyze the results, and have the test migration approved before real migrations begin
- Alternatively, run the migration in batch, with Commit mode set to E

In case of an error during a test migration, continue to provide valid responses to the various collisions (do **not** END or F3); none of the responses have any effect.

Recover from a Failed Migration of a Workstation

Should a real migration of a workstation fail during the migration processing, you need to recover the data and be able to start the migration again. At this point, your goal is to return the definitions to their pre-migration state and guarantee that any collisions that occurred or would have occurred during the failed migration attempt will occur again in the next attempt.

New Definitions Inserted

Before the failure occurred, the migration program may have inserted some new definitions into the repository (using the TO/target status). These new definitions are always automatically assigned to the FROM workstation, even if a TO workstation was specified in the migration panel.

Existing Occurrences Placed in the FROM Workstation

In addition, the program may have placed existing occurrences, in the target status, into the FROM workstation. This would have been done before the FROM workstation original definitions (having the FROM/source status) have been removed from the workstation and deleted from the repository. Therefore, if a failure of the migration process happened before the original definitions could be removed, you will find both the original (source status) and new (target status) definitions in the FROM workstation.

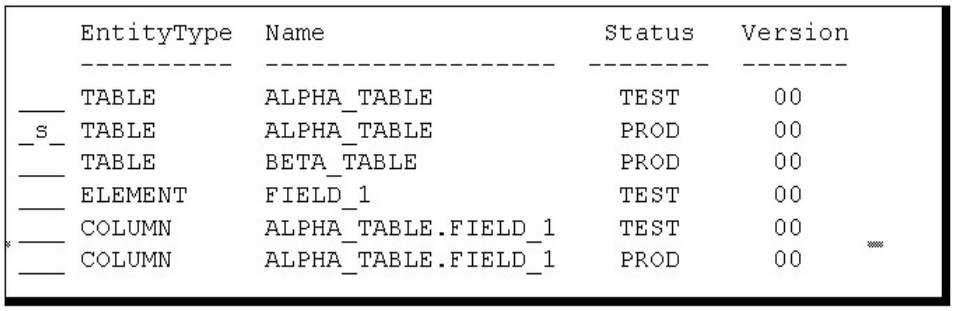
You should not attempt to remigrate the workstation at this point. The original/source definitions would not collide with the new/target definitions because the new/target definitions are in the workstation being migrated. Definitions in a workstation are not collided with definitions found in the same workstation. However, to mirror the previous migration attempt the original/source definitions *must* collide with the new/target definitions. Therefore, you must *clean out* the workstation in question by removing the new/target definitions.

Clean Out the Workstation

Cleaning out the workstation can best be accomplished by removing from the FROM workstation the occurrences that meet the following criteria:

- The occurrence is in the TO/target status.
- The occurrence has a corresponding definition in the FROM/source status which is also found in the displayed workstation.
- The occurrence is an occurrence of an entity type. You can tag occurrences of associations and relationship as well, but this is not mandatory.

For instance: In the preceding list, only the second row in the list must be tagged. It is the only definition that is found in the PROD (target) status, has a corresponding definition in the TEST (source) status, and is an occurrence of an entity type. The last row could also be tagged, although this is not necessary.



	EntityType	Name	Status	Version
___	TABLE	ALPHA_TABLE	TEST	00
_s_	TABLE	ALPHA_TABLE	PROD	00
___	TABLE	BETA_TABLE	PROD	00
___	ELEMENT	FIELD_1	TEST	00
___	COLUMN	ALPHA_TABLE.FIELD_1	TEST	00
___	COLUMN	ALPHA_TABLE.FIELD_1	PROD	00

Use Sample Queries to Identify Definitions in a Workstation

Some sample queries are provided with the product to help you identify the definitions in a given workstation that meet the previous criteria.

The following query, which must be executed outside the Repository (that is, in SPUFI, QMF, and so on), identifies all of the different entity types, not individual entity type definitions, that are found in the specified workstation.

Note: The names of relationship and association types are excluded from the results of this query.

```
SELECT   E.ENT_NAME,  W2.NAME
         FROM DBXREL30.DBX_XREF X,
         DBXREL30.DBX_ENT_TYPE_DESC E,
         DBXREL30.DBX_WKSN_XREF W1,
         DBXREL30.DBX_WORKSTATION_D W2
WHERE    X.ENT_TYPE = E.ENT_TYPE
         AND E.ASSOCIATION = '
         AND E.SOURCE_ENT_TYPE = 0
         AND X.ENT_ID = W1.ENT_ID
         AND W2.ENT_ID = W1.WKSN_ID
         AND W2.NAME = <FROM workstation name>
GROUP    BY E.ENT_NAME, W2.NAME;
```

The second query should be executed in the Repository.

- It is suggested that you save the following WHERE statement in all of the meta-entity types using the QUERY.SAVEALL command found in the SQL Search Criteria Panel.
- This statement returns every occurrence of the current entity type that is found in the specified workstation.
- You must supply the name of the FROM workstation to this query and execute it in every one of the entity types mentioned in the results of the first query.

  ```
  Where    E.ENT_ID IN (SELECT W1.ENT_ID
               FROM   DBXREL30.DBX_WKSN_XREF W1
                  DBXREL30.DBX_WORKSTATION_D W2,
               WHERE  W2.ENT_ID = W1.WKSN_ID
                 AND W2.NAME = `FROM workstation name')
  ```

  You should use the OPTIONS.WORKSTN.DELONLY command to remove the definitions from the workstation.
- You should not delete these definitions from the repository.
- After removing these occurrences from the workstation, you repeat the second query for every entity type mentioned by the first query.

Once the appropriate definitions are removed from the workstation and the cause of the failure is identified and resolved, you can start the migration process again.

Recover from a Failed Migration of Individual Definitions

Should a real migration of individual definitions, migration without a workstation, fail during the migration processing, you need only begin the migration process again.

The pertinent FROM/source definitions recollide with the existing TO/target definitions or collide for the first time with the TO/target occurrences that were inserted before the failure of the previous migration. In both situations, you control how the collisions are handled.

Have Any Definitions Been Lost?

Ask these questions: Should you be concerned about any original (source) definitions that were deleted before the migration failed? What must you do to recover those definitions? You do not have to do anything.

Source definitions are removed from the FROM workstation and deleted from the repository only at the very end of the process-when migrations fail, the original definitions are left intact.

Back Up a Workstation

Prior to migrating workstations, it is a good idea to back it up using the CLONE command.

If the migrate fails, make the backup of the source workstation the source workstation, and delete the original one at your convenience. The CLONE WORKSTATION command can be run online or in batch. To run the clone in batch, select Batch Clone of Workstation from the Repository Utility menu. Enter both the From and the To workstation names on the panel. For more information about the clone workstation command, see the *Administration Guide*.

Migration Reports

Each time you use the Migration Facility to migrate individual occurrences or workstations, the Repository populates the Migration Report Table. This table lists all the actions hypothetically or actually performed against the Repository metadata during a test or real migration, respectively, and is used to generate a Migration Report.

To print the Migration Report, select OPTIONS.REPORT.MIGRATE.

Order of Collision Information in the Migration Report

The Migration Report is ordered by the Repository internal identifier of the entity type involved in the collision.

- In the case of associations and relationships, the internal identifier of the source occurrence determines the order in which the collision information appears.

  Note: This is not necessarily the same order in which the information appears in the migration collision windows.

- This method is used so that association and relationship occurrences are listed directly beneath the mention, if any, of their source occurrences, making the report easier to read.

The Migration Report

The Migration Report contains four sections:

- User-selected responses (ACT column and REASON column). Actual Repository actions are in the IO column.
- Automatic responses (ACT column and REASON column). Actual Repository actions are in the IO column).
- List of minor differences.
- List of deleted occurrences.

User-Selected Responses to Collisions

The first section of the Migration Report lists all the collisions that occurred during the migration that had to be resolved through user responses. The actions taken in response to the options chosen by the user performing the migration are also recorded.

The ACT column lists the actions that were performed on each of the occurrences. Possible values are:

Action	Description
UPD	The occurrence was updated to the new statuses. This can happen if you chose to move a dependent relationship or association to the TO status.
DEL	A dependent relationship or association was deleted (a TO definition with no FROM match).

Action	Description
KEEP	A dependent relationship or association in the TO status with no FROM match was retained.
IGNR	A FROM occurrence (either entity or relationship definition or dependent relationship or association) was ignored and deleted.
MOVE	The FROM entity or relationship definition was moved over the TO definition.
SAME	The FROM entity or relationship occurrence was determined to be the same as the TO occurrence, so the TO occurrence was not updated. The FROM occurrence was deleted.
NEW	The FROM entity or relationship occurrence was moved to the TO status as a new version.

The REASON column details what each action was taken. Possible values include:

Reason	Description
USER SEL	The user-selected the action.
USER NEW	The user specified that a new version of the occurrence should be created.

Automatic Responses to Collisions

The second section of the Migration Report lists all the collisions that occurred during the migration that the Migration Facility was able to handle automatically. The actions that were automatically taken by the Migration Facility are also recorded.

The ACT column lists the actions performed on each of the occurrences. Possible values are:

Action	Description
UPD	The occurrence was updated to the new statuses. This can happen if the occurrence had no match in the TO status.
MOVE	The FROM definition was automatically moved over the TO definition.

The REASON column details the action taken. Possible values include:

Reason	Description
NO MATCH	No match was found for the occurrence.
AUTO REL	A relationship or association was automatically moved to the new status. This can occur when an occurrence is moved over because it has no match or because it was moved over as a new version.
AUTOCHAIN	A relationship or association that is dependent on another relationship or association that was automatically moved over has also been moved over as part of the *chain* of relationships.

Actual Repository Action

The IO column, found in both of the report sections discussed previously describe the actual Repository action performed on the occurrence. Possible values are I (insert) or U (update).

List of Minor Differences

When you specify a TO workstation, it is assumed that the TO and FROM workstations are very similar, with relatively minor differences between them. This is often the situation when you use the Migration Facility to manage models created in a CASE tool.

The Migration Facility attempts to assemble a list of the *minor differences* in the third section of the Migration Report. This list only appears if both a FROM and a TO workstation were specified *and* there were some occurrences in the TO workstation that did not have a corresponding match in the FROM workstation.

Lack of Corresponding Occurrences

This lack of corresponding occurrences can result from the following:

- The deletion of the occurrence (in the CASE tool model, the FROM workstation
- The repository in general
- The change of the name of the occurrence

Deletions During the Cleanup Process

The last section of the report lists all the occurrences that were deleted from the repository during the final cleanup process. These include the following:

- FROM occurrences that were deleted because they matched TO occurrences
- FROM and TO occurrences tagged for deletion by the user during migration processing

Note: During a test migration, the DELETIONS section may list more relationships or associations than will appear when the actual migration is performed. This is because the deletion of occurrences during a real migration causes the immediate deletion of their respective dependent occurrences, before the Migration Facility recognizes the dependent occurrences as metadata that needs to be processed. The test migration would have processed these dependent occurrences.

Empty the Migration Report Table

After you print the report, use the EDIT.MGRATION.MIGRPDEL command to empty the Migration Report Table. MIGRPDEL deletes all the migration information stored under the TSO logon ID of the user submitting the command.

When you issue the EDIT.MGRATION.MIGRPDEL command, the Migration Facility verifies that the Migration Report was generated. If the Migration Facility determines that a report on the migration results that are about to be deleted has not been generated, it displays a warning message to the user. You cannot execute the Migration Facility again until the migration results are deleted from the report table.

Note: An important note about the CASE interfaces: It is possible to run a test migration during the CASE model upload process. This step immediately follows the importation of CASE-designed metadata. Take care when initiating a CASE import with an immediate test migration, because the migration deletes all the information stored in the Migration Report Table under your TSO logon ID without giving you the opportunity to prevent the deletion.

CASE Environments

When using the Repository to manage CASE environments, you can use workstations as logical units of work when uploading and downloading information. You can also use these workstations during migration to aid in CASE model management.

It is far quicker and easier to migrate one workstation containing scores of occurrences than it is to migrate those scores of occurrences individually.

You can gain additional benefits by specifying both FROM and TO workstations for a *model-management* migration because possible name changes or other differences can be identified (discussed earlier) and, more importantly, many unnecessary collisions can be avoided.

CASE Model Management

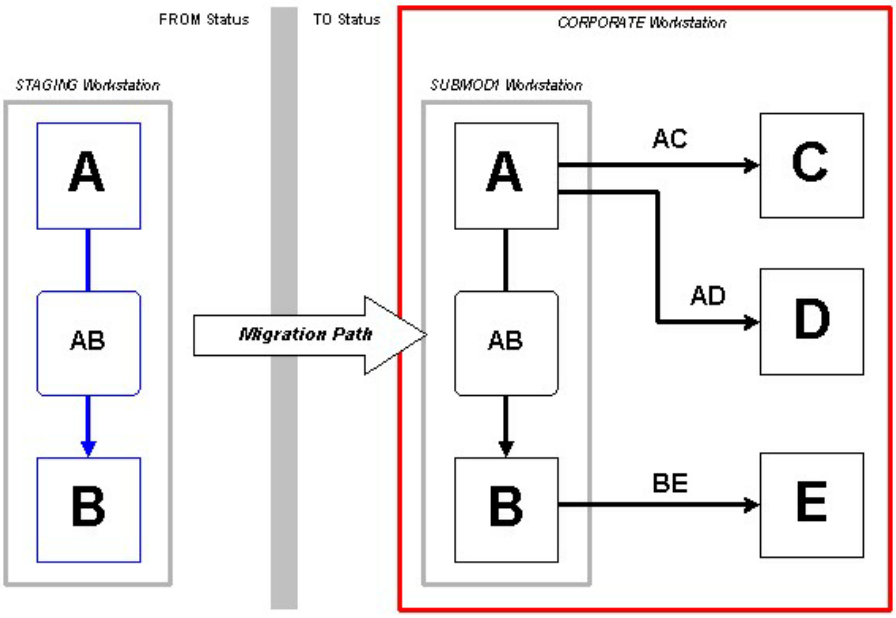
CASE model management usually entails merging two or more slightly different versions of the same base model. These versions should have been assigned to their own workstations when imported into the Repository.

It is safe to surmise that these workstations are very similar, with only relatively minor differences between them.

- When this is the case, it is assumed that the only collisions of importance from the viewpoint of the FROM workstation (containing the latest version of the CASE model) will occur in the TO workstation (containing the previous incarnation of the model)
- Other collisions could possibly occur, but they may not be considered significant in the scope of this migration

Note: During a standard, *non-model management* migration, it is *strongly* suggested that the TO workstation be left unspecified. One does not normally want to restrict the potential collisions resulting from a migration, for fear of missing a very important collision.

The following diagram shows this point. It is assumed that every occurrence to the right of the thick arrow is in the same status, the TO status. The STAGING workstation contains a recently modified and recently imported version of a CASE-designed model, while the SUBMOD1 workstation contains the previous version of that same model.



Previous Model Compared to Corporate Model

If you are concerned simply with merging the most recent version of the CASE-designed model with its previous incarnation, then you should specify the STAGING workstation as the FROM Workstation field in the Migration Control window and specify the SUBMOD1 workstation in the TO Workstation field. For the bulk of CASE-model management situations, this should be the normal course of action.

If you specify both the FROM and TO workstations in this manner, the associations AC, AD, and BE would not be shown as collisions.

This is:

- Acceptable because they do not exist from the viewpoint of the CASE-designed model. They are not included in the SUBMOD1 workstation.
- Desirable because it reduces the number of collisions and therefore the required analytical time and effort.

If you do not specify a TO workstation, they all would be listed as differences because the occurrences that could be involved in a collision would be limited not by workstation, but by status only.

Corporate Model

Also shown in the previous figure, is the site global *Corporate* model (which may be composed of multiple CASE models), found in the CORPORATE workstation, which in this case includes the original version of the CASE-designed model.

Note: The Corporate model may not be wholly contained in a single workstation, but may instead be defined across several individual workstations.

- If the CORPORATE workstation was specified as the TO workstation, then AC, AD, and BE would appear in the Association/Relationship Collision window, because they are found within the specified scope of the migration. However, this method would cause more harm than good.
- If the CORPORATE workstation was specified as the TO workstation, then the occurrences of the STAGING workstation would be merged into the CORPORATE workstation. The STAGING workstation would cease to exist and the last version of the CASE-designed model would not be considered an independent grouping in the repository-which is not desirable at all, if the model needs to be viewed as a discrete unit in the future. None of the benefits of the Repository workstations would be available to the last version of this model, including, significantly, the ability to export the model independently.

What Should Be Included in the FROM Workstation?

Note: For every entity type or relationship type occurrence in the FROM workstation, every association and relationship involving the occurrence is evaluated for collisions in the TO status, whether or not the associations and relationships are actually found in the FROM workstation.

However, if the definitions that are connected to those occurrences by those associations and relationships are not also included in the FROM workstation, then they will not be investigated for collisions. Therefore, for a complete analysis of the potential effects of a migration, it is suggested that you take care to include in the same workstation all entity and relationship occurrences that logically belong in the conceptual group.

Index

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X