CA Gen

Client Server Encyclopedia Construction User Guide

Release 8.5

CA Technologies Product References

This document references the following CA Technologies products:

- CA Gen
- AllFusion® Gen

Contact CA Technologies

Contact CA Support

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

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

Providing Feedback About Product Documentation

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

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

Contents

Chapter 1: Introduction

This guide provides an overview of Client Server Encyclopedia (CSE) Construction. The purpose of this guide is to help you understand the process of construction using the CSE Construction Client component of CA Gen on the workstation.

This book also contains procedural instructions to help you get started. The online help provided with the software explains the individual menus and windows.

This section contains the following topics:

Processing and Output

The following table provides the platform type on which certain applications are processed:

In Processing Type	Application Type	Executes On
Batch	Batch applications	Host or Server
Component	Operations Library	Workstation, Server
Cooperative	Client/server applications	Workstation, Host, Server
Online	Online applications (block mode or character-based)	Workstation, Host, Server
Proxy	Proxies for Procedure Steps that reside in server managers	Workstation and Server
Window	Windows, ASP.NET, Web	Workstation
z/OS Library	z/OS Library	Host

Chapter 2: Construction Concepts

This chapter provides information about construction concepts.

This section contains the following topics:

Construction Stage

Construction is the stage of the CA Gen development process where the diagrams created during Planning, Analysis, and Design are used to build an executable application system. The various application types that you can construct include:

- Online (character-based or block mode) applications
- Batch applications
- Windowed or Graphical User Interface (GUI) applications
- Client/Server applications
- Operations Libraries
- Proxies
- Web Clients
- z/OS Libraries

Processing and Output

The following table provides a high-level view of the database and modules produced when you perform construction processing on components and component types.

Process This Remote File Type	To Create
Data Definition Language (DDL)	Application database
Cascade	Referential Integrity (RI) trigger modules
Load Module	Application executable modules
Operations Library	A DLL or shared library
z/OS Library	A DLL or shared library for z/OS

The Client Server Encyclopedia (CSE) Construction Client offers these construction options:

- Code - perform packaging, generate and install source code
- DDL - generate and install Data Definition Language

Each of these options allows you to set environment and configuration parameters and paths for its components. Each option uses the information from certain CA Gen diagrams as input to define and create components used in the construction process. The diagrams must be complete and pass a Consistency Check with no errors in the Toolset or the CSE before starting construction.

Construction Client Installation

You can install and execute the Construction Client, by itself, on a workstation. Users must have Generate authority on both an encyclopedia and a model to perform generation tasks and Update authority to perform other packaging tasks.

Remote Installation

The remote installation option is available for CSE Construction. Remote installation creates a set of remote files that contains all of the components necessary to compile and install the application components on a target system. These remote files must be moved to the target system and installed using Build Tool (BT) before using the application.

Installation Target

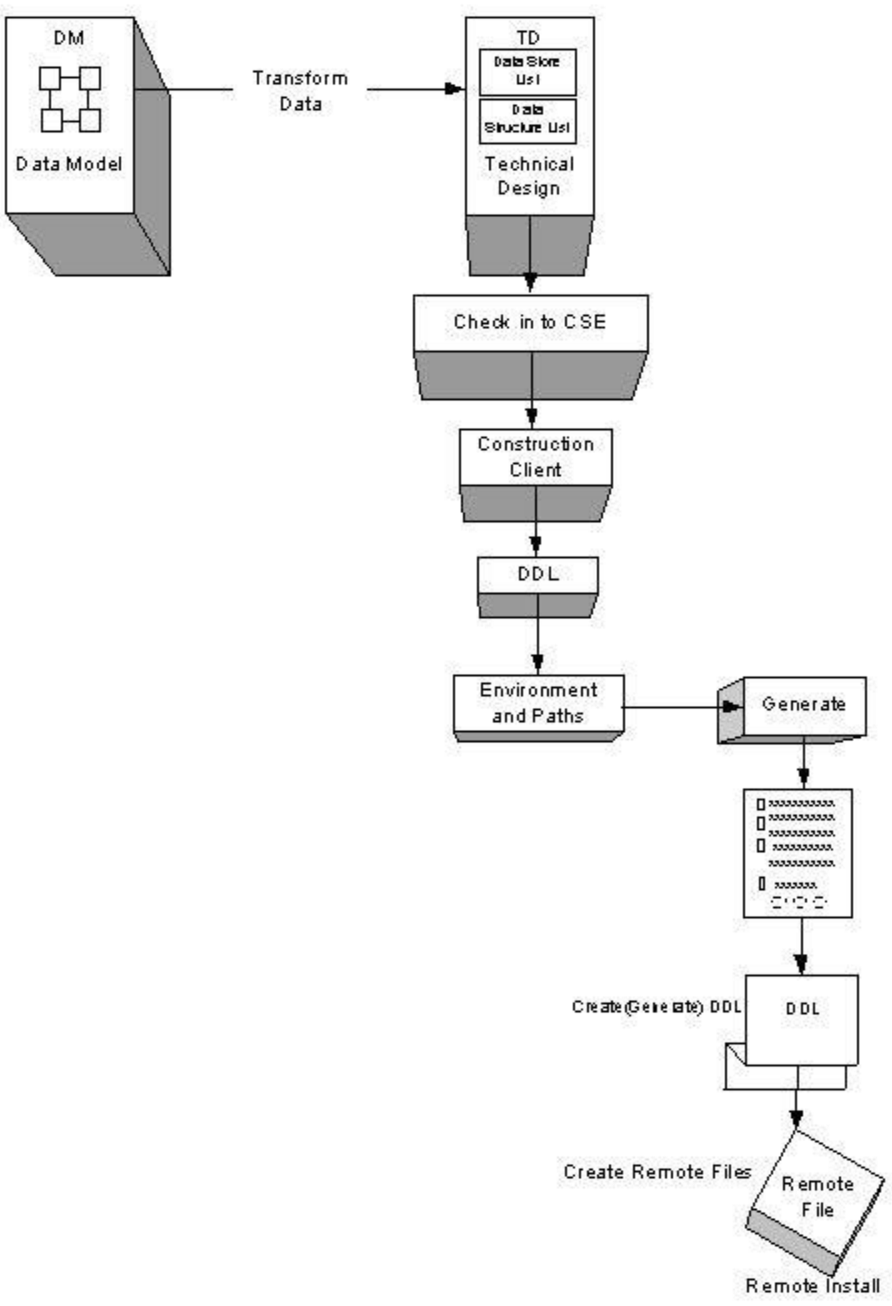
In a remote installation, a CA Gen Build Tool is used to compile, link, and bind the source code generated by the CSE Construction Client. The remote installation target environment may or may not be the same as the Construction Client's environment.

DBMS

The installation of your application requires a Database Management System (DBMS) and a language compiler. The DBMS and compiler must be on the target system. For installation and operation, see the documentation for your specific operating system, DBMS, and compiler.

Overview of Construction

The following illustration shows an overview of Construction using the CSE and the Construction Client:

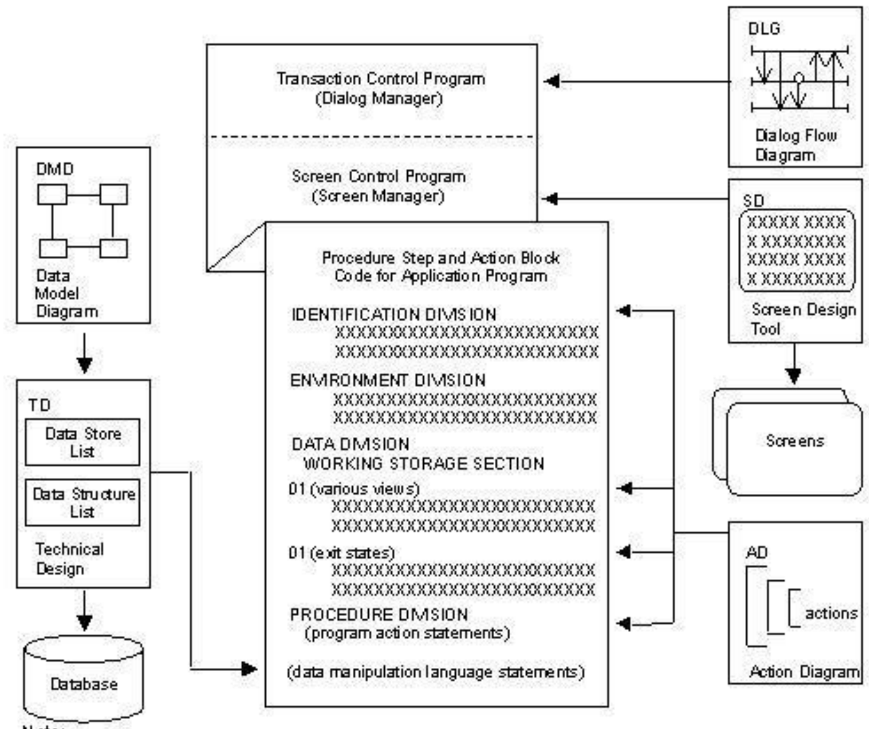


The following illustration gives a high-level overview of the relationships between certain CA Gen diagrams and the structure of a generated application. The application in this example is an online, character-based (block mode) application generated in COBOL. The illustration neither shows all of the diagrams nor all components of a generated application.

Entity types in the Data Model (DM) are implemented as records in the Technical Design (TD). These records become physical components of the database. The application program references and manipulates these records.

Transfers and links on the Dialog Flow Diagram (DLG) are incorporated into a transaction controller appropriate to the application type:

- Dialog Manager for character-based (block mode) applications
- Window Manager for GUI applications and Web Clients
- Server Manager and Window Manager for C/S applications
- Batch Manager for z/OS batch applications



Note:

This figure shows the relationships between major CA Gen componenents of a generated application. (The figure does not show all diagram components.)

Prerequisites

The Construction Client uses certain diagrams in construction processing:

- Data Model Diagram (DMD)
- Technical Design (TD)
- Dialog Flow Diagram (DLG)
- Action Diagram(s) (AD)
- Screens built using the Screen Design (SD) Tool (for a character-based application)
- Windows and Web clients built using the Window Design functionality for:
  - GUI application (GUI application with local database)
  - GUI client component of the Client/Server application
- Web Client application with local database
- Web Client component of Client/Server application

Other prerequisites must also be met before you can complete the Construction process:

- Model must pass a Consistency Check with no errors and the required diagrams must be complete. (This process is done within the Toolset or the CSE.)
- Upload the model to the encyclopedia.
- You must be the encyclopedia administrator or model owner, or have Update (for packaging) or Generate (for generation) authority for a model.

External Action Blocks

If used, external action blocks must be compiled manually prior to remote installation of the remote files. External action blocks are not compiled as part of the use of the Build Tool.

Construction Process

During the Construction process, your CA Gen model is processed into the components of a remote file.

Components

An application contains one remote file for the database (DDL), one remote file for the Referential Integrity (RI) trigger modules, and one remote file for each load module, proxy, operations library, or z/OS library in the business system. These objects are then transferred to a remote target for installation with a remote BT, or used in place for installation on the local system using a local BT.

Transfer Method

The method you use for transfer of the files depends on your target system. For more information, see Build Tool documentation.

DDL Option

The CSE Construction DDL option generates DDL as database source code and installs the source into a DDL remote file. The DDL option uses the transformed data model.

During construction, specify the following for the application:

- Execution environment
- Packaging

Then select the portion of your application to generate. A complete application includes a database, Referential Integrity (RI) trigger modules and load modules (groupings of procedure steps and action blocks). The generated application components then concatenate into remote files for transfer to a target system for installation.

Tasks in the Construction Process

The following is a list of the main activities in the Construction process:

- Specify environment and paths for DDL
- Generate the DDL for the database
- Package procedure steps into load modules
- Specify environment and paths for code
- Generate the source code
- Install into remote files
- Install external action blocks

Generate the DDL

You generate Data Definition Language (DDL) for the database, tables, and indices of your application system. The Technical Design (TD) is the main source of information for DDL generation. Installing the DDL with the appropriate Build Tool creates the database on the target platform.

You generate and install a database with the Generate option in the DDL menu.

Load Module Packaging Steps

Load Module Packaging is the process of specifying the combination of procedure steps that comprise a fully resolved executable application. Packaging also allows you to name the source code and transaction codes manually or with the Complete option. For more information about transaction codes, see the chapter Packaging (see page 41). Each packaged Load Module becomes an executable file after completing the installation process on the target platform.

Note: Regardless of the platform, the term Load Module is used in the Packaging process as a generic way to refer to the set application code that is contained in a fully resolved executable application. Starting with AllFusion Gen 7, applications when installed on z/OS are built as fully resolved executables that reside in DLLs, except for those application components marked for Compatibility. Components marked for Compatibility are built as z/OS non-DLL load modules. Before AllFusion Gen 7, z/OS executables reside in z/OS load modules.

You package load modules with the Package option in the Code menu. Packaging for z/OS Libraries and Operations Libraries are available only on the Toolset.

Specify the Environment

You must specify the environment in which the generated application will execute. The parameters are specific to each business system. Different business systems within the same model can have different environment parameters. Examples of the environment parameters include the operating system, language (source code), and DBMS.

Code environment and path are specified at the business system level unless you are packaging and generating a Cooperative application. You specify parameters for both window managers and server managers for Cooperative applications. These configuration areas are in the Code menu. The DDL has its own environment and path specification areas in the DDL menu. It is also possible to override paths for a specific generation activity.

Generate the Source Code

CA Gen generates the source code for your application. The language choices for your system depend on the options you purchased for your CA Gen configuration. The source code generated includes Referential Integrity (RI) trigger modules, transaction managers, screens, procedure steps, Action Blocks, and other modules.

You generate source code with the Generate option in the Code menu.

Implementation Package

The Implementation Package (also known as the remote file) is produced by the CSE generators when the INSTALL option is selected. Remote installation on the Construction Client creates a set of remote files that contains all of the components necessary to compile and install the application components on a target system. These remote files must be transferred to the target system and installed using the Build Tool before the application can be used.

Testing is accomplished on the target platform after installation with the appropriate Build Tool. For more information, see Build Tool documentation.

Install External Action Blocks

External action blocks allow you to access logic created outside of CA Gen (user-written subroutines). You will need to generate, complete, and install external action blocks if your CA Gen-generated application accesses any information or processes not generated using the Construction Client. You must transfer all external action blocks to the target platform, and then compile them manually *before* you install the load modules that invoke them. The Build Tool does not compile external action blocks. For more information about installing external action blocks, see the documentation for the appropriate Build Tool.

Note: For more information on z/OS applications, see the *Host Encyclopedia Construction User Guide* and the z/OS *Implementation Toolset User Guide*.

Start the CSE Construction Client

Follow these steps:

Note: You must have an authorized user ID to access models on the encyclopedia. For more information, see *Client Server Encyclopedia Guide.*

1. Start the Construction Client. Click Start, All Programs, CA, Gen *xx*, CSE Clients, Construction Client.

   Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*.

2. The Construction Client Logon dialog appears. Enter your user ID and password. Also specify the Host Name and Service/Port for the Message Dispatcher connection.

   Note: The Hostname field can contain a hostname, a hostname with domain name, or an Internet Packet version 4 (IPv4) or version 6 (IPv6) address. The hostname can be a maximum of 1024 characters to accommodate host and domain names that contain non-ASCII characters.

3. Click OK.
4. If multiple encyclopedias are present, the Encyclopedia List dialog appears. Select an encyclopedia and click OK. The Encyclopedia Construction Client window appears.
5. Select the appropriate construction option for the functions you need.

Open a Model

Use this procedure to open a model for construction.

1. Start the Construction Client. The Encyclopedia Construction Client window appears.
2. Select Model. The Construction Model Selection window appears.
3. On the Actions menu, choose Open.
4. Select a model from the Model List dialog box and select Open.

Filter the Display List

The Construction Client allows you to filter the display of components in windows during packaging, code generation, and DDL generation. Use the filter to refine (limit) the number of items in the display list. You can filter on the current window, or a window that displays expanded objects. To enable filtering, use these methods:

- Click a Filter push button (located at the bottom of a window or panel)
- Select a Filter check box

- Select Filter from a menu

In each case, by selecting the Filter option you invoke a filter selection window in which you specify a filter value. You must select the check box to turn on filtering. If you do not, the value you enter is ignored. Once filtering is turned on, it remains on through expansions to lower-level objects. The Construction Client prompts you for a new filter value at each expansion level and retains the previous filter value going back to the level from which you selected the filter option.

Filter Pushbutton

If you invoke the filter selection window from a pushbutton, select the check box so that a check mark appears. This activates the filtering option. Type a filter value in the filter selection window. When you select OK, the filter applies to the display where you invoked the filter option.

Filter Check Box

If you invoke the filter selection window using a check box, select the check box so that a check mark appears. This activates the filtering option. Type a filter value in the filter selection window. When you click OK, the next display uses the filter value you specified.

Filter Menu Item

The filtering option is invoked by selecting the Filter check box so that a check mark appears. This activates the filtering option. Type a filter value in the filter selection window. When you click OK, the filter applies to the display where you invoked the filter option.

Specifying Filter Values

The filter value can contain special characters to assist you in establishing display criteria:

- Use the percent sign (%) to indicate any string of zero or more characters.
- Use an underscore (_) to indicate any single character, except the last character in the string.

The percent sign (%) means any string or no string. For example, the filter value %1 means any string ending with 1. The values ENTITY1 and ET1 would both match.

The underscore (_) means any single character. For example, the filter value E_1 means E followed by any character, followed by 1. Both ET1 and E11 would match. ET111 would *not* match since it has too many characters. Trailing underscores are not supported. Use the percent sign (%) instead.

Examples:

- 378_A selects five-character objects with the first three digits 378 and the last character A.

  378AA

  3788A

- H% selects objects of any character length with the first letter H.

  HELPMODEL

  HOFFENSTEIN

  HILO

- %CENTER% selects objects of any character length that contain the string CENTER at any point in the object name.

  OFFCENTER

  CENTERONE

  ATCENTEROFFICE

Component Selection

When using the Packaging, Code Generation, and DDL Generation options, you can either generate all components or select components individually and add them to a list. When the list is first displayed, it is empty because no objects have been selected. Selection of objects varies slightly for different objects and for different options. See the sample procedures in the chapters that follow. They are organized by option name.

You can use the filter feature described earlier in this chapter to control the display contents during component selection.

Chapter 3: Generating DDL

This chapter provides information about generating the DDL.

This section contains the following topics:

Creating Data Definition Language

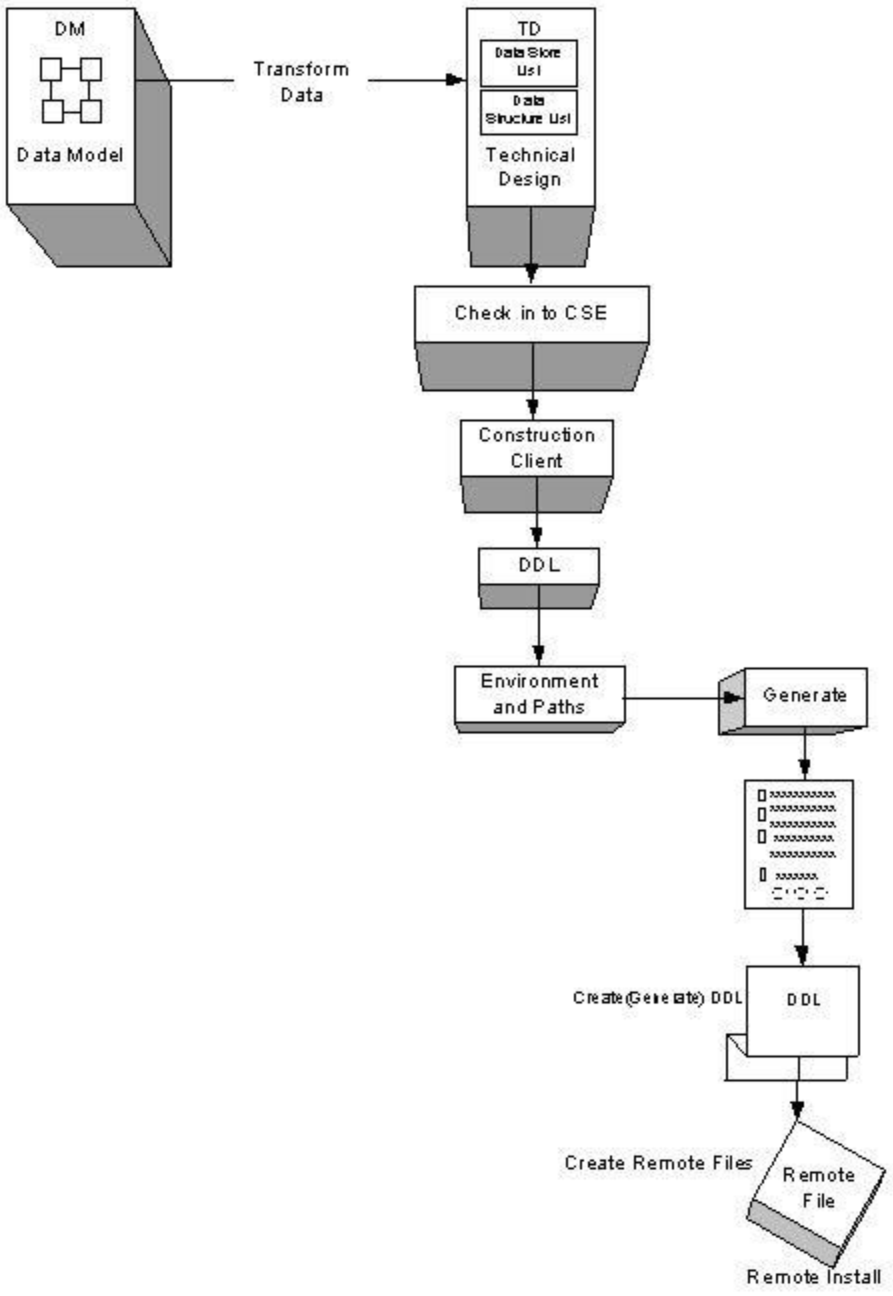
The Technical Design (TD) is used by Client Server Encyclopedia (CSE) Construction to create the Data Definition Language (DDL). The TD contains the Data Store List and the Data Structure List. The type of DDL generated by the CSE Construction Client is Structured Query Language (SQL).

A database must be present before you can install an application if:

- The application has database access statements (SQL calls) in it.

- The application is a character-based (block mode) application that contains links between procedure steps. This restriction exists only if the RPROF table (profile manager) is used. This is not true for CICS applications that use the TSQ profile manager or for UNIX-based applications, if the RPROF table has been placed in a file.

The Construction Client supports DDL generation for remote installations. For a remote installation, the CSE Construction Client creates a remote file, which you transfer to the target system. On the target system, the Build Tool processes the remote file and creates the database. This process is described in the following illustration:



Prerequisites

Before you can successfully create a database from a model or subset, you must:

- Verify consistency of the model

- Transform the DM

Verify Consistency of Model

To verify that the data model is consistent and does not violate any of the rules that govern source generation, use the Consistency Check Report to identify any error-level inconsistencies.

A consistent model is necessary to ensure successful database generation and the TD is the main source of information for database generation. Before CSE Construction can successfully generate the DDL, the model's TD must be complete and consistent. In the Database Design tool, you have the opportunity to make manual adjustments to the transformed TD using the Data Structure List and Data Store List. For example, you might add another entry point or change a table name. If you make changes, run another consistency check, because generation fails if any components are inconsistent.

If only a portion of the database definition is to be generated (for example, a table and its indexes), the portion of the DM and TD that describe the objects to be defined must be complete and consistent.

Transform the Data Model

The Transformation process creates the data structure objects for the TD using the Data Store List and Data Structure List. The Construction Client then uses these objects to generate database definitions, which include the DDL statements necessary to allocate and construct database objects. You transform the DM into a TD using the Design Toolset. This information is stored on the Toolset in one of the files for the model.

Retransform the Data Model

After the initial transformation of the DM, you can use retransformation to implement changes to the TD. Retransformation implements only what has changed. It has no effect on those portions of the TD that were not modified.

Required Activities Outside of CA Gen

Before you can install a CA Gen generated database using the Build Tool, you must install the selected DBMS on the target system. In addition, you must have appropriate authority to create databases to complete the installation.

How to Create a Database

Use the following steps to generate DDL for a database:

1. Set generation configuration parameters
2. Set generation paths
3. Select components for generation
4. Select a generation action
5. Set DDL generation defaults
6. Verify completion

Set Generation Configuration Parameters

You must set Generation Options before generating a database. Review the options for subsequent generations if the target or purpose changes.

To set generation configuration (environment) parameters:

1. Select DDL, Environment, and Configuration.
2. Select environment parameters from the fields available.

You will receive an error message if the combination of environment parameters you select is not valid.

Setting values from this location provides default parameters used for DDL generation actions. You can override these defaults from within the DDL Generation window. At a minimum, you must define the operating system and the DBMS that you are using for this generation.

Several flags affect the generation of a database. You toggle flags on and off by selecting the icon above the column. The default value, No, is represented by a blank. The specific actions that occur as a result of selection appear in the following sections. The flags are summarized in the following table:

Set This Flag	If You Want To
DDL (D) to Y	Generate source for the DDL for the database and/or its subordinate objects with the options set for the database.
DROP (R) to Y	Ensure the database installs even if a database already exists with that same name.

Set This Flag	If You Want To
INSTALL (I) to Y	Install the database objects and create the remote file for remote installation.
APPLY TO ALL SUBORDINATES (A) to Y	Generate source for the DDL for the database and all of its subordinate objects.

Generate DDL

Select this option if you want the CSE Construction Client to generate the DDL as source for the database. If you do not select this option for at least one of the objects, no source code is generated.

Include Drop Statements in DDL

Many database management systems cannot create a database, table, or index if it already exists, and cannot create new databases unless a DROP statement is generated into the DDL. The DROP statement that appears in the generated DDL is a logical data definition statement that removes the description of a database or its components from the system. For each DBMS, you must include a DROP statement to recreate (regenerate) an existing database's components.

You can also specify generation of the DROP statement for one or more database components from the DDL Generation window. When a component is dropped, all data associated with that component is dropped as well.

Important! The DROP statement can delete a database and all of its data. If you need the data, back up the database before you DROP it.

Downstream Effects: If you are creating a new database remote file, do not generate DROP statements in the DDL. If DROP statements appear in a DDL remote file that was not previously installed on the remote platform, the implementation may fail. This occurs because the BT is attempting to delete a database that does not exist.

Install DDL

Select this option if you want the CSE Construction Client to install the generated DDL. The generated DDL is placed into the remote file for remote installation using the install option, discussed later in this chapter.

Apply to All Subordinates

Select this option if you want the CSE Construction Client to generate the DDL as source for all of the subordinate objects in the database, with the options selected for the database. If you do not select this option, you must select each of the subordinate objects manually.

Set Generation Paths

You must specify fully qualified path names for the locations of the DDL for the application database. Special and wildcard characters, including the character that references the current drive for UNIX, are not supported. You set path names for the source and remote files. These paths exist on the server and must conform to the platform naming conventions in use there. For example, UNIX paths are case sensitive. Some paths may require all path names to be eight characters or less.

Note: For more information, see the chapter Target Environment and Construction Paths (see page 75). Be sure the paths you set are correct for the type of server platform you are using. If the validation processing encounters an error, you receive an error message. If the paths have been specified but do not yet exist, the Construction Client will create them. If the paths have not been specified, you receive an error message and generation does not continue.

CA Gen appends the name of the DBMS and the operating system to these paths.

Overriding DDL Paths

You can override the DDL paths by selecting Defaults from the DDL Generation window. Select Override Paths then select Paths to set the override paths. You must set any desired override paths from this location. The dialog box shows that the paths are override paths.

Select Components for Generation

You must select each of the database components for which you want to generate DDL. The components you can select include:

- Database
- Tablespace
- Data Table
- Link Table
- Indexspace

Note: Selecting a database component does not automatically select all of its subordinate objects. For example, selecting a tablespace does not include the records, index spaces, and indexes that are subordinate to it. To automatically select the subordinate objects, set the Apply to All Subordinates option (A) to Y.

From the DDL Generation window, select Select New Objects from the Edit menu. This displays a list of all of the types of components within a database. As you select component object types and List them, you can Add or Expand specific component objects of the application database to the list of items you want to generate.

Choose a component object type and select the List command. From the list that displays, you can Expand to lower level objects and add them to the generation list or you can select the Apply to All Subordinates option to select all of the subordinate objects. After you have added all of the desired components, you can generate the flagged components. You can also generate all databases and their components by selecting Model from the Generate menu.

Each of the lists of expanded objects allows you to select the DDL generation options (Generate DDL, Drop, Install, and Apply to All Subordinates) before you add the component to the list. Selecting Apply to All Subordinates causes all subordinate objects to be generated with the options selected for that component. As an alternative, you can select these options on the generation list for each component on an individual basis.

Important! In this release, if you exceed the maximum cardinality (50 objects) for the list, the extraneous objects are not updated or generated. This may require generation of the objects in two or more sessions.

Filtering

If you have many components at a given level, you may want to filter the display. You can enable filtering from the object type list or from the various object occurrence lists. If you select Filter using a check box, you are filtering the contents of the next display window. If you select Filter using a push button, you are filtering the current display window.

More Information:

Construction Concepts (see page 13)
Construction Stage (see page 13)
Processing and Output (see page 14)
Construction Client Installation (see page 14)
Remote Installation (see page 14)
Installation Target (see page 15)
DBMS (see page 15)
Overview of Construction (see page 16)
Prerequisites (see page 18)
External Action Blocks (see page 18)
Construction Process (see page 18)
Components (see page 19)
Transfer Method (see page 19)
DDL Option (see page 19)
Tasks in the Construction Process (see page 19)
Generate the DDL (see page 20)
Load Module Packaging Steps (see page 20)
Specify the Environment (see page 20)
Generate the Source Code (see page 21)
Implementation Package (see page 21)
Install External Action Blocks (see page 21)
Start the CSE Construction Client (see page 21)
Open a Model (see page 22)
Filter the Display List (see page 22)
Filter Pushbutton (see page 23)
Filter Check Box (see page 23)
Filter Menu Item (see page 23)
Specifying Filter Values (see page 23)
Component Selection (see page 24)

Select a Generation Action

You can choose to have the Construction Client automatically create all database components (tables, indexes) for you, or you can select individual components to create.

The first time you generate the database, select Model from the Generate menu because it provides the quickest way to accomplish the task.

Generating All Database Components

If you choose to have the Construction Client automatically create the components, you can either:

- Generate DDL for all components but not install the DDL (Select Model from the Generate menu with the Install option left blank)
- Generate and install all components of the database (Select Model from the Generate menu with the Install option set to Y)

Generating Flagged Components

To generate and install database objects selectively, you must select the objects individually and then set the generation flags. You select the objects by highlighting one or more components from the object occurrence lists and selecting Add or Expand. This adds the object(s) to a list of components you want to generate. You can set the generation flags in the object occurrence lists or in the Application Generation window. The DDL source code generation flags are:

- Generate DDL source code with trace enabled (D)
- Generate DDL source with DROP statements (R)
- Install code (I)
- Apply to All Subordinates (A)

As an example, a database contains one or more tablespaces, which contain data tables and link tables. The next level of expansion displays indexspaces. Select tablespace from the object type list. Select a tablespace from the list and then select Add. Then select a data table and Add.

Continue this process of expanding and adding down to the indexspace level. Return to the DDL Generation window by selecting Continue until it displays. Set the DDL (D) and install (I) flags. When you select Flagged from the Generate menu, this tells the Construction Client to generate source for the flagged objects and to install the code into a remote file.

More information:

Generate the DDL (see page 20)

Regenerating Database Components

When recreating components that are already installed, you must DROP the old components. You can add DROP statements globally by selecting R for the database in the DDL Generation window. You can also specify DROP for individual components after expansion. When a component is dropped, all the data associated with that component is dropped as well.

Important! If you are creating a new database, do not generate DROP statements in the DDL. DROP statements that appear in a DDL remote file that was not previously installed could cause the implementation to fail.

Set DDL Generation Defaults

These items appear on the DDL Generation Defaults window.

Actions	Select this Check Box
Override the paths set for DDL generation for this operation only	Override paths
Have more than one person testing with the same workstation and you want each person to have a unique database	Qualify Tables and Indexes With Owner ID
Specify storage groups to be used for tables in DB2	Create Storage Group in DDL
Create alternate referential integrity primary and foreign key triggers for comparison with an existing database	Create RI Alter Primary/Foreign Keys & Triggers

Override Paths

Select this option to override the default path information entered in the DDL Paths dialog box. You must set up any desired override paths by selecting Paths in the DDL Generation Defaults dialog box.

Qualify Tables and Indexes with Owner ID

An owner ID is a string used as an implicit qualifier for table names and index names. The ID applies to the execution of every SQL statement. In the DDL, owner IDs appear as a prefix to table and index names in CREATE and DROP statements.

This feature is useful for allowing multiple individuals to have their own test databases on the same server encyclopedia.

Important! Be careful when using the Qualify Tables and Indexes with OwnerID check box for remote installations. When the DDL is generated with the owner ID included in the table and index names, there may be difficulty installing the DDL unless the ID of the user installing the DDL is exactly the same as the user ID found in the DDL. To avoid the possibility of errors during installation, do not specify qualification of tables and indexes with your owner ID.

Create Storage Groups (DB2 only)

If desired, select this option to generate DDL statements, to take advantage of the Storage Group feature of DB2.

Create RI Alter Primary/Foreign Keys and Triggers

If desired, select this option to create alternate referential integrity primary and foreign key triggers for comparison with an existing database.

Verify Completion

If CSE Construction encounters errors during generation and installation, it stops. Messages regarding success or errors are placed into a log file. The log file name is genstat.ctl and is stored in ASCII format.

Note: Most error conditions can be avoided by ensuring the model is complete and consistent before generating.

Generate a DDL

This procedure explains how to select database component objects for DDL generation. During the selection process, you can enable the generation options (DDL generation, include DROP statements, or install) in these ways:

- From the object occurrence windows
- From the DDL Generation window

You may find it useful to enable the desired code generation options as you go.

1. Start the Construction Client.
2. Open a model by selecting Model.
3. Select Actions, then Open in the Construction Model Selection window.
4. Highlight a model from the list and select OK.

5. Select DDL, then Generate.
6. Select Edit, then Select New Objects.

   This displays the database component object type list.

   You can select one or more objects at a time. The client automatically flows to the next object type when you select more than one.
7. Select Database and select List.

   This action displays the DDL Generation Object Type Occurrence List.
8. Select one or more databases and select Add. Select a database and select Expand. (You can also select Continue to return to the object type list.)

   The expansion displays a list of tablespaces in the DDL Generation Expanded Object Occurrence list.

   If you want to generate all components in a database automatically, without having to manually select them, mark the Apply to All Subordinates generation option.
9. Select one or more tablespaces and then select Add. Select one tablespace and then select Expand.

   These two actions add the tablespaces to the generation list and expand to the next subordinate level for the selected tablespace.

   If you want to generate all components in a tablespace automatically, without having to manually select them, mark the Apply to All Subordinates generation option.
10. Select one or more data tables from the list and then select Add. Select one data table from the list and select Expand.

    If you want to generate all components in a data table automatically, without having to manually select them, mark the Apply to All Subordinates generation option.
11. Select one or more indexspaces from the list and then select Add. There is no expansion beyond this level.
12. Select Continue until you return to the list of tablespaces. Select the next tablespace if you are generating more than one in this database. Perform the steps to add that tablespace and all of its subordinate objects to the list.
13. Select Continue until you return to the DDL Generation window.
14. Select the appropriate generation flags for all desired components and then select Generate, then Flagged.

You can use the steps in this procedure to generate the entire DDL or to perform selective regeneration on changed components. When generating the entire database, it is better to use the Model option found in the Generate menu. For more information, see Select a Generation Action.

Results of DDL Generation

As a result of generation and installation, the Construction Client creates directories subordinate to the directories you named in the DDL paths window and concatenates the subdirectories to the paths you entered. (The directories you entered included source path and remote install path.) This allows proper identification of source code by operating system and DBMS type. The user ID is the name used to log on to the Construction Client.

Use these path definitions in the following tables:

- Source = source path + operating system + DBMS type
- Remote Installation = remote install path + operating system + DBMS type

Source Path

The following table describes the path definitions for Source:

File Name	Description
<DB name>.ddl	SQL statements without the installation information (such as the BIND statements). If you select Install only or No Install and No Drop, this file contains no Drop statements. If you select DROP only, this file contains the Drop statements.
<DB name>.gen	Data used by the DDL generator.
<DB name>.icm	Installation control module. The .icm file is a set of instructions on how to install all DDL components. The file is in Standard Generalized Markup Language (SGML) format. This format allows you to write your own installation procedure from the file should you have a particular installation requirement.
<DB name>.ins	SQL statements including the installation information (such as the BIND statements and CREATE TABLE RPROFX statement). If you select Install only, this file contains no Drop statements. If you select DROP only or No Install and No Drop, this file is empty.

Remote Installation Path

The following table describes the path definitions for Remote Installation.

File Name	Description
<DB name>.rmt	If you selected generation for remote installation, the .rmt file is the remote file. Occurs for remote installation only.
<DBname>.err	Messages resulting from a failed DDL generation.

Temporary Files Created by Generation

The files in the following table are temporary and may or may not be present after generation.

Remote Install Path and User ID

File Name	Description
genstat.ctl remote.ctl	Information used by the Construction Client when creating a remote file. Occurs for remote installation only. Log file for generation and installation messages to indicate success or failure. Contains results of the *last* generation action initiated. Data used by the DDL installation process.
rfgddl.txt	Information used by the Construction Client when creating a remote file. Occurs for remote installation only.

Chapter 4: Packaging

This chapter provides information about packaging process through CA Gen.

This section contains the following topics:

Packaging Process

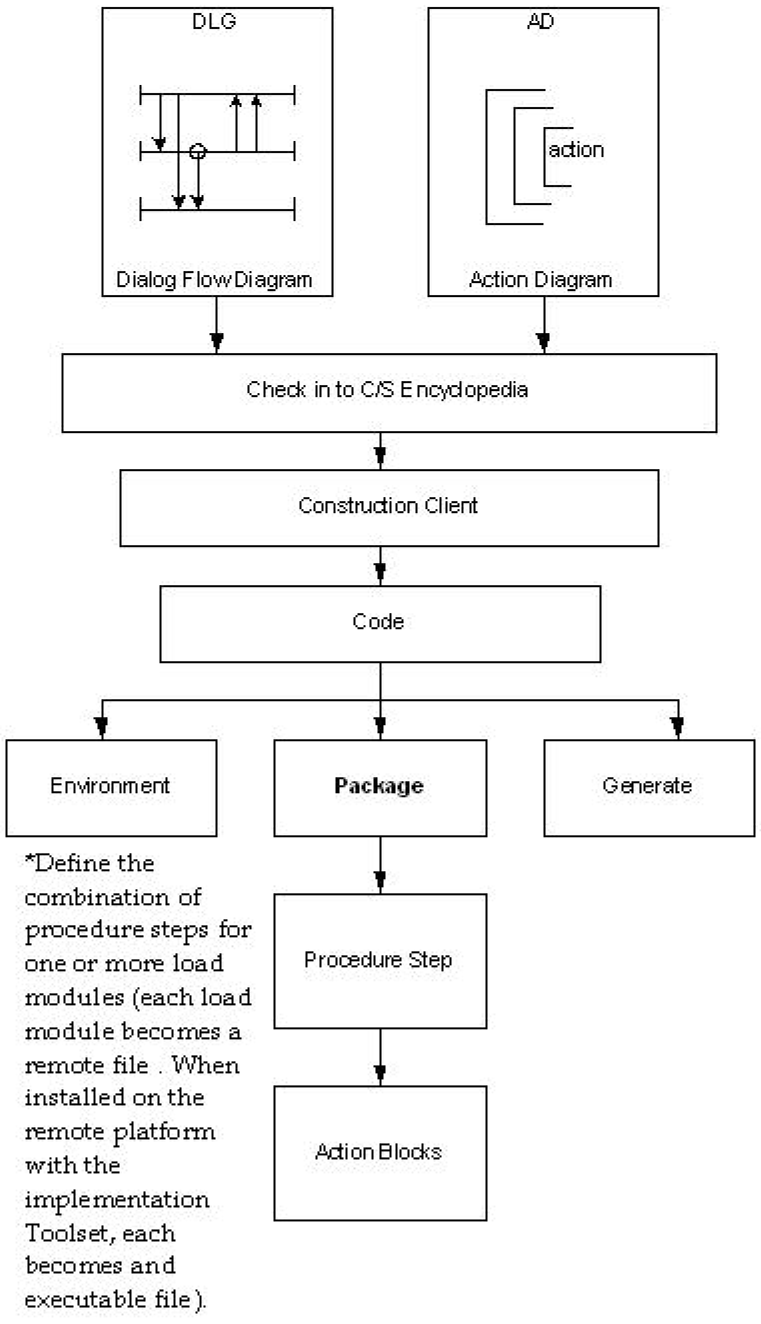
An application generated using a CA Gen model may contain many procedure steps. These procedure steps are the building blocks of executable applications, but they need to be combined into logical groupings for execution. Load module packaging is the process of combining the procedure steps of the application into units that can be executed. Before you can generate an application, the Construction Client requires that these units and their contents be identified.

Load module packaged units are themselves called load modules. The act of packaging does not create the load modules. Packaging simply defines the contents of a load module. A load module (or executable program) is created when the remote file is installed on the target system. An installed load module contains the executable code for all components included in its packaging definition plus CA Gen special function routines and system routines. An example is a UNIX executable.

Note: Regardless of the platform, the term Load Module is used in the Packaging process as a generic way to refer to the set application code that is contained in a fully resolved executable application. Starting with AllFusion Gen 7, applications when installed on z/OS are built as fully resolved executables that reside in DLLs, except for those application components marked for Compatibility. Components marked for Compatibility are built as z/OS non-DLL load modules. Before AllFusion Gen 7, z/OS executables reside in z/OS load modules.

Packaging of operation libraries and z/OS libraries are performed by the Toolset and is not covered in this guide.

The following illustration provides an overview of the packaging process:


DLG
Dialog Flow Diagram
AD
action
Action Diagram
Check in to C/S Encyclopedia
Construction Client
Code
Environment
Package
Generate
*Define the combination of procedure steps for one or more load modules (each load module becomes a remote file . When installed on the remote platform with the implementation Toolset, each becomes and executable file).
Procedure Step
Action Blocks

Types of Packaging

The Client Server Encyclopedia (CSE) Construction Client supports packaging for these types of applications:

- Online (character-based or block mode) applications
- Window-based (GUI) applications
- Batch applications (targeting z/OS)
- Cooperative (Client/Server) applications

Prerequisites

Prior to performing load module packaging, you must:

- Be the encyclopedia administrator or model owner, or have Update authority for a model
- Create the dialog flow of the procedures using the Dialog Design Tool on the workstation
- Check in the changes to the CSE
- Start the Construction Client and open a model

The Package option requires a definition of procedures and procedure steps (in the Dialog Flow Diagram or DLG) to define a load module. Load module packaging defines the load module for a procedure step. The load module is created when it is generated and installed.

This load module definition automatically includes the action blocks used by that procedure step. The load module also contains any action blocks added to that procedure step at a later time. You do *not* have to complete the procedure step or its action diagrams before packaging. (You do have to complete the procedure steps and action diagrams before generation.)

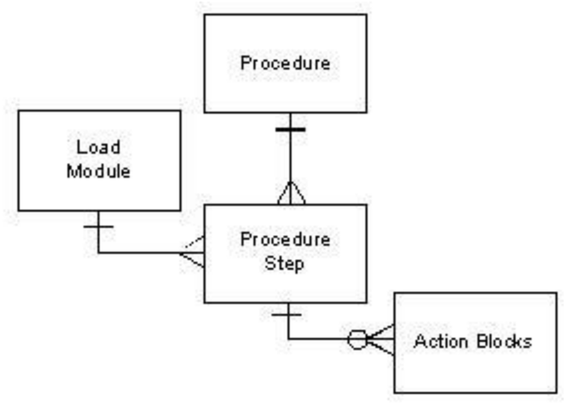
How to Package

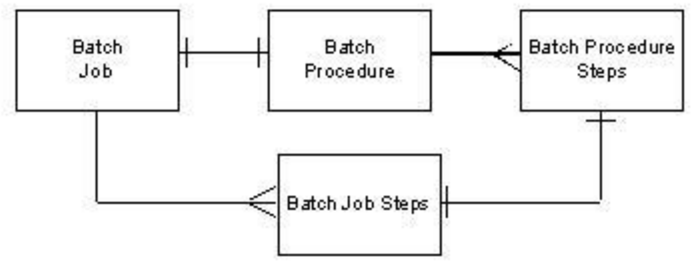
Load Module Packaging varies according to the type of application being generated, and according to the environment for which you are generating. Most of the packaging decisions are accomplished using the Dialog Flow Diagram in Design. Packaging is the implementation of this design. If you change the environment, the new target may have different packaging requirements, making repackaging necessary.

When performing packaging, you must select unpackaged procedure steps. Business Systems may contain one or more load modules, which may contain one or more procedure steps. (Packaging assumes that action blocks are part of the procedure step to which they are referenced. You do *not* explicitly package action blocks.)

Object Relationships

The following illustration explains object relationships:





Select the objects you want to package in a single load module or batch job and then detail those objects. Repeat this procedure for each load module or batch job in the business system.

The basic steps in load module packaging are:

- Create load module or batch job
- Modify existing load module or batch job
- Detail objects

The sections that follow explain the aspects of packaging multiple application types. Information specific to each application type appears later in this chapter.

Create Load Module or Batch Job

The first step in packaging is to create the load module or batch job. Both load modules and batch jobs must have names and at least one procedure step or procedure, respectively.

Creating a Load Module

To create a load module that contains at least one procedure step, you must start the Construction Client, select a model, select Code and Package, and then choose a packaging type. Then select a business system.

After selecting a business system, select Edit, then Add Load Module. In the dialog box that displays, enter a load module name that meets the naming conventions discussed in this chapter. Select at least one procedure step from the list of unpackaged procedure steps, then click OK. This creates the load module packaging that defines its contents for generation. From here, go on to detailing the components in the load module.

Add Objects for Packaging

This procedure explains how to add objects for packaging at different levels. These instructions are valid for online, window (GUI), and cooperative application packaging. Packaging for batch applications is done differently and is discussed in a later procedure.

Follow these steps:

1. Start the Construction Client.
2. Open a model by selecting Model.
3. Select Actions, then Open in the Construction Model Selection window.
4. Highlight a model from the list and click OK.
5. Select Code, then Package and choose a packaging type: Online, Window, or Cooperative.

   This displays a list of business systems in the model.
6. Select a business system and then select Edit, then Add Load Module.

   This displays the list of unpackaged procedure steps in this business system.
7. Specify a load module name and then select a procedure step from the list.

   A load module must have at least one procedure step but can contain more than one procedure step. You can also package procedure steps across procedures.

8. Click OK.

   If you attempt to add a procedure, the Construction Client automatically includes every procedure step in the procedure.

   This defines the load module and returns you to the list of business systems. The business system you had selected remains highlighted.

9. Add another load module by repeating the last three steps.

   If you receive an error message indicating that no such objects exist in the model (after selecting Add Load Module from the Edit menu), all of the procedure steps have been packaged.

Creating a Batch Job

To create a batch job (z/OS) that contains only one procedure, you must start the Construction Client, select a model, and choose the batch packaging type. Then select a business system.

After selecting a business system, select Edit, then Add Batch Job. In the dialog box that displays, enter a job name that meets the naming conventions discussed in this chapter. Select one procedure from the list of unpackaged procedures, then click OK. This creates the batch job packaging that defines its contents for generation. From here, go on to detailing the components in the batch job.

Add Components for Batch Packaging

This procedure explains how to add components for batch packaging.

Follow these steps:

1. Start the Construction Client.
2. Open a model by selecting Model.
3. Select Actions, then Open in the Construction Model Selection window.
4. Highlight a model from the list and select Open.
5. Select Code, then Package, then Batch.
6. This displays a list of business systems in the model.
7. Select a business system and then select Edit, then Add Batch Job.

   This displays the list of *unpackaged* batch procedures in this business system.

8. Specify a batch job name and then select a procedure from the list.

9. Click OK.

   This defines the batch job and returns you to the business system list.

   Only one procedure can be packaged in a batch job. The procedure steps that belong to that procedure are automatically defined in that batch job.

10. Add another batch job by repeating steps 6 through 8.

Modify Existing Load Module or Batch Job

Modifying an existing load module or batch job requires completion of the required diagrams in the toolset and check in of the model to the encyclopedia. You must have the appropriate authority to update the model's packaging. The procedures are similar but have an important difference for batch jobs. After adding the new procedure steps, you must detail them.

Modifying an Existing Load Module

You can modify an existing load module by adding or deleting procedure steps from the packaging. You can add a procedure step as long as it is available in the list of unpackaged procedure steps. If you delete the only procedure step in a load module, you delete the load module as well. Deleting a procedure step returns it to the list of unpackaged procedure steps.

Add a Procedure Step to an Existing Load Module

This procedure explains how to add a procedure step to an existing load module.

Follow these steps:

1. Start the Construction Client.
2. Open a model by selecting Model.
3. Select Actions, then Open in the Construction Model Selection window.
4. Highlight a model from the list and select Open.
5. Select Code, then Package and choose a packaging type: online, window, or cooperative.

   This displays a list of business systems in the model.

6. Select a business system and then select View, then Expand to view the load modules in the business system.

7. Select a load module and then select Edit, then Add Procedure Step to add procedure steps to the existing load module.

   This displays a list of *unpackaged* procedure steps in this business system.

8. Select the procedure steps and click OK.

   This adds the procedure step to the load module definition and returns you to the list of load modules. The load module you selected remains highlighted.

Delete a Procedure Step in an Existing Load Module

This procedure explains how to delete a procedure step in an existing load module.

Follow these steps:

1. Start the Construction Client.
2. Open a model by selecting Model.
3. Select Actions, then Open in the Construction Model Selection window.
4. Highlight a model from the list and click Open.
5. Select Code, then Package and choose a packaging type: online, window, or cooperative.

   This displays a list of business systems in the model.

6. Select a business system and then select View, then Expand to view the load modules in the business system.
7. Select a load module and then select View, then Expand to view the procedure steps in the load module.
8. Select a procedure step and then select Edit, then Delete to delete the procedure step from the load module.

   If the load module contains only one procedure step, this action deletes the load module as well.

   The procedure step returns to the list of *unpackaged* procedure steps.

Modifying an Existing Batch Job

You can add a procedure step to an existing batch job but you cannot delete a procedure step from an existing batch job. Batch procedure steps are only available for packaging in the batch job for which they were defined. When packaging for a different batch job, the newly created procedure step does not appear in a list of unpackaged procedure steps.

Add a Procedure Step to an Existing Batch Job

This procedure explains how to add a procedure step to an existing batch job (after completing the requirements for creation and check in).

Follow these steps:

1. Start the Construction Client.
2. Open a model by selecting Model.
3. Select Actions, then Open in the Construction Model Selection window.
4. Highlight a model from the list and select Open.
5. Select Code, then Package, then Batch.

   This displays a list of business systems in the model.
6. Select a business system and then select View, then Expand to view the batch jobs in the business system.
7. Select a batch job and then select Edit, then Add Procedure Step.

   This displays the list of *unpackaged* batch procedure steps for this batch job. Procedure steps are automatically defined to a batch job when the procedure, to which they belong, is packaged as the batch job.

   An unpackaged procedure step for a batch job will exist only if that procedure step was created *after* the batch job was defined.
8. Select one or more procedure steps and click OK.

   This adds the procedure step to the batch job definition and returns you to the batch job list.

Delete a Batch Job or Batch Job Step

This procedure explains how to delete a batch job step or batch job. (Removal of a procedure from the model would be one reason to delete a batch job in packaging. The batch job would have to be deleted prior to deleting the procedure step from the model.)

Follow these steps:

1. Start the Construction Client.
2. Open a model by selecting Model.
3. Select Actions, then Open in the Construction Model Selection window.
4. Highlight a model from the list and click Open.
5. Select Code, then Package, then Batch.

   This displays a list of business systems in the model.

6. Select a business system and then select View, then Expand to view the batch jobs in the business system.

7. To delete a batch job, select a batch job and then select Edit, then Delete.

8. To delete a batch procedure step, select a batch job then select View, then Expand to display the procedure steps.

Detail Component Objects

The Construction Client requires certain detail information about each of the component objects before beginning generation. This information includes:

- Member names
- Transaction codes

You can supply these names using your own naming conventions. As an alternative, the Construction Client can do it for you when you use the Complete option.

Note: If you do not specify the member names or use the Complete option, the Construction Client executes the Complete function before beginning generation. You should specify the names yourself to be sure that the transaction codes and member names have some meaning. For more information, see Complete Option.

Member Names

The following items have naming conventions you must follow:

- Source, screen, and load modules
- MFS device names
- Special code members generated for specific platforms or TP monitors

Source, Screen and Load Module Naming Rules

CA Gen uses these standard naming conventions for source, screen and load module member names regardless of the type of packaging used:

- One to eight alphanumeric characters as the load module name
- First character must be alphabetic
- Source, screen, and load module names must be unique within the set of names for all such objects in the model
- Source, screen, and load module names must not be a reserved word such as DATE or TIME

MFS Source Naming Rules

CA Gen uses these naming conventions for MID, MOD, and FMT source names for use with MFS devices.

- MID, MOD, and FMT names must be unique within the set of MID/MOD/FMT names for all such screen objects in the model
- One to eight alphanumeric characters for MID and MOD
- One to six alphanumeric characters for FMT
- First character must be alphabetic

Transaction Codes

These types of transaction codes are available for use with procedure steps:

- Clear screen
- Dialog flow

See the following sections for descriptions of the transaction codes and their use. This section also contains a discussion of transaction code naming rules.

Clear Screen Transaction Codes

A clear screen transaction code allows the user to invoke a transaction (procedure step). A clear screen is a blank screen that contains no input other than the transaction code and any free-form (unformatted) data passed with it. The clear screen transaction code may also be used to invoke the transaction from another transaction (using CA Gen NEXTTRAN).

If you want to be able to invoke a procedure step from a clear screen or another transaction, you must assign a *unique* clear screen transaction code to the procedure step. This allows the TP monitor and Dialog Manager to correctly identify the procedure step to execute. The Dialog Manager is discussed in detail in the chapter on system generation and installation.

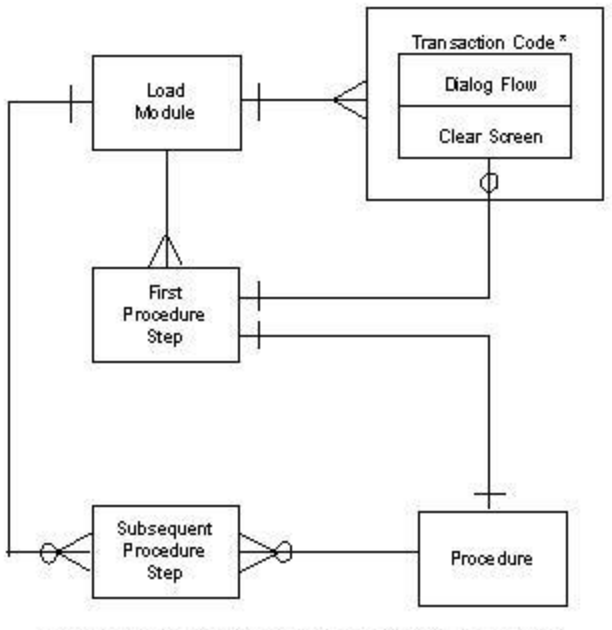
If you do not want a transaction to be invoked from a clear screen, but you still want to be able to access that transaction, ensure that the procedure step is flowed to and assign a dialog flow transaction code.

More information:

Load Module Structure (see page 107)

Dialog Flow Transaction Codes

A dialog flow transaction code is used by the Dialog Manager to manage flows (links and transfers) between procedure steps. The following illustration shows the relationship between procedure steps and dialog flow transaction codes.



* Partitioned on type, where type is Dialog Flow or Clear Screen

Each load module has one or more dialog flow transaction codes assigned to it. A dialog flow transaction code can belong to only one load module. Each load module has one or more procedure steps defined to it. The dialog flow transaction codes are assigned to the procedure steps (if they are flowed to from another procedure step). The same dialog flow transaction code can be assigned to more than one procedure step.

The TP monitor uses the dialog flow transaction code to determine which load module to execute. The transaction manager uses the input data (which identifies the procedure step name at runtime) to determine which procedure step to execute.

A sample mapping of transaction codes for a single load module appears in the following table:

Load Module Contains:	Clear Screen Transaction Code	Dialog Flow Transaction Code	Description
P1 PS1	A	A	Can be invoked directly or through dialog flow
P1 PS2		A	Second procedure step in P1 cannot be invoked directly, only through dialog flow
P2 PS1	B	A	Can be invoked directly or through dialog flow
P3 PS1		A	Cannot be invoked directly; no clear screen transaction code
P4 PS1		C	Cannot be invoked directly; no clear screen transaction code

Note: Pn = Procedure; PSn-Procedure step; where n represents the name of the component.

Procedure steps that cannot be invoked from a clear screen (P3 and P4) must have a dialog flow transaction code assigned. This type of procedure step is always accessed (flowed to) from another procedure step or a "formatted" screen. For example, in a multistep procedure (P1), the second and subsequent procedure steps can only be invoked by dialog flow transaction codes. Only the first procedure step in this example may be accessed by a clear screen transaction code, a dialog flow transaction code, or both.

A load module with a single procedure step assigned to it may have both types of transaction codes:

- The procedure step must have a clear screen transaction code if it is to be invoked from a clear screen.
- The procedure step must have a dialog flow transaction code if it is flowed to from another procedure step.

Note: The dialog flow transaction code is not used to determine the load module to execute on a return from a link. The transaction code used on a return from link is the transaction code that was active when the link was initiated.

Transaction Code Naming Rules

CA Gen uses these standard naming conventions for transaction codes regardless of the type of packaging used.

- Transaction code names must be unique within the set of all such names in the model
- One to eight alphanumeric characters
- First character must be alphabetic

Note: CICS transaction code names must not start with the letter C and must not exceed four characters in length.

If you want the application user to have a meaningful transaction code with which to invoke certain procedure steps, assign the clear screen transaction code manually. The Complete option uses a numeric transaction code that may not be as useful as a mnemonic one.

Complete Option

As an aid in packaging, the CSE Construction Client provides a Complete option that you can select at any of several points in the packaging process. The Complete option automatically assigns a unique name to each object. You can select this option at the business system, load module, procedure step, action block, or batch job levels.

At whatever level you select the Complete option, the Construction Client finishes required packaging for the selected object and all of its subordinates for that application type. For example, if you select Complete during online packaging for a business system, the Construction Client software:

- Assigns file names for procedure step and action block source code, screen code and any other required code, such as Message Format Service (MFS) for the IMS teleprocessing monitor
- Assigns clear screen and dialog flow transaction codes for each packaged online procedure step

Default File (Member) Names

If you do not specify a name, CA Gen assigns a default member name at generation. For procedure steps, CA Gen action blocks, screens, and MID and MOD names, the default name is the rightmost seven digits of the CSE object ID of the component preceded by:

- IMP - procedure step, action block, and screen names
- MID - MID names (for MFS member names)

- MOD - MOD name (for MFS member names)
- FMT - FMT name (for MFS member names)

For example, IMP49508 is a default member name for a procedure step with an object ID of 49508.

Detailing Load Modules

The following procedure explains how to add transaction codes and source member names after packaging at least one load module. You can use the Complete option to provide source member names and transaction codes automatically. However, the naming scheme used by the Complete option may not provide the best transaction code names for the application.

Follow these steps:

1. Start the Construction Client.
2. Open a model by selecting Model.
3. Select Actions, then Open in the Construction Model Selection window.
4. Highlight a model from the list and select Open.
5. Select Code, then Package and select a packaging type.

   This displays a list of business systems in the model.
6. Select a business system and then select View, then Expand to view the load modules in the business system.

   This displays the list of packaged load modules in this business system. If you receive an error message indicating that no such objects exist in the model, none of the procedure steps have been packaged into load modules.
7. Select a load module and then select Detail, then Properties.

   Detail information appears in the Load Module Detail window.
8. Select Trancodes (short for transaction codes).

   The Trancode Maintenance window appears. If no transaction codes have been specified this list is blank. You will need to add a transaction code for each procedure step within this load module, and any that you know you will package later.

   Note: This action creates a list of transaction codes for this load module only. It does not assign the created transaction codes to the procedure step(s) packaged in the load module. That is done in a step that appears later in this procedure. You must create a transaction code list for each load module in the business system.
9. Select Add to create new transaction codes. Type the name of the transaction code in the Add Trancode window and click OK.

10. Select Cancel to return to the Load Module Detail window when all of the trancodes have been defined.
11. Click OK to return to the List Load Modules window.
12. Select a load module and then select View, then Expand to list the procedure steps.
13. Select a procedure step and then select Detail, Properties.
14. Specify the source name, screen name (if applicable), the Dialog Flow trancode, and the Clear Screen trancode (if used) and click OK.
15. Select a procedure step and select View, then Expand to list the action blocks.
16. Select an action block and select Detail. Specify the source name for the action block and click OK.

Detailing Batch Jobs

The following procedure explains how to add source member names after packaging a batch job. You can also use the Complete option to provide source member names automatically.

Follow these steps:

1. Start the Construction Client.
2. Open a model by selecting Model.
3. Select Actions, then Open in the Construction Model Selection window.
4. Highlight a model from the list and select Open.
5. Select Code, then Package and select Batch packaging.

   This displays a list of business systems in the model.
6. Select a business system and then select View, then Expand to view the batch jobs in the business system.

   This displays the list of packaged batch jobs in this business system. If you receive an error message indicating that no such objects exist in the model, none of the procedures have been packaged.
7. Select a batch job and then select View, then Expand to list the procedure steps.
8. Select a procedure step and then select Detail then Package.

   Detail information appears in the Batch Job Step Detail window.
9. Specify the source name, member name, step name, PSB name, and DB2 attach type and then click OK.

   The PSB name is only used for IMS, and for DB2 attach type set to DLIBATCH or IMS BMP.

Load Module Size

Load module size is influenced by the number of procedure steps the load module contains. Actual load module size is machine and environment dependent.

When a load module is called, it must be loaded into memory. In some environments, calling the next load module causes the previous one to be removed from memory. Other environments can load multiple load modules simultaneously, or use dynamic linking to improve performance.

In some environments, a performance penalty occurs when loading and reloading load modules that are too small. However some environments, such as z/OS and CICS, are designed to manage small load modules very efficiently.

Some drawbacks of large load module size are that the executable file may be too large to fit into available memory, or the load module may exceed the size limitations of the linker.

z/OS Dynamic Linking

A program call to a routine that resides in the same load module is known as *static linking*; a program call to a routine that resides in a separate executable is known as *dynamic linking*. CA Gen generates a COBOL CALL literal statement (CALL 'module') when calling routines that are statically linked and when calling routines that are dynamically linked..

Dynamic linking is implemented by CA Gen using a z/OS specific packaging option that indicates how a routine is built. The way in which the routine is built determines how it will be called. The specific dynamically link packaging property associated with each procedure step, screen, or action block (including external action blocks) identifies how that component is resolved during the installation of the CA Gen load module in which it is packaged.

The designation of the dynamically link packaging option for a procedure step, screen, or action block can be set to Default. In this case, the dynamically link packaging option is derived from the dynamically link packaging option established in the business system owning the CA Gen load module in which the given procedure step, screen, or action block is defined.

The following values can be explicitly set for each individual procedure step, screen, or action block (or if set to Default, the value is derived from their respective Dynamically Link packaging option obtained from the default value established in the Business System):

- No-the routine is statically linked into the application.

- Yes-the routine is resolved as the target of a dynamic program call and as such is considered to be dynamically linked at runtime. During the installation of the load module, those components that have their associated dynamically link packaging property set, or derived, to Yes are built so they reside in their own separately loadable executables. These application routines reside in DLLs.
- Compatibility-the routine is resolved as the target of a dynamic program call and as such is considered to be dynamically linked at runtime. A routine designated as Compatibility will reside in a non-DLL executable.

A dynamic program call to a routine that resides in a DLL is invoked directly by the generated COBOL CALL statement. A dynamic program call to a routine that resides in a non-DLL module is indirectly invoked by CA Gen z/OS runtime.

Note: Every module that makes a dynamic program call to a routine marked for Compatibility must be regenerated and reinstalled to incorporate the call to the runtime routine that handles the indirect call processing.

For a module that was built before AllFusion Gen 7, identifying it for Compatibility allows that module to be dynamically called by a CA Gen routine that resides in a DLL.

It is possible to migrate procedure steps, screens, or action block routines that were built before AllFusion Gen 7 and must continue to reside in a non-DLL executable. CA Gen allows a module that is explicitly set to, or defaulted to, Compatibility to be built as a non-DLL executable. These migrated non-DLL executables use the same CA Gen z/OS runtime as those application routines that reside in DLLs.

The CA Gen Toolset, CSE, and Host Encyclopedia each provide an option that indicates whether modules marked for Compatibility should, or should not, be processed (generated and/or installed).

- If the intent is to use routines created with a release of CA Gen before Release 7, then Process modules marked for Compatibility should not be set when generating and/or installing a load module that contains the item marked for Compatibility.
- If the intent is to create a current version of the non-DLL routine, then Process modules marked for Compatibility should be set when generating, and installing a load module that contains the item marked for Compatibility. Selecting the Process modules marked for Compatibility check box causes the RI Trigger modules and all Action Blocks statically called by the module marked for Compatibility to be compiled twice-once using the compiler option NODLL and again using the compiler option DLL. If the Action Blocks are External Action Blocks, these must also be compiled with the NODLL option to be included in the Compatibility load module.

  Note: For more information about Process modules marked for Compatibility option, see Process modules marked for Compatibility.

The Compatibility option is intended to enable a phased migration of an existing application. It allows routines that have been migrated and reside in DLLs to interoperate with routines that reside in non-DLL executables. The non-DLL executables can themselves be migrated and built using the current release of CA Gen or they can remain as is, having been built with a release of CA Gen before the Release 7.

Note: It is possible that feature enhancements offered in future releases of CA Gen may require that any routine making use of the feature be built such that it resides in a DLL.

Features of Dynamic Link

The use of the dynamic linking feature has the potential to reduce total memory and CPU resources required by a TP monitor to process a load module. Dynamic linking common routines eliminates the need to link all of the load modules that use it but applications generated for DB2 require a bind or rebind.

The following list shows the key features and requirements:

- Only procedure steps, action blocks, and screens can be linked dynamically.
- Each dynamic linked module must be a fully resolved module. If this fully resolved module is a DLL, this means Referential Integrity triggers, other action blocks called statically within a procedure step, screen, or action block must be compiled as DLL and with the applicable CA Gen runtimes must be linked into the dynamic linked DLL. If this fully resolved module is a Compatibility module, this means Referential Integrity triggers, other action blocks called statically within a procedure step, screen, or action block must be compiled as NODLL and with the applicable CA Gen runtimes must be linked into the dynamic linked Compatibility load module.
- The dynamically link packaging property of individual procedure steps, screens, and action blocks can be set to Default. Default indicates that the value of the dynamically link property for that component is determined by the corresponding default setting, which is established at the business system level.
- As of AllFusion Gen 7, all the generated dynamic linked modules (including EABs), that do not have a dynamically link packaging property of Compatibility, are generated and built as DLLs.
- Marking a module for Compatibility causes its calling module to be generated and installed so that it processes the call using a module provided as part of the CA Gen runtime. The runtime code performs the dynamic program call to the non-DLL load module. In the cases where a module marked for Compatibility issues a dynamic program call to another non-DLL dynamic load module, only the module issuing the first dynamic program call to a Compatibility module requires generation and installation.
- Applications that run under TSOAE and contain modules marked for Compatibility cannot dynamically call modules built with a release prior to AllFusion Gen 7.6.

- Enhanced Map Block Mode applications containing screens marked for Compatibility cannot dynamically call screen managers built with a release before AllFusion Gen 7.6.
- Modules marked for Compatibility must follow standard OS or LE linkage conventions and must operate in the same AMODE as the caller. These modules must be non-DLL and must be stand-alone, fully resolved programs, eligible for a dynamic, OS style call.
- The ability to call a procedure step, action block, or screen dynamically allows application changes to become effective without having to link every load module using these modules. However, changes made to a module that uses DB2 SQL requires that the DB2 plan corresponding to the load module containing that module be rebound.
- In CICS, a PPT entry is required for each module called dynamically. For more information about defining CICS programs, see CICS documentation.

Considerations for Using the Dynamic Linking Option

The decision to dynamic link CA Gen modules, as opposed to generating one executable load module by means of a static link, should be based on the following factors:

- Compatibility-CA Gen creates DLLs with the exception of those modules marked for Compatibility. The Compatibility option uses specific CA Gen runtime to enable DLLs to issue a dynamic program call to non-DLL load modules. This requires that the module issuing the dynamic program call be regenerated and reinstalled. CA Gen allows modules marked for Compatibility to be rebuilt in such a way that they are able to use CA Gen runtime DLLs. This requires that the modules marked for Compatibility be reinstalled and, if necessary, regenerated as specified in z/OS Application Migration in the *Release Notes*. RI triggers and action blocks, including EABs, statically called by Compatibility modules must be built using the NODLL compiler option. When these RI triggers and action blocks are also used in CA Gen applications that are built as DLLs, they must be compiled using the DLL option. Selecting the option Process modules marked for Compatibility causes the RI triggers and the statically called action blocks to be generated and precompiled once but compiled twice, once as NODLL and again as DLL.
- DLLs-CA Gen runtimes are DLLs. These DLL runtimes are no longer included in generated modules, except for a few that are statically linked with the CA Gen Batch, Online, or Server Managers. This applies to both static and dynamic linked CA Gen modules.
- Size-A dynamic linked module must be fully resolved and include any component that it calls using a static call. This includes some CA Gen runtimes and Referential Integrity triggers. When more than one dynamic linked modules use the same RI modules, these RI modules are included in each dynamic linked module. Most of the CA Gen runtimes are no longer included in each of the DLL applications built by CA Gen so they do not increase the size of these modules.

- Volatility-When a static linked module is changed, the calling module containing the statically linked module must be relinked. Linking a frequently changed module dynamically eliminates the requirement to link every load module that calls it.
- Shared modules-Action blocks that are widely reused within an application environment are good candidates for dynamic linking. Linking a shared module dynamically decreases the amount of storage required during execution.
- Frequency of use-Dynamic linking may be considered for load modules that are invoked infrequently. Dynamic linking modules that are called infrequently would decrease the size of the base load module. Examples of modules called infrequently are exception handling routines and processing options that are rarely selected by the user.

Online Packaging

Online Packaging consists of the following basic activities:

- Selecting the Business System
- Defining load modules
- Defining source names
- Assigning transaction codes

Selecting the Business System

The first step in load module packaging is to select the business system. When you defined business systems with either the Business System Definition Tool or Matrix Processor in Analysis, you named the systems. These names appear in the Online Packaging window when you select Code, then Package, then Online.

Defining Load Modules

Defining a load module consists of the following activities:

- Name the load module
- Add procedure steps

Name the Load Module

The name you choose for the load module becomes the name of the executable file. When you package for remote installation, the load module name also becomes the remote file name.

Downstream Effects: When generating for a remote installation, each load module becomes a separate remote file and is linked on the target system into a separate executable module. The Referential Integrity modules are packaged into a single remote file. The DDL associated with an application is packaged as a single remote file.

Add Procedure Steps

Select the procedure steps to be packaged in each load module from the list of all the unpackaged procedure steps defined for the business system. You do not explicitly package action blocks. By adding procedure steps to the load module, all action blocks used by the procedure step are also included in the load module, as well as all action blocks used by the action blocks.

Flow definitions are independent of packaging options. When you create a Dialog Flow Diagram you do not need to consider how the procedures will be packaged. Likewise, packaging is not dictated by the dialog flow. An application may, however, run more efficiently when each load module contains procedure steps that are to be used together.

You can use the Filter feature to control the display of the unpackaged procedure steps.

More information:

Construction Concepts (see page 13)
Construction Stage (see page 13)
Processing and Output (see page 14)
Construction Client Installation (see page 14)
Remote Installation (see page 14)
Installation Target (see page 15)
DBMS (see page 15)
Overview of Construction (see page 16)
Prerequisites (see page 18)
External Action Blocks (see page 18)
Construction Process (see page 18)
Components (see page 19)
Transfer Method (see page 19)
DDL Option (see page 19)
Tasks in the Construction Process (see page 19)
Generate the DDL (see page 20)
Load Module Packaging Steps (see page 20)
Specify the Environment (see page 20)
Generate the Source Code (see page 21)
Implementation Package (see page 21)
Install External Action Blocks (see page 21)
Start the CSE Construction Client (see page 21)
Open a Model (see page 22)
Filter the Display List (see page 22)
Filter Pushbutton (see page 23)
Filter Check Box (see page 23)
Filter Menu Item (see page 23)
Specifying Filter Values (see page 23)
Component Selection (see page 24)

Defining Source Names

Each procedure step, action block, or screen must have a source member name for the Construction Client to use when storing the generated code. You can assign source member names during packaging, or the Construction Client will assign them for you during generation of the load module. The Construction Client software verifies the uniqueness of source member names within the model or subset on the workstation. For information about names supplied by the Complete option see the Default File (Member) Names section.

Assigning Transaction Codes

For certain operating system and teleprocessing monitor combinations, other generated code needs to be named. You can add transaction codes to a load module and then assign them to the packaged procedure steps during packaging. If you do not, the Construction Client adds and assigns transaction codes for you during load module generation.

If you want the application user to have a meaningful transaction code with which to invoke certain procedure steps, assign your own clear screen transaction codes to them.

The Construction Client displays a window for entry of the transaction code and source member name information. The fields contained in the window depend on the environment and configuration options you specify. See the following table:

Environment and Configuration Options	Information Required by Panel
No display (all environments)	Source member name Dialog flow transaction code
Display, z/OS with IMS MFS	Source member name Screen member name Clear transaction code Dialog flow transaction code MID, MOD, and FMT member names
Display (other environments)	Source member name Screen member name Clear screen transaction code Dialog flow transaction code, if procedure is flowed to

Batch Packaging

A batch job is a group of one or more batch job steps. A job step equates to a procedure step and has an associated load module.

Note: Batch Processing is implemented in z/OS environments as a batch job using JCL. In other environments, batch processing is implemented as online with no display.

Batch Packaging consists of the following basic activities:

- Selecting the business system
- Defining batch jobs
- Defining procedure step properties

Selecting the Business System

The first step in batch load module packaging is to select the business system. When you defined business systems with either the Business System Definition Tool or Matrix Processor in Analysis, you named the systems. These names appear in the Batch Packaging window when you select Code, then Package, then Batch.

Defining Batch Jobs

Defining a batch job consists of the following activities:

- Name the batch job
- Add procedure

Name the Batch Job

Batch job names must be unique within the set of names for all such objects in the model. Select a business system. When you select Edit, then Add Batch Job, the Construction Client displays a list of the unpackaged batch procedures. Specify a job name in the entry field provided.

Add Procedure

After specifying a job name, select at least one unpackaged batch procedure, and then click OK. This creates the batch job and packages the procedure and its procedure steps. Once the procedure is packaged, you can view the procedure steps by expanding the batch job.

Note: Batch packaging is unique in that you can not package more than one procedure into a load module.

You can also review the detail information by selecting Detail. This window allows you to enter job step names. You may want to assign step names to a procedure step if more than one is present.

You can use the Filter feature to control the display of the unpackaged batch procedures.

More information:

Construction Concepts (see page 13)
Construction Stage (see page 13)
Processing and Output (see page 14)
Construction Client Installation (see page 14)
Remote Installation (see page 14)
Installation Target (see page 15)
DBMS (see page 15)
Overview of Construction (see page 16)
Prerequisites (see page 18)
External Action Blocks (see page 18)
Construction Process (see page 18)
Components (see page 19)
Transfer Method (see page 19)
DDL Option (see page 19)
Tasks in the Construction Process (see page 19)
Generate the DDL (see page 20)
Load Module Packaging Steps (see page 20)
Specify the Environment (see page 20)
Generate the Source Code (see page 21)
Implementation Package (see page 21)
Install External Action Blocks (see page 21)
Start the CSE Construction Client (see page 21)
Open a Model (see page 22)
Filter the Display List (see page 22)
Filter Pushbutton (see page 23)
Filter Check Box (see page 23)
Filter Menu Item (see page 23)
Specifying Filter Values (see page 23)
Component Selection (see page 24)

Defining Job Step Properties

Batch job step names must be unique within the set of generation objects in the model.

Packaging done with the Host Encyclopedia Construction Toolset can be checked out to the CSE and used for generation of source code.

Window Packaging

Window packaging consists of the following basic activities:

- Selecting the Business System
- Defining load modules

- Defining source names
- Assigning transaction codes

Window procedure steps for GUI applications can be packaged in one or more load modules. GUI applications may consist of one or more load modules, each of which may contain one or more procedure steps.

Selecting the Business System

The first step in window load module packaging is to select the business system. When you defined business systems with either the Business System Definition Tool or the Matrix Processor in Analysis, you named the systems. These names appear in the Window Packaging window when you select Code, then Package, then Window.

Defining Load Modules

Defining a load module consists of the following activities:

- Name the load module
- Add procedure steps

Name the Load Module

The name you choose for the load module becomes the name of the executable file. Note that when packaging for remote installation, the load module name also becomes the remote file name.

Add Procedure Steps

Select the procedure steps to be packaged in the load module from the list of all the unpackaged procedure steps defined for the business system. You do not explicitly package action blocks. By adding procedure steps to the load module, all action blocks used by the procedure step are also included in the load module, as well as all action blocks used by the action blocks.

An application usually runs most efficiently when each load module contains procedure steps that are likely to be used together. Flow definitions are independent of packaging options. When you create a Dialog Flow Diagram you do not need to consider how the procedures will be packaged. Likewise, packaging is not dictated by the Dialog Flow.

Downstream Effects: When generating for remote installation, each load module becomes a separate remote file and is linked on the target system into a separate executable module. The Referential Integrity modules are packaged into a single remote file. The DDL associated with an application is packaged as a single remote file.

You can use the Filter feature to control the display of the unpackaged procedure steps.

More information:

Construction Concepts (see page 13)
Construction Stage (see page 13)
Processing and Output (see page 14)
Construction Client Installation (see page 14)
Remote Installation (see page 14)
Installation Target (see page 15)
DBMS (see page 15)
Overview of Construction (see page 16)
Prerequisites (see page 18)
External Action Blocks (see page 18)
Construction Process (see page 18)
Components (see page 19)
Transfer Method (see page 19)
DDL Option (see page 19)
Tasks in the Construction Process (see page 19)
Generate the DDL (see page 20)
Load Module Packaging Steps (see page 20)
Specify the Environment (see page 20)
Generate the Source Code (see page 21)
Implementation Package (see page 21)
Install External Action Blocks (see page 21)
Start the CSE Construction Client (see page 21)
Open a Model (see page 22)
Filter the Display List (see page 22)
Filter Pushbutton (see page 23)
Filter Check Box (see page 23)
Filter Menu Item (see page 23)
Specifying Filter Values (see page 23)
Component Selection (see page 24)

Defining Source Names

Each procedure step or action block must have a source name for the Construction Client to use when storing the generated code. You can assign source names during packaging, or the Construction Client will assign them for you during generation of the load module. The Construction Client verifies the uniqueness of source names within the model.

Assigning Transaction Codes

You can add transaction codes to a load module and then assign them to the packaged procedure steps during packaging. If you do not, the Construction Client adds and assigns transaction codes for you during load module generation.

If you want the application user to have a meaningful transaction code with which to invoke certain procedure steps, assign your own clear screen transaction codes to them.

Cooperative Packaging

Packaging for a client/server application, which consists of a client component and a server component, is performed under Cooperative packaging.

Cooperative Packaging consists of the following basic activities:

- Selecting the business system
- Defining load modules
- Defining source names
- Assigning transaction codes
- Specifying environment parameters

Selecting the Business System

The first step in load module packaging for a client/server application is to select the business system. When you defined business systems with either the Business System Definition Tool or the Matrix Processor in Analysis, you named the systems. These names appear on the Cooperative Packaging window when you select Code, then Package, then Cooperative.

Defining Load Modules

Defining a load module consists of the following activities:

- Name the load module
- Add procedure steps

Name the Load Module

The name you choose for the load module becomes the name of the executable file. Note that when packaging for remote installation, the load module name also becomes the remote file name.

Also, you need to select the load module type. The load module types are *server manager* for the server procedures and *window manager* for the client procedures.

More information:

Construction Concepts (see page 13)
Construction Stage (see page 13)
Processing and Output (see page 14)
Construction Client Installation (see page 14)
Remote Installation (see page 14)
Installation Target (see page 15)
DBMS (see page 15)
Overview of Construction (see page 16)
Prerequisites (see page 18)
External Action Blocks (see page 18)
Construction Process (see page 18)
Components (see page 19)
Transfer Method (see page 19)
DDL Option (see page 19)
Tasks in the Construction Process (see page 19)
Generate the DDL (see page 20)
Load Module Packaging Steps (see page 20)
Specify the Environment (see page 20)
Generate the Source Code (see page 21)
Implementation Package (see page 21)
Install External Action Blocks (see page 21)
Start the CSE Construction Client (see page 21)
Open a Model (see page 22)
Filter the Display List (see page 22)
Filter Pushbutton (see page 23)
Filter Check Box (see page 23)
Filter Menu Item (see page 23)
Specifying Filter Values (see page 23)
Component Selection (see page 24)

Add Procedure Steps

For client/server applications, procedure steps are packaged according to load module types.

For the server manager load module type, select the procedure steps to be packaged in the load module from the list of unpackaged procedure steps defined for the business system. Procedure steps that are online with no display appear in this list.

For the window manager load module type, select the procedure steps to be packaged from the list of unpackaged procedure steps defined for the business system. The procedure steps that are to be packaged in window load modules are online with display and with no display.

For both server manager and window manager load modules, you do not explicitly package action blocks. By adding procedure steps to a load module, all action blocks used by the procedure step are also included in the load module, as well as all action blocks used by other action blocks.

Packaging is dependent on the flow definitions for a client/server design. You cannot package the client and server procedure steps together because they will reside on different platforms. This may affect the procedure step definition and dialog flow design.

Downstream Effects: When generating for a remote installation, each load module becomes a separate remote file and is linked on the target system into a separate executable module. The Referential Integrity modules are packaged into a single remote file. The DDL associated with an application is packaged into a single remote file.

You can use the Filter feature to control the display of the unpackaged procedure steps.

Defining Source Names

Each procedure step or action block must have a source name for the Construction Client to use when storing the generated code. You can assign source names during packaging, or the Construction Client will assign them for you during installation of the load module. The Construction Client software verifies the uniqueness of source names within the model or subset on the workstation.

For the client procedures, the source name is generated into the code as the function name in C. For the server procedures, the source name is generated into the code as the PROGRAM-ID in COBOL and the function name in C.

Assigning Transaction Codes

You can manually add transaction codes to a load module and then assign them to the packaged procedure steps during packaging. Otherwise use the Complete option. The Construction Client will automatically add and assign transaction codes for you during load module generation.

These transaction codes allow navigation between procedure steps. If you want the application user to have a meaningful transaction code with which to invoke certain procedure steps, you may want to assign your own clear screen transaction codes to them.

For the server procedures, specify the dialog flow transaction code and a clear screen transaction code. Although clear screen transaction codes are not required for servers since they do not have screens, currently if you do not specify a transaction code, a TIRM629E message will display.

For the client procedures, specify the clear screen transaction codes. You can also specify the dialog flow transaction codes, which are optional but are used by the generator if specified.

Note: CICS transaction code names should not start with the letter C and should not exceed four characters in length.

Specify Environment Parameters

The client and server parts of the application reside on different platforms. As a result of this, you must set the environment parameters for each part separately. This is done during packaging. To set the environment parameters, select Code, then Package, then Cooperative. Select a business system and select View, then Expand to see the window (client) and server managers. Select a client or server manager and then select Detail, then Configuration to set the environment parameters.

Note: Setting the environment parameters at the business system level has no effect for Cooperative applications. Paths for all application types are set at the business system level.

Results of Packaging

To view the results of each type of packaging, select a business system and select View, then Expand. The resulting display shows the load modules packaged in the business system. To see the procedure steps packaged in the load module, select a load module and then select View, then Expand.

To see the action blocks used by the procedure step, including all action blocks used by these action blocks, select and expand a procedure step.

To check for any unpackaged procedure steps, select Edit, then Add Load Module. This displays any unpackaged procedure steps in this business system. If you receive an error message stating that no such objects exist in the model, or if the resulting list is blank, all procedure steps have been packaged.

Chapter 5: Target Environment and Construction Paths

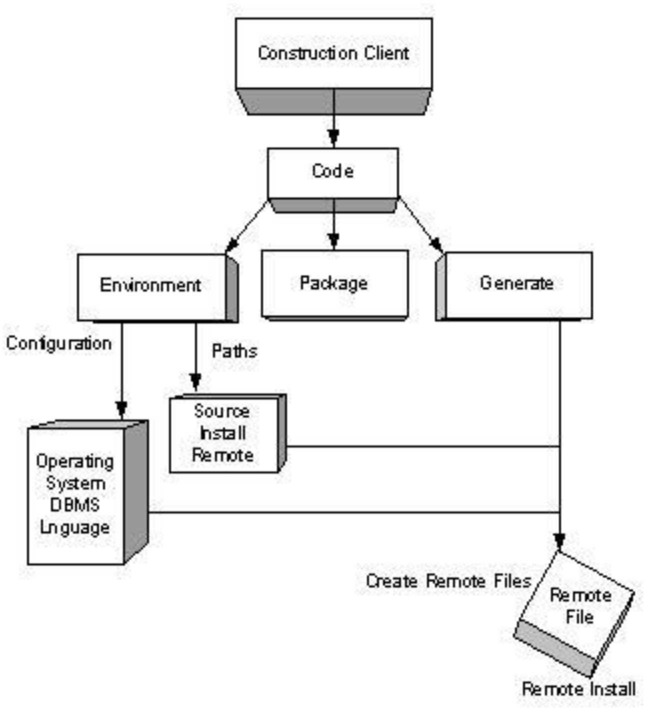
This chapter explains the specifications for the target environment and construction paths.

This section contains the following topics:

Target Environment Specifications

Setting the specifications for the target environment and construction paths is one of the activities you must complete before generating the IPs. The Environment option is used to specify the parameters for the environment in which a CA Gen-generated application executes (the production environment). The Construction Paths provide locations for the files created during generation.

The following illustration depicts an overview of environment and path parameter usage:



The target configuration and paths for load modules and batch jobs are specified at the business system level for online, batch, and windowed applications. For client/server (Cooperative) applications, you must set environment parameters for both window manager (client) and server manager (server) for each load module.

Application Component	Environment	Paths
Load Module	model (encyclopedia)	model (encyclopedia)
DDL	local (client)	model (encyclopedia)
RI triggers	local (client)	model (encyclopedia)

Note: Regardless of the platform, the term Load Module is used in the Packaging process as a generic way to refer to the set application code that is contained in a fully resolved executable application. Starting with AllFusion Gen 7, applications when installed on z/OS are built as fully resolved executables that reside in DLLs, except for those application components marked for Compatibility. Components marked for Compatibility are built as z/OS non-DLL load modules. Before AllFusion Gen 7, z/OS executables reside in z/OS load modules.

Check the environment and paths for the RI trigger modules before each trigger generation, to be sure that they are correct. RI trigger environments and DDL environments are shared. If you set override for either one, you are also changing the environment for the other.

The following sections describe the rules for setting environment parameters and construction paths.

Prerequisites

Before you set the environment and path parameters:

- Obtain at least Update authority to set and save environments and paths.
- Define at least one business system.
- Check in the model to the encyclopedia.
- Start the Construction Client and select a model.

How to Set the Target Environment

To set the target environment parameters, start the Construction Client, open a model, and select Code, then Environment, then Configuration. You can also access the environment parameters from the generation and packaging windows and dialogs. This allows you to set the parameters before you select components for generation or during component selection.

You can use the same specifications for all business systems in a model, or you can use different specifications for separate business systems. In either case, the following rules apply:

- The current release supports remote installation.
- You must define target environment parameters before you can generate remote files.
- Environment parameters are checked against a list of valid combinations on the server. Errors are reported before the settings are saved.
- Environment parameters apply to a particular business system and are stored as part of the business system for a model.
- You can provide temporary overrides of target environment parameters from the Generation Defaults window. Override values are stored locally (on the client machine).
- Environment parameters for the DDL and Referential Integrity (RI) triggers are specified separately. These objects apply to an entire model and do not belong to a particular business system.

- RI trigger and DDL target environment parameters are not stored on the model. RI trigger and DDL environment parameters are stored on the client machine.
- Environment parameter option Process modules marked for Compatibility applies to RI triggers when the target platform is z/OS. This option must be selected when the applications that use these RI triggers contain modules marked for Compatibility.
- Target environment parameters for the client and server portions of a client/server application may be different. These parameters are set during Packaging. You set window manager parameters for the client application and server manager parameters for the server application.
- Environment parameters for MFS devices are set at the Encyclopedia level.

Downstream Effects: The downstream effects of the environment parameters occur during generation and installation and during execution of the application system. The environment parameters enable the generated code to take advantage of certain features of the production environment. This usually results in improved application performance.

Environment Parameters List

Before you can use the construction functions to generate an application system, you must specify the target environment for the generated application. The Construction Client requires the following information about the environment:

- Operating System
- Database Management System (DBMS)
- Source Language
- Teleprocessing (TP) Monitor
- Profile Manager
- Screen Type
- Clear Screen Default
- Maximum Segment Size (IMS)
- Restartable Application
- Extended Attribute Support
- Enforce Data Modeling Constraints
- Optimize Import View Initialization

For CICS targets only:

- Handle Command ABENDS
- XCTL for Flows When Possible
- Pseudoconversational Support

Dynamic Link Defaults (MVS):

- Procedure steps
- Action blocks
- Screen managers

Dynamic Linking Options (for z/OS operating systems only)

The Dynamically Link packaging option is available for applications built for the z/OS execution environments. The Dynamically Link packaging property of each individual Procedure steps, Screens and Action Blocks (including EABs) can be set to Default, Yes, No, or Compatibility. When set to Default, the actual value is derived from the value entered for the Dynamic Link Default option of the specific component.

For a business system the following values can be specified:

- Yes-indicates the component is resolved as the target of a dynamic program call and as such is considered to be dynamic linked at runtime. The component is generated and built so they reside in their own separately loadable executables. These applications are built as DLLs.
- No-indicates that the component will be statically linked into the application.
- Compatibility-indicates that the component will be dynamically called at runtime and will reside in a non-DLL module. The Compatibility option uses specific CA Gen runtime to enable DLLs to issue a dynamic program call to non-DLL load modules. This requires that the module issuing the dynamic program call be regenerated and reinstalled. CA Gen allows modules marked for Compatibility to be rebuilt in such a way that they are able to use CA Gen runtime DLLs. This requires that the modules marked for Compatibility be reinstalled and, if necessary, regenerated as specified in z/OS Application Migration section in the *Release Notes*.

More information:

Packaging (see page 41)
Packaging Process (see page 41)
Types of Packaging (see page 44)
Prerequisites (see page 44)
How to Package (see page 44)
Object Relationships (see page 45)
Create Load Module or Batch Job (see page 46)
Creating a Load Module (see page 46)
Add Objects for Packaging (see page 46)
Creating a Batch Job (see page 47)
Add Components for Batch Packaging (see page 47)
Modify Existing Load Module or Batch Job (see page 48)
Modifying an Existing Load Module (see page 48)
Add a Procedure Step to an Existing Load Module (see page 48)
Delete a Procedure Step in an Existing Load Module (see page 49)
Modifying an Existing Batch Job (see page 49)
Add a Procedure Step to an Existing Batch Job (see page 50)
Delete a Batch Job or Batch Job Step (see page 50)
Detail Component Objects (see page 51)
Member Names (see page 51)
Source, Screen and Load Module Naming Rules (see page 51)
MFS Source Naming Rules (see page 52)
Transaction Codes (see page 52)
Clear Screen Transaction Codes (see page 52)
Dialog Flow Transaction Codes (see page 53)
Transaction Code Naming Rules (see page 55)
Complete Option (see page 55)
Default File (Member) Names (see page 55)
Detailing Load Modules (see page 56)
Detailing Batch Jobs (see page 57)
Load Module Size (see page 58)
z/OS Dynamic Linking (see page 58)
Features of Dynamic Link (see page 60)
Considerations for Using the Dynamic Linking Option (see page 61)
Online Packaging (see page 62)
Selecting the Business System (see page 62)
Defining Load Modules (see page 62)
Name the Load Module (see page 63)
Add Procedure Steps (see page 63)
Defining Source Names (see page 64)
Assigning Transaction Codes (see page 65)
Batch Packaging (see page 65)
Selecting the Business System (see page 66)
Defining Batch Jobs (see page 66)
Name the Batch Job (see page 66)
Add Procedure (see page 66)
Defining Job Step Properties (see page 67)

MFS Devices

MFS device characteristics are only used with the z/OS IMS environment and are stored at the encyclopedia level. They are not project-specific. The device characteristics are set as a result of the encyclopedia installation and should be reviewed against site requirements.

Note: Only an encyclopedia administrator can change the device characteristics.

If multiple encyclopedia administrators manage projects from a single encyclopedia, they must coordinate the configuration of the MFS device characteristics. If you change the device characteristics, the change affects all users of the CSE targeting IMS.

Follow these steps:

1. Start the Construction Client.
2. Select Encyclopedia, then Terminal Characteristics, then Modify MFS Devices.
3. Select a device type and then choose Select.

   This displays the Current MFS Device window.
4. Select the desired extended attribute from the pull-down list.
5. Select the Selected for Generation check box to enable code generation for the selected MFS device, then click OK.

Available Options

Your environment parameter choices depend on which options you purchased for your CA Gen configuration. A description of each option appears in the online help for the Construction Client.

Select z/OS Screen Generator Options

The Enhanced Screen Generator is the default for all block mode applications when z/OS is the target platform. The Enhanced Screen Generator is not available for other target platforms. You can use the previous version of the screen generator by setting the environment variable TIMAPGEN to 1. The enhanced screen generator is used when the environment variable is set to 2, or when the variable is undefined. The variable can be set in a command file, a shell script, or a batch file that starts the CA Gen software. To permanently override the default selection of the Enhanced Screen Generator, add the TIMAPGEN variable to the system environment variable definitions. See your operating system documentation for more specific information on setting environment variables.

When the TIMAPGEN variable is defined in the same session that starts CA Gen, the definition is local to that session and all its child sessions. The definition is not local to other sessions on the system. It is possible to have more than one version of CA Gen, each with a different environment variable.

How to Set Construction Paths

To set the construction paths, start the Construction Client, open a model, and select Code, then Environment. Select a business system or Referential Integrity triggers from the displayed list and select Detail, then Paths. You can also access the construction paths from the generation and packaging windows. Use a naming convention that reflects the application when setting paths for the various generated components. For example, use the same source path for load module and RI trigger source, so that all source code is grouped in one location. Use the same remote installation path so that the load module and RI trigger remote files (.rmt files) are in a single location.

Use the following rules when setting paths:

- Only remote installation is available in this release. Be sure to set the remote installation path to provide a location for the remote files.
- If the Construction Client appends the user ID to a path name, it is the name used to log on to the client.
- You must set construction paths before you generate remote files. Not entering a generation path results in an error that stops generation processing.
- The paths reference locations on the server machine, so you must enter the paths using the appropriate naming conventions and restrictions for your server. Using the incorrect path type results in an error.

  Paths are checked on the server and errors are reported before the path names are saved.

  Construction paths are associated with business systems as part of the business system for a model.

- You can provide temporary overrides of construction paths from the Generation Defaults window. Overrides are stored locally (on the Client).
- z/OS construction library definitions are included when complete models are transferred between encyclopedias. (z/OS construction library definitions are not included in subsets transferred between encyclopedias.)
- Construction paths for the DDL and Referential Integrity triggers are specified separately. These objects apply to an entire model and do not belong to a particular business system.

Path Types

When setting paths, you provide locations for objects created during generation. These locations are directories on the server and must reflect the naming conventions and restrictions in use for that platform. For example, path names on UNIX servers are case-sensitive.

When you specify paths, you must do so for each of these types of objects:

- Source code
- Installation control files
- Remote files (one remote file for each component, operations library, z/OS library, or proxy)
- Bitmaps (optional, for GUI applications only)

The generation procedure appends certain information to the path names depending on what is being generated. This helps to differentiate between different versions of the same application generated for more than one platform.

For load module, RI trigger source, and the installation control file, generation appends the name of the operating system and source code language. For DDL, generation appends the operating system and DBMS names (in that order).

Source Code

The Generate option uses this path to contain the files created during generation of the source code.

Installation Control Files

The Installation Control files contain information used to install the remote files on the target platform. The Build Tool uses the install control information to link-edit and bind the components into load modules that can be executed.

Remote Files

The remote file contains the generated source and instruction for installation. Each remote file has a .rmt file extension. Before you can install the application into the target environment, you must transfer the remote files to the target platform. When all generation steps are complete, the remote files that exist will include the following files:

- One DDL remote file for the database
- One RI trigger module remote file
- One remote file for each load module
- One remote file for each proxy
- One remote file for each operations library
- One remote file for each z/OS library

Generated MFS

Generation uses the path you specify here for the locations of the IMS source files when you target z/OS operating systems with IMS-MFS as the teleprocessing monitor. After generation, this library contains the source members for MID, MOD, and FMT source using either system assigned names, or the names you specify when detailing procedure steps (during packaging).

More information:

Packaging (see page 41)
Packaging Process (see page 41)
Types of Packaging (see page 44)
Prerequisites (see page 44)
How to Package (see page 44)
Object Relationships (see page 45)
Create Load Module or Batch Job (see page 46)
Creating a Load Module (see page 46)
Add Objects for Packaging (see page 46)
Creating a Batch Job (see page 47)
Add Components for Batch Packaging (see page 47)
Modify Existing Load Module or Batch Job (see page 48)
Modifying an Existing Load Module (see page 48)
Add a Procedure Step to an Existing Load Module (see page 48)
Delete a Procedure Step in an Existing Load Module (see page 49)
Modifying an Existing Batch Job (see page 49)
Add a Procedure Step to an Existing Batch Job (see page 50)
Delete a Batch Job or Batch Job Step (see page 50)
Detail Component Objects (see page 51)
Member Names (see page 51)
Source, Screen and Load Module Naming Rules (see page 51)
MFS Source Naming Rules (see page 52)
Transaction Codes (see page 52)
Clear Screen Transaction Codes (see page 52)
Dialog Flow Transaction Codes (see page 53)
Transaction Code Naming Rules (see page 55)
Complete Option (see page 55)
Default File (Member) Names (see page 55)
Detailing Load Modules (see page 56)
Detailing Batch Jobs (see page 57)
Load Module Size (see page 58)
z/OS Dynamic Linking (see page 58)
Features of Dynamic Link (see page 60)
Considerations for Using the Dynamic Linking Option (see page 61)
Online Packaging (see page 62)
Selecting the Business System (see page 62)
Defining Load Modules (see page 62)
Name the Load Module (see page 63)
Add Procedure Steps (see page 63)
Defining Source Names (see page 64)
Assigning Transaction Codes (see page 65)
Batch Packaging (see page 65)
Selecting the Business System (see page 66)
Defining Batch Jobs (see page 66)
Name the Batch Job (see page 66)
Add Procedure (see page 66)
Defining Job Step Properties (see page 67)

Bitmaps

Generation uses the path you specify here for the location of the bitmap files. Use of bitmaps is an option for GUI applications. Bitmap paths should be assigned at the business system level.

Note: For more information about the use of bitmap files, see the *Design Guide.*

Chapter 6: Generating and Installing Remote Files

This chapter provides information about generating and installing remote files.

This section contains the following topics:

Remote Installation

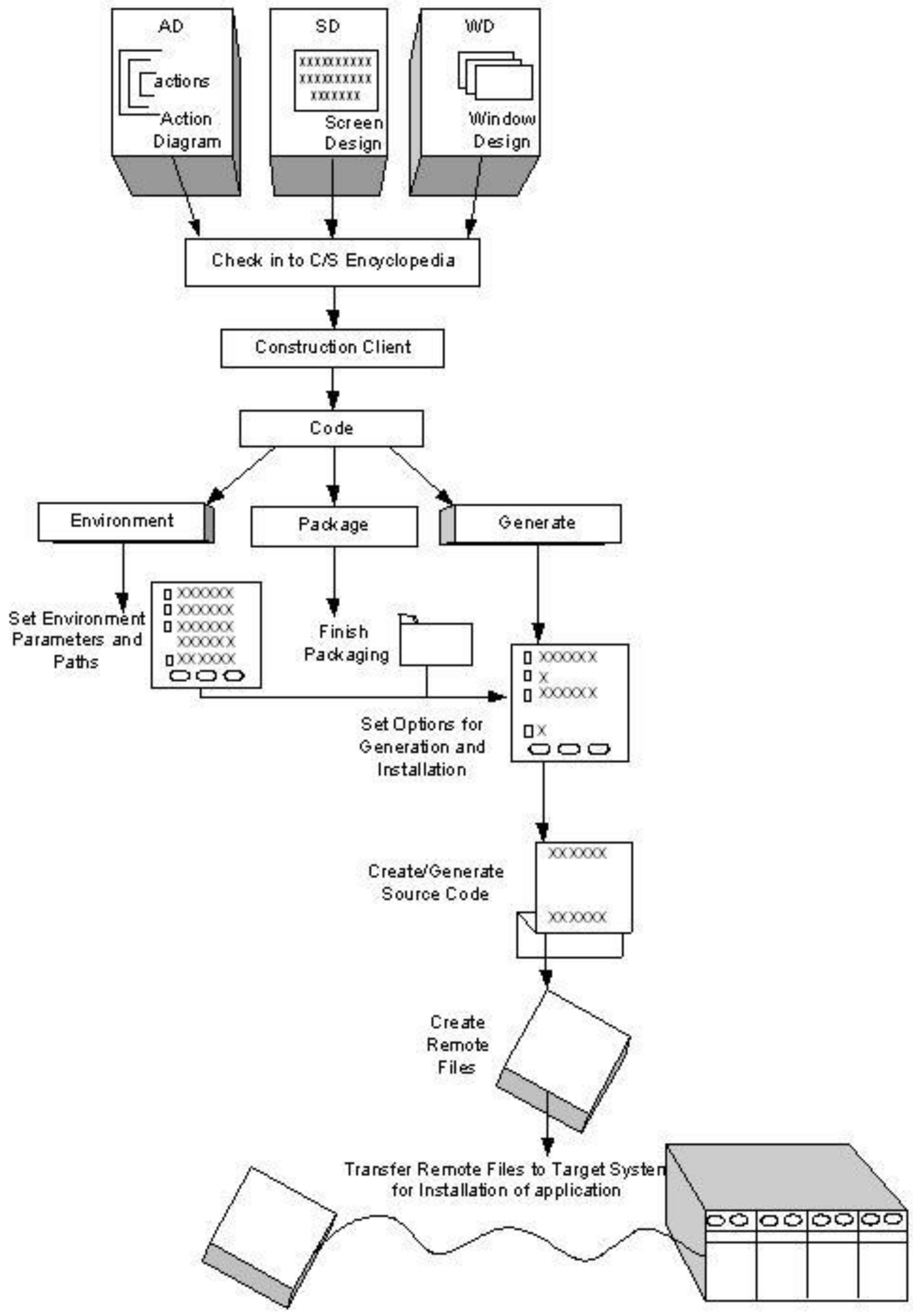
The Client Server Encyclopedia (CSE) Construction Client currently allows remote installations only. A remote installation is one in which an Installation Toolset is used to compile, link, and bind the source code generated by the CSE or a workstation toolset. The remote installation target environment may or may not be the same as the Construction Client's environment.

After the actual code is generated, remote installation creates a set of remote files that contain all of the components necessary to compile and install the application components on a target system. You must move these remote files to the target system and install them in an environment supported by CA Gen before using the application. Remote installation requires the presence of certain CA Gen components on the selected target system.

If you are using External Action Blocks in the application, you must complete, compile, and link edit them outside of the Construction Client. Move them to the target environment and add them to the application using the appropriate methods for the Installation Toolset. For more information, see the documentation for the Installation Toolset being used.

If an application, including Cooperative Server applications that target CICS or IMS, is to be used on a z/OS platform and you have the Host Encyclopedia Construction Toolset installed on the host where the application is to execute, it is not necessary to create remote files using the CSE Construction Client. Instead, you can check in the model and perform construction on the Host to generate and install the application. Ensure the Cooperative Servers are packaged before checking in the model.

The following illustration provides an overview of the process for generating remote files:


AD
actions
Action Diagram
SD
Screen Design
WD
Window Design
Check in to C/S Encyclopedia
Construction Client
Code
Environment
Package
Generate
Set Environment Parameters and Paths
Finish Packaging
Set Options for Generation and Installation
Create/Generate Source Code
Create Remote Files
Transfer Remote Files to Target System for Installation of application

Remote Files

Three different types of remote files are created using CSE Construction. Each of the remote files contains a different type of information necessary for the application.

Type	Number of remote files	Description of Contents
Load Module	One or more per application	Application source code for a single load module. One remote file for each load module, proxy, operations library, or z/OS library.
Database	One per database	DDL for an application database. Some environments allow more than one database per application. In this instance, one DDL remote file exists for each database.
RI Triggers	One per model	Referential Integrity trigger modules for an application. These routines implement referential integrity for the entire application.

A complete CA Gen generated application for remote installation contains a database remote file, a Referential Integrity module remote file, and one or more load module remote files.

The process discussed in this chapter generates the load module and Referential Integrity remote files. The remote file that becomes the application database is generated separately.

More information:

Generating DDL (see page 25)
Creating Data Definition Language (see page 25)
Prerequisites (see page 27)
Verify Consistency of Model (see page 28)
Transform the Data Model (see page 28)
Retransform the Data Model (see page 28)
Required Activities Outside of CA Gen (see page 28)
How to Create a Database (see page 29)
Set Generation Configuration Parameters (see page 29)
Generate DDL (see page 30)
Include Drop Statements in DDL (see page 30)
Install DDL (see page 30)
Apply to All Subordinates (see page 30)
Set Generation Paths (see page 31)
Overriding DDL Paths (see page 31)
Select Components for Generation (see page 31)
Filtering (see page 32)
Select a Generation Action (see page 33)
Generating All Database Components (see page 34)
Generating Flagged Components (see page 34)
Regenerating Database Components (see page 35)
Set DDL Generation Defaults (see page 35)
Override Paths (see page 35)
Qualify Tables and Indexes with Owner ID (see page 35)
Create Storage Groups (DB2 only) (see page 36)
Create RI Alter Primary/Foreign Keys and Triggers (see page 36)
Verify Completion (see page 36)
Generate a DDL (see page 36)
Results of DDL Generation (see page 38)
Source Path (see page 38)
Remote Installation Path (see page 39)
Temporary Files Created by Generation (see page 39)
Remote Install Path and User ID (see page 39)

Prerequisites for Generating Remote Files

Before you can generate remote files, you must have the CSE Construction Client and the appropriate construction options for remote generation and installation installed on your workstation.

In addition, you must complete the following activities:

- Complete appropriate diagrams
- Transform the Data Model (DM)

- Check in any changes to the CSE
- Have the proper administrative authority or have Generate and Update authority for a model
- Specify environment parameters and paths for the business system(s) and RI trigger modules
- Perform packaging
- Select Generation options

Complete Diagrams

The data files of the models on your workstation are the main source of information for generation. Generation uses information from the diagrams created during Analysis and Design.

Before you can generate source code successfully, the diagrams for the part of the model you are generating must be complete and pass Consistency Check with no errors. These diagrams include:

- Entity Relationship Diagram
- Dialog Flow
- Action Diagram
- Technical Design (both Data Structure List and Data Store List)
- Screen Design (for character-based applications)
- Window Design functionality in the Navigation Diagram for:

  GUI applications

  Client/Server applications

  Web Client applications
- Components (CBD) for Operations Libraries
- z/OS Libraries

The CA Gen Workstation Toolset allows you to check consistency using options in the Analysis and Design Toolsets. Consistency Check Reports are also available on CSE.

Transform Entity Relationship Diagram

Before you can generate source code or create a database, you must first transform the Entity Relationship Diagram into a TD. You must also complete any necessary changes to the Data Structure List and Data Store List as part of completing database design using the Toolset.

Transformation does not create the database remote file. This is done during the database generation and installation process on the Construction Client.

Specify Environment Parameters and Paths

Before you generate an application, you must specify its target environment and paths. You can specify the environment parameters and paths by selecting Code, then Environment. You can select from a number of operating systems and DBMSs. You will receive error messages if you select incorrect combinations of environment parameters. The specific selections available on your workstation depend on the generation options purchased.

For all applications, the paths belong to a business system. For online, window, and batch applications, the environment parameters belong to a business system. For cooperative applications, you must set the environment for the window manager (client) and server manager (server) for *each* load module. For operations library and z/OS library, you must set the environment parameters for each one. Different business systems within the same model can have different environment parameters. RI triggers are model level objects whose environment and paths are stored locally (on the client machine). The environment parameters you set in the Construction Environment window are stored on the model and are saved as properties of the business system. You can override environment information on the Generation Defaults window.

Note: Both RI trigger modules and DDL are model-level objects not associated with any one business system. They share environment settings. If you change the environment for one, you change it for the other as well.

If you select the override options, you are changing the environment parameters for the DDL and RI triggers.

Package Load Modules

Before you generate an application, you must package the load modules.

Note: For more information, see the chapter Packaging.

More information:

Packaging (see page 41)
Packaging Process (see page 41)
Types of Packaging (see page 44)
Prerequisites (see page 44)
How to Package (see page 44)
Object Relationships (see page 45)
Create Load Module or Batch Job (see page 46)
Creating a Load Module (see page 46)
Add Objects for Packaging (see page 46)
Creating a Batch Job (see page 47)
Add Components for Batch Packaging (see page 47)
Modify Existing Load Module or Batch Job (see page 48)
Modifying an Existing Load Module (see page 48)
Add a Procedure Step to an Existing Load Module (see page 48)
Delete a Procedure Step in an Existing Load Module (see page 49)
Modifying an Existing Batch Job (see page 49)
Add a Procedure Step to an Existing Batch Job (see page 50)
Delete a Batch Job or Batch Job Step (see page 50)
Detail Component Objects (see page 51)
Member Names (see page 51)
Source, Screen and Load Module Naming Rules (see page 51)
MFS Source Naming Rules (see page 52)
Transaction Codes (see page 52)
Clear Screen Transaction Codes (see page 52)
Dialog Flow Transaction Codes (see page 53)
Transaction Code Naming Rules (see page 55)
Complete Option (see page 55)
Default File (Member) Names (see page 55)
Detailing Load Modules (see page 56)
Detailing Batch Jobs (see page 57)
Load Module Size (see page 58)
z/OS Dynamic Linking (see page 58)
Features of Dynamic Link (see page 60)
Considerations for Using the Dynamic Linking Option (see page 61)
Online Packaging (see page 62)
Selecting the Business System (see page 62)
Defining Load Modules (see page 62)
Name the Load Module (see page 63)
Add Procedure Steps (see page 63)
Defining Source Names (see page 64)
Assigning Transaction Codes (see page 65)
Batch Packaging (see page 65)
Selecting the Business System (see page 66)
Defining Batch Jobs (see page 66)
Name the Batch Job (see page 66)
Add Procedure (see page 66)
Defining Job Step Properties (see page 67)

Select Generation Options

Review the generation options before generating or installing an application. Some of the generation options can override the environment parameters.

The generation options are discussed in the following sections.

How to Create a Remote File

Use the following steps to create the remote files for a remotely installed application.

Set Date and Time

Before generating and installing an application, verify that the time and date are correct on the server. The CSE Construction Client uses them when creating files. The code generator also creates a date and time field in the source code. This allows you to scan the code to verify the date and time that the source was generated.

Specify Generation Options

The options you set on the environment parameters windows (in the Environment and Package tools) can be checked in to the encyclopedia and are saved as properties of the business system. Generation Defaults are not tied to a specific business system or model. They are tied to your workstation and are retained from session to session by the CSE. For this reason, when you check in work to the Encyclopedia and then check out, whether from the same model or another, the defaults last used are still in effect. It is, therefore, important that you check these options before generating.

Define an operating system, DBMS type, language, and TP monitor. All settings are verified against a list of valid combinations before they are saved. To access the generation defaults, select a model and then select Code, Generate, <generation type>. Select a business system and then select Defaults.

The following check boxes are available:

- Override Business System Target Environment
- Override Paths
- Delete generated source after install
- Generate Foreign Action Blocks
- Process modules marked for Compatibility

Override Business System Target Environment

Selecting this check box means the values in the drop-down lists on this window for operating system, DBMS type, source language, and TP monitor are used instead of the values stored as environment parameters for any selected business system.

Note: When the override check box is selected, the business system environment parameters in the Environment option are overridden for character-based (block mode) and GUI applications. The client and server environment parameters in the Cooperative packaging option are overridden for client/server applications.

Override Paths

You can override paths for the business system by selecting Defaults from the Application Generation window. Select Override Paths and then select Paths. Type the override path information and click OK. The override path information is stored locally (on the Client).

Delete Generated Source after Install

You can delete all generated source code immediately after installation by selecting this check box.

Downstream Effects: Use of the Delete Source after Install option may cause the installation of load modules to fail when common action blocks are used.

This option causes the source code to be deleted from the Construction Server when it is installed into a remote file. In this case, the source code for a common action block will be packaged into the first remote file to be created and then deleted. All subsequent remote files will only contain references to the common action block, not the actual code.

Another way to make the common action blocks available is to split all the remote files on the target platform before installing any of them.

This same problem can occur when using the Build Tool (BT). The BT also has an option that causes the deletion of source code during an installation. If the common action block source code is deleted, then subsequent installs which use that action block may fail.

Generate Foreign Action Blocks

Enabling this option causes the selected foreign action block (an action block owned by a different business system than the parent load module's business system) to be generated using the configuration and paths of the parent load module's business system, unless overridden. If the option is not enabled, the selected foreign action block will not be generated.

During install, if the generate foreign action blocks option is selected, the source code generated for the foreign action block will be copied from the path where the parent load module's business system source code was generated (unless the path name has been overridden) for inclusion in the .rmt file. If the option was not enabled, the source code for the foreign action block will be copied from the path where the owning business system's source code was generated (unless the path was overridden).

Process Modules Marked for Compatibility

This option applies to MVS applications, which have components that either explicitly or by default are configured with their Dynamically Link packaging property set to Compatibility. Compatibility is intended to be used by applications that make use of one or more dynamically called modules that for some reason must reside in a non-DLL module.

Typically the components marked for Compatibility include:

- Routines that were built for dynamic linking using a release of CA Gen before Release 7
- External Action Blocks (EABs) that contain a dynamic program call to a non-Gen routine that resides in a non-DLL module

- Routines that are the target of a dynamic program call that is initiated from a routine that resides in a non-DLL module

Use the *Process modules marked for Compatibility* option to generate and install modules marked for Compatibility. If the option is not set, the processing of components marked for Compatibility will be bypassed. Selecting the Process modules marked for Compatibility option causes all Action Blocks statically called by a module marked for Compatibility to be compiled twice-once using the compiler option NODLL and again using the compiler option DLL.

The result of the NODLL compile is linked into the Compatibility module. The result of the DLL compile is provided so that it can be linked into any DLL applications that statically call these action blocks. If the action blocks are External Action Blocks, they must be compiled using the appropriate compiler options and made available to be included in the install of the final load module.

Selecting the Process modules marked for Compatibility option when generating and installing RI triggers causes the RI trigger modules to be generated and precompiled once but compiled twice in a similar way to Action Blocks.

Separate libraries are provided in the target environment to hold the separate NCAL modules resulting from the two compile steps.

Select a Generation Action

You have two ways to generate code using the CSE Construction Client. You can choose to have the Construction Client automatically create all the components for you or you can choose the components to create manually.

- Generate the model (without selecting components)
- Generate flagged (selected) components

The first time you generate remote files for the application, select Generate, then Model to generate and to install all code. Be sure to set the install option to Yes. This action provides the quickest way to accomplish the installation.

Generating the Model

Do not select component objects when generating with this option.The procedure generates all component objects in the model for the type of packaging specified when selecting Code Generation from the client.

You have two further options at this point:

- The Trace option allows you to select whether to include trace code in the generated source code.

- The Install option allows you to select whether to generate remote files.

Note: It is the installation process that actually creates the remote files. Selecting Model from the Generate menu with the Install option set to Yes creates one remote file for all Referential Integrity trigger modules, and one remote file for each load module in the application.

Generating Flagged Components

To generate and install code selectively, you must select the objects individually and then set the generation flags. You select the objects by highlighting one or more components from the object occurrence lists and selecting Add or Expand. This adds the objects to a list of component objects you want to generate. You can set the generation flags in the object occurrence lists or in the Application Generation window.

The code generation flags are:

- Generate source code with trace enabled (T)
- Generate code (C)
- Generate screen (S)
- Install code (I)
- Apply to all subordinates (A)

As an example, business systems contain load modules, which contain procedure steps, which contain action blocks. Select a load module, its procedure step and its action blocks, and Add them. From the Application Generation window, set the code (C), install (I), and apply to all subordinates (A) flags. When you select Flagged from the Generate menu, this tells CSE Construction to generate code for the flagged objects and all of their subordinate objects, and to install the code into a remote file.

Note: If you select the Apply to All Subordinates option for an object and then manually add one of its subcomponents to the window with different generation options, the subcomponent will be generated using the generation options set for it.

For more information about the selection process, see the chapter Construction Concepts (see page 13).

Note: Referential Integrity (RI) trigger modules are necessary for the application to work properly. Be sure to generate the RI triggers when you generate remote files.

Generate Source Code with Trace

Trace support tells the Construction Client to generate *additional* source code, to allow you to *step through* the execution of the action diagrams for the application after you generate and install the source code. If you intend to test a load module or its components with the Trace option, you *must* generate Trace support at the time you generate the load module.

Enabling the Trace support flag first automatically enables the Code generation flag. (You cannot have trace code without source code.) Selecting the Code flag once it is set to Y turns trace and code off for the same reason. Select the Code flag first if you do not want Trace support enabled.

If you do not choose this option, you can still test the application, but you cannot view the underlying logic. The default choice is blank (No).

Trace code does not work in all environments, such as IMS or CICS. Trace code is unrecognizable to TP monitors native to operating systems. See the appropriate Build Tool documentation for instructions on invoking an application that contains Trace code.

Downstream Effects: The special code included to allow Trace significantly increases the size of each remote file. If space is a potential problem on either the code generation platform or the target system, consider adding trace selectively rather than globally.

Generate Code

Use this option to generate code. If you do not select this option for each component, the component is not generated.

Source names are generated into the code as the PROGRAM-ID in COBOL and the function name in C.

Generate Screen

This option only appears at the procedure step level, since they are the only components which have screens associated. If you do not select this option, the screens are not generated and the application cannot be tested after installation.

Install

Use this option to create the remote files. If you do not select this option for each component, only source code is generated and no remote file is created. (You cannot install and execute the application on the target platform without the remote file.)

Apply to All Subordinates

Use this option to generate code for the selected component and all of its subordinate objects using the current generation options selected for the component. For example, if you want to generate all subordinate components of a business system with Trace, and then install all of its load modules, you can do so with the selection of the Apply to All Subordinates option at the Business System level. If you do not select this option, you must manually select each of the desired subordinate objects desired for generation.

Note: If you select the Apply to All Subordinates option for an object and then manually add one of its subcomponents to the window with different generation options, the subcomponent will be generated using the generation options set for it.

Verify Completion

You can verify completion of the generation process by the messages that appear in the pop-ups and message window. If an error occurs, generation or installation processing stops and messages are written to the generation status file.

Use this file to determine which components generated successfully and which did not. You can also use this file to determine the final outcome of the processing.

Code Generation Procedure

This procedure explains one method for selecting components for source code generation. The important concept is the expansion to the next level. During the selection process, you can enable the generation options (Trace support, Code, Screen, Install, or Apply to All Subordinates) in these ways:

- From the object occurrence windows
- From the Application Generation window

You may find it useful to enable the desired code generation options as you go. If you change the generation options for an item and add them again, the last option chosen is the one used. There exists a maximum number of objects (maximum cardinality) that can be displayed in the list.

Note: For more information on generation options and cardinality issues, see Select Generation Options.

Any object added after this limit is reached is not generated. If you have more objects than can be displayed, you may have to generate in two sessions.

Follow these steps:

1. Start the Construction Client.
2. Open a model by selecting Model.
3. Select Actions, then Open in the Construction Model Selection window.
4. Highlight a model from the list and select Open.
5. Select Code, then Generate and select a generation type.

   This opens the Application Generation window which will be empty initially. After making a selection, this window contains the selected objects.
6. Select Edit, then Select New Objects.

   You can select one or more objects at a time. You automatically flow to the next object type when you select more than one.
7. Select RI Triggers from the Generation Object Type List and then select List.

   This action displays the Generation Object Occurrence List which contains the RI Triggers.
8. Select the RI Triggers and select Add, then select Continue.

   This returns you to the Generation Object Type List for further selections. The order of expansion from the Business system down is Business System, then Load Module, then Procedure Step, then Action Block.
9. Select Business System and then select List.
10. Select a business system from the list and select Expand.

    This displays the Generation Expanded Object Occurrence List from which you add and expand other objects.

    If multiple dialects (French, German, and so on) are used by windows in procedure steps packaged in any Window Load Module, an entry for each dialect is displayed.

    If you want to generate all components in a business system automatically rather than manually, mark the Apply to All Subordinates generation option.

11. Select one or more load modules and select Add. Select a load module and select Expand.

    This adds the load modules to the list of objects for generation and expands to the next level of objects for the selected load module.

    If you want to add load modules containing non-default dialects, you may select from the list.

    If you want to generate all components in a load module automatically rather than manually, mark the Apply to All Subordinates generation option.

    You can use Select All if you want to select all objects in the display.

12. Select one or more procedure steps and select Add. Select a procedure step and select Expand.

    This adds the procedure steps to the list of objects for generation and expands to the next level of objects for the selected procedure step.

    If you want to generate all components in a procedure step automatically rather than manually, mark the Apply to All Subordinates generation option.

13. Select one or more action blocks and select Add.

    This adds the action blocks to the list of objects for generation. There is no expansion beyond this level.

14. Select Continue until you return to the list of load modules. Select the next load module if you are generating more than one in this business system, and did not select the Apply to All Subordinates generation option. Repeat steps 10 through 14 to add that load module and all of its desired subordinate objects to the list.

15. Select Continue until you return to the Application Generation window.

16. Select the appropriate generation flags for all desired components and then select Generate, then Flagged.

    You must select an object before you can set the flags for it. Choose Edit, then Select All to select all objects in this window or select each object individually.

You can use the steps in this procedure to generate the entire application or to perform selective regeneration on changed components. When generating the entire model, it is better to use Model from the Generate menu. This option only generates applications of the type you chose when you entered the generation dialog. If you choose Generate, then Model for a Cooperative application, the Construction Client generates all Cooperative applications in the model. It does not generate online, window, or batch applications in the model.

Results of Generating Remote Files

Listed in the following tables are subdirectories and files generated by the Construction Client for character-based (block mode) applications and GUI applications (including GUI standalone and GUI clients) during application generation.

Subdirectories and Generated Files

During generation and installation, the Construction Client creates directories subordinate to the directories you named in the Construction Paths window. (The path directories you entered included source path, install control path, and remote install path for source and RI triggers.) This allows proper identification of the source code by operating system and source code language. The user ID is the name used to log on to the Construction Client.

When load modules containing non-default dialects are selected for generation, a specific directory subordinate to the language subdirectory is created for each dialect source code. The default dialect source code is stored in the language subdirectory.

These path definitions appear in the table that follows:

- Source = source path + operating system + language + dialect(s)
- Installation Control = install control path + operating system + language + dialect(s)
- Remote Installation = remote install path + operating system + language + dialect(s)

If you are using MFS devices, the following separate path entry exists:

- MFS Source = MFS source + operating system + language + dialect(s)

Various files listed in the following table are created during generation and installation:

Path	File Name	Description
Source	*.c *.cbl	Generated COBOL or C source code without embedded SQL. The extension indicates the language used to generate the source code. The file names can be: ■ Load Module name ■ Member source name (procedure steps, action blocks) ■ Dialog Manager ■ Batch Manager ■ Window Manager name (Cooperative and GUI) ■ Batch Job Step name
MFS Source	*.cbl	IMS-specific source for MID, MOD, and FMT members. This file is only present if you generate the screen.
Source	*.sqb	Generated COBOL source code with embedded SQL.
Source	*.sqc	Generated C source code with embedded SQL.
Source	*.err	Messages resulting from a failed generation.
Source	*.gml	General Markup Language source file used with online help created for GUI applications.
Source	*.h	Header file created for GUI applications.
Source	*.java	Java source file
Source	*.html	Hypertext markup language file
Source	*.css	Cascade style sheet file
Source	*.jsp	Java Server Page
Source	*.js	Java Script
Source	*.rme	Read Me file
Source	*.lst	List file
Source	*.xml	Extend Markup Language file

Path	File Name	Description
Installation Control	*.icm	Install control information. The install control file is a set of instructions on how to install all procedure steps, screens, and action blocks for the load module. The install control file is in Standard Generalized Markup Language (SGML) format.
Remote Installation	*.rc	Resource control file created for GUI applications.
Remote Installation	*.rmt cascade.rmt	A remote file which contains a concatenation of the .icm and source files. A separate .rmt file exists for each load module. The RI triggers are placed in a file called cascade.rmt.
%%%	.asp	
	.bas	
	.cpp	
	.def	
	.idl	
	Rgs	

If you generate a complete application, the result is one remote file for the RI triggers, named cascade.rmt, and one remote file for each load module. A complete application also has a database remote file, which is created separately.

Note: For more information, see the chapter "Generating DDL (see page 25)."

The files in the following table are temporary and may or may not be present after generation:

Path	File Name	Description
Source	*.ctl	Information used by the Construction Client when creating a remote file.
	genstat.ctl	Generation status log file that contains messages about the last action initiated.
remote install path + user ID	rfg.txt rfgcdr.txt	Status messages issued during the generation of remote files.

Chapter 7: Load Module Structure

This appendix provides information about load module structure and its different components.

This section contains the following topics:

Understanding Load Module Structures

The CA Gen software does not require an understanding of load module structure to generate and install application systems. However, an understanding of the structure of load modules is an important aid in performance tuning and in the investigation of runtime error messages.

Creating a load module that is ready for execution in a specific environment has the following stages:

- Creation of procedure steps and action blocks using the Action Diagramming Tool
- Creation of load module packaging definitions using Packaging
- Creation of load module source code using Generation
- Creation of load module object code using one of the remote Build Tools

In the first stage, you implement the business rules of your organization with action statements.

In the second stage, you package the procedure steps and action blocks into load modules or batch jobs. This is influenced by the characteristics of the target operating system. Some systems are more efficient with fewer but larger load modules while other environments are more efficient with more but smaller load modules.

In the third stage, you generate source code which integrates:

- Procedure steps and action blocks
- Any CA Gen-supplied special functions used by the action blocks
- External action blocks
- Provisions for interfacing with a teleprocessing monitor.

In the fourth stage, TP monitor-specific constructs are included within the load module when it is installed. CA Gen-supplied user exits are linked to by the load module during execution.

An analogy can be made between the top-down approach to programming and the usage of procedure steps and action blocks. In a Top-Down Structured program, all code is contained within paragraphs. Control does not simply flow from the top of the program to the bottom, and there are no explicit GO TO commands. One main paragraph placed at the beginning of the procedural section controls the flow within the program by calling paragraphs. Each paragraph returns to the main paragraph upon completion. A procedure step performs the function of the main control paragraph by issuing USE statements which invoke specific action blocks.

The term load module means the same thing in the IBM host environment as it does in CA Gen. Other operating systems have their own terms for executable code.

Load Module Components

In general, a CA Gen generated load module contains the following components:

- A transaction manager (Dialog and Screen Manager, Batch Manager, Window Manager, or Server Manager)
- One or more procedure steps from the same business system and the action blocks used by the procedure steps
- Referential Integrity trigger modules
- CA Gen provided special functions (if used in action blocks)
- CA Gen provided runtime routines
- User-provided external action blocks, optionally

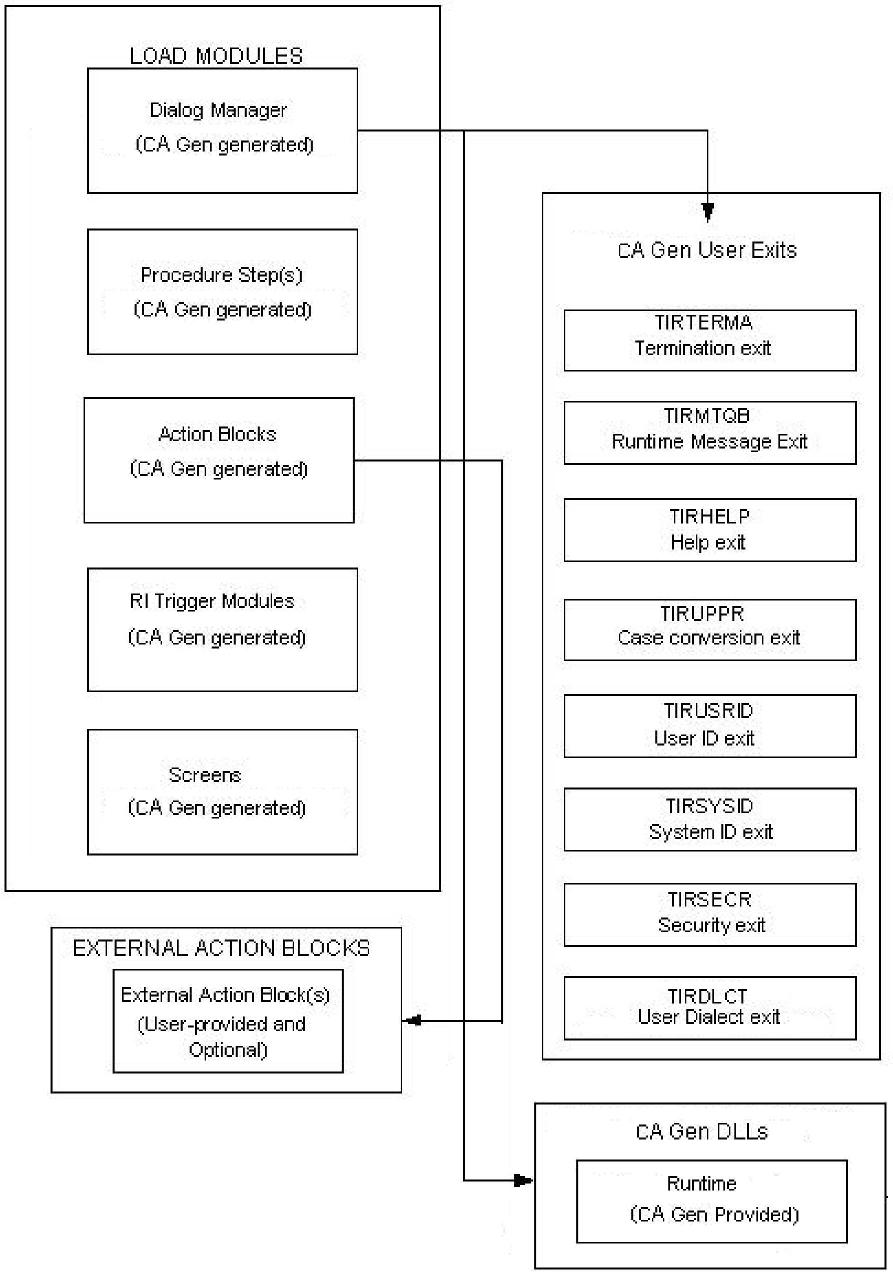
External action blocks are not generated by CA Gen.

Some of the CA Gen provided runtime routines can be customized for your environment.

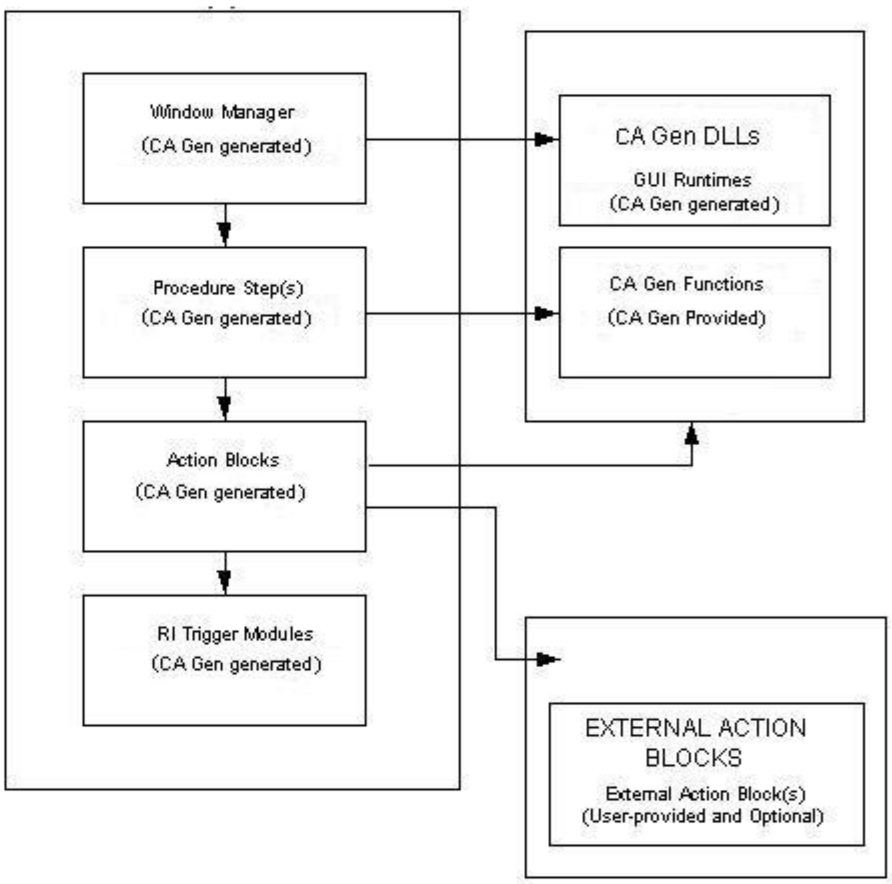
Note: For more information, see the Windows *Implementation Toolset User Guide, the z/OS Implementation Toolset User Guide,* or the *Host Encyclopedia Construction User Guide*.

The structure of online load modules for character-based (block mode) applications and GUI applications (including GUI standalone and GUI client) in the workstation environment, and how the components work together, are shown in the following illustrations.

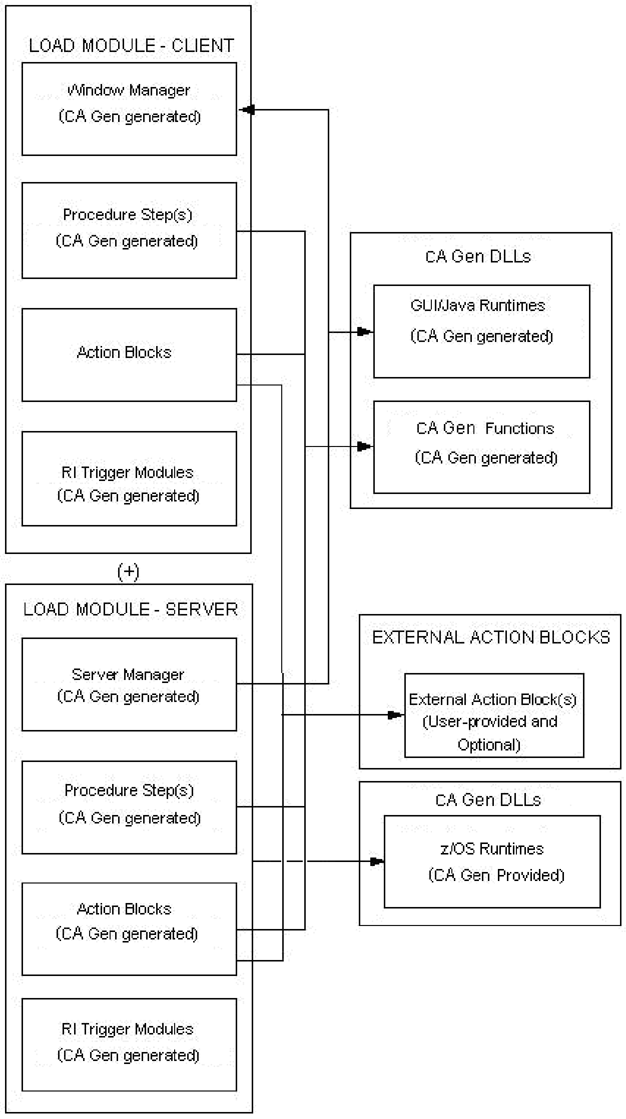
Load Module Structure for Online Applications



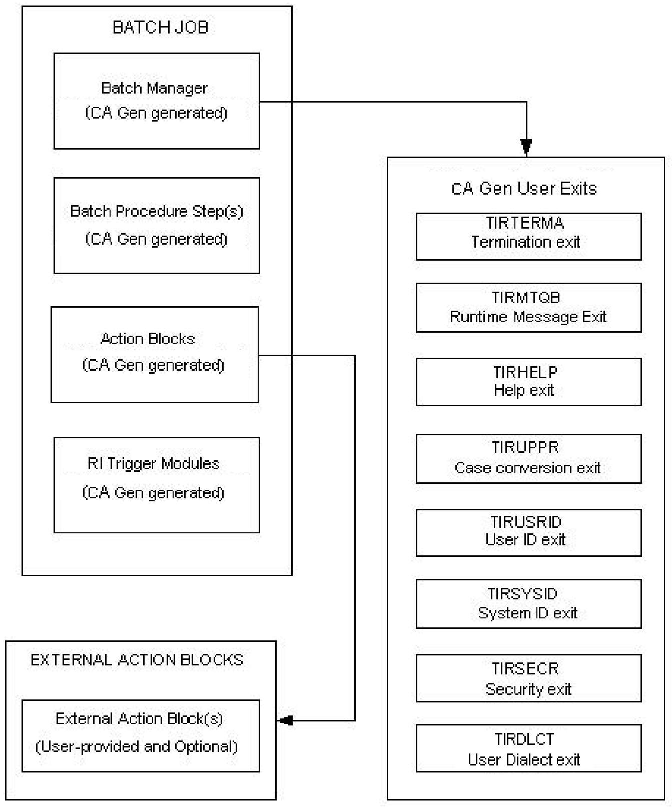
Load Module Structure for GUI Applications



Load Module Structure for Cooperative Applications


LOAD MODULE - CLIENT
Window Manager (CA Gen generated)
Procedure Step(s) (CA Gen generated)
Action Blocks
RI Trigger Modules (CA Gen generated)
(+)
LOAD MODULE - SERVER
Server Manager (CA Gen generated)
Procedure Step(s) (CA Gen generated)
Action Blocks (CA Gen generated)
RI Trigger Modules (CA Gen generated)
CA Gen DLLs
GUI/Java Runtimes (CA Gen generated)
CA Gen Functions (CA Gen generated)
EXTERNAL ACTION BLOCKS
External Action Block(s) (User-provided and Optional)
CA Gen DLLs
z/OS Runtimes (CA Gen Provided)

Load Module Structure for Batch Applications


BATCH JOB
Batch Manager (CA Gen generated)
Batch Procedure Step(s) (CA Gen generated)
Action Blocks (CA Gen generated)
RI Trigger Modules (CA Gen generated)
EXTERNAL ACTION BLOCKS
External Action Block(s) (User-provided and Optional)
CA Gen User Exits
TIRTERMA Termination exit
TIRMTQB Runtime Message Exit
TIRHELP Help exit
TIRUPPR Case conversion exit
TIRUSRID User ID exit
TIRSYSID System ID exit
TIRSECR Security exit
TIRDLCT User Dialect exit

Transaction Manager

The Transaction Manager, which is generated during installation of each load module, acts as an executive to control the flow between procedure steps. The following lists shows the types of transaction managers:

- Dialog Managers and Screen Managers (character-based application)
- Batch Managers (batch application)
- Window Managers (GUI and Web Client applications and the client component of the client/server application)
- Server Managers (server component of a client/server application)

TIRMAIN is a CA Gen-supplied runtime module that is the entry point of all load modules. TIRMAIN calls the appropriate Transaction Manager, which then takes over the control of the dialog flow (links and transfers) between procedure steps.

The Transaction Manager maintains execution context using a mechanism called the Profile Manager. To accomplish this, the Transaction Manager uses information from procedure step import and export views, screen definitions, dialog flow, and load module packaging.

For character-based (block mode) applications, the Transaction Manager contains cross-reference tables of procedure steps, exit states, and transaction codes for the procedure steps packaged in the load module. The procedure steps to which the load module procedure steps link or transfer are also cross-referenced. The Transaction Manager uses all of this information to select the target procedure step within its load module. If the target procedure step is in another load module, the information is used to select the transaction code that invokes the load module containing the target procedure step.

When a load module is executed, the Transaction Manager determines the procedure step to execute. It then populates the import view of the procedure step using screen input and data passed from another procedure step or data taken from the Profile Manager. This allows each procedure step to reference data in its input view without having to perform the processing required to gather it.

Note: Window Managers do not use the names TIRMAIN and TIRMSG, but the transaction concepts for GUI applications are the same.

Dialog Manager

The Transaction Manager is named as the Dialog Manager for online (character-based) applications.

For online applications with screens, a Screen Manager is generated for every procedure step. The Screen Manager is called by the Dialog Manager to perform functions such as input parsing, function key translation, input/output view mapping, scrolling and Help/Prompt command processing. The Screen Manager calls runtime routines to edit input and output, and write output buffers.

Batch Manager

The Batch Manager is similar in function to the Dialog Manager. The Batch Manager controls the dialog flow between procedure steps. Only forward transfers are allowed in batch processes (links are not allowed).

Batch applications use three special files. They are referenced by the DDNAMES (data definition names) TIRIOVF, TIRMSGF, and TIRERRF. CA Gen accesses these special files through the CA Gen-provided runtime routine.

- TIRIOVF is the data set used to pass data (views and Batch Manager control information) from one procedure step to another on a dialog flow.
- TIRMSGF is the Batch Manager Message File. It provides a trace of procedure steps executed, exit states set, exit state messages issued, and dialog flows taken.

- TIRERRF is the Batch Manager Error Message File. In the event of a failure, error messages and diagnostic information are written to this file. The messages and diagnostic information will help you determine the cause of the failure.

Window Manager

The Window Manager is similar in function to the Dialog Manager. The Window Manager maintains an execution profile similar to the Dialog Manager, with two notable differences:

- The Profile Manager's data is maintained in memory so there is no profile database.
- GUI applications are not restartable since there is no stored profile to retrieve. A GUI task can be interrupted by simply minimizing the current window and performing another task. You can resume by restoring the minimized window and continuing the dialog.

The Window Manager can be created only in C and Java languages.

Server Manager

The Server Manager is similar in function to the Dialog Manager. The Dialog Manager sends a screen runtime message (TIRFAIL) when it encounters an error. The Server Manager formats and sends the error message into the data view buffer. This error is then passed back to the client application and displayed in a dialog box.

The Server Manager can be created in the COBOL and C languages depending on the target platform for the server application.

Profile Manager

The Profile Manager is a CA Gen provided runtime routine that saves the export views of procedure steps. It is used to save information between procedure step executions, to support implicit scrolling, hidden fields, links to other procedure steps, Help/Prompt requests and application restarts. These features are supported by a push-down, pop-up run-time stack. The routine maintains one stack per business system per user. If you are working in three business systems, you have three stacks. Different implementations of the stack are available and include SQL databases, temporary storage queues (TSQ) for CICS, sequential files, and others depending upon the environment used.

When you *transfer* to a procedure step, the stack is cleared and the export view of the procedure step that you are transferring to is the only entry on the stack. When you *link* to a procedure step, existing entries remain on the stack and the export view of the procedure step that you are linking to is placed on top of the stack.

The Profile Manager imposes some overhead on the application. You can reduce this overhead by requesting that the application be non-restartable. Transfers and screen displays will not cause the Profile Manager to be invoked unless implicit scrolling, hidden fields, links to other procedure steps or Help/Prompt requests are used.

Note: For more information about specifying your application as non-restartable, see the *CA Gen Encyclopedia Client Help*.

If your application is restartable, the capabilities of the Profile Manager let you interrupt an application and return to it without losing data. You can interrupt a business system to use another business system or to perform activities unrelated to the applications.

Restart and Reset (for Character-Based Applications Only)

If you wish to return to a business system after an interruption, you have two options. You can resume the dialog where it was interrupted by using your Profile database stack for the business system (RESTART). Or, you can start the dialog at the beginning by clearing the existing stack (RESET).

To resume the dialog, clear the screen and enter any transaction code for the business system followed by the word RESTART. The format is:

```
<trancode> RESTART
```

If the clear screen default command specified for the target environment is RESTART, simply enter the transaction code.

The Profile Manager displays the screen that was active, and all data that was present, when the dialog was interrupted. (The screen and data are taken from the top entry of the stack.) The top entry is always the last screen active for the business system, regardless of the transaction code entered. At this point, you may continue the dialog as if it had never been interrupted.

If you prefer to start the original dialog from the beginning, enter the clear screen transaction code for the procedure step you wish to execute followed by the word RESET. The format is:

```
<trancode> RESET
```

If the clear screen default command specified for the target environment is RESET, simply enter the transaction code. This clears the stack and begins the dialog.

Procedure Steps

A business system consists of procedures. During load module packaging you select the procedure steps to be packaged in each load module from a list of all the procedure steps defined for the business system.

Character-based (block mode) applications, GUI and Web Client applications, and client/server applications have no restrictions on the number of procedure steps that you can package in each load module.

Batch applications require that you package only one procedure step for a single load module.

Every procedure has a first procedure step. In one-step procedures, the first procedure step is the only procedure step. It is shown on the Dialog Flow Diagram as a procedure line. In multiple-step procedures, the first procedure step is the procedure step that appears immediately beneath the procedure line on the Dialog Flow Diagram.

Action Blocks

An executable load module contains every action block used by the procedure steps or by other action blocks in the load module. A single copy of the action block is packaged in the load module, regardless of how many procedure steps within the load module use it.

CSE Construction supports the use of common action blocks. This means that any action block defined in a model can be used by any other action block or diagram in that model. This applies even if the action block is not scoped within a business system that USEs the action block, provided that both action blocks are generated for the same target environment.

The following table shows the effect of generating a common action block with different override paths enabled. In the table, the same common action block was selected by expanding different procedure steps from two separate business systems.

Override Target Environment	Override Paths	Effects
Yes	No	Common action block is displayed twice on the list. The common action block is generated twice, once for each business system.
No	Yes	Common action block is displayed twice on the list. The common action block is generated twice, once for each business system, into the override path's location.

Override Target Environment	Override Paths	Effects
Yes	Yes	Common action block is displayed twice on the list. The common action block is generated once into the override paths location, using the override target environment.
No	No	Common action block is displayed twice on the list. The common action block is generated twice, once for each business system into its separate paths.

When the option Process modules marked for Compatibility is selected, the code is still generated as per this table but the same code is compiled twice, once with the compiler option NODLL and again with the compiler option DLL.

If a common action block is not generated within its associated business system, you receive an error message from the Construction Client noting unresolved external references.

Referential Integrity Trigger Modules

When two entity types participate in a relationship, each is said to reference the other. This reference is implemented by storing the primary key for one entity type as a field in the record for the other entity type. Referential Integrity (RI) rules verify that a reference exists when a relationship pairing is required, and that the reference is removed when the record to which it points is removed. Depending on rules in the Data Model, this may mean setting the reference (foreign key) to null or deleting a record that is left without a required reference.

RI triggers are generated modules that enforce referencing rules defined by the Data Model. RI triggers are generated and maintained for each model separately, and they are based on the complete Data Model.

When you display generation component objects in the Code option, the RI triggers appear first in the list of objects. You must generate the RI triggers *before* installing any source language code. Failure to generate the RI trigger modules first can result in unresolved references when linking load modules.

When the option Process modules marked for Compatibility is selected, the RI triggers are generated only once but compiled twice, once with compiler option NODLL and again with the compiler option DLL.

Generation of RI triggers results in the following types of modules:

- ENTITY trigger modules
- RELATIONSHIP trigger modules

ENTITY Trigger Modules

ENTITY trigger modules execute the deletion of a row, as well as enforcing restrict logic and acting as a manager for processing associated rows. One ENTITY trigger module exists for every entity type.

An ENTITY trigger module is called by an action block or procedure step to manage the delete action, or by a RELATIONSHIP trigger module when the RI rules imply that delete.

RELATIONSHIP Trigger Modules

RELATIONSHIP trigger modules execute the update (set null) of foreign keys to implement a DISASSOCIATE or TRANSFER in an action diagram. They also identify additional processing required, based on the DMD rules specified and the type of action requested (DELETE, DISASSOCIATE or TRANSFER). Further deletions are processed by calling the appropriate ENTITY trigger module. The RELATIONSHIP trigger module also manages the delete or disassociate of link records required for many-to-many relationships. One RELATIONSHIP trigger module exists for every relationship.

Chapter 8: Installing and Using the CA Gen Sample Model

This appendix provides information about installing and using the CA Gen Sample Model.

This section contains the following topics:

Using the Sample Model

The CA Gen Client Server Encyclopedia (CSE) software includes a sample model with character-based (block mode), cooperative, and GUI business systems. You can use the model to:

- Test the installation and configuration of the Client Server Encyclopedia (CSE) Construction Server.
- Ensure that the CSE installation and configuration is correct.

Your installation is correct if you can:

- Generate and install data definition language (DDL).
- Generate and install code.
- Use the Application Execution Facility (AEF) with the character-based (block mode) business system.

Character-Based Generation, Installation, and Testing

Use the following procedures to install and test character-based (block mode) generation.

Installing and Testing Character-Based Generation

Follow these steps:

1. Install the server software and upload the CA Gen Sample Model into your CSE.

2. Start the Construction Client. Open and log on to the encyclopedia.
   a. Select Encyclopedia, then Open from the Encyclopedia Construction Client window.
   b. Select the correct Encyclopedia from the Encyclopedia List window and press Logon.

   Note: If only one encyclopedia exists, no list will display but the encyclopedia will open when you select Open from the pull-down menu.
3. Select and open the CA Gen Sample Model.
   a. Select Model from the Encyclopedia Construction Client window.
   b. Select Actions, then Open from the Construction Model Selection window.
   c. Select the CA Gen Sample Model from the Model List window and select Open.

Perform Online Packaging of the Model

Follow these steps:

1. Select Code, then Package, then Online from the Construction Model Selection window.
2. Select the business system CORPORATE_MANAGEMENT.
3. View “current” Online Packaging for CORPORATE_MANAGEMENT:
   a. Select View, then Expand from the Online Packaging window.
   b. Verify that several Online Load Modules are listed on the List Load Modules window.
4. Remove “current” Online Packaging for CORPORATE_MANAGEMENT:
   a. Select one Online Load Module.
   b. Select Edit, then Delete from the List Load Modules window.
   c. You are prompted to confirm the deletion of the load module.
   d. Perform steps a through c for each load module.
   e. Select Edit, then Exit to close the List Load Modules window.
5. Select the business system CORPORATE_MANAGEMENT from the list of business systems on the Online Packaging window.
6. Add Online Packaging for CORPORATE_MANAGEMENT:
   a. Select Edit, then Add Load Module from the Online Packaging window.
   b. Enter a load module name.
   c. Select all procedure steps listed on the Add Load Modules window.
   d. Click OK to create the load module.

7. Complete Online Packaging for CORPORATE_MANAGEMENT:
    a. Select Edit, then Complete from the Online Packaging window.
    b. For each Online Load Module in CORPORATE_MANAGEMENT, Complete will assign source names to generation type components; add trancodes to the load module; and then assign dialog flow/clear screen trancodes to each procedure step packaged into the load module.
    c. Select Edit, then Exit to close the Online Packaging window.

Set DDL Configuration Parameters

Follow these steps:

1. To set the DDL configuration parameters
2. Select DDL, then Environment, then Configuration from the Construction Model Selection window.
3. Set the configuration properties and click OK.

Assign DDL paths

Follow these steps:

1. Select DDL, then Environment, then Paths from the Construction Model Selection window.
2. Assign the DDL paths. Paths will be validated based on the server platform. Paths do not have to exist on the server at time of validation but path names must be valid for the server platform.

Verify DDL Generation Defaults

Follow these steps:

1. Select DDL, Generate then Defaults from the Construction Model Selection window.
2. Verify the Target Environment settings.
3. Verify the remaining DDL Generation Defaults as listed in the following table and click OK.

Check Box	Value
Override Paths	No
Create storage groups	No

Check Box	Value
Qualify tables and indices with Owner ID	No
Create RI Alter Primary/ Foreign Keys & Triggers	

Generate DDL

Follow these steps:

1. Select DDL, then Generate from the Construction Model Selection window.
2. Select Generate, then Model from the DDL Generation window.
3. Verify the Generate Model Options as listed in the following table and click OK.

Check Box	Value
Install	Yes
Include Drop Statements	No

The Generation Progress Indicator windows appear as the DDL statements are generated. Once generation is complete, the Continue button in the Progess Indicator is enabled. When you click on the Continue button, the GENSTAT.CTL file is presented in the file browser. The GENSTAT.CTL file contains the generation status for all DDL components in the model. Close the Review window by double-clicking on the system menu icon in the upper left corner.

Set Configuration Parameters for Character-Based Generation

Follow these steps:

1. Select Code, then Environment from the Construction Model Selection window.
2. For Referential Integrity Triggers:
   a. Highlight the Referential Integrity Triggers.
   b. Select Detail, then Configuration.
   c. Set the configuration properties and click OK.
3. For Business Systems:
   a. Highlight the business system CORPORATE_MANAGEMENT.
   b. Select Detail, then Configuration.
   c. Set the configuration properties and click OK.

Assign Paths for Character-Based Code Generation

Follow these steps:

1. Select Code, then Environment from the Construction Model Selection window.
2. For Referential Integrity Triggers:
   a. Highlight the Referential Integrity Triggers.
   b. Select Detail, then Paths.
   c. Assign the Paths and click OK.
   d. Paths will be validated based on the server platform. Paths do not have to exist on the server at time of validation but path names must be valid for the server platform.
3. For Business Systems:
   a. Highlight the business system CORPORATE_MANAGEMENT.
   b. Select Detail, then Paths.
   c. Assign the Paths and click OK.

   Paths will be validated based on the server platform. Paths do not have to exist on the server at time of validation but path names must be valid for the server platform.

Verify Code Generation Defaults

Follow these steps:

1. Select Code, then Generate, then Non-Cooperative from the Construction Model Selection window.
2. Select Defaults from the Application Generation window.
3. Verify the Target Environment.
4. Verify the remaining Generation Defaults as listed in the following table and click OK.

Check Box	Value
Override Business System Target Environment	No
Override Paths	No
Delete generated source after install	No
Generate Foreign Action Blocks	Yes
Process modules marked for Compatibility	No

Generate Character-Based Source Code

Follow these steps:

1. Select Code, then Generate, then Non-Cooperative from the Construction Model Selection window.
2. To select objects, select Edit, then Select New Objects from the Applications Generation window.
3. Select Referential Integrity Triggers and Online Load Modules from the Generation Object Type List. Click OK.
4. On the Generation Object Occurrence List window:
   a. Select Referential Integrity Triggers.
   b. Select Code and Install. When selected, a Y appears.
   c. Select Add, then Continue.
   d. Select ALL Online Load Modules.
   e. Select Code, Screen, Install, and Apply to All Subordinates. When selected, a Y appears.
   f. Select Add, then Continue, then Cancel.
   g. Select Generate, then Flagged from the Application Generation window.

      The Generation Progress Indicator window appears as the source code is generated. Once generation is complete, the Continue button in the Progess Indicator is enabled. When you click on the Continue button, the GENSTAT.CTL file is presented in the file browser. The GENSTAT.CTL file contains the generation status for all Referential Integrity Triggers and Non-Cooperative Online generation components (Dialog Managers, Action Blocks, and Screens) in the model.

   h. Close the Review window by double-clicking the system menu icon in the upper-left corner.
5. Select Edit, then Exit to close the Application Generation window.

Transferring Remote Files to the Target System

Generation produces remote files, which contain the generated source and instruction for installation. Each remote file has a .rmt file extension. Before you can install the application into the target environment, you must transfer the remote files to the target platform. For detailed instructions about transferring remote files, see the Build Tool documentation.

GUI Generation, Installation and Testing

Use the following procedures to install and test GUI generation.

Installing and Testing GUI Generation

Follow these steps:

1. Install the server software and upload the CA Gen Sample Model into your CSE.
2. Start the Construction Client, open and log on to the encyclopedia.
    a. Select Encyclopedia, then Open from the Encyclopedia Construction Client window.
    b. Select the correct Encyclopedia from the Encyclopedia List window and press the Logon pushbutton. If only one Encyclopedia exists, it is auto selected.
3. Select Model from the Encyclopedia Construction Client window.
    a. Select Actions, then Open from the Construction Model Selection window.
    b. Select the GUI CA Gen Sample Model from the Model List window and click Open.

To Perform Window Packaging of the Model

Follow these steps:

1. Select Code, then Package, then Window from the Construction Model Selection window.
2. Select the business system GUI_CORPORATE_MANAGEMENT.
3. View “current” Window Packaging for GUI_CORPORATE_MANAGEMENT:
    a. Select View, then Expand from the Window Packaging window.
    b. Verify that one Window Load Module is listed on the List Load Modules window.
4. Remove “current” Window Packaging for GUI_CORPORATE_MANAGEMENT:
    a. Select the desired Window Load Module.
    b. Select Edit, then Delete from the List Load Modules window.
    c. You are prompted to confirm the deletion of the Window Load Module.
    d. Perform steps a through c for each load module.
5. Select Edit, then Exit to close the List Load Modules window.

6. Select the business system GUI_CORPORATE_MANAGEMENT from the list of business systems on the Window Packaging window.
7. To add Window Packaging for GUI_CORPORATE_MANAGEMENT:
    a. Select Edit, then Add Load Module from the Window Packaging window.
    b. Enter a load module name.
    c. Select all procedure steps listed on the Add Load Modules window.
    d. Click OK to create the Window Load Module.
8. To view "current" Window Packaging for GUI_CORPORATE_MANAGEMENT:
    a. Select View, then Expand from the Window Packaging window.
    b. Verify that one Window Load Module is listed on the List Load Modules window.
9. Select Edit, Complete from the Window Packaging window to complete Window Packaging for GUI_CORPORATE_MANAGEMENT.

    For each Window Load Module in GUI_CORPORATE_MANAGEMENT, Complete will assign source names to generation type components; add trancodes to the load module; and then assign clear screen trancodes to each procedure step packaged into the Window Load Module.
10. Select Edit, then Exit to close the Window Packaging window.

To Set DDL Configuration Parameters

Follow these steps:

1. Select DDL, then Environment, then Configuration from the Construction Model Selection window.
2. Set the configuration properties and click OK.

To Assign DDL Paths

Follow these steps:

1. Select DDL, then Environment, then Paths from the Construction Model Selection window.
2. Assign the DDL paths. Paths will be validated based on the server platform. Paths do not have to exist on the server at time of validation but path names must be valid for the server platform.

To Verify DDL Generation Defaults

Follow these steps:

1. Select DDL, then Generate from the Construction Model Selection window.
2. Select Defaults from the DDL Generation window.
3. Verify the Target Environment.
4. Verify the remaining DDL Generation Defaults as listed in the following table and click OK.

Check Box	Value
Override Paths	No
Qualify tables and indices with Owner ID	No
Create storage groups	No
Create RI Alter Primary/Foreign Keys & Triggers	No

To Generate DDL

Follow these steps:

1. Select DDL, then Generate from the Construction Model Selection window.
2. Select Generate, then Model from the DDL Generation window.
3. Verify the Generate Model Options as listed in the following table and click OK.

Check Box	Value
Install	Yes
Include Drop Statements	No

   The Generation Progress Indicator windows appears as the DDL statements are generated. Once generation is complete, the Continue button in the Progress Indicator is enabled. When you click Continue, the GENSTAT.CTL file is presented in the file browser. The GENSTAT.CTL file contains the generation status for all DDL components in the model.

4. Close the Review window by double-clicking the system menu icon in the upper left corner.

To Set Configuration Parameters for GUI Code Generation

Follow these steps:

1. Select Code, then Environment from the Construction Model Selection window.
2. For Referential Integrity Triggers:
   a. Highlight the Referential Integrity Triggers.
   b. Select Detail, then Configuration.
   c. Set the configuration properties and click OK.
3. For Business Systems:
   a. Highlight the business system GUI_CORPORATE_MANAGEMENT.
   b. Select Detail, then Configuration.
   c. Set the configuration properties and click OK.

To Assign Paths for GUI Code Generation

Follow these steps:

1. Select Code, then Environment from the Construction Model Selection window.
2. For Referential Integrity Triggers:
   a. Highlight the Referential Integrity Triggers.
   b. Select Detail, then Paths.
   c. Assign the Paths and click OK.

   Paths will be validated based on the server platform. Paths do not have to exist on the server at time of validation but path names must be valid for the server platform.
3. For Business Systems:
   a. Highlight the business system GUI_CORPORATE_MANAGEMENT.
   b. Select Detail, then Paths.
   c. Assign the Paths and click OK.

   Paths will be validated based on the server platform. Paths do not have to exist on the server at time of validation but path names must be valid for the server platform.

To Verify GUI Code Generation Defaults

Follow these steps:

1. Select Code, then Generate, then Non-Cooperative from the Construction Model Selection window.
2. Select Defaults from the Application Generation window and verify the Target Environment.

   Verify the remaining Generation Defaults as listed in the following table and click OK.

Check Box	Value
Override Business System Target Environment	No
Override Paths	No
Delete generated source after install	No
Generate Foreign Action Blocks	No
Process modules marked for Compatibility	No

To Generate GUI Source Code

Follow these steps:

1. Select Code, then Generate Non-Cooperative from the Construction Model Selection window.
2. To select objects to generate:
   a. Select Edit, then Select New Objects from the Applications Generation window.
   b. Select Referential Integrity Triggers and Window Load Module from the Generation Object Type List and click OK.
   c. On the Generation Object Occurrence List window, select Referential Integrity Triggers.
   d. Select Code and Install. When selected, a Y appears.
   e. Select Add, then Continue.
   f. Select ALL Window Load Modules.
   g. Select Code, then Screen, then Install, then Apply to All Subordinates. When selected, a Y appears.
   h. Select Add, then Continue, then Cancel.

i. Select Generate, then Flagged from the Application Generation window.

The Generation Progress Indicator window appears as the source code is generated. Once generation is complete, the Continue button in the Progress Indicator is enabled. When you click on the Continue button, the GENSTAT.CTL file is presented in the file browser. The GENSTAT.CTL file contains the generation status for all Referential Integrity Triggers and Non-Cooperative Window generation components (window load modules and action blocks) in the model.

3. Close the Review window by double-clicking the system menu icon in the upper-left corner.

4. Select Edit, then Exit to close the Application Generation window.

Transferring Remote Files to the Target System

Generation produces remote files that contain the generated source and instruction for installation. Each remote file has a .rmt file extension. Before you can install the application into the target environment, you must transfer the remote files to the target platform. For detailed instructions about transferring remote files, see the Build Tool documentation.

Index