CA Gen

Workstation 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 section contains the following topics:

Construction

Construction is the stage of CA Gen development where the activities of Planning, Analysis, and Design can be used to build an executable application system. The various application systems that you can construct include:

- Character-based (block mode) applications
- Graphical user interface (GUI) applications
- Client/server applications
- Batch applications
- Proxies
- Components

Note: The ability to generate GUI and client server applications depends on the CA Gen options on your workstation.

You can also cross-generate to available Implementation Toolsets.

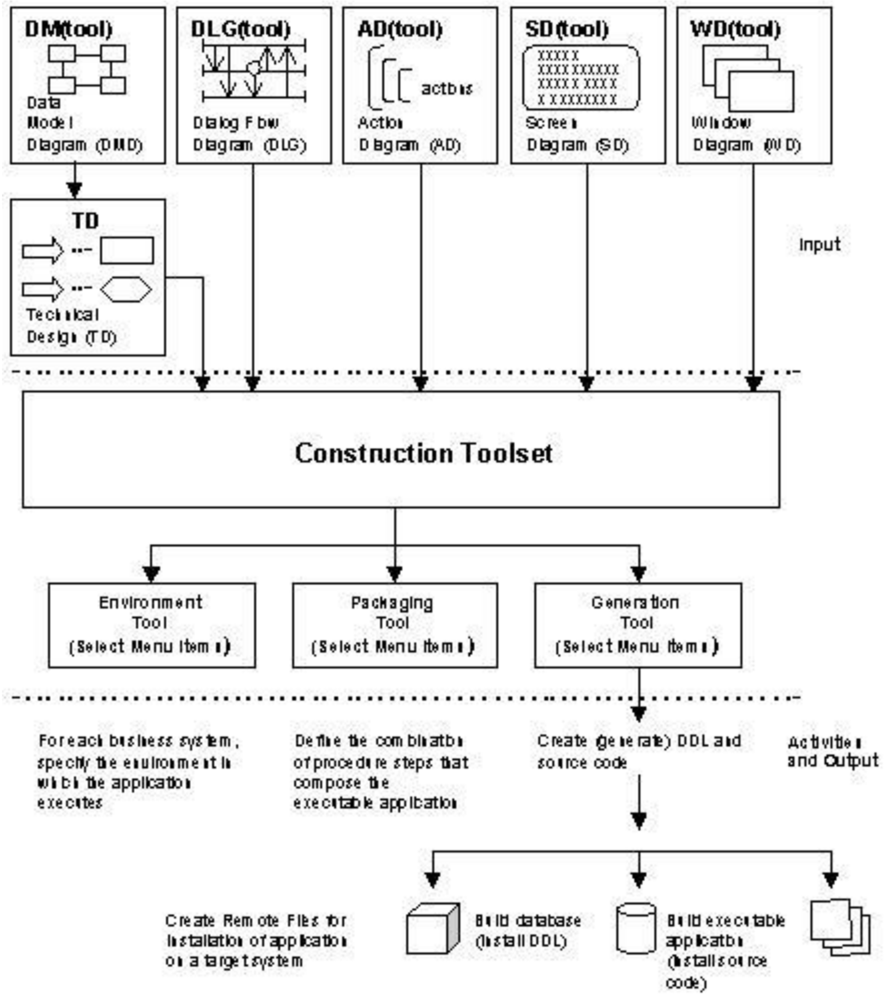
The Construction Toolset consists of three tools. They are:

- Environment,
- Packaging
- Generation

Each tool uses information from certain CA Gen diagrams, to define and create components used in the construction process. The diagrams must be complete and consistent.

The Construction Toolset cannot be installed and run by itself on a workstation. It requires the presence of the Analysis and Design Toolsets. The installation of your application will require a compiler and may require other third party software, like a Database Management System and middleware components. These software components may be on a workstation or on the target system. The location depends on where you intend to run your application. Because the installation uses software other than CA Gen, you may encounter problems that are not related to CA Gen software. You may need to see the vendor documentation for a specific operating system, DBMS, middleware, and compiler.

The following diagram presents an overview of construction:


DM(tool)
Data Model Diagram (DMD)
DLG(tool)
Dialog Flow Diagram (DLG)
AD(tool)
actions
Action Diagram (AD)
SD(tool)
Screen Diagram (SD)
WD(tool)
Window Diagram (WD)
TD
Technical Design (TD)
Input
Construction Toolset
Environment Tool (Select Menu Item)
Packaging Tool (Select Menu Item)
Generation Tool (Select Menu Item)
For each business system, specify the environment in which the application executes
Define the combination of procedure steps that compose the executable application
Create (generate) DDL and source code
Activities and Output
Create Remote Files for installation of application on a target system
Build database (Install DDL)
Build executable application (Install source code)

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

Information about the Dialog Flow Diagram (DLG) is incorporated into a transaction controller appropriate to the following application types:

- Dialog Manager for character-based (block mode) applications
- Window Manager for GUI applications
- Server Manager and Window Manager for client/server applications
- Server Manager that can be accessed by a proxy
- Batch Manager for batch applications
- Component for Component-Based Development

Screen layouts are implemented as screen definitions and controlled by a Screen Manager. Action diagrams contribute to several areas of a generated application.

Prerequisites for Construction

Construction requires the following diagrams:

- Data Model Diagram (DMD)
- Technical Design (TD)
- Dialog Flow Diagram (DLG)
- Action Diagrams (AD)
- Screens built using the Screen Design (SD) Tool (for a character-based application)
- Windows built using the Window Design functionality for the following:
  - GUI application (GUI application with local database)
  - GUI client component of the client server application

You must also meet the following prerequisites before you can complete the construction process.

- Model must be complete and consistent.
- A local or remote installation requires a DBMS supported by CA Gen installed on the target platform or database server.
- Client server installation requires a supported DBMS installed on the target server platform.
- A local or remote installation requires a compiler for the type of high-level languages generated by CA Gen.
- If used, external action blocks must be compiled outside of CA Gen for an installation on the target platform.

- Target system (remote) generation requires CA Gen options that enable generation for your specific target system.
- Selecting the Screen Generation Options is required for block mode applications with z/OS as the target platform.

Workstation Construction Process

During the workstation construction process, your CA Gen model is transformed into the components of an executable application (code and database definitions). These components are then prepared for execution on the workstation, or they are gathered into groups for transfer to the system where it will execute.

During the Construction process, you build and install a database from your transformed data model. You also translate the logic in your action diagrams, screen or window definitions, and dialog flow diagram into the executable code for an application.

Your CA Gen model is complete and consistent at the beginning of construction. During Construction, ensure that the application you generate will work properly in your selected environment. To do this, specify:

- The environment in which the application will run
- How the application is physically organized

Then select the portion of your application to generate. A complete application includes a database, referential integrity trigger modules, load modules (groupings of procedure steps), operations libraries (if the generated application will be a component), and, optionally, external action blocks. The application components can then be:

- Installed on your workstation for test purposes
- Concatenated into Remote Files for transfer to a target system for remote installation

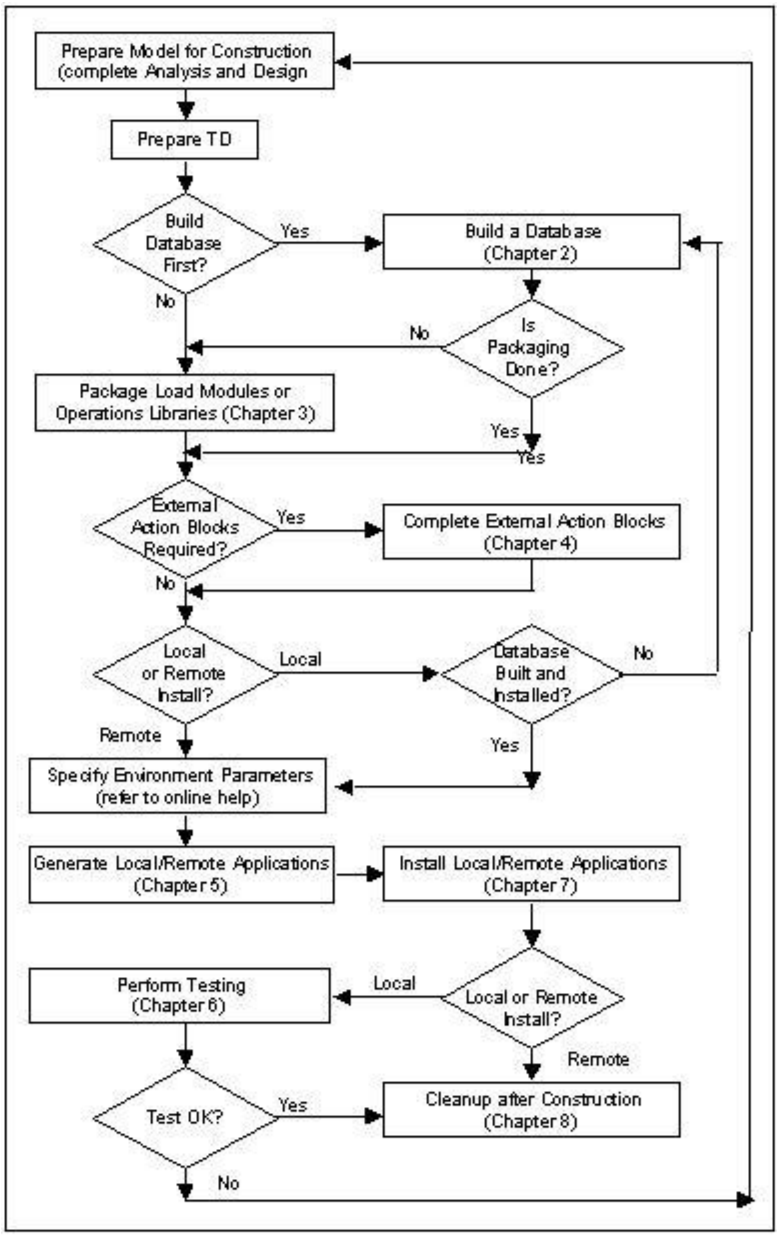
Summary of the Construction Process

The main activities in the Construction process include:

- Building the database
- Defining the combination of procedure steps that compose the executable application (referred to as packaging)
- Completing external action blocks
- Specifying the environment
- Creating the source code for the application

- Installing the application
- Testing the application

The following diagram illustrates the construction process:


Prepare Model for Construction (complete Analysis and Design
Prepare TD
Build Database First?
Yes
No
Build a Database (Chapter 2)
Is Packaging Done?
No
Yes
Package Load Modules or Operations Libraries (Chapter 3)
Yes
External Action Blocks Required?
Yes
No
Complete External Action Blocks (Chapter 4)
Local or Remote Install?
Local
Remote
Database Built and Installed?
No
Yes
Specify Environment Parameters (refer to online help)
Generate Local/Remote Applications (Chapter 5)
Install Local/Remote Applications (Chapter 7)
Local or Remote Install?
Local
Remote
Perform Testing (Chapter 6)
Cleanup after Construction (Chapter 8)
Test OK?
Yes
No

More Information:

Building a Database (see page 15)
Packaging (see page 27)
Using External Action Blocks (see page 43)
Generating Local and Remote Applications (see page 55)
Application Testing (see page 71)
Installing Local and Remote Applications (see page 73)
Cleaning Up After Construction (see page 81)

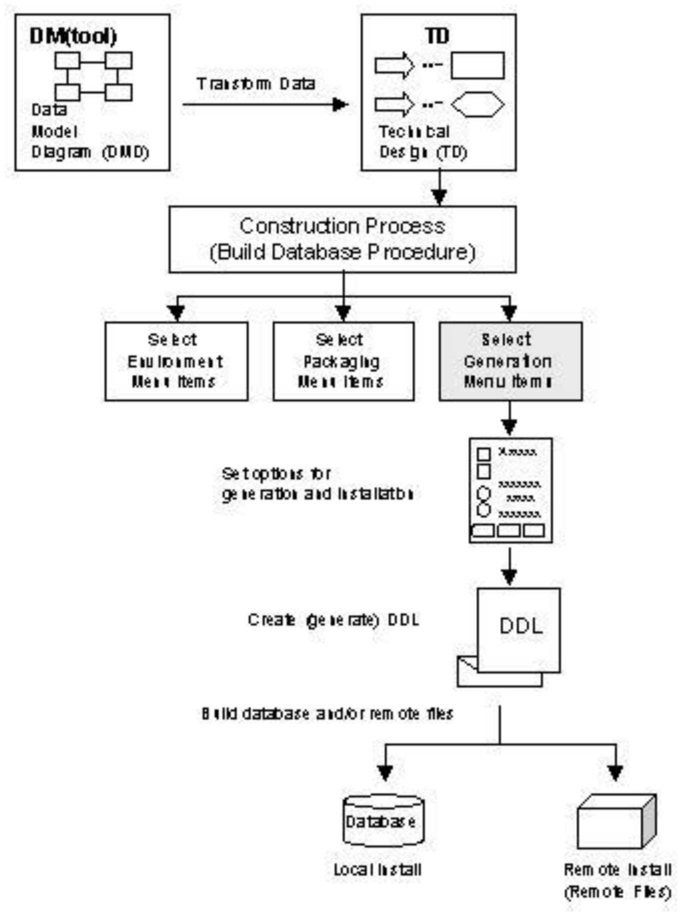
Chapter 2: Building a Database

This section contains the following topics:

Design Construction Process

The design used by the Construction process to build the database Data Definition Language (DDL) is the Technical Design (TD) as shown in the following illustration. The type of DDL generated by CA Gen is Structured Query Language (SQL).

The following diagram shows an overview of a database build:


DM(tool)
Data Model Diagram (DMD)
Transform Data
TD
Technical Design (TD)
Construction Process (Build Database Procedure)
Select Environment Menu Items
Select Packaging Menu Items
Select Generation Menu Items
Set options for generation and installation
Create (generate) DDL
DDL
Build database and/or remote files
Database
Local Install
Remote Install (Remote Files)

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

- The application has database access statements (SQL calls) in it.
- The application is a character-based (block mode) application that requires a profile table. For more information, see the appendix "Load Module Structure (see page 89)."

You can install the generated DDL for a local installation or a remote installation. For a local install, installation builds the database, tables, and indexes on your workstation. For a remote install, installation creates a remote file that you then transfer to your target system. On the target system, the Implementation Toolset unpacks the remote file and builds the database.

Note: You must choose Install to get a remote file when generating for remote installations.

Building for Local and Remote Data Services

You can build a database for either local data services or for remote data services (depending on the DBMS).

Building for Local Database Services

For local database services, the database is installed and accessed on the same workstation as the application. You can set the profile token OPT.DBCONNECT in the Build Tool to LOCAL for local database services.

Building for Remote Services

For remote data services, the database is installed on a database server (networked server with a DBMS installed on it) but is accessed on the workstation. The access from a workstation to a database on a database server is also considered a local installation. You can locate (or point to) the database on the server by specifying the drive of the server on the workstation for the database location. You do not need any special parameters for generation or installation.

To access a remote database with a CA Gen generated application:

- You must set the profile token OPT.DBCONNECT in the Build Tool to the name of the remote database.
- The DBMS must provide remote data services functionality. For information about configuring your workstation and the server to provide remote data services, see the vendor DBMS documentation.

Prerequisites

Before you can successfully build a database from a model or subset, you must perform the following tasks:

- Verify that the Data Model Diagram (DMD) is consistent
- Transform the DMD
- Verify that the Technical Design (TD) is consistent
- Access and start the DBMS, if necessary

It is also recommended that you verify that the date and time are correct for your workstation so that the proper timestamp is associated with the generated components.

Verify Consistent Data Model

A consistent model is necessary to ensure successful database generation. You must use the Construction-level consistency check to verify that the DMD is consistent and does not violate any of the rules that govern source generation.

Correct any inconsistencies before proceeding.

Transform the Data Model

You transform the DMD into a TD using the Design Toolset. The transformation process creates the data structure objects for the TD. These objects are then used by CA Gen to generate database definitions that include the DDL statements necessary to allocate and construct database objects. This information is stored on your workstation in one of the files for the model.

After initial transformation of the DMD, you can use retransformation to implement changes to the TD. For more information about transformation and retransformation, see the *Toolset Help*.

Verify Consistent Technical Design

The TD is the main source of information for database generation. As CA Gen generates DBMS-specific technical designs, you must select a DBMS consistent with the model, the technical design, and the database. Before CA Gen can successfully generate the DDL, the model's TD must be complete and consistent. In the Database Design tool, you can make manual adjustments to the transformed TD using the Data Structure List (DSL) and Data Store List. For example, you might add another entry point or change a table name. If you make changes, run a construction-level consistency check again because generation fails if any components are inconsistent.

If you intend to generate only a portion of the database definition (for example, a table and its indexes), the portion of the DMD and TD that describes the objects to be defined must be complete and consistent.

Required Activities Outside of CA Gen

Before you can install CA Gen generated database, you must be able to access and write to your selected DBMS. Usually, this involves steps such as accessing and starting the DBMS and ensuring that you have the appropriate permission for the task you want to perform.

How to Build a Database

The following tasks describe how to build a database.

Set Generation Parameters

Set Generation parameters before generating a database. Review the options for subsequent generations if your target or purpose changes.

Follow these steps:

1. Select Construction.
2. Select Generation.
3. Select Options.
4. Select Generation Defaults.

CA Gen provides the following dialogs to configure your application for generation:

- Environment Parameters (choose the Environment menu option)
- Generation Defaults (choose the Generation menu option)

- Client and Server Environments Parameters (choose the Packaging menu option and for more information about procedures, see *Distributed Processing - Overview Guide*).

Note: The Environment Parameters window applies only to a business system, not to the model. You must specify the configuration for your database DDL, Referential Integrity (RI) triggers, and the type of installation (local or remote) for the model on the Generation Defaults window.

At a minimum, you must define the operating system and the DBMS that you are using for this generation, and whether this particular generation is for the local workstation or for a remote (target) environment.

Several other options are presented for your selection. For more information about these options, see the *Toolset Help*.

Several options affect the generation of a database. The specific actions that occur because of selection are explained in the following table.

The additional options do not apply to database generation.

The database generation check boxes and their functions are as follows:

Include Drop Statements in DDL (Gen All)

Ensures that your database installs even if a database already exists with that same name.

Note: This check box applies only to the Generate, All action and the Generate, All and Install action

Qualify Tables and Indexes with Owner ID

Enables each person to have a unique databse when more than one person is testing with the same workstation.

Create Storage Group in DDL

View the SQL statements for storage groups in DB2.

Run Consistency Check for each item generated

Ensures consistency of the component you are generating.

Create RI Alter Primary/Foreign Keys & Triggers

Creates an alternate referential integrity primary and foreign key triggers for comparison with an existing database.

Include DROP Statements in DDL (Gen All)

Many database management systems cannot build a database, table, or index if one already exists. 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 certain DBMSs, a DROP statement is necessary before a new database or its components can be rebuilt.

You can also specify generation of the DROP statement for one or more database components from the DDL Generation window. The DROP column on the DDL Generation window overrides the DROP selection on the Generation Defaults window. In addition, when a component is dropped, all the data associated with that component is dropped as well.

Important! The DROP statement deletes a database and all of its data. Back up the data, if you need it before you DROP a database.

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 workstation.

Downstream Effects

Do not use this check box for target system implementation (remote installation). When the DDL is generated with the owner ID in the table and index names, there may be difficulty installing the DDL unless the ID of the user installing the DDL is the same as the user ID found in the DDL. To avoid the possibility of errors during installation, specify not to qualify tables and indexes with your owner ID.

Create Storage Group in DDL (DB2 only)

When applicable, selecting this option causes generation of DDL statements that let you take advantage of the Storage Group feature of DB2.

Run Consistency Check for Each Item Generated

This option verifies that each item generated is consistent before generation occurs. If the objects or model is not consistent, CA Gen specifies each inconsistency and indicates the level of severity (error or warning).

If you do not select this option, CA Gen does no checking before generation or installation. Generation or installation may fail if a component is not consistent.

Create RI Alter Primary/Foreign Keys and Triggers

If required, select this option to create alternate referential integrity primary and foreign key triggers for comparison with an existing database. This option generates RI constraints and primary key statements only.

Perform Build Tool Profile Setup

The Build Tool utilizes profile files that are used to customize a model's build environment. A set of tokens is used to define various environment settings (DBMS, System, Target and so on). These DBMS tokens are used when building the database.

Note: For more information, see the *Build Tool User Guide*.

Select a Generation Action

You can either choose to have CA Gen automatically build all database components (tables, indexes) for you, or you can choose the components to build.

The first time you generate the database, use the DDL, Generate All and Install option because it provides the quickest way to accomplish the task.

Generating All Database Components

If you choose to have the CA Gen automatically build the components, you can do either of the following:

- Generate DDL for all components but not install the DDL (DDL, All option).
- Generate and install all components of the database (DDL, All and Install option).

Generating Selected Database Components

If you choose to select the components individually, first open the database DDL, select the database, and expand all. Then toggle on each component and specify the action to be performed. For more information about DDL, Selected option, see the *Toolset* Help. After your choices are complete, select the DDL, Selected option.

Regenerating Database Components

When recreating components that were already installed, you must DROP the old component. You can add DROP statements globally using a selection on the Generation Defaults window, or you can do it for each component generated by toggling on the DROP column for the component. The DROP column overrides the DROP selection on the Generation Defaults window. When a component is dropped, all the data associated with that component is dropped as well.

Verify Completion

If you are installing a database on your workstation, the Build Tool continuously displays a status line of installation progress, giving you the option to review the results.

To review the results, highlight the status line and select Review. A log appears with additional information. If there is an error, once you have corrected the error condition, you can regenerate the database or selected components.

Note: Most error conditions can be avoided by assuring your model is consistent before generating. If the error log does not display, check for available disk space.

If you generated for remote installation, view the progress of the generation on the window that automatically appears when the generation begins. A successful or failed completion is reported as the final status.

Stopping the Generation Process

If you want to stop the generation process at any time, click STOP on the Generation Status window. Generation will stop within two seconds.

Results of Database Build

Because of the database build or installation process, CA Gen creates a sub-directory named DDL subordinate to the directory that contains the model. For example, a model named ORDRENT is contained in the ORDRENT.IEF directory. The associated database components generated by CA Gen are therefore stored in the DDL sub-directory under ORDRENT.IEF.

The list of various files that result when a database is built is as follows:

Note: Each file uses the database name as the filename.

- CTL

  Contain information used by CA Gen when creating a remote file. Occurs for remote installation only

- DDL

  Contains SQL statements without the installation information (such as the BIND statements).

- ERR

  Contains messages resulting from a failed generation.

- GEN

  Contains data used by the DDL generator.

- ICM

  Contains install deck information. The install deck is a set of instructions on how to install all procedure steps, screens, and action blocks for the load module. The install deck is in Standard Generalized Markup Language (SGML) format. This format lets you write your own installation procedure from the file should you have a particular installation requirement.

- INS

  Contains SQL statements including the installation information (such as the BIND statements and CREATE TABLE RPROFX statement).

- OUT

  Contains a log of the activities that occurred during installation. This log, called a review file, can be viewed with a text editor or from the Build Tool with the Review option.

- RMT

  Occurs for remote installation only. If you selected generation for remote installation, the .RMT file is the remote file.

- TXT

  Contains information used by CA Gen when creating a remote file. Occurs for remote installation only.

- TGT

  Builds tool set-up file per module.

- CMD, BAT

  Installs batch command files.

Database Directories

If you select local installation, your DBMS will store your databases as a set of files on physical storage. The DBMS, not CA Gen, controls the location of the database.

Installing DDL with the Database Loader

After the database build process, CA Gen lets you install DDL (database) using a database loader program. CA Gen installs loader programs for each supported DBMS.

DDL installation using a database loader program is supported on Windows, UNIX, and Linux operating systems.

Recompiling the Database Loader Program

If you are using a DBMS whose version is different from the DBMS version used to generate the DDL, it is necessary to rebuild the database loader program for the current DBMS version.

For more information, see the appendices “Rebuilding DBMS DLLs and Executables” in *Windows Implementation Toolset User Guide* or “Rebuilding DBMS Shared Libraries” in *UNIX and Linux Implementation Toolset User Guide.* .

Chapter 3: Packaging

This section contains the following topics:

Load Module Packaging

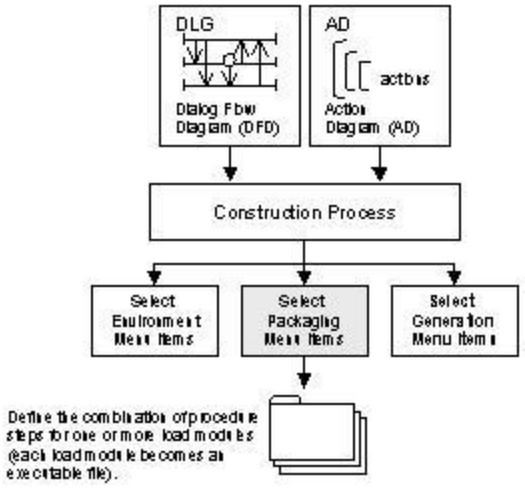
Load module packaging lets you specify the components to create an executable program. The act of packaging does not create the load modules. Packaging defines the contents of a load module. A load module (or executable program) is generated, and then built during installation. An installed load module contains the executable code for all components included in its packaging definition plus special function routines and system routines.

Note: If you are building a component for use in Component-Based Development, specify Operations Libraries, rather than Load Modules, to package and generate the component.

For the Code, All and Code, All and Install actions, CA Gen does the packaging for you when you generate code. For the Code, Selected action, you must package manually.

If you want CA Gen to package for you, you can skip this chapter. However, if you let CA Gen package for you, you have little control over the packaging names and scheme.

The following diagram describes you an overview of Packaging:


DLG
Dialog Flow Diagram (DFD)
AD
actions
Action Diagram (AD)
Construction Process
Select Environment Menu Items
Select Packaging Menu Items
Select Generation Menu Item
Define the combination of procedure steps for one or more load modules (each load module becomes an executable file).

Prerequisites

Before you perform load module packaging, you must:

- Open a business system.
- Create the dialog flow of the procedures using the Dialog Design Tool.
- Create the action diagrams using the Action Diagramming Tool.

The Packaging Tool requires a completed Dialog Flow Diagram (DFD) for you to create a load module. For this reason, a completed Action Diagram (AD) is also a prerequisite. The load modules are nothing more than file names. That is, the files, at this point, are empty. The contents (procedure steps) for load modules come from the AD (including any supporting action blocks) that you completed for each procedure step.

Why This Step Is Necessary

An application generated using CA Gen generated model contains many procedure steps. These procedure steps are the building blocks of executable applications, but they need to be combined into logical groupings or units for execution.

Packaging combines the procedure steps of the application into units that can be executed. Before you can generate an application, CA Gen requires that these units and their contents be identified.

CA Gen refers to the packaged units as load modules. The term load module, which is commonly associated with the IBM mainframe world, refers to an executable program. In other environments, similar concepts are:

- Windows executable (.EXE) file
- UNIX, Linux, or NonStop executable

Load Modules are designed to run within a shell or an environment, commonly called a TP monitor. The TP monitor manages screen or window handling functions and some execution control functions. As a result, each executable program does not have to contain the code required to perform these tasks.

How to Package

Packaging varies according to the type of application being generated. They are as follows:

- Online packaging for character-based (block mode) applications
- Batch packaging for batch applications
- Window packaging for GUI applications
- Cooperative packaging for Client server applications
- Component packaging to build components for use in Component-Based Development

Packaging also varies according to the environment for which you are generating. If you change the environment, the new target may have different packaging requirements, making repackaging necessary.

All packaging can be performed on the workstation. All packaging is uploaded and stored on the encyclopedia. Packaging done on the encyclopedia is downloaded with the model or subset and must be changed if it is not appropriate for your workstation environment.

Note: You may see an asterisk (*) next to a procedure step when you package load modules. The asterisk indicates that the procedure step is checked out to another subset. You cannot package procedure steps that are checked out to another subset.

Using the Complete Option

As an aid in packaging, CA Gen provides a Complete option that you can select at several points during packaging.

The Complete option finishes packaging for the selected object. The Complete option also 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 level.

At whatever level you pick the Complete option, CA Gen finishes required packaging for the selected object and its subordinates.

For example, when you complete online packaging for a business system, CA Gen software:

- Creates one or more load modules (based on your selection of multiple modules or one module) and assigns load module names.
- Assigns unpackaged online procedure steps (with called action blocks) to the load modules.
- 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 TP monitor.
- Assigns clear screen and dialog-flow transaction codes for each procedure step packaged during transformation.

You can view the results of packaging by selecting the business system and selecting VIEW and EXPAND ALL. For more information, see Results of Packaging *(see page 41).*

EJB Option

The EJB Generation Option lets you generate, build, and assemble Server Procedure Steps as Stateless Session Enterprise Java Beans (EJBs). This feature enables traditional CA Gen Servers to run as an Enterprise Java Bean (EJB) in a J2EE container. The following sections provide some brief information about the EJB option. The TP monitor will be able to select the EJB option. The EJB properties tab will be available in server details and within the EJB property tab is the URL of the naming and directory server and the Always Start a new Transaction flag.

Note: The default is for EJB procedure steps to participate in an existing transaction.

For more information about installing prerequisite software and functionality of the EJB option, see *Distributed Processing - Enterprise JavaBean User Guide.*

Server Procedure Steps

In the TP monitor, the server procedure steps are responsible for performing most of the application's business logic and database accesses. If allowed by the TP monitor, a server procedure step can invoke other server procedure steps. Server procedure steps do not have a user interface.

Runtime Environment

The EAR files generated by CA Gen are designed to assemble any application server that conforms to the J2EE 1.4 standard.

Generating a Model in Java

The following sections describe the process of generating a CA Gen model in Java and assembling it onto an application server.

Generating

The generation of Java Web Clients and EJB Servers can take place on the CA Gen Workstation Toolset or the Client Server Encyclopedia (CSE). If the generation takes place on a CSE, the remote files must be moved to a Windows Professional workstation to be built.

Building

After generating the application in Java, it can be built only on a Windows Professional workstation system using the CA Gen Windows Build Tool. The application is ready for assembly after all the CA Gen Load Modules have been successfully built.

Basic Packaging Activities

Packaging consists of the following basic activities:

1. Selecting the Business System
2. Defining load modules
3. Defining source names
4. Assigning transaction codes

Selecting the Business System

The first step in load module packaging is to select or name the business system where you define load modules and their components. You most likely did this when you defined the business system using either the Business System Definition Tool or Matrix Processor in Analysis.

Defining Load Modules

Defining a load module consists of the following activities:

- Naming the load module
- Adding procedure steps

Downstream Effects

When generating for remote installation, each load module or operations library becomes a separate remote file that is linked on your target system into a separate executable program. The Referential Integrity modules are packaged into a single remote file. The DDL associated with an application database is packaged as a single remote file.

Name the Load Module

The name you choose for the load module becomes the name of the executable file. Therefore, you may want to choose meaningful names. CA Gen software accepts up to eight alphanumeric characters as the load module name. The first character must be alphabetic. When you package for remote installation, the load module name also becomes the remote filename.

Note: Server load module names should not start with the letter C if the target system platform is CICS for the server application.

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 your load module, all action blocks used by the procedure step are also included in the load module by CA Gen, as well as all action blocks used by the action blocks.

- GUI Applications

  All procedure steps must be packaged into one or more load modules.

- Client Server Applications

  Procedure steps are packaged according to load module types. The load module types are server manager for the server procedures and window manager for the client procedures.

  For the server manager load module type, only the procedure steps that are online, with no display, appear in the list of all the unpackaged procedure steps defined for the business system.

  For the window manager load module type, the procedure steps that are to be packaged in window load modules are usually online with display.

  For the most part, flow definitions for a client server design are dependent on packaging options. That is, when you create a Dialog Flow Diagram, you do need to consider which procedures you will package as clients and which procedures you will package as servers. Therefore, cooperative packaging for client server applications tends to be dictated by the Dialog Flow Diagram.

- Batch Applications

  Flow definitions for batch procedure steps are limited to sequential transfers from one procedure step to the next, with no returns to prior procedure steps. Each batch procedure step must be packaged into a separate load module.

- Online Applications

  Flow definitions for online applications are independent of packaging options. That is, when you create a DLG, you do not need to consider how the procedures will be packaged. Conversely, packaging is not dictated by the dialog flow. However, an application runs most efficiently when each load module contains procedure steps that are to be used together.

- Components

  A collection of transactional or sub transactional operations are grouped together in CBD packaging. This collection, called an Operations Library, is packaged into a DLL or a shared library, which in turn is linked to (consumed by) one or more applications.

Online and Window Tradeoffs

For local testing (testing performed on the workstation) of character-based (block mode) applications, the recommended method is to package several procedures (and all of their procedure steps) into one load module. This recommendation has to do with performance of generated code because of the number and size of load modules.

During generation, each load module becomes an executable file. When a load module is called, it must be loaded into memory. When the next load module is called, it, in turn, is loaded while the previous load module is removed from memory.

If the load modules are too small, a performance penalty occurs when loading and reloading executable files.

In contrast, if the load modules are too big (for example, if you package all procedure steps into one load module), the executable file may be too large to fit into available memory, or the load module may exceed the size requirements of the compiler.

Defining Source Names

Each procedure step or action block must have a source name for CA Gen to use when storing the generated code. You can assign source names during packaging, or CA Gen will assign them for you during installation of the load module. CA Gen software verifies the uniqueness of source names within the model or subset on the workstation. The source name is also generated into the code as the procedure name in C and Java.

For cooperative packaging, this stands true for server procedures as well. For the client procedures, the source name is generated into the code as the function name in C and Java.

Source names can be up to eight characters long, and the first character must be alphabetic.

Assigning Transaction Codes

CA Gen, during generation of the load module, assigns transaction codes to each of the procedure steps. (This is true only for the Code, All and Code, All and Install actions.) The two kinds of transaction codes are clear screen and dialog flow.

Of the two transaction codes, the dialog flow transaction code is optional but will be used by the generator if specified. However, 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 code names to them.

For server procedures, specify the dialog flow and no clear screen transaction codes.

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

Note: If the target system platform is CICS, the transaction codes for the server load modules cannot start with the letter C and can only be four characters long.

CA Gen software accepts names of up to eight characters and the first character must be alphabetic.

CA Gen supplied teleprocessing monitors (commonly called TP monitors) use a special file named AEENV to store clear screen transaction codes. CA Gen adds to this file as each load module is installed. You must, therefore, maintain the file manually because CA Gen software does not purge or prevent duplicate names. Each clear screen transaction code should be unique within the AEENV file.

For remote installations using CA Gen supplied TP monitor such as IEFAE, AEF, or NonStop AEF, each target configuration has one file.

UNIX and Linux have one AEENV file per model. The AEENV file is located in the directory or subdirectory where you have installed CA Gen.

Note: GUI applications require the AEENV file only when diagram testing under the Build Tool.

Packaging Options

The Packaging Tool is used to identify the minimum number of objects required to generate and install the source code for a load module or operations library. Packaging does not create the source code for an executable load module or operations library; that is done by generation.

The following options are available with the Packaging Tool:

- Online
- Batch
- Window
- Cooperative
- Component
- z/OS Library

To perform packaging

1. Select Construction (from either the CA Gen tree view or the menu bar).
2. Select Packaging.
3. Select Diagram.
4. Select Open.
5. Select the required Packaging option.

For online, batch, Window, and cooperative packaging, a packaged load module contains at least the procedure steps and names for the generated source code files. A packaged load module also contains other information depending on its type. For Component Packaging, a packaged operations library contains at least the action blocks and names for the generated source code files.

The packaging information for all packaging types is transferred to the host when a model is checked in.

For more information about Packaging, see the *Toolset Help*.

z/OS Dynamic Linking

CA Gen generates different COBOL code when calling a routine that is statically linked within an executable as opposed to when calling a routine that resides in a separate executable. For calls to routines that are statically linked within an executable, CA Gen generates a COBOL CALL literal statement. For calls to routines that reside in an executable separate from the calling executable, CA Gen generates a COBOL CALL identifier statement that references a variable. The variable contains the name of the routine that is to be dynamically invoked. A program call to a routine that resides in a separate executable is known as *dynamic linking*.

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 AllFusion Gen before Release7, 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. 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 key features and requirements are:

- 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 dynamically 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 section in the Release Summary. 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 be 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.

Local Testing

Testing a client server application on your workstation (called local testing) before you place the application into production is highly recommended. For local testing, you need to package all of the client and server procedures into window load modules under cooperative packaging. After the local test runs successfully, then you must repackage the server procedures into server load modules.

Note: Local testing is not applicable if you are using a mainframe DBMS or asynchronous cooperative processing.

Results of Packaging

To view the results of packaging, select a business system, VIEW, and EXPAND ALL. The resulting display shows the load modules and the procedure steps and action blocks that comprise them.

To check for any unpackaged procedure steps, you can attempt to ADD to a load module. If the resulting list is blank, all procedure steps were packaged.

Chapter 4: Using External Action Blocks

This section contains the following topics:

Importance of EAB

You will need to generate, complete, and install external action blocks (EABs) if the CA Gen generated application you are going to install will access any information or processes not generated by CA Gen. External action blocks must be completed and installed before the load modules that invoke them are installed.

For remote installation, external action block interface routines need to be generated on your workstation. The action block interface routines then need to be moved to your target system for completion and installation. For more information, see the Implementation Toolset Guide that applies to your specific target system such as the *Windows Implementation Toolset User Guide*, the *UNIX and Linux Implementation Toolset User Guide* and the *NonStop Implementation Toolset User Guide.*

Note: If there are no external action blocks in your application, bypass this chapter.

How External Action Blocks Are Used

External action blocks let you access logic created outside of CA Gen. For example, your load module may need to use a standard date manipulation subroutine that was defined for your organization. Alternatively, the load module may need to access files created with a DBMS not supported by the CA Gen product.

External action blocks are defined as CA Gen action blocks that do not contain program logic. External action blocks are defined using the Action Diagramming Tool. When you define an external action block, you enter the name and define the views, as with any other action block.

The import and export views in the external action block are the interface between the CA Gen procedure steps (or action blocks) and the logic created outside of CA Gen. An external action block provides a structure that is used to match the views of a procedure step with the views (input and output arguments) of a non-CA Gen generated subroutine.

Action Block Components

Conceptually, the external action block involves the following items listed in the following table:

- Procedure step or action block that uses the external action block

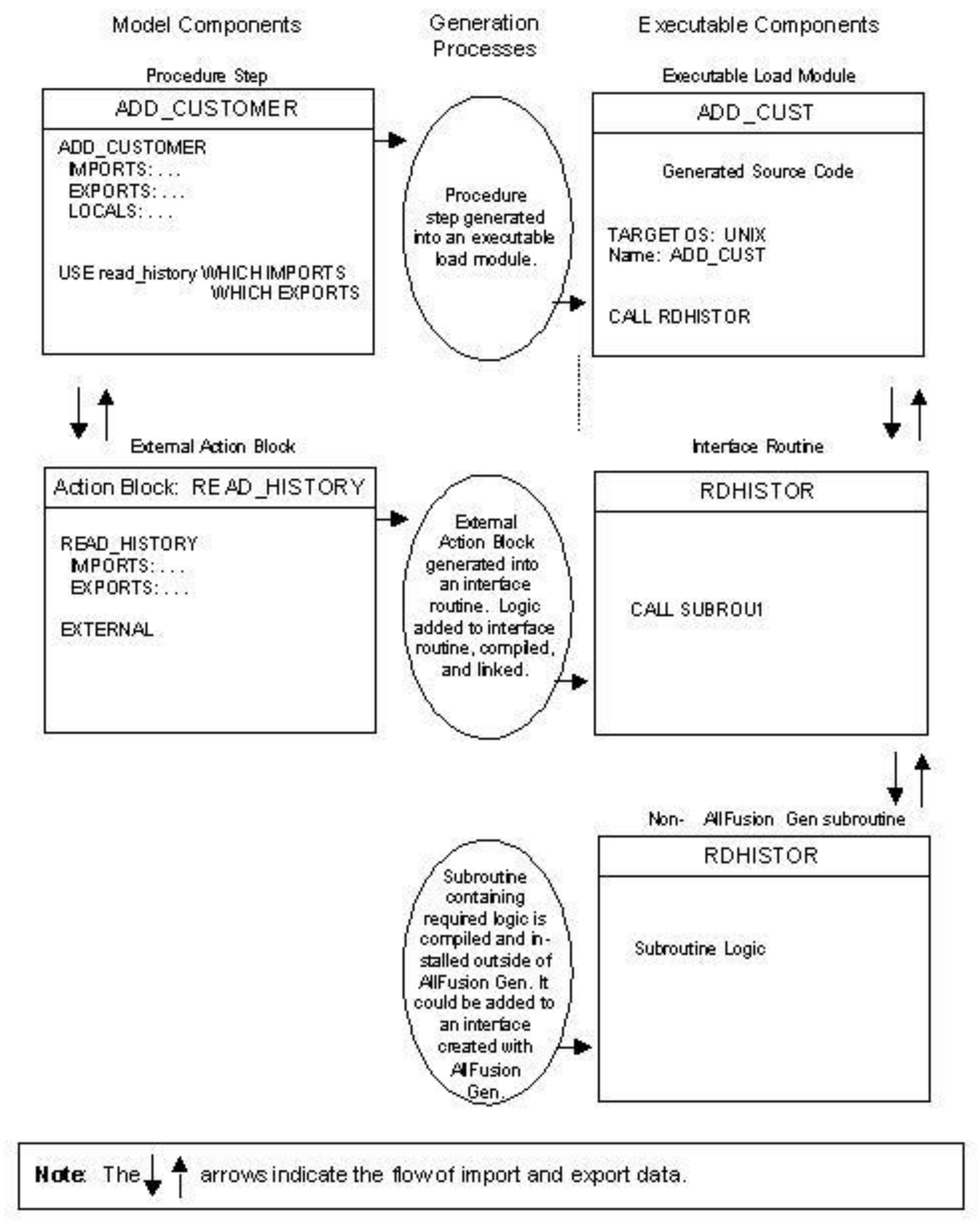
  Created in Action Diagramming Tool (Design)
- External action block
- Created in Action Diagramming Tool (Design)
- Interface routine

  Created in Action Diagramming Tool (Design)
  - The interface routine is generated from the external action block using the Generate option from the PAD.
  - The interface routine can also be generated using the Construction Toolset.
- Non-CA Gen subroutine logic

  Outside of CA Gen

CA Gen uses the views defined in the CA Gen external action block to generate the interface routine. The interface routine includes much of the view structures needed to transfer data to and from the externally generated code (the user-written code). Manually written code is then added to the interface routine either in the form of a call to an existing subroutine or in the form of handwritten, non-CA Gen subroutine logic.

The following diagram illustrates the elements and processes involved in external action block execution:


Model Components
Generation Processes
Executable Components
Procedure Step
ADD_CUSTOMER
ADD_CUSTOMER
IMPORTS: . . .
EXPORTS: . . .
LOCALS: . . .
USE read_history WHICH IMPORTS
WHICH EXPORTS
Procedure step generated into an executable load module.
Executable Load Module
ADD_CUST
Generated Source Code
TARGET OS: UNIX
Name: ADD_CUST
CALL RDHISTOR
External Action Block
Action Block: READ_HISTORY
READ_HISTORY
IMPORTS: . . .
EXPORTS: . . .
EXTERNAL
External Action Block generated into an interface routine. Logic added to interface routine, compiled, and linked.
Interface Routine
RDHISTOR
CALL SUBROU1
Subroutine containing required logic is compiled and installed outside of AllFusion Gen. It could be added to an interface created with AllFusion Gen.
Non- AllFusion Gen subroutine
RDHISTOR
Subroutine Logic
Note: The ↓ ↑ arrows indicate the flow of import and export data.

Prerequisites

Programming experience and a knowledge of the programs or information being accessed using the external action block are required for completing external action blocks.

Why this Step is Necessary

External action blocks are necessary to access logic or databases created outside of CA Gen.

Viewing the action diagram that contains the action block can identify external action blocks. The action diagram contains only one key statement, EXTERNAL. This statement identifies the action block as external.

If external action blocks are not properly compiled, linked, and stored, any action blocks and procedure steps that use an external action block will fail to install. CA Gen is not responsible for installing EABs; the EAB install process is dependent on the programming language and the specific content and logic contained in the completed EAB.

How to Use External Action Blocks

Follow these steps:

1. Generate external action block interface routines.
2. Locate the external action block code.
3. Determine the input and output requirements for your external action.
4. Create the appropriate external action logic.
5. Compile the external action block.

Except for Step 1, each of these tasks is performed outside of CA Gen.

Generating External Action Block Interface Routines

External action block interface routines are generated individually and separately from the load module that contains the action diagram for the external action block. This is done so that the interface routines can be completed before the rest of the code is generated and installed.

External action block interface routines can also be generated when the code for a load module or selected load module components is generated.

If you selected local installation, you can generate code for procedure steps and action blocks that use external action blocks, but you will not be able to successfully install that code until the external action blocks are completed and installed.

Locating External Action Block Interface Routines

The external action block interface routine generated by CA Gen is located in the source-code component subdirectory associated with the model being implemented. The file containing external action block code is named using the action block source member name.

If you select Delete Source After Install on the Generation Defaults panel, all source code is deleted except the external action block interface routines.

It is recommended that you move or copy the external action block interface routine to another directory on your workstation when you add logic to it. (You may want to create a subdirectory under the EXTRN directory for this purpose.) This will eliminate the possibility of overwriting your completed external action block source code by generating the EAB interface routine again.

Determine Input and Output Requirements for External Action Blocks

Determine the data the action block will receive from the CA Gen generated application, the data that must be returned to the CA Gen generated application, and the procedures that will be required to implement the external action block.

CA Gen uses the views defined in the external action block to generate the interface routine. The import and export views shown in the interface routine are supplied by the CA Gen procedure step or action block that uses the external action block. The import views define the data that is passed from the CA Gen generated application to the interface routine. The export views define the data returned to the CA Gen generated application from the interface routine.

Decimal Precision Attributes

Import and Export views in External Action Blocks may contain any of the supported attributes by CA Gen. This section is intended to give you a better understanding for handling views that contain attributes of type decimal precision.

The data structure for an attribute that is implemented with decimal precision is a DPrec array whose size is the length of the attribute plus three (for the sign, decimal point, and null terminator). A DPrec is a typedef of char. For example, an attribute that is defined as a number of 18 digits will be implemented as DPrec[21].

The number represented within the DPrec array may consist of the following depending on the definition of the attribute it implements:

- A minus sign
- Zero or more decimal digits with a decimal point, one or more decimal digits without a decimal point

- A decimal point
- One or more decimal digits
- A null terminator

For the import view, a decimal precision attribute may be its simplest form or may contain a plus sign and/or leading and trailing zeros.

For the export view, all decimal precision attributes must adhere to the following rules:

- The character representation of the number placed in the DPrec array may contain a fewer number of digits than that defined for the attribute. However, it must not contain more digits to the left or right of the decimal than that defined for the attribute.
- The DPrec array must be null terminated.
- If number of digits to the right of the decimal equals the total number of digits, the resulting string must not contain a leading zero prior to the decimal point.

Creating External Action Logic

To implement an external action block, you may prefer to add a call to an existing subroutine rather than including the subroutine logic itself. Either method is acceptable.

You can complete an external action block two ways:

- Code it completely by hand
- Add logic to a CA Gen generated interface routine

You should use the CA Gen generated interface routine because much of the work is already done for you in a format that is acceptable to CA Gen.

Interface-routine logic can be written in any language. This language does not have to be the same as that generated by CA Gen. If this
interface-routine is used by a mainframe application, it must be written in a language that follows LE linkage conventions.

However, the interface routine logic must follow specific requirements. The CA Gen generated load module passes the following parameters to the EAB interface routine. They are passed in the order indicated in the following table.

For C Without High Performance View Passing	**For C With High Performance View Passing**	**For Java**
IEF-RUNTIME-PARM1	IEF-RUNTIME-PARM1	in_runtime_parm1

For C Without High Performance View Passing	For C With High Performance View Passing	For Java
IEF-RUNTIME-PARM2	IEF-RUNTIME-PARM2	in_runtime_parm2
w_ia (import view C)	PSMGR-EAB-DATA (null array)	in_globdata
w_oa (export view C)	w_ia (import view C)	import_view
PSMGR-EAB-DATA (null array)	w_oa (export view C)	export_view

IEF-RUNTIME-PARM1 and IEF-RUNTIME-PARM2 are the first two parameters passed from the CA Gen generated load modules. These parameters must be coded on the entry statement of the interface routine and must always be passed in this order to any subordinate routines that are called.

For the C language, the position of the import and export views (w_ia and w_oa) depends on whether High Performance View Passing is used (see the previous table that describes the order in which parameters are passed from load modules to EABs). By default, High Performance View Passing is set ON.

Note: For more information about High Performance View Passing, see the Host Encyclopedia Construction User Guide or the Toolset Online Help.

For component development, High Performance View Passing must be set ON for both the component and the consuming model.

PSMGR-EAB-DATA is an array set to zeroes (Null array). For target system implementation, this array is passed but not used. (The array is used for IMS and CICS applications on the mainframe.) For more information about the use of this array, see *Host Encyclopedia Construction User Guide.*

Modify the stub using any editor that can save files in the appropriate character set format. The modified stub cannot contain any control codes or header information unique to the editor.

Note: After you add logic to an external action block interface routine, remember to move or copy it to another directory. This ensures that changes are not overwritten if you generate the action block again.

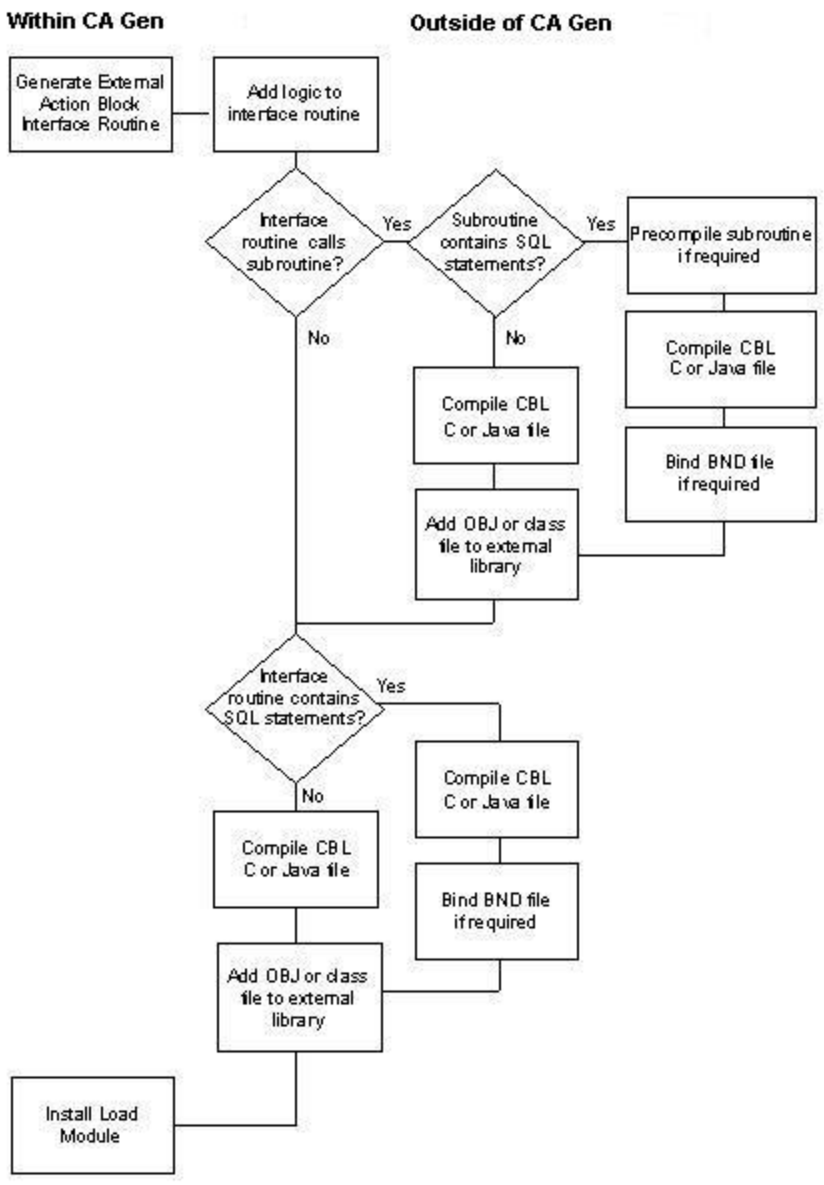
Compiling, Binding, and Storing an External Action Block

There are four steps to follow in creating an external action block:

- Precompile the external action block (depending on the DBMS).
- Compile the external action block.
- Bind the external action block (depending on the DBMS).
- Add the external action block to the External Library.

In addition, there are usually subroutines called by the external action block. These four steps also apply to these subroutines.

The following figure illustrates these steps. Note that all of these activities, with the exception of generating the external action block interface routine, are performed outside of CA Gen. After these steps are completed, you can return to CA Gen and install the load module that uses the external action block.


Within CA Gen
Outside of CA Gen
Generate External Action Block Interface Routine
Add logic to interface routine
Interface routine calls subroutine?
Yes
Subroutine contains SQL statements?
Yes
Precompile subroutine if required
No
No
Compile CBL C or Java file
Compile CBL C or Java file
Bind BND file if required
Add OBJ or class file to external library
Interface routine contains SQL statements?
Yes
No
Compile CBL C or Java file
Compile CBL C or Java file
Bind BND file if required
Add OBJ or class file to external library
Install Load Module

Precompile the External Action Block

There are certain conditions where a precompile step may be necessary. For example, if your external action block contains SQL statements used to access your DBMS, you may need to precompile it.

Compile the External Action Block

Compile the external action block interface routines to create object files (.obj) or class files (.class). The object files must be added to a special library as described later in this section. For more information about usage of the compiler, see the compiler documentation.

Bind the External Action Block

This step is associated with precompile steps and is only necessary if your external action block interface routine accesses a DBMS that requires a bind.

The precompiler creates a bind file with an extension of .BND. Bind this module to the database using the SQLBIND command.

Add the External Action Block to the External Library File

When you install a CA Gen load module that uses external action blocks, the installation process requires an external library file. This file contains all compiled external action blocks and all subroutines called by them.

You must create the external library file in the extrn directory for UNIX and Linux. The directory in which the external library file resides is located in the same directory level in which CA Gen models are stored.

For UNIX and Linux, the library file must be named with the extension .a for C applications. You create the library file outside of the CA Gen with the ar command. For more information about the ar command and usage, see the UNIX and Linux compiler documentation.

If the external library file resides in a different location other than the extrn directory, use the Build Tool token LOC.EXTERNAL_LIB to point to the location of that library.

When you install an application from the Build Tool and set the LOC.EXTERNAL_LIB token, the Build Tool uses the external library file pointed to by that token.

The Build Tool adds optional levels to external library management. Traditionally, a single external library is referenced at the model level. Models referencing external objects to perform differing tasks may cause a conflict. The CA Gen Build Tool will additionally reference an external library in the EXTRN subdirectory subordinate to the generated code directory so that model-specific objects are managed better.

You can also process external action blocks for remote installations. For more information, see the *Windows Implementation Toolset User Guide*, the *UNIX and Linux Implementation Toolset User Guide*, the *NonStop Implementation Toolset User Guide*, or the *z/OS Implementation Toolset User Guide.*

Results

The result of External Action Block development and installation process is a compiled, executable action block that resides in the appropriate action block library. The location can be determined by listing the library's contents.

When related to your application, however, the activities involving external action blocks conclude within CA Gen. When you finish installing all the required external action blocks, you can install the load module that uses the external action block.

If the load module that uses the external action block installs successfully, you are ready to test the load module.

For more information about testing a load module, see the chapter "Application Testing (see page 71)."

Chapter 5: Generating Local and Remote Applications

This section contains the following topics:

Local Applications

Generation is the activity that creates source code. A local application is one that is installed and executes in the same environment in which you generated the source code.

Remote Files

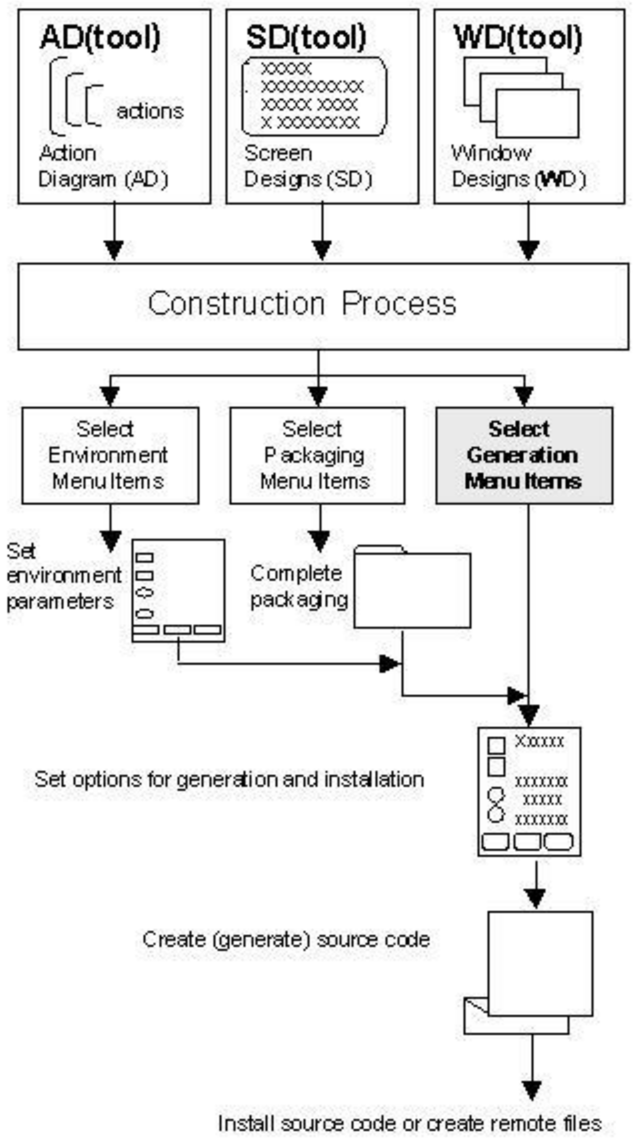
A remote file is created on one system, and installed and executed on another. There are four types of remote files. Each of the files contains a different type of information necessary for the application. They are described in the following table:

Type	Number	Description of Contents
Load Module	One or more per application	Application source code for a single load module.
Database	Usually one per application	DDL for an application database. For remote installation, all load modules for a CA Gen application usually share a common database.
RI Triggers	One per application	Referential Integrity trigger modules for an application. These routines implement referential integrity for the entire application.
Operations Libraries	One or more per application	Public action blocks for a single operations library

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

The remote process generates the Load Module and Referential Integrity remote files. The remote file that becomes the application database is generated separately. For more information about building a database, see the chapter "Building a Database."

The following diagram shows a general overview of the generation process:


AD(tool)
actions
Action Diagram (AD)
SD(tool)
Screen Designs (SD)
WD(tool)
Window Designs (WD)
Construction Process
Select Environment Menu Items
Select Packaging Menu Items
Select Generation Menu Items
Set environment parameters
Complete packaging
Set options for generation and installation
Create (generate) source code
Install source code or create remote files

Prerequisites

Several activities must occur before you can successfully generate source code for a local application or for remote files. These activities include:

- Completing appropriate diagrams
- Transforming the Data Model Diagram (DMD)
- Specifying environment parameters for the business systems
- Packaging load modules
- Selecting Generation options
- Creating Referential Integrity (RI) trigger modules

Complete Diagrams

The local encyclopedia, a workstation repository that stores models and subsets that are not checked into the Client Server or Host Encyclopedia, is 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 construction-level Consistency Check. These diagrams include the following:

- Data Model Diagram (DMD)
- Dialog Flow Diagram (DFD)
- Action Diagram (AD)
- Technical Design (TD)
- Screen or Window Designs (depending on the application)

CA Gen provides several ways to verify consistency:

- The Analysis and Design Toolsets include a Consistency Check option that you can use to check a model.
- The Construction Toolset lets you check consistency before you generate each component. This is done using Generation Defaults.

Transform Data Model

Before you can generate source code, you must first transform the DMD into a TD using the Design Toolset. The transformation process creates the data structure objects. These data structure objects are then used by CA Gen to generate database definitions that include the DDL statements necessary to allocate and construct database objects. This information is stored in the local encyclopedia on your workstation.

Transformation does not build the actual database. This is done during the database generation process. For more information about transformation, see *Toolset Help*.

CA Gen lets you generate the source code for an application before you build the database that the application references. You cannot, however, test your application until you build the database. For more information about building a database, see the chapter "Building a Database (see page 15)."

Specify Environment Parameters

Before you generate an application, you must specify its target environment. You can specify the environment parameters using the Environment Tool.

Follow these steps:

1. Select Construction.
2. Select Environment.
3. Select your target business environment.
4. Select Detail.
5. Select Properties.

Environment parameters belong to a business system. Different business systems within the same model can have different environment parameters. The parameters you set on the environment window are uploaded to the encyclopedia and are saved as properties of the business system.

The parameters you will set include the operating system, DBMS type, language, TP Monitor, Profile Manager, and screen type. You also have the option to clear the screen default, restart the application, enforce data modeling constraints, extend attribute support, optimize import view initialization, and select various MVS parameters.

Note: You specify the environment parameters for a client server application in the Client Environment and Server Environment using the Packaging Tool. For more information, see *Toolset Help.*

You can also specify environment information in the Generation Defaults window using the Generation Tool. The basic difference is that the environment parameters using the Environment Tool (for character-based or GUI applications) and the environment parameters using the Packaging option (for client/server applications) are meant to indicate the configuration where the application will run in production. The Generation Defaults window lets you override those selections for your workstation.

The Generation Defaults dialog also presents generation and installation options that are used to control generation activities. For more information about generation and installation options, see Creating an Executable Application (see page 60).

There are two situations in which you would want to set parameters again on the Generation Defaults window:

- For purposes of testing, you may temporarily override the environment windows and reset the parameters. The Generation Defaults window lets you do this. For more information, see Specify Generation Default Parameters.
- If you are generating RI trigger modules or DDL, both of which are model-level objects not associated with any one-business system, you must set the parameters on the Generation Defaults window.

Package Load Modules

Before you generate an application, you must package the load modules. For information about packaging, see the chapter “Packaging (see page 27).”

Specify Generation Default Parameters

You should review the Generation Default parameters before you generate or install an application. Some of the generation parameters can override the environment parameters.

The parameters you set on the environment parameters windows (in Environment and Packaging) can be checked into the encyclopedia and 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 retained from session to session by CA Gen.

For this reason, when you upload and download work to and from the encyclopedia, whether from the same model or another, the defaults you last used stay in effect. It is, therefore, important that you check these options before generating.

For more information about generation parameters, see Creating an Executable Application (see page 60).

Create Referential Integrity Trigger Modules

All RI trigger modules must be generated and installed before an application can be installed. Failure to do so can result in unresolved references when linking load modules.

RI triggers are automatically generated if they do not exist for the generated actions. Optionally, you can select RI triggers as one of the components to generate or install with the Code, Selected action.

Note: RI trigger modules are necessary for your application to work properly. RI trigger generation adds the source member names for the trigger modules to the model. The names are required for generating action blocks that call the RI trigger modules.

When generating RI triggers to create a remote file to be installed on MVS, the option Process modules marked for Compatibility must be selected if the RI triggers are to be used by modules marked for Compatibility. For more information about Process modules marked for Compatibility option, see Process modules marked for Compatibility (see page 64).

Creating an Executable Application

Generating and installing load modules is the process that creates an executable application. The application can then be tested. The following sections provide the information necessary to create the executable application.

Set the Date and Time

Before generating and installing an application, you must verify that the time and date are correct for your workstation.

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. You can set it in a command file, a shell script, or a batch file that starts CA Gen. For more information about setting environment variables, see the operating system documentation.

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.

Select Generation Default Parameters

Before selecting parameters, confirm the generation defaults last saved, for the following reasons:

- Some parameters can be set only using the Generation Tool.
- The specifications for the operating system, DBMS, language, and TP monitor can be overridden for a load module. Unless overridden, these specifications come from the environment parameters for the business system.
- The specifications for the operating system, DBMS, language, and TP monitor used for the generation and installation of Referential Integrity trigger modules or DDL can be set only from the Generation Defaults window.

Assuming you specify parameters on the Environment Parameters window, the only required selection for the generation and installation of a load module is the type of installation. Optionally, for a load module, you may want to override the operating system, DBMS, language, and TP monitor. Selecting an operating system determines what choices are available for the other three options.

Accessing the Generation Defaults Window

Follow these steps:

1. Select a business system.
2. Select Construction.
3. Select Generation.
4. Select Options.
5. Select Generation Defaults.

Setting Options for Local and Remote Applications

You can apply several options to the generation or installation of a load module. These options are as follow:

- Override Business System Target Environment Parameters with above defaults

  Specifies a different operating system, DBMS, language, or TP monitor than the ones set for environment parameters

- Run Consistency Check for each item generated

  Ensures consistency of the component you are generating.

- Generate source code with trace (GEN ALL)

  'Step through' your action diagrams during execution of the application

- Delete generated source after remote install

  Provides more storage space on a physical disk after you install your application

- Process modules marked for Compatibility

  Generates and/or builds non-DLL load modules migrated from releases of CA Gen before Release 7

Downstream Effects

Selecting the Delete Generated Source After Remote Install option may indirectly cause the installation of load modules to fail. When more than one load module uses the same common action block, the first load module installed contains the actual code. After this first module is formatted into a remote file, the source code is deleted from the code generation platform and all other remote files contain only the reference to this action block.

The remaining Generation options apply to database generation (for more information, see the chapter "Building a Database (see page 15)") and not directly to load module generation and installation, and thus, are not listed in the following sections.

Override Business System Target Environment using Above Defaults

Selecting this option means that the values in the drop-down list displayed on this window, for the operating system, DBMS type, source language, and TP monitor are used instead of the values stored as Environment parameters for this business system.

Unless you are certain that the environment parameter values for each business system are what you want, select this option.

Note: When you select the override option, you override the business system environment parameters previously set in Environment for character-based (block mode) and GUI applications. You also override the client and server environment parameters previously set in Packaging for client/server applications.

Run Consistency Check for Each Item Generated

This option verifies that each item generated is consistent before generation occurs. If the objects or model is not consistent, CA Gen specifies each inconsistency and indicates the level of severity (error or warning).

If you do not select this option, CA Gen does no checking before generation or installation.

Generate Source Code with Trace (Gen All)

Trace support allows Diagram Testing. It tells CA Gen to generate additional source code, which allows you to 'step through' the execution of the procedure-action diagrams, for your 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.

Note that this option has no effect if the Generation action is Code, Selected. During selective generation, the TRCE (trace) column lets you include Trace code only in the load module components you designate.

When you do not choose this option, you can still test your application, but you cannot view the underlying logic. The default choice is to not generate trace code.

Downstream Effects

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

Delete Generated Source After Remote Install

This option automatically deletes the generated source code after remote installation. The purpose is to provide additional disk space for other CA Gen models and other applications. The default choice is not to delete the source. If you select this option, you must regenerate the source code for a load module before you can reinstall the load module.

Note: When you are generating and installing multiple load modules, the source code is deleted after each compile, or at least after each install. If you are concerned about running out of space during generation, consider generating and installing load modules one at a time.

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 can either choose to have CA Gen automatically create all the components for you or you can choose the components to create.

The first time you install the application, we recommend the Code All and Install action that generates and installs all code. This action provides the quickest way to accomplish the installation.

Generating the Complete Application

If you choose to have CA Gen automatically create all of the components, you can either:

- Generate the source code for all components of the application, but not install the code
- Generate and install the source code for all components of the application

Note that the remote installation process actually creates the remote files. If you generate a complete application, the generation action Code, All and Install creates one remote file, for all Referential Integrity trigger modules, and one remote file for each load module in your application.

A complete application also has a database remote file that is created separately. For more information about building a database, see the chapter "Building a Database (see page 15)."

Generating Selected Components

If you choose to select the components yourself, first open one of the code options in the Generation Tool. For more information, see *Toolset Help*. After your choices are complete, select the Code, Selected action.

The Code, Selected action is inactive (or dimmed) until you open Online Code, Batch Code, Window Code, Cooperative Code, Proxy Code, Component Code, or DDL.

Note: Ensure that you generate the RI triggers once by selecting individual components when you generate or regenerate an application (the Code, Selected action).

Operations Library Considerations

You must generate and build an application with one or more operations libraries in a specific sequence.

If the application has operations consisting of common action blocks, you must generate and build the operations library you generated and build the load modules that reference it.

If the application references an operations library that is built from another model, you must build and specify the operations library in the Build Tool before you generate and build the load modules that reference it. The Build Tool profile entry set is OPT.CBDLIST located under Options. For more information about setting tokens, see the *Build Tool User Guide*.

Installing Code Generated with Other CA Gen Tools

You can also install source code that you generated from CA Gen Toolsets other than Construction. The generation action Install All Changes accomplishes this. This action is active only if you generate code from a Toolset other than Construction.

Verify Completion

You can verify the completion statuses of the generation and installation processes.

Status of Generation

You can verify completion of the generation process by viewing the messages that appear in the Generation Status window. The window displays columns of information about the generation process. They are as follows:

- Lines

  Displays the number of lines of source code generated per component.
- Status

  Displays the status of the component being generated:
  - In progress
  - Completed
  - Failed
- Type

  Displays the type of component being generated. For example:
  - Transaction Controller (Dialog/Window/Server/Batch Manager)
  - Install Deck
  - Procedure Step
  - Action Block
  - Triggers (Referential Integrity modules)
  - Screen
- Name

  Displays the name of the component.

The Status column on the Generation Status window indicates the load module components that complete or fail generation. The word *failed* appears if a component does not generate successfully. The Number Failed indicator shows how many components failed.

The Show command allows you to display the error file for a failed component. The error file contains the message or messages that explain why the component failed. The extension of the error file is .ERR. The file name is the name you previously specified in packaging or CASCADE for RI trigger modules.

Various messages may display if a component fails generation. If you perform a construction-level Consistency Check on your model and remove all errors before generation, the following conditions will not occur.

Entity (entity type name) not implemented in TD

The named entity type does not exist as a table in the TD. Review the DMD and TD for the missing entity type.

Relationship (relationship name) not implemented in TD

The named relationship does not exist as a linkage on the TD. Review the DMD and TD for the missing relationship.

No linkage: Relationship (relationship name)

This is similar to the message Relationship not implemented in TD, but indicates a more serious failure such as a corrupted model or possibly an internal CA Gen error.

Function (function name) not available in target SQL environment

You have used a CA Gen supplied function that is not allowed in the target database management system. Review the table of CA Gen supplied functions in Action Diagramming to identify which functions are allowed in which database management systems.

One of the inner nested repeating groups has a cardinality greater than the number of times it is placed on the screen

This message indicates that the cardinality of an inner nested repeating group must be the same as the number of times it is placed on the screen.

Failing current screen generation because no fields have been placed on this screen. Complete specification of this screen and retry generation.

Screens require the placement of one or more fields to be generated.

Results of Generating Local and Remote Applications

The following sections lists the subdirectories and files that are generated by CA Gen for character-based (block mode) applications and GUI applications (including GUI stand-alone and GUI clients) during application generation.

Subdirectories and Generated Files

CA Gen automatically creates subdirectories for the generated code. These subdirectories are subordinate to the directory created to hold the model. For generated source code, CA Gen creates a subdirectory named C, C#, Java, or COBOL. For generated proxy source code, CA Gen creates a subdirectory named proxy/c, proxy/c#, proxy/com, or proxy/java.

The different files that are created during generation are as follows:

- C

  Generated C source code without embedded SQL.

- C#

  Generated C# file.

- CTL

  Information used by CA Gen when creating a remote file.

- JAVA

  Generated Java source code.

- CLASS

  Java Byte code

- HTML

  Hypertext Markup Language

- JS

  Java Script

- CSS

  Cascading Style Sheets

- XML

  eXtensible Markup Language

- CS

  Generated C# code

- Config

  Configuration file

- JAR

  Java Archive

- EAR

  Enterprise Application Archive

- MSI

  Microsoft Installer

- VDPROJ

  Visual Studio .NET Setup and Deployment Project

- ICM

  Install deck information. The install deck is a set of instructions on how to install all procedure steps, screens, and action blocks for the load module. The install deck is in Standard Generalized Markup Language (SGML) format.

- RMT

  A remote file, which contains a concatenation of the ICM and source files.
- TXT

  Status messages issued during the generation of remote files.
- SQB, SQC

  Generate C and COBOL source code with embedded SQL statements.
- RC

  Generated window and dialog definitions for UNIX, and Windows.
- GML

  Generated online help information.
- H

  GUI header file. Contains IDs that are shared among the C, RC, and GML files.
- DEF

  DLL dynamic-link library specifications default file.
- TGT

  Build tool set-up file per module.
- CMD, BAT

  Install batch command files.
- MAK

  Make files.
- LIB

  Import library pointers to reference DLL.
- EXP

  Export files for DLLs.
- SSE

  OLE object information for load modules using OLE embedding.
- CBL

  Generated COBOL source code without embedded SQL.

Chapter 6: Application Testing

This chapter describes the various testing tools and methods used for testing character-based, GUI, and client server applications.

Diagram Testing

CA Gen gives you the ability to perform diagram testing. After you complete construction, you can perform end-to-end testing from data modeling to data storage for block-mode applications under UNIX, Linux, NonStop, Windows and TSO, GUI applications under Windows, Distributed Processing Client (DPC) applications under UNIX, Linux, Windows, and Distributed Processing Server (DPS) applications under z/OS CICS, UNIX, Linux, Windows, and NonStop.

You can view the logic behind your application while stepping through the application's execution one statement at a time. If the application does not perform as you expected, you can immediately identify the action diagram statement being executed. You can then alter this statement in the CA Gen software so that processing will perform as required.

The Distributed Processing Client and Diagram Trace Utility provide an interactive debug facility for clients and servers generated and constructed using CA Gen. An application developer can step through the generated application using CA Gen action diagrams.

Using trace-mode debugging, you can:

- View the execution of a load module.
- Stop the debugging process to examine or modify the contents of variable or loop counters.
- See transfers of control between statements.

The load module can be set to execute only one line at a time, so that the effect of calculations on variables can be reviewed. This is also the concept behind diagram testing. The CA Gen software enables you to test the action diagrams that you build with using diagram testing. Diagram testing does not show the execution of source code.

For more information on application testing for a particular platform, see the following documents:

- Mainframe

  z/OS Implementation Toolset User Guide for or

  Host Encyclopedia Construction User Guide

- UNIX and Linux

  UNIX and Linux Implementation Toolset User Guide
- Windows

  Windows Implementation Toolset User Guide
- NonStop

  NonStop Implementation Toolset User Guide

Chapter 7: Installing Local and Remote Applications

This section contains the following topics:

Install Process

Installation is the process that converts the source code into executable files (called load modules or operations libraries). Installation involves compiling, link-editing, and possibly binding the source code to a database. You specify under the Generation Default options in the Generation Tool whether the installation is local or remote.

Local Installation

A local installation is one in which the application executes in the same environment in which you generated the source code.

Remote Installation

When targeting remote environments, installation creates the completed remote file, which is moved to the remote machine. In other words, a remote installation is one in which the application source code is generated on one computer and then installed for execution on another.

For example, a remote installation for UNIX or Linux and Oracle, although generated on Windows, produces an application that accesses Oracle and can be installed under UNIX or Linux.

Remote installation begins using the same generation process as used for local installation. The generation process creates the source code.

However, after the actual code is generated, the installation process for remote is very different. Remote installation creates a set of remote files that contain all of the components necessary to compile and install the application on a target system. These remote files must be moved to the target system and installed in an environment supported by CA Gen before the application can be used.

Note: The term remote file has superseded the use of the term Implementation Package (IP). There may still be references to IPs within some CA Gen documentation, or on some screens. The terms are synonymous.

Remote installation requires the presence of certain CA Gen components on the selected target system.

The installation of a generated GUI application or client, of a client/server application on another workstation, is also considered a type of remote installation. Unlike the type of remote installation that generates remote files and requires the presence of a remote installation tool, remote installation for GUI applications and clients means transferring (or copying) generated application executables (.EXE files), dynamic-linked libraries (DLLs), and other required files to a directory on the target workstation.

Prerequisites

You must generate and install all RI trigger modules before you install an application. For more information, see the chapter "Generating Local and Remote Applications (see page 55)."

You must also complete several additional activities to successfully install an application:

Follow these steps:

1. Log on to the operating system (if required)
2. Start the Database Management System (DBMS)
3. Build the database referenced by the application
4. Complete any external action blocks used by the application

Prerequisite Details

The following section describe the prerequisites that must be satisfied before you install an application. Read each requirement carefully and execute the tasks requested.

Log On to the Operating System and Start the DBMS

Before installing an application, you must log on to the operating system if the operating system requires it.

The DBMS must be running before you can install a database. In most cases, you can start the DBMS on your workstation yourself. For more information, see the vendor DBMS documentation.

Build the Database

Before you can test an application, you must build the database that the application references. For more information about building the database, see the chapter "Building a Database (see page 15)."

Resolve External Action Blocks

External action blocks define the interface between CA Gen procedure steps or action blocks and handwritten subroutines. You use external action blocks to access subroutines that were created outside CA Gen. You must compile any external action blocks outside the CA Gen software. This must be done before you install the CA Gen procedure step or action block that uses the external action block. If not, you will receive an unresolved external reference error condition.

For more information about how to code external action blocks for use with a CA Gen application, see the chapter "Using External Action Blocks (see page 43)."

Note: The installation process accesses software other than CA Gen. Errors during installation can occur from these other software applications, such as compilers, or from the operating system itself.

Local Build Status

CA Gen summons the Build Tool as a separate session to handle the installation of an application. The Build Tool serves several purposes:

- Reports on the status of the installation
- Lets you test the application
- Lets you review a text file that summarizes the results of the build

The way to invoke the Build Tool is to install an application or database, or from the command line, type the Build Tool executable name. There can be only one Build Tool session at a time.

You can also use the Build Tool icon, which allows you to invoke the Build Tool manually. By manually invoking the Build Tool, you can install an application both locally and remotely.

The Build Tool runs independently of CA Gen, allowing you to continue working within applications, such as CA Gen, during build. You can even exit CA Gen if you want.

The Exit button lets you close the Build Tool window. However, until you are finished with your testing, you may prefer to minimize the window rather than close it. Otherwise, you lose access to the Test option on the Build Tool window. For more information about application testing, see the chapter "Application Testing (see page 71)."

Build Status Messages

Various status messages can appear during the building of an application. A representative list is contained in the table Build Status Messages.

If the installation fails, select the status line listing the error. Select the Review action to display a log of the installation. The log contains information about why the error occurred.

Note: The build process accesses software other than CA Gen. Errors during build can occur from these other software applications, such as compilers, or from the operating system itself.

Results of Installing Local and Remote Applications

The following table lists status messages that are generated by CA Gen for character-based (block mode) applications and GUI applications (including GUI stand-alone and GUI clients) during application generation.

The following messages represent only a subset of the actual messages that may exist, as messages can be added by using scripts.

Waiting

The generated item is waiting for build.

Installing

The build process is installing the components of the application.

Compiling (component name)

The build process is compiling the named component.

Binding SQL modules

The build process is binding the Structured Query Language (SQL) modules to a database.

Linking (load module name)

The build process is linking the components for the named load module.

Installing transaction codes

The build process is installing the transaction codes that have been defined for the procedures in the load module.

Complete: Ready for testing

The application has installed successfully and is ready for execution (testing).

Completed test. Ready to run.

The application has completed its execution and is ready for execution again.

Completed test with errors. Can retest.

The application failed to terminate normally, but you can select Test to try the execution again.

Awaiting installation: Exit test of duplicate load module

You tried to install a load module that is currently executing (being tested).

(End the execution of the load module that is being tested by pressing Ctrl+Home. CA Gen will automatically delete the duplicate load module from the Installation Window and proceed with the build of the new load module.)

Testing

A load module is currently executing (being tested).

Error: Review results for details

The build process was unable to install the load module.

(Select Review to browse a listing of the errors and problems encountered during the build.)

Error: Unable to create install command file

Either an invalid Install Control Module (ICM) file was generated or the file was not generated because of a generation failure.

Error: No trancode defined for (load module)

CA Gen cannot locate the requested transaction code in the AEENV file.

Subdirectories and Built Files

CA Gen automatically creates subdirectories for the built code. These subdirectories are subordinate to the directory created to hold the model. For installed source code, CA Gen creates a subdirectory named C.

The file extensions and contents of files that are built permanently (P) for assembly or temporarily (T) as artifacts for both character-based (block mode) applications and GUI (including GUI stand-alone and GUI client applications) are as follows:

Note: This is a representative list, only, of the most important resultant files.

- BND

  Information used to bind the file to a database and user. (P)

- ERR

  Messages resulting from a failed generation or installation. (T)

- EXE

  Executable code produced from a packaged load module. (P)

- LNK/RSP

  A listing of the object files and link libraries used in the link-edit process (a response file). (T)

- OBJ

  Object code produced by the compiler. (T)

- OUT

  A log of the activities that occurred during installation. This log, called a review file, can be viewed with a text editor or from the Install Monitor with the Review option. (T)

- DLL

  Object code for procedure steps and the RI file. (P)

- HLP

  Object code for the GML file (online help) for UNIX and Windows. (P)

- DEF

  DLL dynamic-link library specifications default file. (T)

- CTL

  Information used by CA Gen when creating a remote file. (T)

- MAK

  Make files. (T)

- PRJ

  Helps to compile project file - Windows and HP (T)

- SST

  OLE object data for Windows only (P)

- RES

  Compile RC files (T)

- REG

  Registration for Windows (P)

Chapter 8: Cleaning Up After Construction

This section contains the following topics:

Cleanup Process

After all construction and testing activities are completed for a particular project on the workstation, the cleanup process can begin. Cleaning up involves returning your model to the encyclopedia and removing all special files created for construction and testing of your model on the workstation. If you generated remote files for installation on a target system, cleaning up includes identifying these files for transfer to your target system.

Each step of workstation generation results in some sort of output. Some of these procedures produce output in the form of files stored on your hard drive. Other output occurs in the form of additions to existing files.

During generation, the definition of your application database is created. Source code is generated for each load module in your application, as are RI trigger modules.

If you install your application on the workstation and perform testing, then your application database is installed using the DBMS on your workstation. The RI triggers and load module code are compiled and linked (and bound to a database, if necessary). Entries are made into the AEENV file to make the new load modules accessible to the AEF for testing. Each load module (both local and remote) has an Install Control Module (ICM) file generated.

If you installed your application by creating remote files for transfer to another system, the definition of your database is copied into a remote file, as is control information from the ICM. The components for each load module are copied into a remote file along with the ICM that defines those components. Likewise, all RI trigger modules for an application are combined into a remote file.

When you have completed generation and testing of your model, you no longer need many of the files created by generation. These files should be deleted from your workstation so you can reuse the disk space. Additionally, deleting files avoids confusion if you work with the same model, or one with similar component names, later.

How to Clean Up

The following steps are part of the cleanup process:

Check In Model

When you complete generation and test for a particular model or subset, you need to check in your model. This returns your model to the controlled environment of the encyclopedia where it can be accessed for further development and maintenance.

The load modules and databases generated using a CA Gen model are independent of the model. The model can be checked in before testing occurs, although the model must be checked out again if changes are required as a result of testing.

Check in your model as soon as possible after work is completed on it. This ensures that your changes are recorded in the encyclopedia and allows others to access and use the model or subset.

When you check in a model, a read-only copy of the model remains on the workstation. You can delete that read-only copy to conserve space, or keep it for reference. Note that you can generate code from a read-only model, but you cannot save or check in any changes.

Transfer Remote Files

When you select remote installation, the Generate and Install processes create remote files. These files contain all the necessary elements to install an application on CA Gen equipped target system.

There are three different types of remote files created for each application. These are listed in the following table along with a description of their contents and location.

Type	**Number**	**Workstation Location**	**Naming Convention**	**Description**
Load Module	One or more per application.	Depending on the language selected for your application, in the C subdirectory under your model directory.	The name of your load module followed by the file extension .RMT	Application source code for a single load module.

Type	Number	Workstation Location	Naming Convention	Description
Database	Usually one per application.	In the DDL subdirectory under your model directory.	The name of your database followed by the file extension .RMT	DDL for an application database. For target system implementation, all load modules for a CA Gen application usually share a common database.
RI	One per application	Depending on the language selected for your application, in the C subdirectory under your model directory.	CASCADE .RMT	RI trigger source for an application. These routines implement referential integrity for the entire application.

The remote files need to be copied or moved from their location on the workstation to the target system for installation, further testing, and production execution.

After the load modules are copied to the target system, they can be deleted from the workstation, along with all the source components used to generate them. However, if you prefer, the source components can be stored on the workstation so that only the changed components need to be regenerated if testing on the target system reveals some problems.

Chapter 9: Troubleshooting

This section contains the following topics:

Creating and Installing a Database

Follow these steps:

1. The steps for creating and installing a database are as follows:
2. Ensure that you have specified the correct operating system and database management system (DBMS) to be used for implementation.
3. If you are creating a new database remote file for remote installation, 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.
4. If you are creating a database remote file for remote installation, specify not to qualify tables and indexes with your owner ID. This is useful only when testing an application on your workstation.
5. 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 the same as the user ID found in the generated DDL.
6. If you receive a Consistency Check error message during construction that relates to database structure inconsistencies, return to the Design Toolset and retransform your Technical Design (TD). For more information about retransformation, see *Toolset Help*.

Packaging

Follow these steps:

1. Flow definitions are independent of packaging options, except for client/server applications. When creating the Dialog Flow Diagram (DLG) for a client/server design, you do need to consider which procedures are packaged as clients and which are packaged as servers.
2. An application runs most efficiently if each load module contains procedure steps that are likely to be used together.

3. Packaging an entire application into one load module is desirable in some situations. However, it can result in a large load module that may be too large to fit in available memory and therefore cannot be executed.
4. If you want to package your application into multiple load modules, each must be accessed separately. However, you do not have to completely close one application to invoke another; you can minimize one load module into an icon and access another load module. This keeps both load modules in RAM for faster access, so the only limitation is the availability of memory.

Installing External Action Blocks

Follow these steps:

1. External action blocks are associated with a load module in the same manner as other CA Gen action blocks. When generation occurs for that load module, information identifying the action block is included in the resulting load module or remote file. However, the action block interface routine is not included in the resulting load module or remote file. It should be completed and compiled outside of CA Gen, preferably using the same compiler options as CA Gen .cmd and .bat files.
2. Modify the interface routine using any editor that can save files in the appropriate character-set format. The modified interface routine cannot contain any control codes or header information unique to the editor.

Note: When creating EABs outside of CA Gen, you must take care to follow defined naming standards, as duplicate names can cause errors. EABs generated by CA Gen have unique names.

Generating an Application

Follow these steps:

1. Ensure that you have specified the correct operating system, programming language, database management system (DBMS), screen format type, and teleprocessing monitor to be used for implementation.
2. Generate Referential Integrity trigger modules before generating load modules. Generating the RI triggers adds the source member names for the trigger modules to the model. The names are required when you generate action blocks that call the RI modules.

3. If you select Delete Source After Install when installing load modules, ensure that the load modules are installed in the same order in which they are generated. When more than one load module uses the same common action block, the first load module to be generated contains the actual code. After this first module is generated, the source code is deleted and all other load modules contain only the reference to this action block. If the load modules are installed in an order different from the order in which they were created, a load module that references the action block may be installed before the load module that actually contains the action block. In that case, the installation will fail.
4. If your application uses external action blocks and you are installing locally, compile all of them before installing the load modules, which reference them. The external action blocks are compiled outside of the CA Gen.
5. If your application uses external action blocks and you are installing remotely, note that external action block code is not included in the remote module. It must be located and moved to your target system separately. You must compile all external action blocks used by your application before installing the load modules that reference them. The external action blocks are compiled outside of CA Gen. They must then be moved to the location defined for them in your Target Configuration so that your application can find them.

Testing Your Application

Follow these steps:

1. The Diagram Trace Utility is usable only if you generated load modules with trace support. The resulting load modules contain all the additional logic necessary to allow the special trace functions to occur.
2. You can generate trace for a single procedure step, an entire load module, or your complete application.
3. The resulting load modules are much larger than they would be without trace. If space is a problem, consider generating trace selectively. During load module generation, you can generate the trace code only for the elements you want to test.
4. To use the trace functions during test, you must enable them before you access the AEF.
5. When you are instructed to press End while on a trace screen, be sure to use the end key assigned to your testing environment, not the one assigned to the application being tested.

If you generated the entire load module or application with trace, you can elect to turn trace off using the TRACE OFF command.

Appendix A: Load Module Structure

This appendix describes the load module structure for character-based and GUI applications and other load module components.

Load Structure

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

The road to creating a load module that is ready for execution in a specific environment can be described as having four stages:

1. Creating Procedure Steps and Action Blocks using the Action Diagramming Tool.
2. Creating Load Module Packaging definitions using the Packaging Tool.
3. Creating Load Module source code using the Generation Tool.
4. Creating Load Module object code using either the Generation Tool or one of the Remote Implementation Toolsets.

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

In the second stage, you build a list that defines how you group procedure steps and action blocks into programs. This is influenced by the characteristics of the target operating system. Some systems are more effective with fewer but larger load modules while other environments are more effective with more but smaller load modules.

In the third stage, source code is generated 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, monitor-specific constructs are included within the load module when it is installed. In the MVS environment, CA Gen supplied user exits in object form are link-edited into DLLS that are invoked during execution. In other environments, these user exits are linked to 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 CA Gen procedure step performs the function of the main control paragraph by issuing USE statements that invoke specific action blocks.

The term load module means the same thing in the IBM mainframe environment as it does to CA Gen. Other operating systems have their own terms for machine readable code that can be executed. Some of them include:

- Windows-executable file (.EXE)
- Java-Archive (JAR) file
- UNIX, Linux, and NonStop executable

Special Load Module Functions

CA Gen supplies more than 50 sophisticated algorithms that can be included as special functions in action diagrams. These are then subsequently included in load modules by the code generator. You can use these functions to manipulate strings, numbers, dates, time, and timestamps. You can also convert data from one domain to another.

Each algorithm is described in the Toolset Help for Functions.

Load Module Components

In general, a CA Gen generated load module contains:

- An application 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, some which can be customized for your environment.

  Note: For more information, see *Host Encyclopedia Construction User Guide* or *Windows Implementation Toolset User Guide* for Windowed Environments.

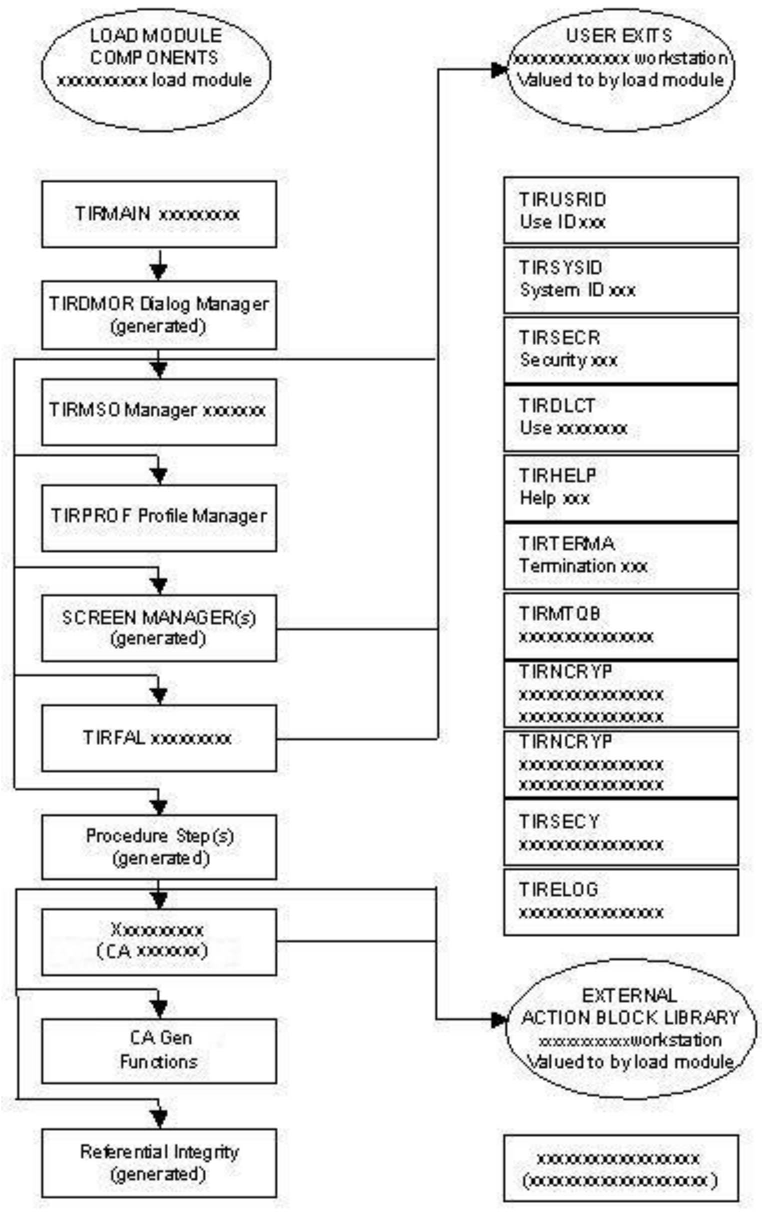
- (Optional) User-provided external action blocks

The only component not generated by CA Gen is the external action block.

The following sections show the structure of online load modules for character-based and GUI applications (including GUI stand-alone and client) and how the components work in a workstation environment.

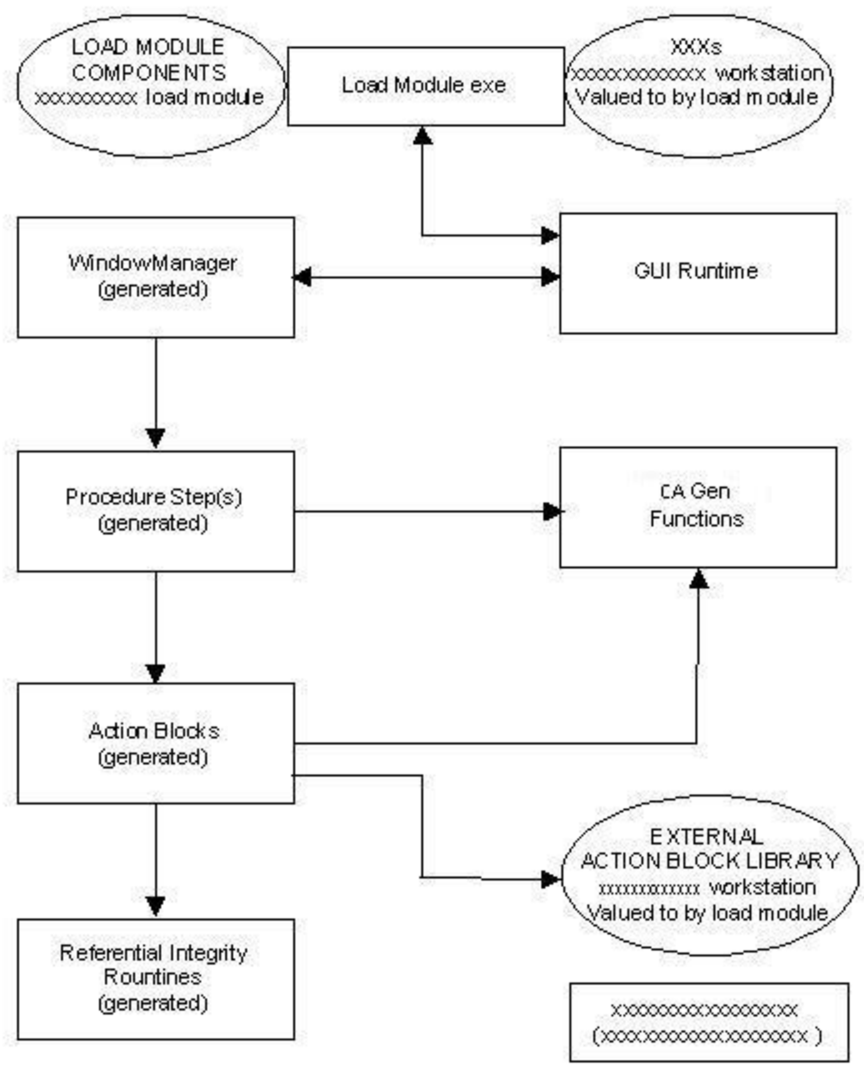
Load Module Structure for Character-Based Applications

The following diagram depicts the load module structure for character-based applications:


LOAD MODULE COMPONENTS xxxxxxxxxx load module
USER EXITS xxxxxxxxxxxx workstation Valued to by load module
TIRMAIN xxxxxxxx
TIRDMOR Dialog Manager (generated)
TIRMSO Manager xxxxxxx
TIRPROF Profile Manager
SCREEN MANAGER(s) (generated)
TIRFAL xxxxxxxx
Procedure Step(s) (generated)
Xxxxxxxxx (CA xxxxxxx)
CA Gen Functions
Referential Integrity (generated)
TIRUSRID Use ID xxx
TIRSYSID System ID xxx
TIRSECR Security xxx
TIRDLCT Use xxxxxxxx
TIRHELP Help xxx
TIRTERMA Termination xxx
TIRMTQB xxxxxxxxxxxxx
TIRNCRYP xxxxxxxxxxxxxx xxxxxxxxxxxxxx
TIRNCRYP xxxxxxxxxxxxxx xxxxxxxxxxxxxx
TIRSECY xxxxxxxxxxxxxx
TIRELOG xxxxxxxxxxxxxx
EXTERNAL ACTION BLOCK LIBRARY xxxxxxxxxxworkstation Valued to by load module
xxxxxxxxxxxxxxx (xxxxxxxxxxxxxxxxx)

Load Module Structure for GUI Applications

The following diagram depicts the load module structure for GUI applications:


LOAD MODULE COMPONENTS xxxxxxxxx load module
Load Module exe
XXXs xxxxxxxxxxxx workstation Valued to by load module
WindowManager (generated)
GUI Runtime
Procedure Step(s) (generated)
CA Gen Functions
Action Blocks (generated)
EXTERNAL ACTION BLOCK LIBRARY xxxxxxxxxxxx workstation Valued to by load module
Referential Integrity Rountines (generated)
xxxxxxxxxxxxxxxxx (xxxxxxxxxxxxxxxxxxx)

Application Manager

The Application Manager, which is generated during installation of each load module, acts as an executive to control the flow between procedure steps, to support terminal input and output, and to maintain execution context. The types of application managers are as follows:

- Dialog Managers and Screen Managers (character-based application)
- Batch Managers (batch application)

- Window Managers (GUI application and the client component of the client/server application)
- Server Managers (server component of a client/server application)

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

The Application Manager maintains execution context in a database called the Profile Manager. To accomplish this, the Application 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 Application 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 Application 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. The CA Gen provided runtime routine TIRMSG is called to invoke the other load module.

When a load module is executed, the Application Manager determines the procedure step to execute and populates the import view of the procedure step from screen input, of 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 concepts for GUI applications are the same.

Dialog Manager

The Application Manager is referred to as the Dialog Manager for character-based (block mode) applications.

For character-based (block mode) 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 CA Gen provided runtime routines to edit input and output, and write output buffers.

Batch Manager

Batch processing, in this context, is a capability on IBM mainframes. This information about Batch Managers, therefore, applies primarily to an environment that supports batch processing. The Batch Manager can be created only in COBOL.

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). For more information about Batch Manager, see *Host Encyclopedia Construction User Guide*.

CA Gen batch applications use three special files. They are referenced in the job control language (JCL) by the data definition (DD) names TIRIOVF, TIRMSGF, and TIRERRF. CA Gen accesses these special files through the CA Gen provided runtime routine TIRBTCH.

- 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 those 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 helps you to 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 execution profile 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 simply by 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 language.

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.

Profile Manager

The Profile Manager is a CA Gen provided runtime routine that manages the Profile database. The Profile database is a push-down, pop-up runtime stack that stores the export views of procedure steps. It is used to save information between procedure step executions, to support scrolling, hidden fields, links to other procedure steps, Help/Prompt requests, and application restarts. CA Gen maintains one stack per business system per user. If you are working in three business systems, you have three stacks.

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.

If your application is restartable, the capabilities of the Profile Manager let you interrupt a CA Gen generated application and return to it without losing data. You can interrupt a CA Gen business system to use another CA Gen business system or to perform activities unrelated to CA Gen generated applications. If your application is not restartable, the Profile Manager only accesses the Profile Database to support scrolling, hidden fields, links to other procedure steps, and Help/Prompt requests. For more information about specifying your application as non-restartable, see the *Toolset* Help.

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 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. Procedure steps for batch applications are actually job steps in JCL.

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 (DLG) 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.

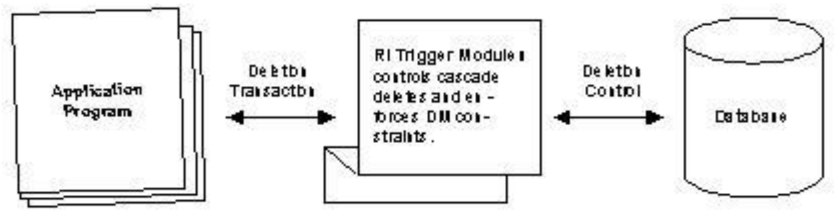
Action Blocks

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.

CA Gen 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, if both action blocks are generated for the same target environment.

Referential Integrity Trigger Modules

Referential Integrity trigger modules are CA Gen generated and implement the cascade delete logic necessary to enforce data modeling constraints specified in the Data Model. The Referential Integrity triggers are model-level objects. They are tied to the Data Model and the resultant database, not a specific business system. When you display code in the Generation Tool, the RI triggers appear on the top line, above the first business system. (The modules are called triggers because a delete can trigger the deletion of something else, which can trigger the deletion of something else, and so on.)


Application Program
Deletion Transaction
RI Trigger Modules controls cascade deletes and enforces DM constraints.
Deletion Control
Database

User-Provided External Action Blocks

An external action block is defined using the Action Diagramming Tool. For more information about defining an external action block using the Action Diagramming Tool, see the *Action Diagramming User Guide.*

An external action block requires just three components:

- Import Views
- Export Views
- The designation that it is external

An external action block contains no action statements of its own. The EXTERNAL designation tells CA Gen that a non-CA Gen generated program is used to obtain the data for the external action blocks export views.

Appendix B: Retrieving SQLCA Data

This section contains the following topics:

Process Continuation During Database Exception

CA Gen provides two action diagramming statements, WHEN DATABASE DEADLOCK OR TIMEOUT and WHEN DATABASE ERROR, that allow your application to continue processing when a deadlock, timeout, or other database exception occurs. If your application uses either of these database exception-handling statements, it must provide its own logic to roll back any previous updates and to ensure data integrity if processing continues.

For more information about using these statements in MVS applications, see the *Host Encyclopedia Construction User Guide*.

Important! The WHEN DATABASE DEADLOCK OR TIMEOUT statement will be inoperative for applications running under IMS on z/OS, because IMS automatically cancels and rolls back any transaction that encounters this type of error, without returning control to the application program.

External Action Blocks

CA Gen also provides source code for external action blocks (EABs) that allow applications to retrieve the SQLCODE and other data from the SQLCA after one of these WHEN DATABASE statements executes. Because the SQLCA data for each DBMS supported by CA Gen is different, a separate source member is provided for each DBMS. This source code is located in your CA Gen directory. Since each member functions as an external action block, you must define and package the external action blocks in your application model.

Using External Action Blocks

To use these EABs in your application, follow this procedure:

1. Open your model using CA Gen Toolset.
2. Select Design, Work Set List.

3. For the DBMS you are using, add a new work set named xxx_SQLCA, where "xxx" identifies the DBMS (Oracle, for example).
4. Add the following attributes to this work set, in the order shown in the table:

Work Set (one per DBMS)	Attribute Name	Length	Domain
DB2_UDB_SQLCA	SQLCODE	8	Numeric
	MESSAGE_LENGTH	8	Numeric
	ERROR_MSG	256	Text, mixed case, varying length
	SQLWARN	11	Text, uppercase, fixed length
	SQLSTATE	5	Text, uppercase, fixed length
	DBMS_TYPE	8	Text, mixed case, fixed length
MSSQL_SQLCA	SQLCODE	8	Numeric
	MESSAGE_LENGTH	8	Numeric
	ERROR_MSG	70	Text, mixed case, fixed length
	SQLWARN	8	Text, uppercase, fixed length
	SQLSTATE	5	Text, uppercase, fixed length
	DBMS_TYPE	8	Text, mixed case, fixed length
	DBMS_TYPE	8	Text, mixed case, fixed length
ORACLE_SQLCA	SQLCODE	8	Numeric
	MESSAGE_LENGTH	8	Numeric
	ERROR_MSG	70	Text, mixed case, fixed length
	SQLWARN	8	Text, uppercase, fixed length
	DBMS_TYPE	8	Text, mixed case, fixed length

Work Set (one per DBMS)	Attribute Name	Length	Domain
ODBC_SQLCA	SQLSTATE	5	Text, uppercase, fixed length
	SQLCODE	8	Numeric
	ERROR_MSG	512	Text, mixed case, fixed length
	MESSAGE_LENGTH	8	Numeric
	DBMS_TYPE	8	Text, mixed case, fixed length
JDBC_SQLCA	SQLSTATE	5	Text, uppercase, fixed length
	SQLCODE	8	Numeric
	ERROR_MSG	512	Text, mixed case, fixed length
	MESSAGE_LENGTH	8	Numeric
	DBMS_TYPE	8	Text, mixed case, fixed length
ADONET_SQLCA	SQLSTATE	5	Text, uppercase, fixed length
	SQLCODE	8	Numeric
	ERROR_MSG	512	Text, mixed case, fixed length
	MESSAGE_LENGTH	8	Numeric
	DBMS_TYPE	8	Text, mixed case, fixed length

5. Select Design, Action Diagram.
6. Create a new action block named GET_xxx_SQLCA with the default action block property settings, such that both the High Performance View Passing and Generate Missing Flags in Code options are enabled.
7. Add an export view to this action block containing the xxx_SQLCA work set, and all its attributes.

   Note: These must be in the order shown in the preceding table.
8. Add EXTERNAL as the only statement in the action block (other than any notes you want to add).
9. Open each existing action diagram that needs to retrieve the SQLCA data.

10. Add a USE GET_xxx_SQLCA statement after the appropriate WHEN DATABASE statements and match a local or export view to the EAB's xxx_SQLCA export view.

    Note: The USE statement does not have to be subordinate to the WHEN DATABASE statement, but the EAB will not work properly unless a WHEN DATABASE statement has actually executed first.

11. Select Construction, Packaging. Open a construction environment (Online, Batch, or Cooperative) that contains an action diagram that USES the new EAB.
12. Expand a load module that USES the new EAB.
13. Select GET_xxx_SQLCA, Detail, Properties.
14. Specify the GETyyyyy source member name that corresponds to your DBMS:
    - ODBC

      GETODBCS
    - DB2 for Windows, UNIX, LINUX

      GETDB22S
    - MS/SQL Server

      GETMSSQL
    - Oracle

      GETORACL
    - JDBC

      GETJDBCS
    - ADONET

      GADONETS
15. Save your model.

    You do *not* need to generate the external action block stub.

16. Copy source member GETyyyyy from the SQLCA samples under the samples directory beneath your CA Gen directory to the EXTRN directory that you use for other external action blocks.
17. Compile the source member and link it into the appropriate EXTRN.LIB for your application.

    The EAB source provided by CA Gen does not require precompilation or binding for workstation targets.

    Note: For more information about EAB, see the chapter “Using External Action Blocks (see page 43).”

18. Regenerate and reinstall any load modules that you modified to USE this EAB.

Modifying External Action Block Sample Code

You can modify the EAB sample code to exclude SQLCA data that you do not need, to return additional data, or to extend the length of the error message. If you do this, you must also modify the view definitions in your model to agree with your changes to the code. If you rename the EAB source member, you must also rename (repackage) it in your model.

Once again, this EAB must not be called except after a WHEN DATABASE statement has executed. The EAB uses the SAVE-SQLCA area in GLOBDATA to obtain the SQLCA data it returns. The generated application only saves the SQLCA in this area under the following conditions:

- A WHEN DATABASE statement executes, in which case the application continues processing.
- An exception of any kind occurs after a database call, and no WHEN statement for that exception was provided, in which case the application would terminate with a fatal error before the EAB could be called.

The SQLCA is not saved when a logical exception statement executes, or when a database call succeeds. So, GET_xxx_SQLCA cannot be used to retrieve SQLCA data associated with WHEN SUCCESSFUL, WHEN NOT FOUND, WHEN ALREADY EXISTS, WHEN NOT UNIQUE, or WHEN PERMITTED VALUE VIOLATION statements.

However, after a WHEN DATABASE statement has executed, the saved SQLCA for that statement is available from then on, unless replaced by the SQLCA for a subsequent WHEN DATABASE statement. So, this EAB may be used at any time to return the SQLCA for the last database exception, regardless of how many intervening database calls were made since that WHEN DATABASE statement executed.

Index