

CA Gen

Build Tool User Guide

Release 8.5



Third Edition

This Documentation, which includes embedded help systems and electronically distributed materials (hereinafter referred to as the "Documentation"), is for your informational purposes only and is subject to change or withdrawal by CA at any time.

This Documentation may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA. This Documentation is confidential and proprietary information of CA and may not be disclosed by you or used for any purpose other than as may be permitted in (i) a separate agreement between you and CA governing your use of the CA software to which the Documentation relates; or (ii) a separate confidentiality agreement between you and CA.

Notwithstanding the foregoing, if you are a licensed user of the software product(s) addressed in the Documentation, you may print or otherwise make available a reasonable number of copies of the Documentation for internal use by you and your employees in connection with that software, provided that all CA copyright notices and legends are affixed to each reproduced copy.

The right to print or otherwise make available copies of the Documentation is limited to the period during which the applicable license for such software remains in full force and effect. Should the license terminate for any reason, it is your responsibility to certify in writing to CA that all copies and partial copies of the Documentation have been returned to CA or destroyed.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in the Documentation is governed by the applicable license agreement and such license agreement is not modified in any way by the terms of this notice.

The manufacturer of this Documentation is CA.

Provided with "Restricted Rights." Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

Copyright © 2015 CA. All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

CA Technologies Product References

This document references the following CA Technologies products:

- CA Gen

Contact CA Technologies

Contact CA Support

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

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

Providing Feedback About Product Documentation

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

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

Document Changes

The following documentation updates have been made since the last release of this documentation:

- [Duplicate Filenames Ignored](#) (see page 117)-Added the topic.
- Updated the following topics:
 - [Environment Variables](#) (see page 18)
 - [Module Panel Fields](#) (see page 37)
 - [Script Names](#) (see page 79)
 - [Token Tables](#) (see page 85)
 - [Options](#) (see page 42)
 - [Using the Build Tool Server](#) (see page 19)

Contents

Chapter 1: Introduction	9
Build Tool Terms.....	10
Build Tool Functionality.....	10
How Build Tool Works.....	12
How to Build an Executable Application.....	14
How to Assemble an Application.....	15
Visual Studio Support.....	15
64-bit Windows Support.....	16
Chapter 2: Prerequisites for Building	17
Generate the Source Code.....	17
Install Third-Party Products.....	17
Environment Variables.....	18
Establish the Target Directory Structure.....	18
Chapter 3: Using the Build Tool Server	19
Security Considerations.....	20
How to Run the Build Tool Server as a Task on Windows.....	21
Chapter 4: Using the Command Line Client	23
Build an Application with the Command-Line Client.....	23
Invoke the Command-line Client.....	24
Command Reference.....	26
BUILD.....	26
CLEAN.....	27
EXIT/QUIT.....	28
HELP.....	29
SPLIT.....	29
STATUS.....	30
SHUTDOWN.....	31
Chapter 5: Using the GUI Client	33
Main GUI Client Panel (Module Panel).....	33
Toolbar and Menu Items.....	34
Entry Fields.....	37

Directory Tree	37
Module Panel Fields	37
Profiles	43
Testing	48
Assembling	49

Chapter 6: Build Scripts 79

Locating Scripts	79
Script Names	79
Tokens	82
Context-Specific Tokens	82
Token Delimiters	83
Customizing Scripts	83
Script Statements	83

Chapter 7: Profile Tokens 85

Token Tables	85
Database-Specific Tokens.....	102
Installation-Specific Tokens.....	103
Script Name-Specific Tokens	103
Optional Tokens	104
Location Tokens	106
GUI Application Tokens for C Applications.....	106
Additional Tokens for C Applications	107
Java Tokens	108
.NET Tokens.....	109
MSI Data Tokens	109

Chapter 8: Troubleshooting 111

Problems and Resolutions for all Supported Systems.....	111
Problems and Resolutions for Windows	111
Stream of Warnings	112
Error - L1093: ... object not found.....	112
Compile Step Failed #1.....	113
Models Will Not Build	113
Load Module Failed to Generate	113
Compile Step Failed #2.....	114
Error - Can't Find MSVCRXX.LL	114
Exception caught: No such file or directory	115
Error - Test FAILED statement in Message Panel	115

Error - ODBC BlockMode Application fails to start.....	116
Build Scheduling Status for All Modules Being Processed	116
Remote Host Not Available Error Message.....	116
Duplicate Filenames Ignored.....	117
Problems and Resolutions for UNIX, Linux, and NonStop.....	117
Unresolved Symbol	117
Command-Line Application	118
Output File Not Updated.....	118
Compile/Link Failed #1.....	118
Make: line too long	119
Make: don't know how to make <member-name>.pc	119
Exception caught: No such file or directory	120
Build Scheduling Status for All Modules Being Processed	120

[Index](#)

[121](#)

Chapter 1: Introduction

Running the Build Tool can be the last step in the process of turning the design in your model into an executable application. The Build Tool compiles that source code and prepares the code for deployment to a specific execution (target) environment. The Build Tool is an element of the Implementation Toolset, which also contains tools for testing and customizing applications.

Note: For more information about testing and customizing applications on Windows, UNIX, or Linux, see the *Windows Implementation Toolset User Guide* or the *UNIX and Linux Implementation Toolset User Guide*.

The Build Tool is written in Java and runs on Windows, Nonstop, UNIX, and Linux. The tool can create and assemble applications in preparation for deployment to supported environments.

Note: The NonStop platform supports the following databases:

- SQL/MX
- SQL/MP

The building of applications targeting SQL/MX are performed using the Build Tool, while applications targeting SQL/MP are built using the Setup Tool. For more information about using the Setup Tool, see the *NonStop Implementation Toolset User Guide*.

There are several ways to execute the Build Tool. The execution runs in three modes:

1. GUI client
2. Command-line
3. Server mode for doing batch jobs in the background.

All three modes perform similar operations. The GUI client on Windows provides an Assemble utility. Assembling is the packaging of applications that are built for redistribution. The Build Tool Server can be started either as a Windows Service, UNIX Daemon, or a Linux Daemon.

An important feature of the Build Tool is that you can control the building of applications across a network remotely from one client. For example, if you are running the Build Tool as a Windows client, you can control the building process on a UNIX or Linux system from your Windows client. Then you can simultaneously start a build on another system from your Windows client.

Note: This guide refers to a Windows client, UNIX client, Linux client, or Java client. All these terms refer to the same software.

The Windows client can be invoked directly from the Workstation Construction Toolset for Windows.

Important! The Workstation Toolset does not run on UNIX or Linux systems. The UNIX and Linux clients must be invoked manually.

Build Tool Terms

To use the Build Tool GUI Client, understand the following terms:

Build

Compiles and links the generated source code.

Clean

Deletes files resulting from building, allows a rebuild to occur.

Configuration

Controls the display of modules by sorting, including, and omitting.

Assemble

Packages applications that are intended for distribution.

Process Object

Requests to do something.

Profile

Collection of a customizable set of parameters that governs building of generated applications.

Split

Unbundles a Remote file (RMT file) and make its Implementation Control Module (ICM) file available for processing by the Build Tool.

Build Tool Functionality

CA Gen Build Tool includes the following functionality:

■ Options management

You can control the display of modules in the GUI client by sorting, including, and omitting attributes of modules being built. A configuration file stores these settings. You can perform the following tasks:

- Change the layout of the GUI client
- Select the default viewing tool for test results

- Set the default FTP utility
- Set build flags.

- Profile management

A Profile table stores the parameters that are used in building. You can have your own named set of profiles.

- Split

Unbundles RMT files to make the ICM file available to the Build Tool.

- Build

Compiles (and to link when applicable) the source code in the ICM files.

- Clean

Removes files that were created as a result of a build operation by the Build Tool. This functionality lets you build again. The files that are removed are as follows:

- .o
- .obj
- .lib
- .dll
- .exe
- .out.

- Review

View the results of a rebuild in a user specified ASCII editor.

- FTP

Transfer files in the client mode to a remote system in either ASCII or binary form using a user-supplied FTP program.

- Test

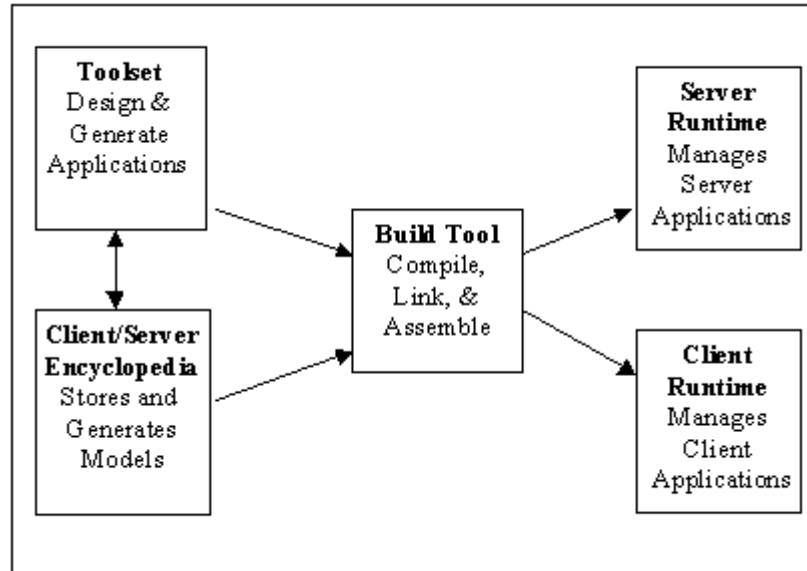
Execute C applications from either a command-line or interactively test with the Build Tool.

- Assemble

Bundle a built application that is intended for distribution.

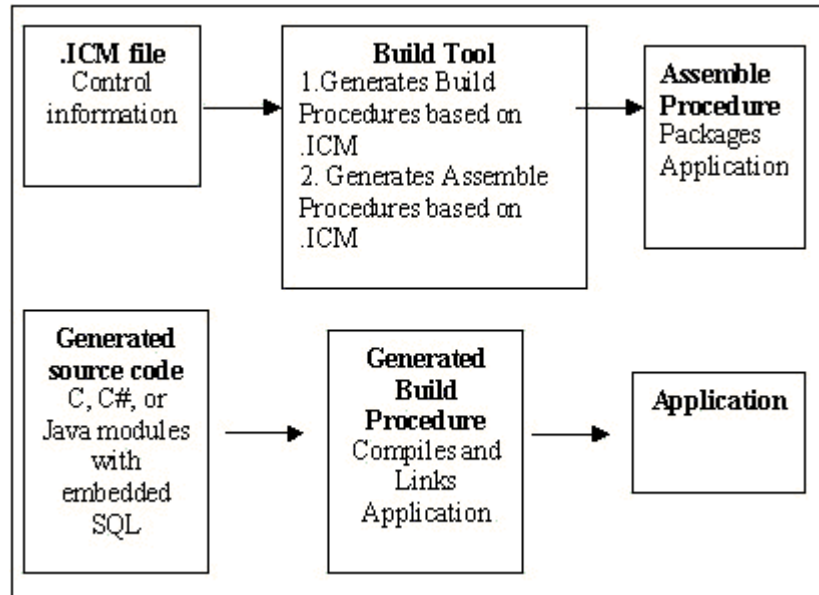
How Build Tool Works

The following illustration describes the relationship of the Build Tool with other CA Gen tools. The Workstation Toolset and the Client-Server Encyclopedia contain code generators that create source code from your model. These source modules are presented to the Build Tool for compiling and linking. This makes your business application available for execution. Separate runtime environments for the client and the server manage the execution of your application.

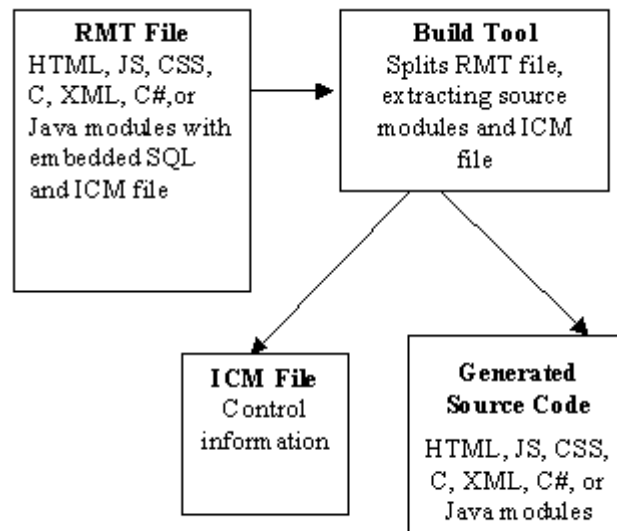


Construction creates a control file with an extension of ICM each time you generate a source code. The ICM file contains control information necessary to compile the source code and build it into an executable system. By reading the ICM, the Build Tool can generate a build procedure for the source code.

A special file called a Remote File (RMT) is created when the application is targeted to a machine other than the local machine. This is called a Remote Build as opposed to a Local Build. The RMT file contains the ICM file and all associated source files. This RMT file is transferred to the target system, where the Build Tool that is installed on the target system processes the RMT file and the ICM file. The following diagram illustrates the handling of the ICM file.



The Build Tool is a client server application and can compile and link applications that are either local or remote. In either case, the build process can be monitored and managed from the Build Tool client on the local machine, as shown in the following illustration.



How to Build an Executable Application

To assemble your business application, the bundled set of generated source modules must be converted to an executable code. The following steps describe the process of converting the generated source code modules to an executable code ready for assembling:

Note: In this scenario, the application is generated on a local Windows machine and the target system is a remote UNIX machine.

1. Start the Build Tool Client (GUI version).

Note: For more information about the Build Tool Client (GUI version), see [“Using the GUI Client”](#) (see page 33).

2. Start the Build Tool Server on the remote system. The Build Tool Server has no interface and work only when the client sends information.

Note: For more information about the Build Tool Server, see [“Using the Build Tool Server”](#) (see page 19).

3. From the Build Tool Client, connect to the Build Tool Server.
4. Transfer the RMT files using FTP to the remote UNIX or Linux system where Build Tool Server is running.

Important! Remote files that are transferred using FTP to UNIX or Linux systems must maintain the case they were generated in, for example, CASCADE.rmt. Altering the case of the generated remote files cause the UNIX or Linux Build Tool to fail.

5. Using the Build Tool Client, select the set of modules to build on the remote system by selecting the associated ICM or RMT file.
6. Build the selected ICMs. The ICMs are processed in sequential order. The Build Tool uses a set of build scripts and profile settings to generate a build procedure that is based on the information in each ICM.

Note: For more information about the build scripts, see [“Using the GUI Client”](#) (see page 33). For more information about the profile settings, see [“Profile Tokens”](#) (see page 85).

7. The generated build procedures are executed in sequential order to compile and link the application.
8. The Build Tool Client displays the build results. The Build Log can be reviewed to examine any build failures.

The Build steps described above can also be invoked using the command-line mode.

How to Assemble an Application

Assemble is the process of packaging applications for redistribution and is available only on Windows for use with GUI clients, Java, and C# applications.

Follow these steps:

1. Identify the set of modules to assemble by selecting the associated ICM from the Build Tool Client.
2. Modify the Assemble settings in the Assemble panel, which is used to configure your deployment bundle.

The ICMs are processed in sequential order. The Build Tool uses a script that is combined with your assemble settings to generate a procedure based on the information in each ICM.

The generated assemble procedure is executed to package the application. The Build Tool Client displays the assembling results.

3. Review the build log to identify any failures.

You now have a package (EAR or MSI) ready for you to deploy on your target environment.

Visual Studio Support

CA Gen supports compiling generated C applications on Windows using Visual Studio. The Build Tool token `OPT.VSVERSION` is used to set the appropriate Visual Studio compiler, and for this release defaults to Visual Studio. For more information about this token, see Appendix B.

The `%GENxx%Gen\VSabc` folder contains a collection of files that support Visual Studio. When setting token `OPT.VSVERSION` to Visual Studio, the shared libraries and executables in the `%GENxx%Gen\VSabc` folder are utilized. Executing a Visual Studio built application from the Build Tool does not require any environment changes (the Build Tool takes care of the environment).

Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.

More information:

[Profile Tokens](#) (see page 85)

64-bit Windows Support

CA Gen supports compiling and executing generated C Blockmode and server applications as 64-bit images on Windows using Visual Studio. Users can build their generated C Blockmode and server applications as 64-bit by setting the new Build Tool token, OPT.BITS, to 64 in their profile. Users must regenerate their code to gain access to 64-bit data types that are used in the Windows X 64 API to create 64-bit images. Regenerated code can be compiled into either 32-bit X86 binaries or 64-bit X64 binaries by setting the OPT.BITS Build Tool token.

The %GENxx%Gen\VSabc\amd64 folder contains a collection of files that support 64-bit Windows. When setting token OPT.BITS to 64, the shared libraries and executables in the %GENxx%Gen\VSabc\amd64 folder are utilized. Executing a 64-bit built application from the Build Tool does not require any environment changes (the Build Tool takes care of the environment).

Note: xx refers to the current release of CA Gen. abc refers to the supported version of Visual Studio. For more information about the release number and the supported version of Visual Studio, see the *Release Notes*.

Chapter 2: Prerequisites for Building

To build and assemble your CA Gen model as an executable application in a production environment, complete the following tasks:

1. Generate the source code
2. Install third-party products
3. Establish the target directory structure

Note: We cannot build a large executable application based on more than one model.

Generate the Source Code

Construction is the process that generates source code from your model definitions. This source code is compiled and made ready for assembling by the Build Tool. Construction also generates an Implementation Control Module (ICM) containing instructions for the Build Tool. If you select Remote Installation during construction, a Remote file (RMT) is generated that contains a concatenation of the ICM file and its load module source files.

Install Third-Party Products

Third-party products such as the JRE, compilers, databases, middleware, and third-party ActiveX or Web Controls must be installed before a build can occur. This includes both the machine where the Build Tool is installed and any target machines where you want to install the application. The only exception is the compiler, which is not required on the target execution machine.

For the complete list of required software, see the *CA Gen Technical Requirements* document in <http://ca.com/support>.

You can customize the database, compiler, and middleware settings that are used by the Build Tool by setting tokens in your user profile. For more information about tokens, see “[Profile Tokens](#) (see page 85).”

Environment Variables

The Build Tool uses the following environment variables:

Windows Systems

Variable	Description
PATH	Must include %GENxx%\Gen and %GENxx%\Gen\VS100
GENxxJRE	Points to installed JRE

Unix, Linux, and NonStop

Variable	Description
PATH	Must include \$IEFH/bin and \$IEFH/bt
IEFJRE	Points to installed JRE
IEFPLATFORM	Used to identify platform-specific requirements when running Build Tool

Establish the Target Directory Structure

By default, the directory used for all files that are created by the Build process is the directory that contains the generated ICM or RMT files from construction. The default directory can be changed by populating your user profile with a target directory structure of your choice before building an application. The Profile Tokens that are used to identify target directories are listed in “[Profile Tokens](#) (see page 85).”

Chapter 3: Using the Build Tool Server

The Build Tool server runs either as a system-started daemon or as a manually started task. The server mode of the Build Tool is designed to handle remote builds and communicate output views to its clients. The server receives requests from Build Tool clients running on Windows, UNIX, and Linux. These clients are not directly logged on to the server machine. This allows Build Tool client users to perform build functions across the network. However, the clients perform all the functions of the Build Tool on local machines without a server.

For UNIX and Linux platforms, the server mode can be started at boot time by adding the S99bldtool script to the /etc/rc3.d directory on UNIX and /etc/rc.d directory on Linux or adding this script to the Cron table. The Build Tool server opens a listening socket using the default port 7776. This listener port can be changed by adding a `-i nnnn` variable, where `nnnn` is a port number. If the listener port is changed, it must be changed in both the client and the server.

You can manually start the Build Tool in server mode, passing an unused port to communicate to the individual server. You can run a server for an individual user, separate from the system server. It is flexible and you can start the Build Tool multiple times, each using a unique port listener address.

The Build Tool Server is invoked manually using the following command syntax:

Windows

```
%GENxx%gen\bldtool.bat -c SERVER
```

Or

```
%GENxx%gen\bldtool.bat -c SERVER [options]
```

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

Unix, Linux, and NonStop

```
$IEFH/bt/bldtool -c SERVER
```

Or

```
$IEFH/bt/bldtool -c SERVER [options]
```

The options are as follows:

-i

Port number. This is the port number that will be used to connect Build Tool Clients to this server.

Default: 7776

There are several possible modes of the server. A Build Tool daemon could be started by the system at boot time and another Build Tool server started manually (batch). Clients use the daemon copy but can also access the batch copy by specifying the batch port. Starting the server as a batch job is most useful when there is no need for a full time server or you want to isolate your work on a remote platform.

The server receives requests from remote clients and queues them. The Build Tool server then scans the input queue looking for RMT files to split. The splitting of multiple remote files is run simultaneously.

After performing a split, the server looks for the highest priority task in the queue. The priority is as follows:

1. Building DDL
2. Building Referential Integrity (Cascade)
3. Building Operations Libraries
4. Build Load Modules

The build process is sequential for each model. If the build request contains more than one model, the builds are processed in parallel.

The FTP server on UNIX is used to confirm user ID and password for remote UNIX Build Tool server sessions. Ensure that the FTP daemon is running on the UNIX platform you want to access as a remote Build Tool server. However, the Build Tool Client login credentials are not checked when the Build Tool Server is running under the Windows platform.

Security Considerations

The client performs a login check whenever a new remote node is requested. If the login fails, a dialog message displays the error. After login, requests can be sent to the server for splits, builds, and to review results. Requests are evaluated for ownership on the remote host and appropriate error messages that are displayed if a request cannot be processed.

The default security on the Build Tool server is none. If you want to secure commands or paths, create a file in the IEFH/bt directory called user.security. Records in this file must be in the following format:

Userid or * followed by at least one blank followed by the command or path to be secured.

The * indicates that all users have access to the path or command.

The path security matches the path to the security rule, allowing the creation of specific rules.

Examples

```
User1 /aaa/bbb/ccd/dd
User2 /aaa/bbb/ccd/
User3 /aaa/bbb/ccd/eee
* /aaa
```

These rules are implemented as follows:

- User1 can access all directories and files under /aaa/bbb/ccd/dd.
- User2 can access all directories and files under /aaa/bbb/ccd/.
- User3 can access all directories and files under /aaa/bbb/ccd/eee, but not files under /dd.
- * /aaa means that all users can access any directories or files under /aaa.

If User4 /bbb/ccd is a profile, User4 would not have access to /aaa

To implement shutdown command

```
Userid/* shutdown
```

To establish security for a path on Windows

```
Userid/* c:\aaa\bbb
```

To establish security for a path on UNIX, Linux, and NonStop

```
Userid/* /aaa/bbb
```

How to Run the Build Tool Server as a Task on Windows

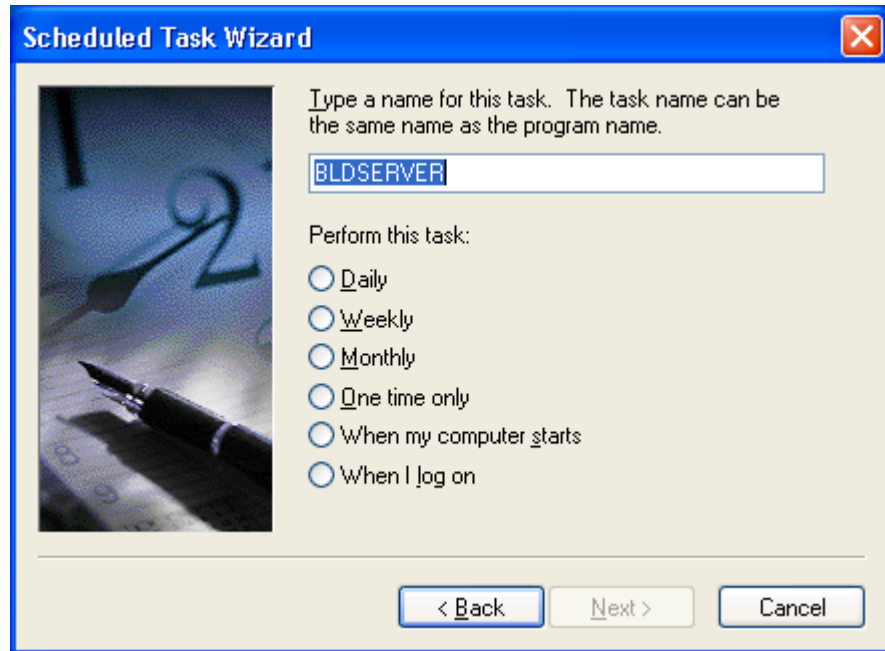
To run the Build Tool server on Windows as a started task, perform the following steps:

1. Go to Control panel, Scheduled Tasks.
2. Click the Add Scheduled Task and click next. Click browse, and traverse the directory tree to the CA Gen directory (the default is: %GENxx%\Gen).

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

3. In the File name field, enter Bldserver.bat and press the Enter key.

The following panel appears:



Under Perform this task, select one of the radio buttons and continue until you reach the Finish button. Clicking Finish schedules the server task to start as specified. If a command prompt appears with the server start, close the window and the server continues running.

Chapter 4: Using the Command Line Client

You can invoke the Build Tool as a Command-line Client. The Build Tool Command-line Client enables you to interact with the Build Tool from a command prompt, as opposed to the GUI client. Command-line usage (as opposed to GUI) is beneficial in the following cases:

- Batch command processing
- Working with terminal devices (no GUI available)
- Remote hosting from one server to another

After you have invoked the Command-line Client, you can key in the commands to perform various Build Tool tasks. These commands are not case-sensitive. Enter the commands in any combination of uppercase and lowercase characters. Embed Build Tool commands in scripts, which can then be used by an interactive user or executed as batch processes.

You can also call the Command-line Client with a set of parameters to perform a set of actions without entering the interactive mode. For a complete list of Build Tool commands, see [Command Reference](#) (see page 26).

Build an Application with the Command-Line Client

Building an application with the Command-line Client involves only a few commands.

Follow these steps:

1. Start the Command-line client.
2. Build the application.
3. Check status.
4. Exit.

Invoke the Command-line Client

The Build Tool Command-line Client is invoked using the following command syntax:

Windows

```
%GENxx%gen\bldtool.bat -c COMMAND
```

Or

```
%GENxx%gen\bldtool.bat -c COMMAND [options]
```

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

UNIX or Linux

```
$IEFH/bt/bldtool -c COMMAND
```

Or

```
$IEFH/bt/bldtool -c COMMAND [options]
```

There are two ways to invoke the Command-line Client, depending on whether you want to provide options:

- When no options are passed when invoked, enter the interactive mode of the Command-line Client. You can enter various commands, as described later in this section.
- When options are passed during the invocation, processing takes place based on these options, and control returns after the processing is completed. This is known as the batch mode.

The options when invoking the Command-line Client are as follows:

-c

Interface Type: One of COMMAND, CLIENT, or SERVER. Use COMMAND to invoke the Command-line Client.

Default: CLIENT

-a

Action to perform: BUILD, CLEAN, SPLIT, SHUTDOWN, STATUS, or HELP. This is a required option. Batch only.

Default: None

Note: SHUTDOWN is the only action that can be used for a remote Build Tool server.

-n

Specifies the ICM or RMT file to process.

Default: All ICM and RMT modules either in the current path or the path that is specified by the `-l` parameter.

-l

Specifies the path (location) used to locate the specified ICM or RMT file to process. The -l parameter must be delimited with double quotes if the path contains spaces. Use of "." (unquoted) is acceptable and is translated to represent the current directory. An environment variable, set to such a location, can also be used.

Default: None

-d

Specifies the dialect to use. If not present, the default is US English.

Default: None (US English)

Important! When building a generated application with dialects, do not modify the default profile manager target location tokens. They must remain set to MODELDIR.

-r

Specifies the name of a remote host machine (UNIX, Linux, or Windows) that is used to perform remote processing. The Build Tool server must be running on this machine before invoking a processing command. Parameters -u and -p are also required. This parameter accepts both IPv4 and IPv6 host names, and is used only for the SHUTDOWN action.

Default: None

-u

Specifies the User ID for the remote host machine referred to with the -r parameter. This parameter must accompany the -p parameter, and is used only for the SHUTDOWN action.

Default: None

-p

Specifies the password for the remote host machine referred to with the -r parameter. This parameter must accompany the -u parameter, and is used only for the SHUTDOWN action.

Default: None

-i

Specifies the port number for the Build Tool Server on a remote host referred to with the -r parameter, and is used only for the SHUTDOWN action.

Default: 7776

-f

Specifies the user profile to use when processing. This user profile must be previously set up using the Build Tool GUI Client and resides in the \$HOME directory (UNIX and Linux) or under the %USERPROFILE% directory (Windows). The profile file is case-sensitive.

Default: Default Profile

Use these options (in addition to -c COMMAND) to invoke the Build Tool Command-line Client through a batch command procedure. When invoked in this manner, the set of options is passed directly to the Build Tool and processing is initiated sequentially. After the processing is complete, the control returns to the calling command procedure.

Command Reference

If you provide only the -c COMMAND option, the interactive mode of the Command-line Client is initiated and the following prompt is displayed:

Enter Command

You are now in the command mode and can enter any of the following commands:

- BUILD
- CLEAN
- EXIT
- HELP
- QUIT
- SPLIT
- STATUS
- SHUTDOWN

The following sections provide a detailed description for each of these commands.

BUILD

The BUILD command submits a process request for one or more ICM and/or RMT files, either locally or remotely. If there are multiple files to process, the processing occurs sequentially. After you submit the BUILD command, control is returned to the interactive mode. You can submit additional BUILD commands for the local machine or for any of the remote machines that have active Build Tool Servers running on them. To retrieve status information about a submitted BUILD command, use the STATUS command. Exit the interactive mode after a BUILD command is submitted, and re-enter interactive mode later to check on the status of the build.

Usage

```
BUILD [-n filename] [-l path] [-d dialect] [-f profile_name]
```

Parameters

For more information about descriptions of the parameters used for the BUILD command, see [Invoke the Command Line Client](#) (see page 24).

Notes

If the `-n` parameter is not used, all ICMs and RMTs in the path are processed in the following order:

- DDL
Creates Database Tables
- CASCADE
Creates Cascade Library
- OPLIB
Creates Operations Library
- LM
Creates Load Module

Examples

```
BUILD -n P302.ICM -l c:\genmodels\coop07
```

Submits a process request for P302.ICM that resides in c:\genmodels\coop07.

```
BUILD -l c:\genmodels\coop07
```

Submits a process request for all ICM and/or RMT files, which reside in c:\genmodels\coop07.

CLEAN

The CLEAN command submits a clean request for one or more ICM and/or RMT files, either locally or remotely. If there are multiple files to process, the processing occurs sequentially. After you submit the CLEAN command, control is returned to the interactive mode. You can submit additional CLEAN commands for the local machine or for any of the remote machines that have active Build Tool Servers running on them. To retrieve status information on a submitted CLEAN command, use the STATUS command. You may exit the interactive mode after a CLEAN command is submitted, and may re-enter interactive mode later to check on the status of the clean.

The CLEAN command removes all produced files that are based on the outcome of the previous BUILD command. This includes, but is not limited to:

- Object files
- Library files
- Executable files
- Any intermediate files
- Make files
- Log files

Only the files that are split from the .RMT (if you are dealing with an .RMT file) or the files that are originally generated locally (if you are dealing with an ICM file) remain. When a CLEAN command is completed, the remaining files are left so that a full build can occur the next time when the BUILD command is issued against the same ICM or RMT file.

Usage

```
CLEAN [-n filename] [-l path]
```

Parameters

For more information about descriptions of the parameters that are used for the CLEAN command, see [Invoke the Command Line Client](#) (see page 24).

Notes

If the `-n` parameter is not used, all ICMs/RMTs in the path are cleaned.

Examples

```
CLEAN -n P302.ICM -l c:\genmodels\coop07
```

Submits a clean request for the produced files that are related to P302.ICM, which resides in c:\genmodels\coop07.

```
CLEAN -l c:\genmodels\coop07
```

Submits a clean request for the produced files that are related to all ICM or RMT files that reside in c:\genmodels\coop07.

EXIT/QUIT

The EXIT or QUIT command ends the Command-line Client in an interactive mode.

Usage

EXIT or QUIT

Parameters

None

Examples

EXIT

HELP

The HELP command displays the usage for the set of commands that are used with the Command-Line Client in an interactive mode.

Usage

HELP

Parameters

None

Examples

HELP

SPLIT

The SPLIT command submits a split request for one or more RMT files, either locally or remotely. If there are multiple files to split, the processing occurs sequentially. The SPLIT command returns a one-line status response message for the selected list of RMTs requested, and then returns the user to the interactive mode.

Usage

SPLIT [-n filename] [-l path] [-f profile]

Parameters

For more information about descriptions of the parameters that are used for the SPLIT command, see [Invoke the Command Line Client](#) (see page 24).

Notes

If the `-n` parameter is not used, all RMTs referenced using the `-l` parameter are split.

Examples

```
SPLIT -n P302.RMT -l c:\genmodels\coop07
```

Submits a split request for P302.RMT, which resides in c:\genmodels\coop07.

```
SPLIT -l c:\genmodels\coop07
```

Submits a split request for all RMT files that reside in c:\genmodels\coop07.

STATUS

The STATUS command submits a status request for one or more ICM or RMT files, either locally or remotely. If there are multiple files to process, the processing occurs sequentially. The STATUS command returns a one-line status response message for the selected list of ICMs or RMTs, and then returns you to the interactive mode.

Usage

```
STATUS [-n filename] [-l path]
```

Parameters

For more information about descriptions of the parameters that are used for the STATUS command, see [Invoke the Command Line Client](#) (see page 24).

Notes

If the `-n` parameter is not used, all ICMs/RMTs in the path are requested to provide status.

Examples

Enter Command:

```
STATUS -n P302.ICM -l c:\genmodels\coop07
```

Submits a status request for P302.ICM, which resides in c:\genmodels\coop07.

Enter Command:

```
STATUS -l c:\genmodels\coop07
```

Submits a status request for all ICM and/or RMT files that reside in c:\genmodels\coop07.

SHUTDOWN

The SHUTDOWN command terminates a remote server. Include a port number for the remote host if the port number is not the default.

Usage

```
SHUTDOWN [-r remote_host] [-u userid] [-p password] [-i port]
```

Parameters

For more information about descriptions of the parameters that are used for the SHUTDOWN command, see [Invoke the Command Line Client](#) (see page 24).

Example

```
SHUTDOWN -r unix01 -u user01 -p user01
```


Chapter 5: Using the GUI Client

This chapter reviews the terminology that is associated with Build Tool functions and describes the panels and procedures you use when invoking the Build Tool from the Graphical User Interface (GUI) Client.

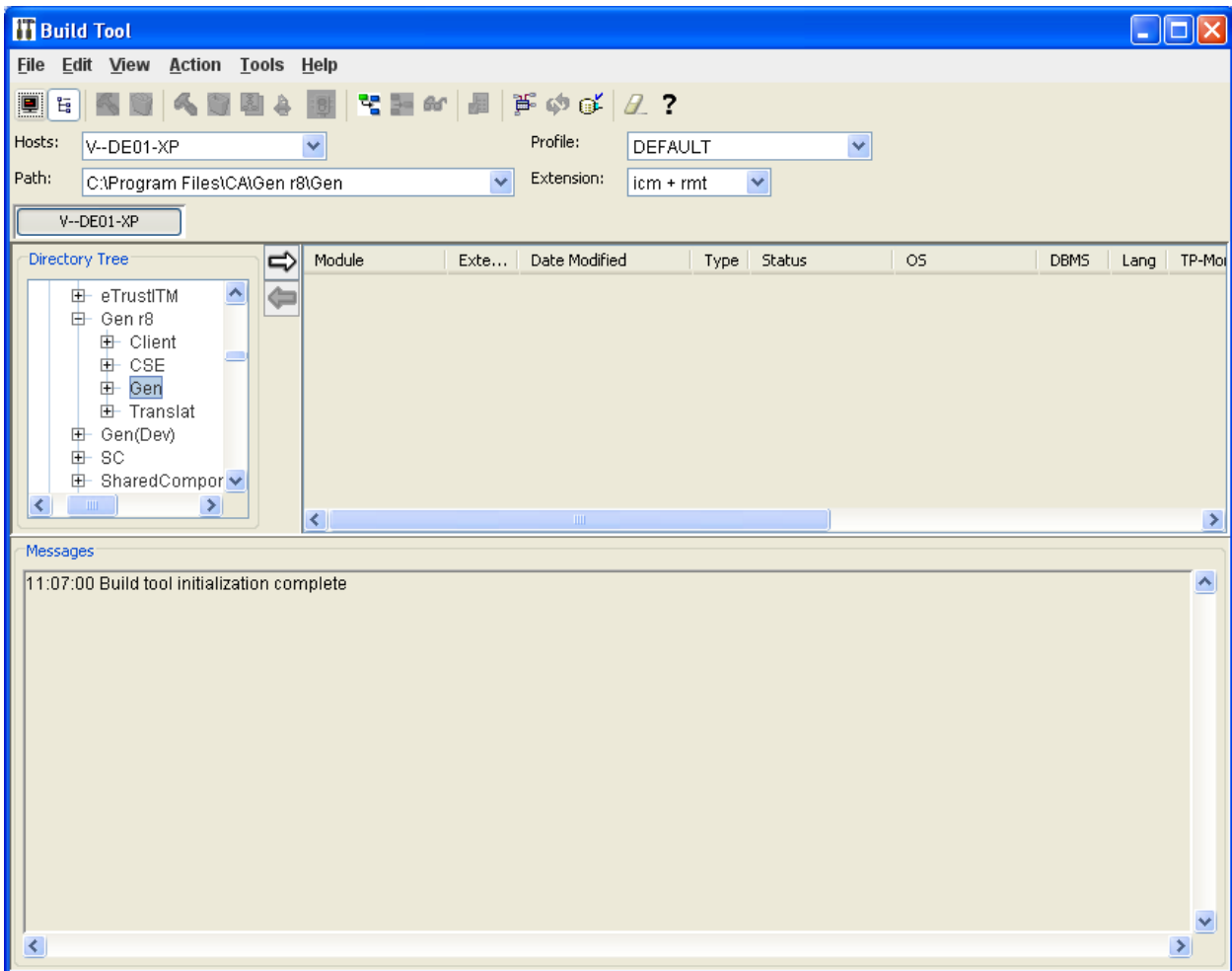
Note: The GUI Client is not supported on NonStop.

Main GUI Client Panel (Module Panel)

The main Graphical User Interface (GUI) Client panel contains the following:

- Menu bar
- Toolbar
- Entry fields
- Directory tree
- Module panel
- Messages panel











The menu bar contains a superset of Build Tool controls available as Toolbar icons. The following screen shot displays the Main GUI Client panel.










Toolbar and Menu Items

The following table lists and describes Build Tool GUI Client menu items and the equivalent toolbar icons:

Menu Item	Description	Toolbar Icon
Edit, Unselect All	Unselects all modules on the Module panel.	N/A
Edit, Select All	Selects all modules on the Module panel.	N/A

Menu Item	Description	Toolbar Icon
Edit, Clear Message Area	Deletes Message panel text.	
View, Full Screen View	Displays Module panel in full screen mode and removes Directory Tree	
View, Tree View	Displays Directory Tree to left of Module panel.	
Action, Build All	Builds all modules on the panel.	
Action, Clean All	Runs Clean for all ICM files displayed to delete all the modules created by these ICMs. RMTs are not Cleaned.	
Action, Build	Compiles and, where applicable, links source code generated by the Construction Toolset, and prepares the modules for Assembling.	
Action, Clean	Runs Clean for selected ICM files.	
Action, Split	Splits selected RMT files. It unbundles an RMT file to make its ICM file available to the Build Tool. The Build Tool also looks for RMT files when builds are requested and determines whether the RMT must be Split.	
Action, Assemble	Bundles applications for distribution. Each of the three types of applications that can be assembled (Java, GUI, and .NET) has its own dialog. Assemble is disabled for modules not yet built successfully. Assemble is not available on UNIX or Linux. You cannot assemble Web Service Definition modules by themselves; they must be packaged as part of an application.	
Action, Cancel	Cancels the execution of a build process.	

Menu Item	Description	Toolbar Icon
Action, Review	<p>Presents the results of a build. It creates an ASCII file (.out) that is a log of the build processing. You can choose one of three ASCII editors for viewing the results from the Review Editor drop-down list of the Options Utility. The defaults are Notepad for Windows and vi for UNIX and Linux. WordPad is also available for Windows.</p> <p>The Review button is available when a build step is complete. Selection of multiple ICMs produces multiple windows, one for each file.</p>	
Action, Test	<p>The Test option button remains disabled if the selected module has not been successfully built. Remote program testing is not supported.</p> <p>If the program has been built, the Load Module Test dialog appears. Enter the Tran code to be tested and any clear screen input to begin the test.</p>	
Action, Add Module	Adds all the ICMs and RMTs found in the selected directory. The suffix setting is checked to determine which files to display.	
Action, Remove Module	Removes selected files from the display area.	
Action, Disconnect Remote Host	Disconnect selected remote Host. This action is only enabled when remote host is in focus.	
Tools, Profile Manager	Creates, deletes, and edits user profiles.	
Tools, File Transfer	Transfer files using FTP.	
Tools, Options ...	Configures GUI client displays.	

Entry Fields

The entry fields below the toolbar that allow you to enter Host, Path, Profile, and Extension information are as follows:

Host

The network name for the computer you are using. The list contains one local host and can contain multiple remote hosts. These names are stored in the configuration properties file. As you connect to more Hosts, their names also appear below the Path drop-down list. You can switch from Host to Host by clicking a host name or selecting the name from the Host drop-down list. This field accepts both IPv4 and IPv6 hostnames.

Path

A fully qualified directory name. The current path is displayed and up to ten previous paths can be stored. The last path that is used is saved in the configuration properties file for use at the next Build Tool startup.

Profile

A default profile and all user profiles are kept in the configuration properties file. Profiles provide a means of storing a set of customizable parameters that govern building of generated applications

Extension

Refers to the three-character suffix at the end of a file name. By setting the Extension, you see only the files you want-ICM or RMT or both.

Directory Tree

On the left side of the panel of the module, the Directory Tree reflects the file system of the active host. Tabs are added as and when remote hosts are added. Each host directory tree, path, and the selected model are maintained until the Build Tool terminates.

Module Panel Fields

The Module panel lists the files available to build and includes the following fields:

Module

Specifies the ICM or RMT file name. The model name appears on a separate line above the files from that model. The PStep Interface Designer in CA Gen Studio generates the Module names with the .WSD suffix.

Extension

Identifies the file type as either ICM or RMT.

Date Modified

Identifies the creation date and time of the ICM or RMT file.

Type

Specifies file type that is represented by the ICM or RMT file. Possible values are:

DLL

Database

RI

Cascade

PR

Proxy

IC

Java

OL

Operations Library

LM

Load Module

SRVR Router

Web Service Definition Router

Status

Specifies the current status of the file. The possible values are:

New

No action has been performed on this file

Clean-Scheduled

Files scheduled to be cleaned

Cleaning

In the process of being cleaned

New-Cleaned

Scheduled files have been cleaned

Selected

File has been identified for action

Split-Scheduled

RMT file that is scheduled for splitting

Splitting

RMT file is splitting

Split Complete

Split process has completed

Split-FAILED

Split process has been unsuccessful

Building

File is in the Build process

Build-OK

File has been successfully built

Build-FAILED

File has not been successfully built

Build-Scheduled

In queue to be built

Assembling

Selected files are in the process of being assembled

Assemble-OK

Files have been successfully assembled

Assemble-FAILED

Assemble process has been unsuccessful

Cancel-Scheduled

Scheduled canceling a process

Cancel

Cancels the execution of the build process

OS

Specifies the operating system that is chosen during construction. The options are:

- UNIX
- Linux
- Windows
- CLR
- JVM
- Nonstop

DBMS

Specifies the database that is chosen during construction. The options are:

- <NONE>
- Oracle
- DB2 z/OS
- DB2 UDB (UNIX, Linux, and Windows)
- MSSQL (Windows only)
- ODBC/ADO.NET (Windows only)
- JDBC
- SQL/MX

TP-MON

Specifies the execution environment. The options are:

- Com
- Comp_Services (Component Services)
- EJB (Enterprise Java Beans)
- IEFAE
- Internet
- NET
- ASP.NET
- Java VM (Java Virtual Machine)
- Web Services
- Windows
- Pathway

Lang

Specifies the language. the options are:

- C#
- CS
- C
- Java

Path

Specifies the path that contains the displayed ICM or RMT.

Module Panel

The Module panel displays the modules ready for Build Tool processing.

Populating the Module Panel

The main panel always appears with a directory tree in the left panel. Using the menu or toolbar, you can replace the directory panel with a Full Screen View. By doing so, the directory tree goes hidden.

Follow these steps:

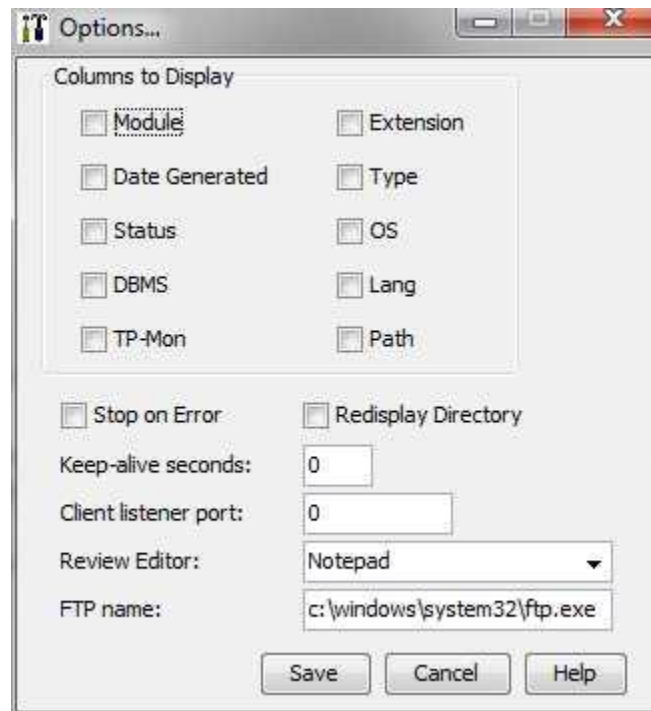
1. Select the model in the Tree View or Path directory.
2. Click the right arrow, use Action, Add to Module panel, or right click the node and select Add Module from the pop-up menu.
3. The model name appears on a separate line.
4. Selecting the model name selects all of the files in the model.

Display Full Screen

You can expand the Module panel to display the full width of the screen. This closes the directory and search panels. To display full screen, select the icon or click View, Full Screen View.

Options

To change the layout of the Module panel, choose either the Options icon or Tools, Options. This displays the Options dialog, which allows you to include or omit fields.



The description of the fields in the Options dialog are as follows:

Columns to Display

Set of checkboxes that represent the columns that are displayed in the Module Panel.

Stop on Error

Terminates ICM processing when a build error is encountered.

Redisplay Directory

Restores the Module panel to the last known environment. The redisplay uses the last known Path, Extension, and Profile to determine which models to display.

Keep-alive seconds

Specifies the time in seconds for a thread to poll any remote host session at a specified interval. The values can range from 1-900 seconds. The default is zero seconds.

Client listener port

Specifies a single port or port range when you have a Firewall. If you specify _0 or leave the default, the Build Tool Client picks up an available port. If you specify a port number or range of ports, the Build Tool tries using these ports for the Client Side Listener.

Review Editor

Specifies the editor that is used for viewing output from builds. The options are:

- Notepad
- WordPad
- UNIX/Linux vi editor

FTP name

Allows you to specify an FTP application to call when you click the File Transfer icon.

Moving Module Panel Fields

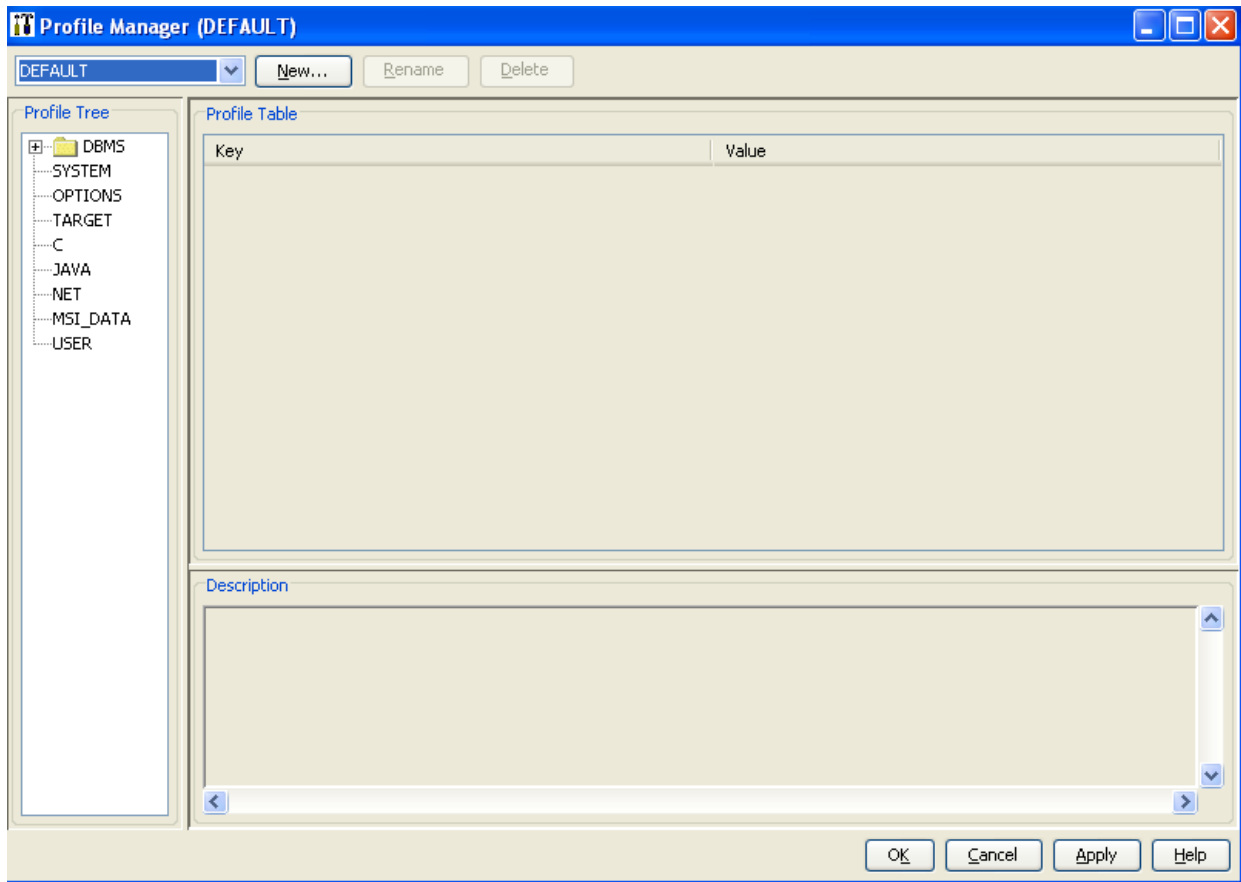
To change the position of the fields, use your mouse to move the fields left or right. The changes are saved and used in future displays.

To expand or contract fields, place the cursor on the vertical line in between fields. Moving the cursor to the left or right collapses or expands a field.

Profiles

Profile Manager allows you to create, edit, or delete your own unique profiles. A profile consists of a collection of settings to tailor application builds to your specifications in the form of tokens consisting of a key and a value that is assigned to the key. The build scripts use tokens and the tokens are substituted with the value assigned when the .mak file is generated at build time.

To open the Profile Manager, click the Profile Manager icon on the Toolbar or select Tools, Profile Manager. The Profile Manager dialog appears.



The Profile Manager window opens with the currently active profile selected. A different profile can be selected from the drop-down in the upper left corner.

The Profile Tree allows you to select the profile categories:

- DBMS
- SYSTEM
- OPTIONS
- TARGET
- C
- JAVA
- NET
- MSI_DATA
- USER

USER is reserved for key values that are created for use with a user-modified script. You can expand the tree by clicking the + plus sign next to a category.

To change the database options, click the + sign next to DBMS in the Profile Tree; the tree expands to display supported databases. Any of the key values can be changed and saved in a profile.

There exists a simple hierarchy of profiles. The default profile is the one delivered with the Build Tool, and contains the default setting for all tokens. All users of the Build Tool use the default profile.

Note: The default profile can be modified, but that changes to the default profile affect all users. Once modified, a default profile file resides on the disk representing any changes that are made to the delivered token values.

A user can create many user-specific profiles. Each of these profiles is saved in the personal directory of the user, cannot be used by other user, and supercede the default profile. The superceeding of tokens is on a token by token basis, starting with the delivered token value, then possibly superceded by the modified default profile token value, and then possibly superceded by the modified user profile token value.

For example, the delivered default value of the token on UNIX is the following:

```
OPT.CMD_FORM = NO
```

The modified value of this token in the default profile is the following:

```
OPT.CMD_FORM = YES
```

The modified value of this token in the user profile being used for this particular build is the following:

```
OPT.CMD_FORM = NO
```

The token OPT.CMD_FORM is initialized to NO as part of the delivered Build Tool token values (on UNIX). Since the default has been modified to YES, this value will supercede the initial value. And since a user-specific profile is being used which has the same token that is modified to NO, this will supercede the token's value once again. The net result is that the OPT.CMD_FORM token is set to NO for the current build activity.

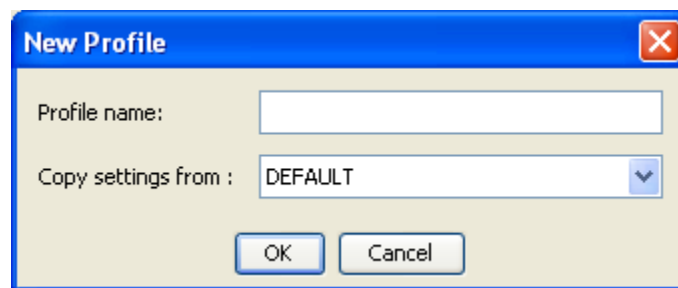
The selected profile is the basis for any new profiles. A profile that is created with a user ID is not shareable with other users. The default profile must be set up by the installer of CA Gen or the system administrator. Changes to the default profile are saved in the Build Tool directory and shared by all users.

When the USER category is selected, cells in both the Key and the Value columns are editable. When any other node is selected, only cells in the Value column are editable. You can edit the values by placing the cursor in the corresponding cell.

When the User node is selected, the last row of the table is always a new line with the <New Key> text in light grey in the Key cell and a blank value cell. You can add a new custom key by entering a name in the Key cell. After the focus shifts from the cell, the corresponding Value cell becomes editable, and another blank line is added as the last line in the table. This process is repeated every time that a new key is entered. Any custom key can be deleted by deleting its name from the Key cell. After the focus leaves an empty Key cell, the entire row is deleted from the table.

Click OK to save the changes and close the window. If you click Cancel, you are prompted to save or discard any unsaved changes before the window is closed.

To create a new profile, click New. The New Profile dialog appears:



Provide a unique name for the new profile. Only alphanumeric characters, with '-' (hyphen) and '_' (underscore), are acceptable for a profile name. An invalid profile name results in the following error message:

Profile name contains invalid characters

You can base the new profile either on the default profile or on one of the custom profiles. Click Save and a new profile is created.

To delete a profile, select the profile name in the combo box in the Profile Manager window and click Delete.

Note: The default profile cannot be deleted.

Format of a Build Tool Profile File

When a Build Tool Profile is modified (either the default profile or a user profile), only the modified tokens are saved to disk.

The file is formatted in token/value pairs, using the following format:

```
<PROFILE_NAME>.<PROFILE_TREE_SECTION>.<PROFILE_SUBTREE_SECTION>.<KEY>=<value>
```

Note: All data to the left of the '=' must be in uppercase. The value to the right of the '=' can be mixed case.

All keys have a preface of *one* of the following:

OPT.

This preface is for option tokens.

LOC.

This preface is for location tokens.

The following example shows a profile file format for the OPT.DBUSER option key set to dbuserid in the MYPROFILE profile with the DBMS profile section tree name and the ORACLE subtree section name

```
MYPROFILE.DBMS.ORACLE.OPT.DBUSER=dbuserid
```

Note: For more information about the list of available tokens, according to their profile tree section and subtree section, see the appendix "[Profile Tokens](#)" (see page 85)."

The location of the default profile file is a fixed location. You can find the file that is named System.profile in the following location:

- For Windows:

%ALLUSERSPROFILE%\CA\Gen xx\cfg\buildtool

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

- For UNIX and Linux:

\$IEFH/bt

You can find the user-specific profile files, named using the format user.<PROFILENAME>.profile, in the following location:

- For Windows:

%USERPROFILE%\AppData\Local\CA\Gen xx\cfg\buildtool

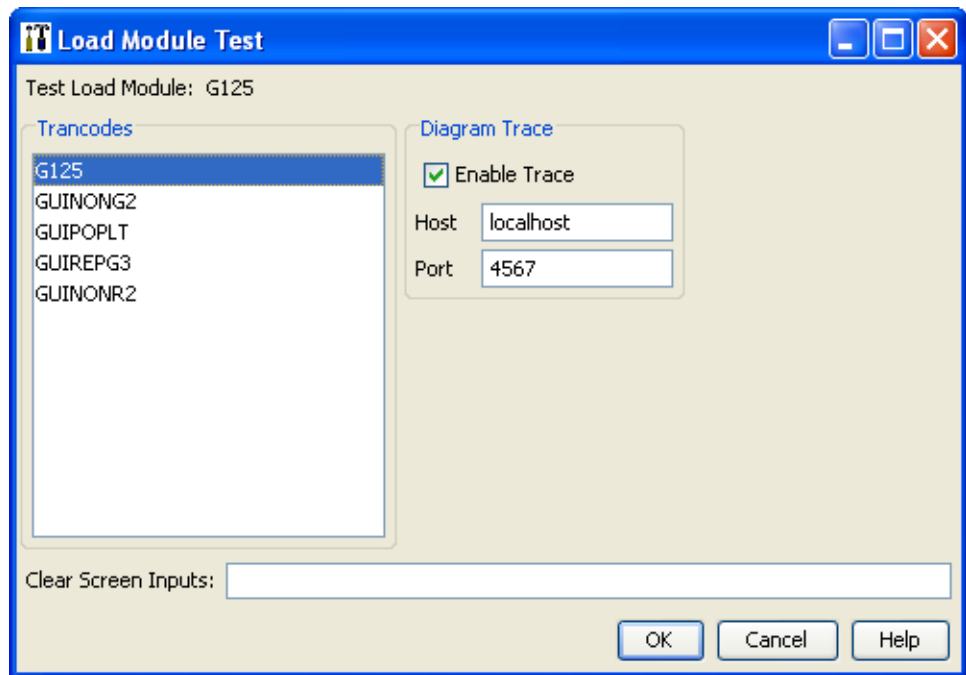
Note: xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.

- For UNIX, Linux, and NonStop:

\$HOME

Testing

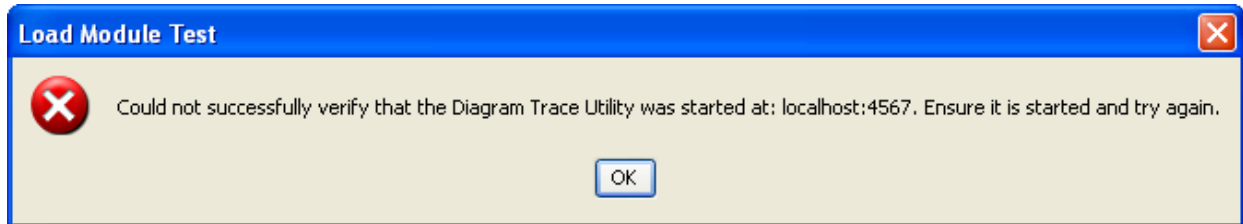
As mentioned earlier, you can use the Test button to execute locally built C applications through the Build Tool. When invoked with a selected load module, the Load Module Test dialog appears. You are provided with a list to select the Trancode and an entry field for Clear Screen Inputs to begin the test. In addition, you can select more options to use the Diagram Trace Utility to trace the application.



Follow these steps:

1. Generate the application with trace.
2. Select the Enable Trace option.
3. Populate and validate the Host and Port entry fields. The Host field accepts both IPv4 and IPv6 host names.
4. Start the Diagram Trace Utility must be started and list on the assigned Port, from the assigned Host.

If the Host field does not match the host name or IP address on which the Diagram Trace Utility is started, or the Port field does not match the port on which the Diagram Trace Utility is listening, you receive the following error message:



When the application is tested, invocation and completion messages are displayed in the Message Panel.

For more information about the Diagram Trace Utility, see the *Diagram Trace Utility User Guide*.

Assembling

The following sections describe the panels that are used for assembling Java, GUI and .NET applications for distribution.

EAR File Assembling

The EAR File Assemble Details panels are a set of dialogs that allow packaging of Java applications. When you select Action, Assemble, you see only the dialogs appropriate for the ICM you selected.

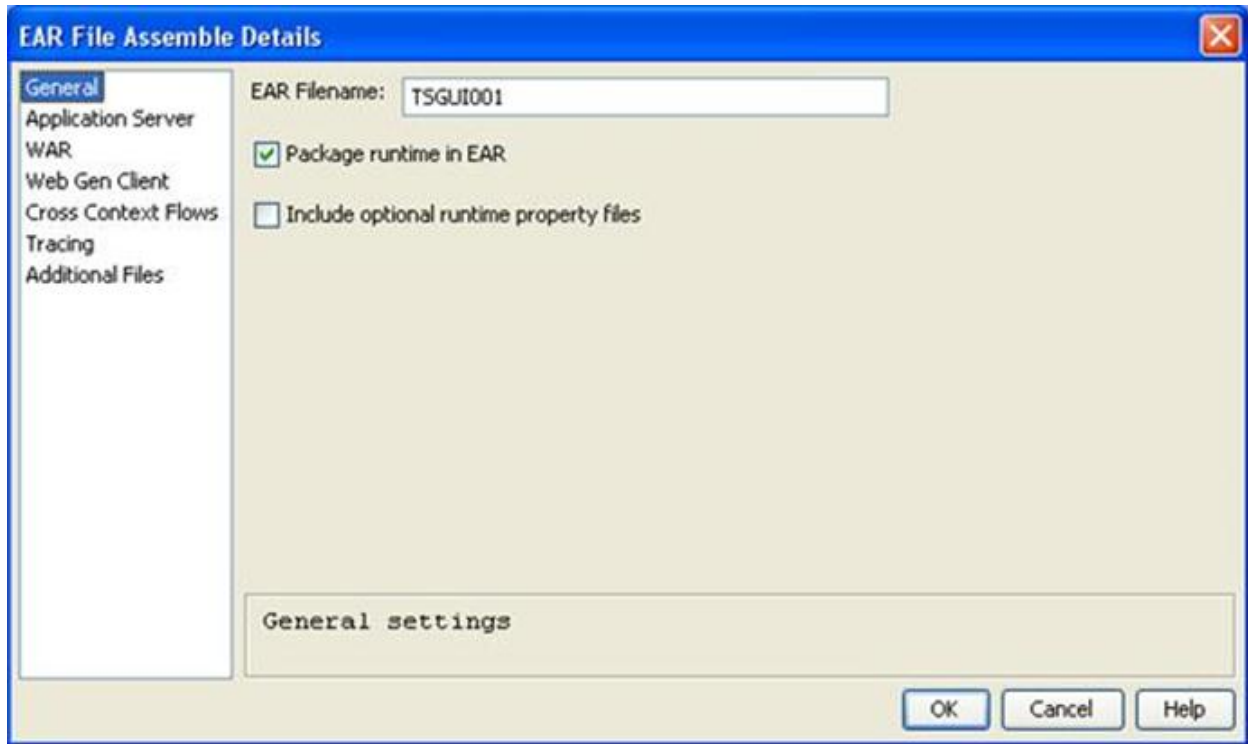
The possible dialogs are as follows:

- General
- Application Server
- WAR (Web Archive)
- Web Gen Client
- Cross Context Flows
- Tracing
- Additional Files
- Web Services

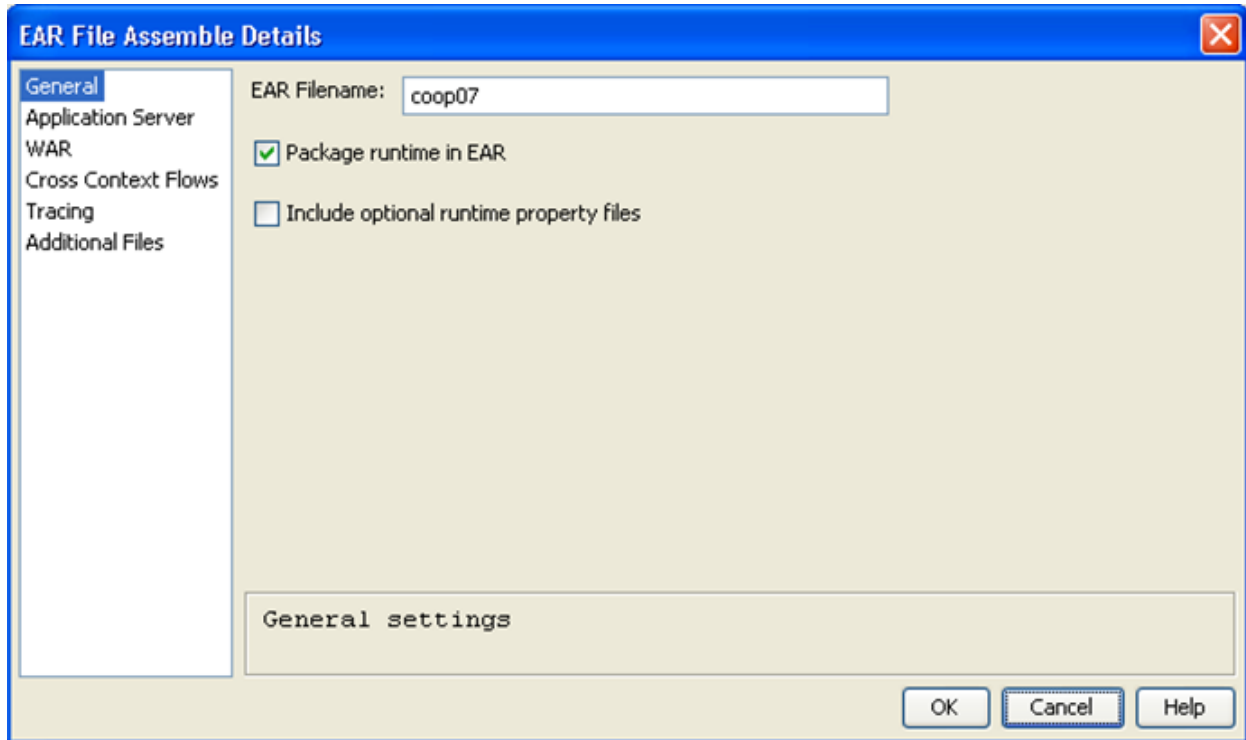
Important! When assembling applications with dialects, select only one Load Module icm for each assembly. The non-dialect specific icm files must be selected with every dialect.

General

When you select General from the EAR File Assemble Details with only Window Load Modules (TP Monitor: INTERNET) selected for assembly and Web Gen clients are selected, the following dialog appears:



When you select General from the EAR File Assemble Details panel with only Server Load Modules (TP Monitor: EJB) selected for assembly and no Web Gen clients are selected, the following dialog appears:



The fields in the General dialog are described in the following list:

EAR Filename

Specifies the name of the EAR file to be created. The length of the EAR file name and directory path cannot exceed 255 characters. The default is the local model name.

Package runtime in EAR

If checked, the CA Gen Java Runtimes are packaged into the EAR file. For more information, see CA Gen Java Runtime/User Exits in the chapter "Pre-Generation Tasks" in *Distributed Processing-Enterprise JavaBean User Guide*.

Include optional runtime property files

If selected, the following files are copied from the CA Gen directory into the genrt##.jar file within the EAR file:

- jdbccfg.properties
- commcfg.properties

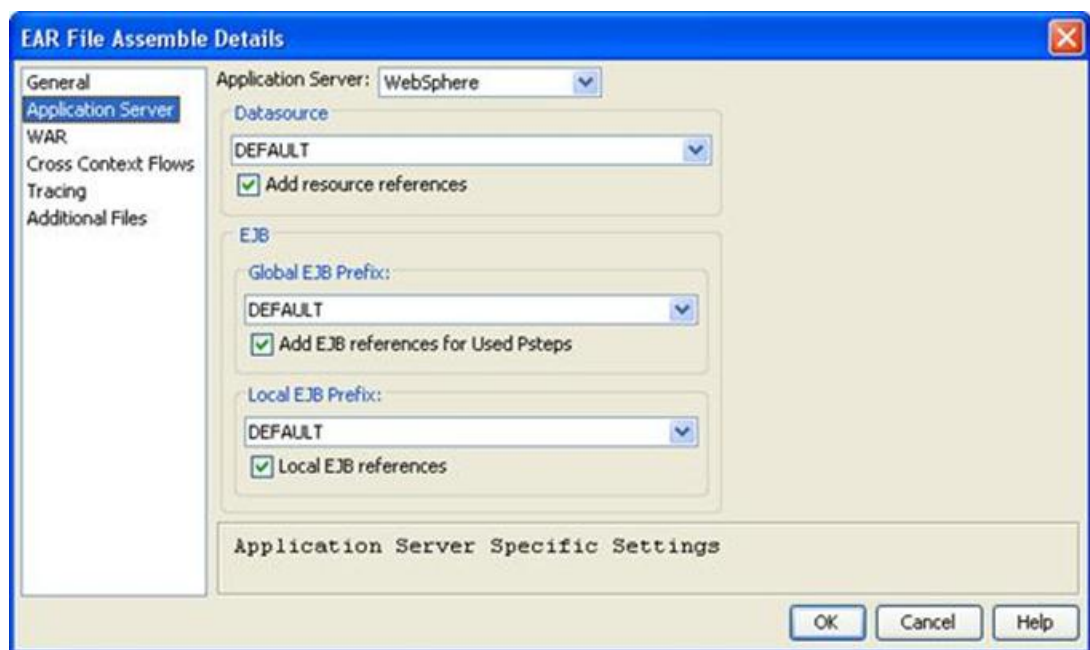
- codepage.properties
- applicationmanager.properties

This option is provided as a convenience to clients who are deploying to an Application Server. However, this option must be used with care, as any change to either file requires redeployment of the application to the Application Server.

Note: This option is enabled only if Package runtime in EAR is selected.

Application Server

When you select Application Server from the EAR File Assemble Details panel, the following dialog appears:



The fields in the Application Server dialog are as follows:

Application Server

Specifies the application server that you are targeting. The options in the drop-down list are:

- Generic
- WebSphere
- WebLogic

Datasource

Specifies the JNDI name of the CA Gen database you want to find.

Add resource references

Allows a datasource to be entered.

Global EJB Prefix

Specifies a prefix name to be used when performing JNDI lookups for EJBs that are defined outside of this EAR file.

Add EJB references for Used Psteps

Allows a Global EJB Prefix to be entered.

Local EJB Prefix

Specifies a prefix name to be used when performing JNDI lookups for EJBs defined in the same EAR file and running in the same JVM.

Local EJB references

Defines all of the Server Beans that the client code can call and allow the actual location of the Server Beans to be obtained when the client module is installed. The client code can use Local References, Remote References, or direct JNDI lookup to find the Server Bean. Local References execute faster.

When Local EJB references is checked, the EJB references are local rather than remote. The local references must refer to Server Beans in the same EAR file as the referring client and the Beans must be installed on the same JVM as the client. The local and remote reference definitions are stored in the XML descriptors of the calling code.

Note: The JMS Configuration fields on this dialog, which are used for Cross-Context Flows, will be disabled when the Generic option is selected from the Application Server drop-down. JMS configuration is unique to each Application Server and Gen has only been certified with WebLogic and WebSphere. Therefore, the Assemble process only supports specifying JMS configuration information for these two Application Servers. This does not preclude the use of Cross-Context Flows when the Generic option is selected, but Application Server specific customizations will have to be made to the EAR file outside of the Assemble process, in addition to the usual JMS configuration steps that are required on the Application Server itself.

WAR (Web Archive)

When you select WAR from the EAR File Assemble Details panel, the following dialog appears:

The screenshot shows the 'EAR File Assemble Details' dialog box with the 'WAR' tab selected. The 'WAR Filename' is 'tsgui001' and the 'Context' is 'tsgui001'. The 'Static Content' section has two checked options: 'Package Static Content within WAR' and 'Package Web Service Access Designer Content within WAR'. The 'Web Server DocumentRoot' field is empty. The 'Web Archive Settings' section is at the bottom. The dialog has 'OK', 'Cancel', and 'Help' buttons.

The fields in the WAR dialog are as follows:

WAR Filename

Specifies the name of the WAR file to be created. The length of the WAR file name and directory path cannot exceed 255 characters. The default is the local model name.

Context

Specifies the name that is to be used to access the generated application. For example, if Context is set to myapp and has a trancode of menu, the application would be started with the following URL:

`http://hostname.com/myapp/menu.jsp`

Use JDBC DataSources

If selected, an Application Server Resource Adapter is generated and placed in the WAR file deployment descriptor. The required information for the adapter is taken from the DBMS connection information that is specified in the Windows Build Tool Setup file using these tokens:

- OPT.DBCONNECT
- OPT.DSUSER
- OPT.DSPSWD
- If cleared, the Java Web Client code attempts to retrieve the connection information from the jdbccfg.properties file.

Use JTA transactions

If selected, the Java Web Clients use the Java Transaction API to manage the database connection and transaction control (two phased commit). This applies to those applications whose Java Web Clients access more than one database in a single client procedure step.

J2EE Application Servers can perform DBMS connections as part of a two phase commit Java transaction.

Note: Not all application servers support this feature.

This feature does not allow the Java Web Client and EJB servers to participate in a distributed transaction. CA Gen does not allow such distributed transactions to be designed.

If the Java Web Client is enabling threads, then the client is not allowed to participate in the JTA transaction. By disabling threading, the Java Web Client is prevented from utilizing Java Beans in the user interface.

Note: This field is available only if Use JDBC DataSources is selected, and Enable Threading is cleared on the Web Gen Client dialog.

Package Static Content within WAR

If selected, all dynamic and static content is packaged in the WAR file. Otherwise, only the dynamic content (JSPs and JAR files) is packaged in the WAR file, while the static content (HTML, JavaScript, GIFs) is copied to the document root of the web server.

Higher performance may be achieved by clearing this option and providing a static content location using the Web Server DocumentRoot field.

Web Server DocumentRoot

Specifies the root document. This must be the root document of the Web server.

If Apache is being used, the document root is <APACHE_HOME>/htdocs.

If IIS is being used, the document root is <IIS_HOME>/wwwroot.

This field is available only when Package Static Content within WAR is cleared, that is, you have chosen to separate the static and dynamic content.

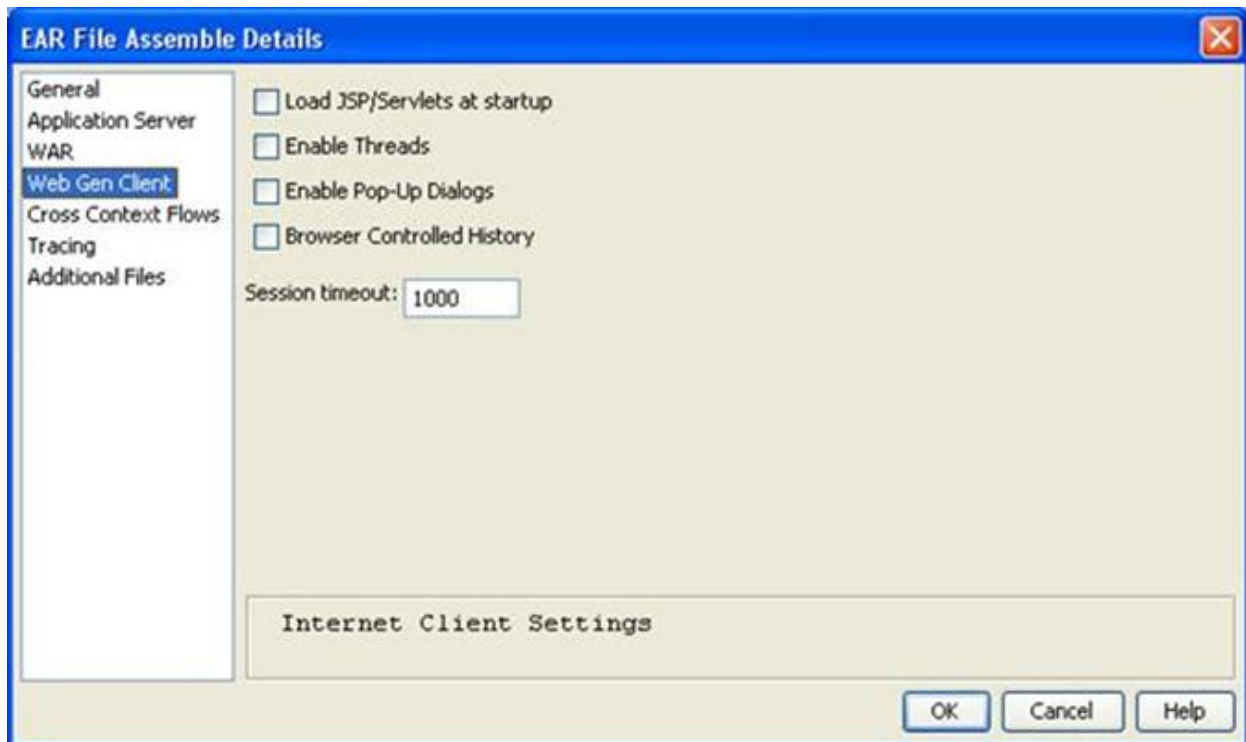
Package Web Service Access Designer Content within WAR

Includes the generated files, which are the Action Steps directory and all its subdirectories, in the Web View application WAR file.

Default: Selected

Web Gen Client

When you select Web Gen Client from the EAR File Assemble Details panel, the following dialog appears:



The fields in the WAR dialog are as follows:

Load JSP/Servlets at startup

Improves the performance of the initial load of the JSP and servlets by having them precompiled and loaded by the application server when the application is loaded. This option is cleared during development and selected for production work.

Enable Threads

If selected, the Java Web Client Runtime uses threads to perform certain functions such as displaying MessageBoxes or accessing ActiveX controls. However, some Application Servers do not work correctly if threads are used.

Enable Pop-Up Dialogs

If selected, dialog support is enabled. Otherwise, all windows are embedded in the browser.

Browser Controlled History

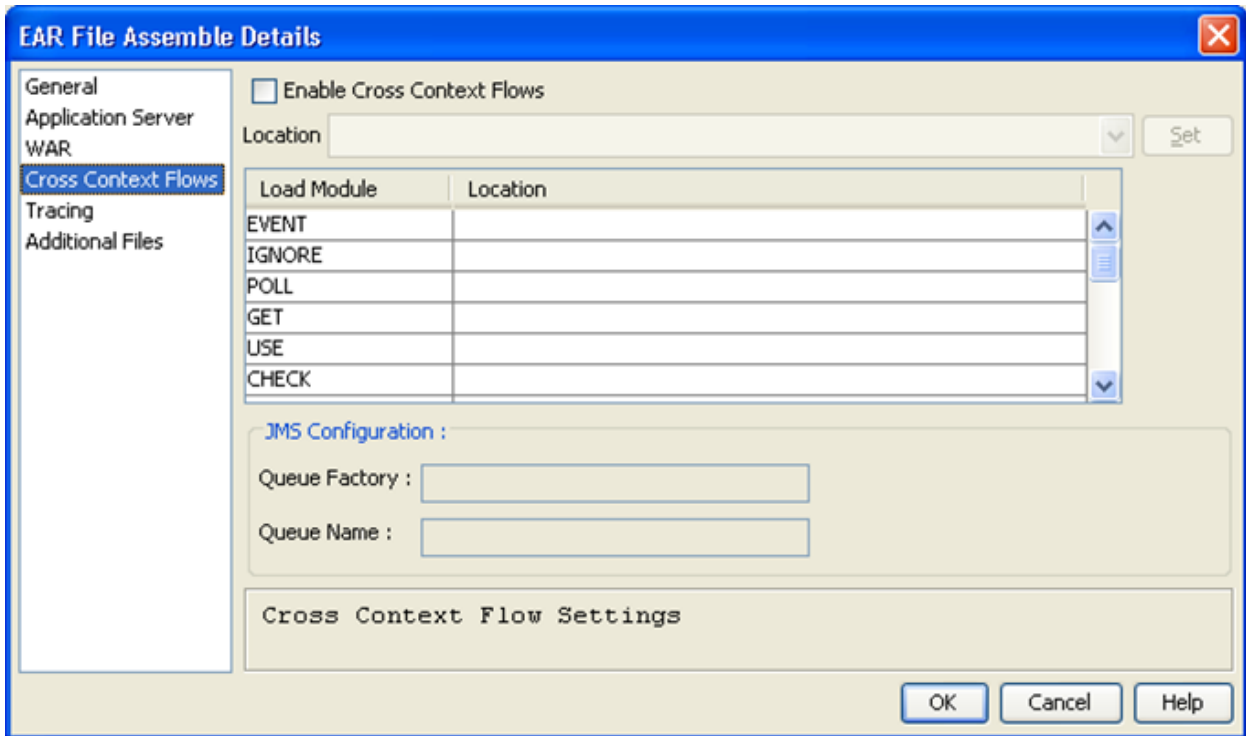
If selected, you can use the Back and Forward buttons of your browser for navigation. The default style only allows access to one page at a time. Although the Back and Forward buttons remain enabled, using them takes you out of the application. This option is not valid for Cross-Context Flows applications.

Session timeout

Specifies the number of minutes a session can be idle before it is abandoned. The default is 30. The Timeout range is from 1 through 1440 minutes.

Cross-Context Flows

The Cross-Context Flows feature allows load modules that are deployed in separate web applications to interact. To support this feature, the location of each load module must be specified.



Selecting the Enable Cross-Context Flows check box activates Cross-Context flows for the current assembly. Browser Controlled History is not allowed for Cross-Context applications. Build Tool does not allow you to select both Browser Controlled History and Cross-Context Flows.

The table provides a list of all of the load modules that are referenced by the load modules that are selected in the main Build Tool panel.

Some of the listed load modules are local to the current web application that is being assembled. There is no need to specify a mapping for local load modules.

A location should be specified for any remote load modules in the corresponding Location field. If the location of a remote load module is not specified, then an error occurs at runtime if an attempt is made to flow to it.

Enter the fully qualified URL of the context where the load module is deployed. For example, when configuring context CBD1 and it references CBD2LM2 in web application CBD2, enter the URL as `http://myserver/CBD2`. The CBD2 in the URL would be the context name that is provided while assembling web application CBD2.

To enter a location URL, type the fully qualified URL into the Location field to the right of the load module name to configure. Alternatively, select multiple load modules to be deployed in the same context and enter the Location URL into the Location field. After entering the URL, click Set to populate the Location fields for each of the selected load modules.

The JMS Configuration section of the dialog contains the following fields:

Queue Factory

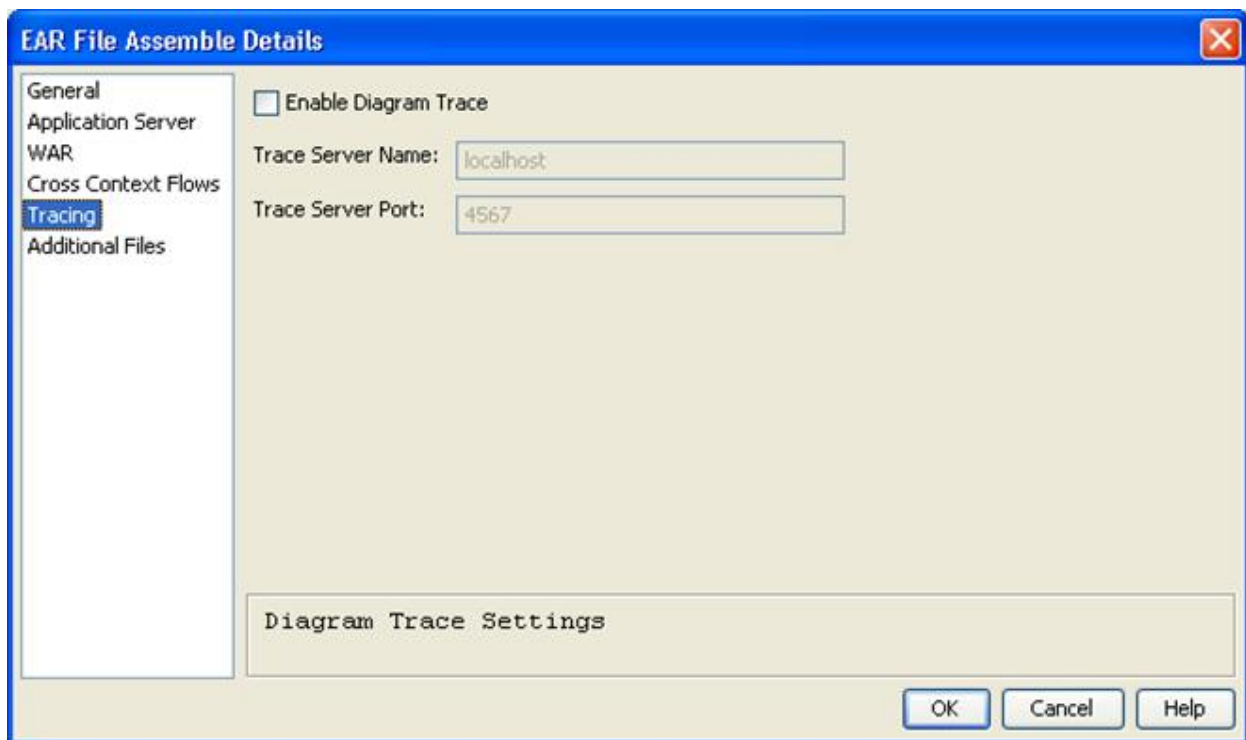
Specifies the JNDI name of the JMS Connection Factory that is configured for the application server.

Queue Name

Specifies the JNDI name of the JMS Queue that is configured for the application server.

Tracing

When you select Tracing from the EAR File Assemble Details panel, the following dialog appears:



The fields in the Tracing dialog are as follows:

Enable Diagram Tracing

If selected, CA Gen generates XML to specify the Trace Server in the Deployment Descriptor. If this flag is not set, the generated code is not traceable, even if it was generated for trace.

Note: The application must also be generated with Trace to use Diagram Tracing.

Trace Server Name

Specifies the name of the Trace Server. This field is only enabled if Enable Diagram Tracing is checked. The IP address can be in IPv4 format or IPv6 format.

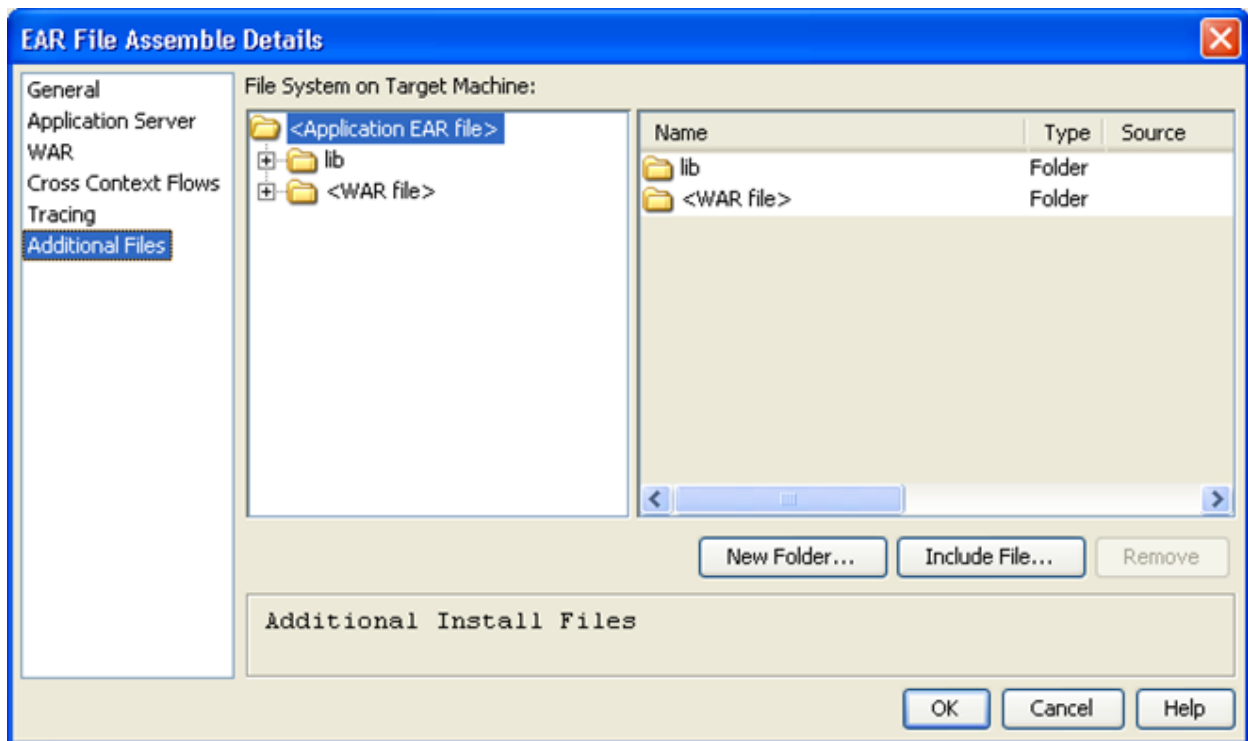
Trace Server Port

Specifies the port number of the Trace Server. This field is only enabled if Enable Diagram Tracing is checked.

Additional Files

To assemble different Operation Libraries that are built in different models, use the Additional Files feature of the Assemble Utility.

When you select Additional Files from the EAR File Assemble Details panel, the following dialog appears:



The fields in the Additional Files dialog are described in the following list:

File System on Target Machine

Displays a tree list showing the folder structure that is installed on the target machine. If an EAR file deployment, it is the folder structure that is created within the EAR file itself. Folders can be added and removed as desired. The folder structure is pre-populated with some default folders that are installed based on the Load Modules that are selected for the assemble.

Detail Table

Displays a table listing the details for the currently selected folder. The contents of the folder are either files (that have been included) or sub-folders. For each item, the name, type (File, Folder), and the source location of the included file is shown.

New Folder...

Enables you to add a new sub-folder below the currently selected folder.

Include File...

Opens a File chooser dialog to allow the assembler to select files off the current machine that are included into the EAR file in the current folder selected.

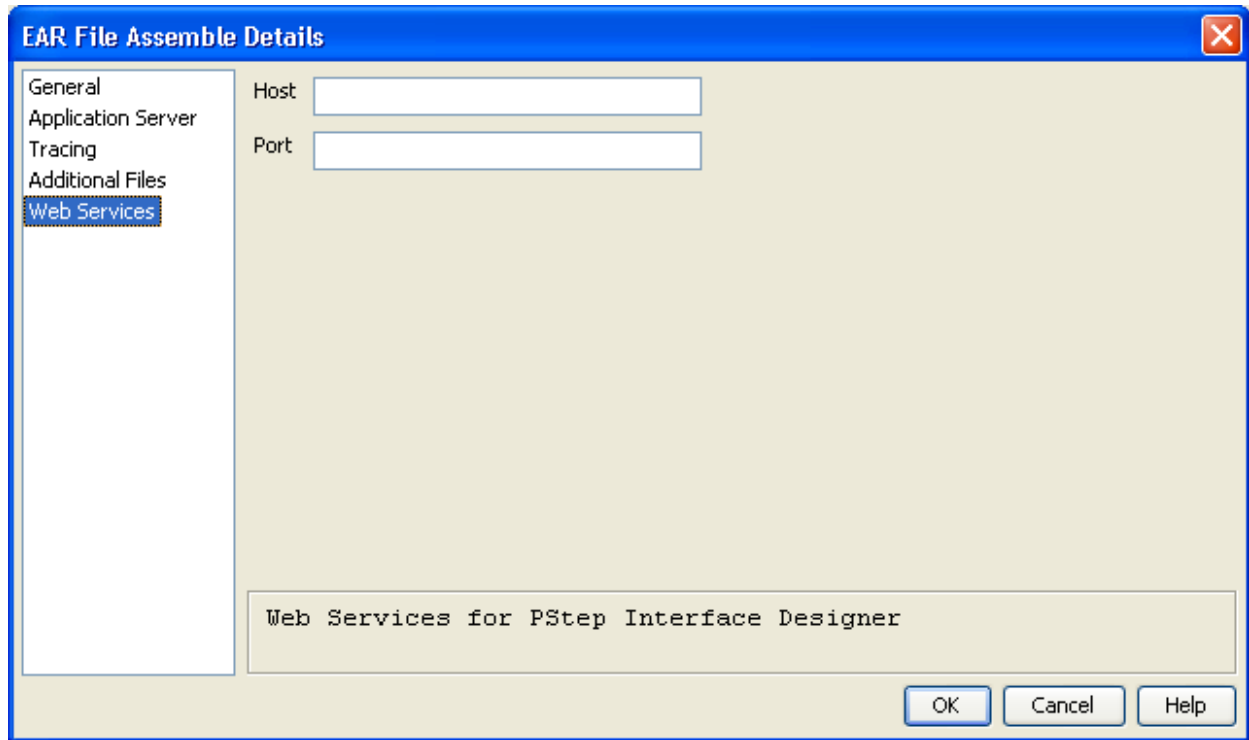
Remove

Removes the currently selected item. If the item is a file, then the file is removed from its folder. If the item is a folder, then the folder and all of its contents (files and sub-folders) are removed.

Note: This dialog makes it possible for you to add any needed files to the EAR file. This feature is provided for your convenience; however, CA has no way of knowing whether these files are properly licensed on your deployment environment. Therefore, it is your responsibility to ensure proper licensing on the target machine.

Web Services

The Web Services panel is visible only if you have included a Web Service Definition module for assembly. When you click Web Services from the EAR File Assemble Details panel, the following dialog appears:



You can override the host and port values of the Web Service Url in the Web Services pane. The Web Service Url is a property of the Web Service Definition that is set when a new Web Service Definition is created in the PStep Interface Designer perspective in CA Gen Studio. The values in this panel are used in the generated WSDL packaged with the application.

MSI Assembling

The MSI Assemble Details panels are a set of dialogs that allow packaging of both GUI and .NET applications. When you select Action, Assemble, the Build Tool Client looks at the content of the selected ICM and displays the appropriate dialogs. The output is an MSI file ready for installation on a Windows system. The possible dialogs follow:

- General (GUI and .NET)
- ASP.NET (.NET only)
- Global Assembly Cache (GAC) (.NET only)
- DataSources (.NET only)

- Tracing (.NET only)
- Additional Files (GUI and .NET)
- GUI (GUI only)
- Commands (GUI only)

Important! While assembling applications with dialects, select only one Load Module icm for each assembly. The non-dialect specific icm files should be selected with every dialect.

Note: We cannot build a large executable application based on more than one model.

For documentation purposes, the MSI .NET Assembly Details panels are illustrated when the dialogs are applicable for both .NET and GUI applications.

Both .NET and GUI applications are assembled from a selected set of load module (LM) and referential integrity trigger (RI) icm files. UNIX is not supported.

A major difference between .NET applications and GUI applications is that GUI assemblies can optionally include database design language (DDL) icm files, giving the installer the opportunity to create the associated database tables locally. If a DDL icm is included in the selections when the *Assemble* command is executed, the assembled MSI file will offer the installing user a *Custom* option to install the DDL feature. A *Typical* install and the default *Custom* install will not install the DDL. The installing user must choose the *Custom* install and then select *DDL* for local installation. The files necessary for the DDL feature are copied to the installing user's TempFolder. The DDL processing is done before any optional *Commands*. The stdout and stderr of the DDL processing is recorded in a log file on the user's desktop with the same basename as the MSI file and a *.log* extension. For DB2 DBMS applications, procedure binding is included after the data tables have been initialized.

Any options that are entered into the Assemble Details panels are stored in an assemble icm file. This file is re-read on each execution of the assemble command to initialize the panel fields. The assemble command is disabled if this assemble icm file is selected.

General (GUI and .NET)

When you select General from the MSI GUI or .NET Assemble Details panel, the following dialog appears:

The screenshot shows a Windows-style dialog box titled "MSI .NET Assemble Details". On the left is a vertical list of tabs: "General" (selected), "ASP.NET", "GAC", "DataSources", "Tracing", and "Additional Files". The main area contains several input fields and checkboxes. The fields are: "MSI Filename:" with text "Sample", "Product Name:" with text "Sample", "Product Version:" with text "1.0", "Manufacturer:" with text "CA", "Product Code GUID:" with text "{10F0B1C1-7568-CD7B-01B2-2D60E3795C0E}", "Upgrade Code GUID:" with text "{9D60F908-DD6F-727B-96C4-3101294642B9}", and "HttpRuntime Execution timeout:" with text "110". To the right of the GUID fields are "Create GUID" buttons. At the bottom left is a checked checkbox labeled "Package CA Gen Runtime in MSI". Below the input fields is a text box labeled "General settings". At the bottom right are "OK", "Cancel", and "Help" buttons.

The fields in the General dialog are described in the following list:

MSI Filename

Specifies the base name of the production MSI database file. The suffix .msi is appended automatically.

Product Name

Specifies the name of the application visible to the end user. Also used as part of the default destination directory path.

Product Version

Specifies the version of the product. Must be in the format ###.###.##### and is visible to the end user. This field and the two GUIDs are used by Microsoft MSI Installer to determine which files are copied from the media to the system during access, updates, upgrades, and revisions. For more information, see the MSI SDK documentation.

Manufacturer

Specifies the manufacturer name. This is used as part of the default destination directory path and visible to the end user.

Product Code GUID and Upgrade Code GUID

Specifies the Product Code GUID and Upgrade Code GUID. Click Create GUID to the right of the fields to populate the fields. You have the choice to enter a value or let one be calculated. Together with Product Version, these fields are used to control MSI behavior when installing newer software. It is important that these values only change when appropriate. For more information, see the Microsoft MSI SDK documentation.

HttpRuntime Execution timeout

Sets the value for the httpRuntimeExecutionTimeout attribute in the web.config file. For the specified number of seconds a browser lets an ASP.NET page run without responding. After the specified number of seconds, the operation is assumed to have failed, and the ASP.NET page is terminated. The default value for this attribute is 110 seconds. The range for httpRuntime executionTimeout is between 1 and 3600 seconds.

Package CA Gen Runtime in MSI

If checked, the CA Gen Runtimes are packaged into the MSI file.

ASP.NET (.NET only)

When you select ASP.NET from the MSI .NET Assemble Details panel, the following dialog appears:

MSI .NET Assemble Details

General
ASP.NET
 GAC
 DataSources
 Tracing
 Additional Files

☒ Compress Dynamic Content ☒ Pre-compile ASPX pages
☐ Browser Controlled History ☐ Threaded
☒ Auto-Include 3rd Party Control Assemblies ☐ Exclude Assemblies in GAC

Session State: StateServer Session State Timeout: 20
 Connection String: tcpip=localhost:42424
☒ Encrypt sessionState section (web.config)

Custom Errors: RemoteOnly Default Redirect:
 Display Mode: Dialogs Idle Threads:
 Theme Name: Theme1

ASP.NET Client Settings

OK Cancel Help

The fields in the ASP.NET dialog are described in the following list:

Compress Dynamic Content

Allows you to compress the dynamic HTML content returned to the browser.

Pre-compile ASPX pages

Allows you to compile an ASP.NET application before deployment.

Browser Controlled History

Allows the Back and Forward buttons of your browser to be used for navigation. The default style allows access to only one page at a time. Although the Back and Forward buttons remain enabled, using them takes you out of the application.

Threaded

Allows the runtime to create an extra thread in addition to the one the Application Server creates for each request. This option is essential for applications that use MessageBox and MessageBoxBeep. In Web Generation Applications, message boxes and OCX controls require threading to be enabled.

Auto-Include Third Party Control Assemblies

If selected, includes any third-party web control assemblies that you used in your application and any assemblies they reference, in the MSI file. You can also use the Additional Files dialog if you want to specify them yourself.

Important! This feature is provided for your convenience, however, CA has no way of knowing whether these third-party web controls are properly licensed on your deployment environment. Therefore, it is your responsibility to ensure proper licensing on the target machine.

Exclude Assemblies in GAC

Enabled only when *Auto-Include 3rd Party Control Assemblies* is selected. When selected and enabled, any third-party web control assemblies that are deployed in GAC are not included in the MSI file. By default, this field is disabled and not selected.

Session State

The Web is a stateless environment. However, most web applications require effective state management. In particular, the session state of a web application is the data that an application caches and retrieves across different requests. A session represents all the requests that are sent by a user during a connection to the site. The session state is the collection of persistent data that the user generated and used during the session. The state of each session is independent from that of another session and does not survive the end of the user session.

The modes that are supported are:

StateServer

Session values are serialized and stored in the memory of a separate process (aspnet_state.exe). The process can also run on another machine.

SQLServer

Session values are serialized and stored in a Microsoft SQL Server table. The instance of SQL Server can run either locally or remotely.

InProc

Session values are stored in the memory of the ASP.NET worker process. Thus, this mode offers the fastest access to these values. However, when the ASP.NET worker process recycles, the state data is lost.

For more information, see the Microsoft documentation about ASP.NET state management.

Session State Timeout

Specifies the number of minutes a session can be idle before it is abandoned. The default is 20. The Timeout range is from 1 through 1440 minutes.

Connection String

Required for both StateServer and SQLServer Session State modes. The IP address can be in IPv4 format or IPv6 format.

For StateServer

Known as stateConnectionString. It specifies the server name or IP Address and port number where session state is stored remotely. For example, tcpip=127.0.0.1:42424.

For SQLServer

Known as sqlConnectionString. It specifies the connection string for a SQL Server. For example, data source=localhost;Integrated Security=SSPI;Initial Catalog=northwind.

Encrypt sessionState section (web.config)

Encrypts the sessionState information. It is selected by default.

Custom Errors

Specifies whether custom errors are enabled, disabled, or shown only to remote clients.

On

Specifies that Custom Errors is enabled. If no defaultRedirect is specified, users see a generic error.

Off

Specifies that Custom Errors is disabled. This allows display of detailed errors.

RemoteOnly

Specifies that Custom Errors are shown only to remote clients and ASP.NET errors are shown to the local host. This is the default.

Default Redirect

Enabled only when Custom Errors is On. It specifies the default URL to direct a browser to if an error occurs. The URL may be absolute (for instance, <http://www.myapp.com/ErrorPage.htm>) or it may be relative. A relative URL such as [ErrorPage.htm](#) is relative to the Web.config file that specified the defaultRedirect URL, not to the Web page in which the error occurred. A URL starting with a slash (/), such as [/ErrorPage.htm](#), means that the specified URL is absolute to the root path of the Application Server.

Display Mode

Dialogs

Enables modal and modeless dialogs in a web application.

Classic

All windows are embedded in the same browser.

Idle Threads

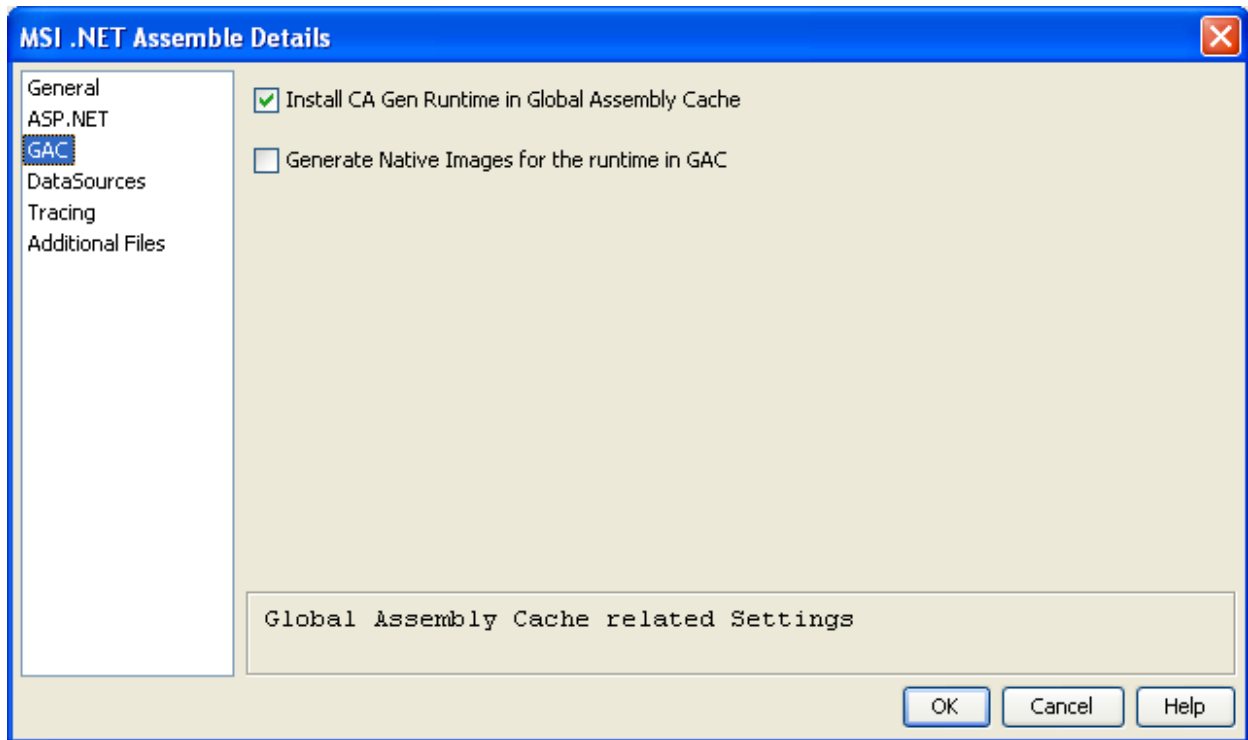
Specifies the minimum number of idle threads that are created and maintained by the ThreadPool. They are managed in a background task and used to serve incoming requests to reduce the time that is spent in creating and destroying threads when new requests are initiated or existing ones complete. Tuning this value can help optimize your application performance. For more information, see the Microsoft documentation.

Theme Name

Specifies the directory where the skin and the images are stored.

Global Assembly Cache (.NET only)

When you select Global Assembly Cache (GAC) from the MSI .NET Assemble Details panel, the following dialog appears:



A description of the fields in the GAC dialog is as follows:

Install CA Gen Runtime in Global Assembly Cache

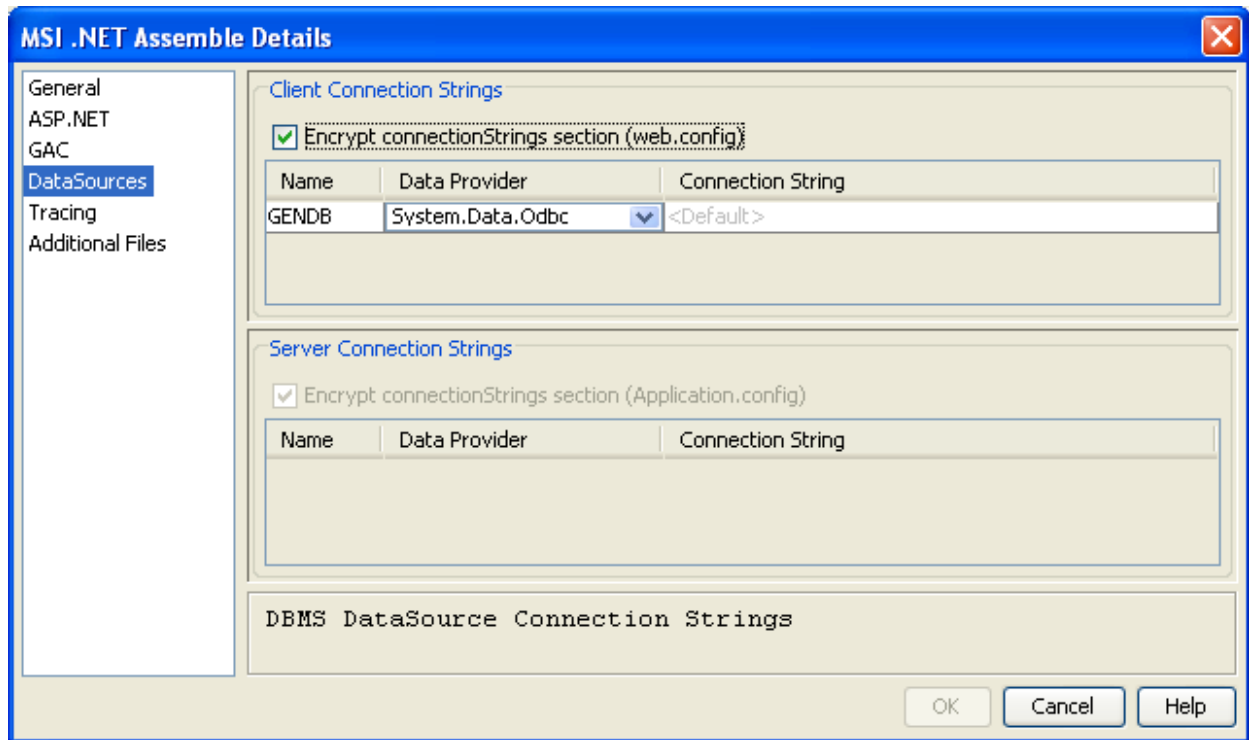
Check this box if you want to include the CA Gen runtime software in your Global Assembly Cache.

Generate Native Images for the runtime in GAC

Check this if you want to generate native images for CA Gen runtime for a .NET application.

DataSources (.NET only)

When you select DataSources from the MSI .NET Assemble Details panel, the following dialog appears:



The fields in the Tracing dialog are as follows:

Client Connection Strings

Specifies the connection string parameters for every logical database name specified with ASP.NET Clients.

Encrypt connectionStrings section (web.config)

If selected, CA Gen encrypts the connection string in the web.config file

Name

Specifies the name of the database

Data Provider

Specifies the data provider. This is an enterable drop down that displays the list of data providers. By default, four data providers are listed. They are:

- System.Data.Odbc
- System.Data.SqlClient

- IBM.Data.DB2
- Oracle.DataAccess.Client

Note: You can enter your own data provider

Server Connection Strings

Specifies the connection string parameters for every logical database name specified with .NET Servers.

Encrypt connectionStrings section (Application.config)

If selected, CA Gen encrypts the connection string in the Application.config file

Name

Specifies the name of the database

Data Provider

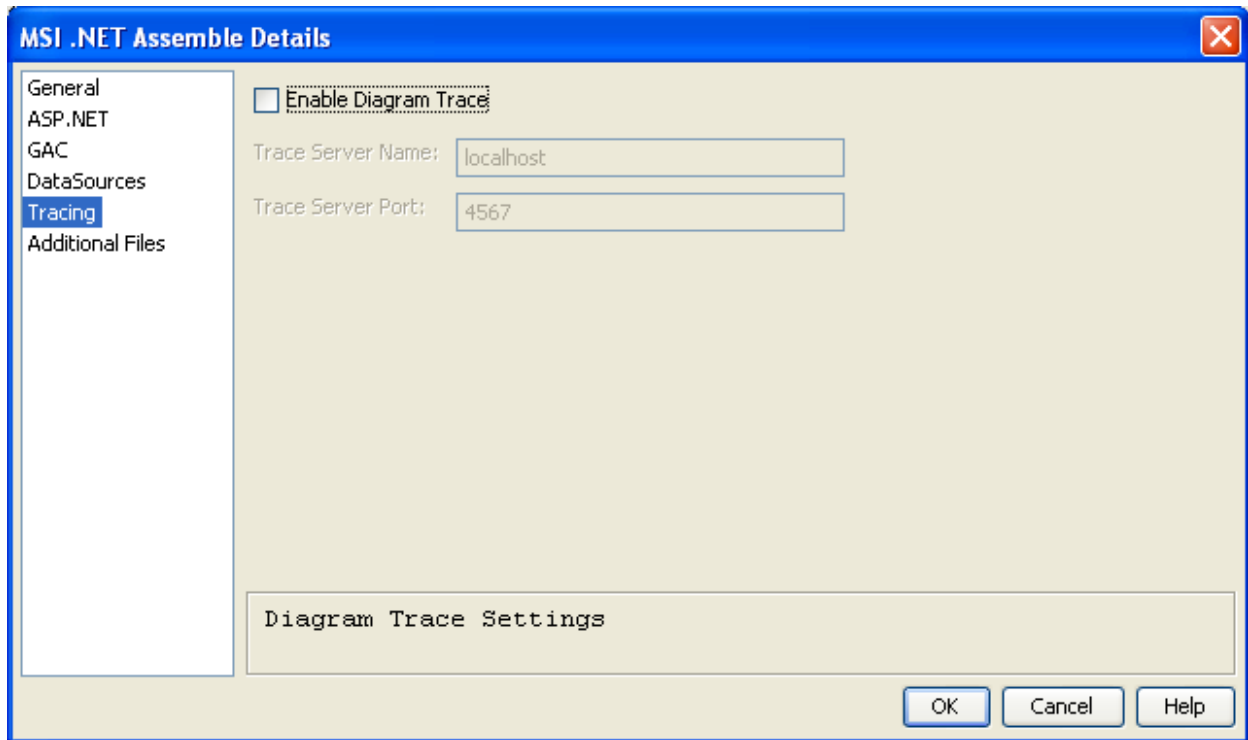
Specifies the data provider. This is an enterable drop down that displays the list of data providers. By default, four data providers are listed. They are:

- System.Data.Odbc
- System.Data.SqlClient
- IBM.Data.DB2
- Oracle.DataAccess.Client

Note: You can enter your own data provider.

Tracing (.NET only)

When you select Tracing from the MSI .NET Assemble Details panel, the following dialog appears:



The fields in the Tracing dialog are described in the following list:

Enable Diagram Trace

If selected, CA Gen generates XML to specify the Trace Server in the Deployment Descriptor. If this flag is not set, the generated code is not traceable, even if it was generated with Trace.

Note: The application must also be generated with Trace to use Diagram Tracing.

Trace Server Name

Specifies the name of the Trace Server. This field is available only if Enable Diagram Tracing is checked. The IP address can be in IPv4 format or IPv6 format.

Trace Server Port

Specifies the port number of the Trace Server. This field is available only if Enable Diagram Tracing is checked.

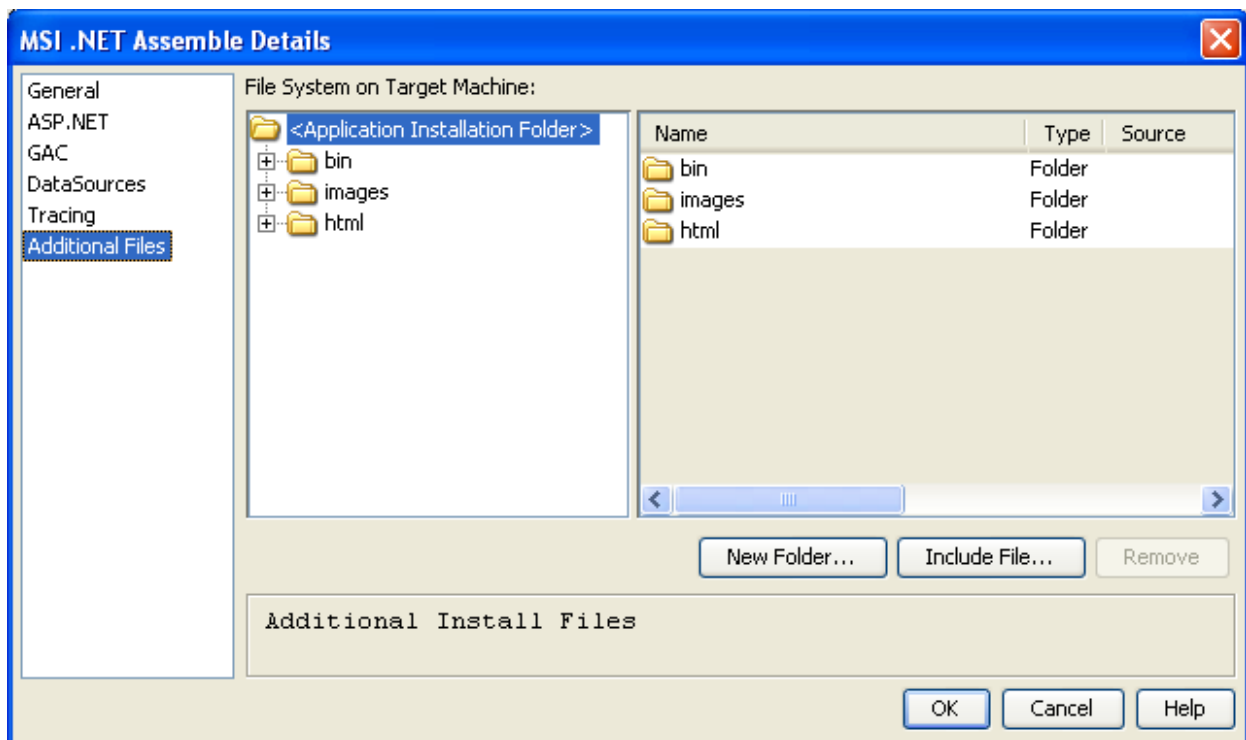
Load Data

Field	Description
Load items	Loads all the items and the list does not contain a scroll bar.
Load items on scroll	Loads all the items and the list contains a scroll bar.

Additional Files (GUI and .NET)

You can use the Additional Files Feature of the Assemble Utility to assemble different Operation Libraries that are built in different models.

When you select Additional Files from the MSI GUI or .NET Assemble Details panel, the following dialog appears:



The fields in the Additional Files dialog are described in the following list:

File System on Target Machine

Displays the tree list showing the folder structure that is installed on the target machine. Folders can be added and removed as desired. The folder structure is pre-populated with some default folders that are installed based on the Load Modules that are selected for the assemble.

Detail Table

Displays a table listing the details for the currently selected folder. The contents of the folder are either files (that have been included) or sub-folders. For each item, the name, type (File, Folder), and the source location of the included file is shown.

New Folder...

Adds a new sub-folder below the currently selected folder.

Include File...

Opens a File chooser dialog to allow the assembler to select files off the current machine that is included into the MSI file in the current folder selected.

Remove

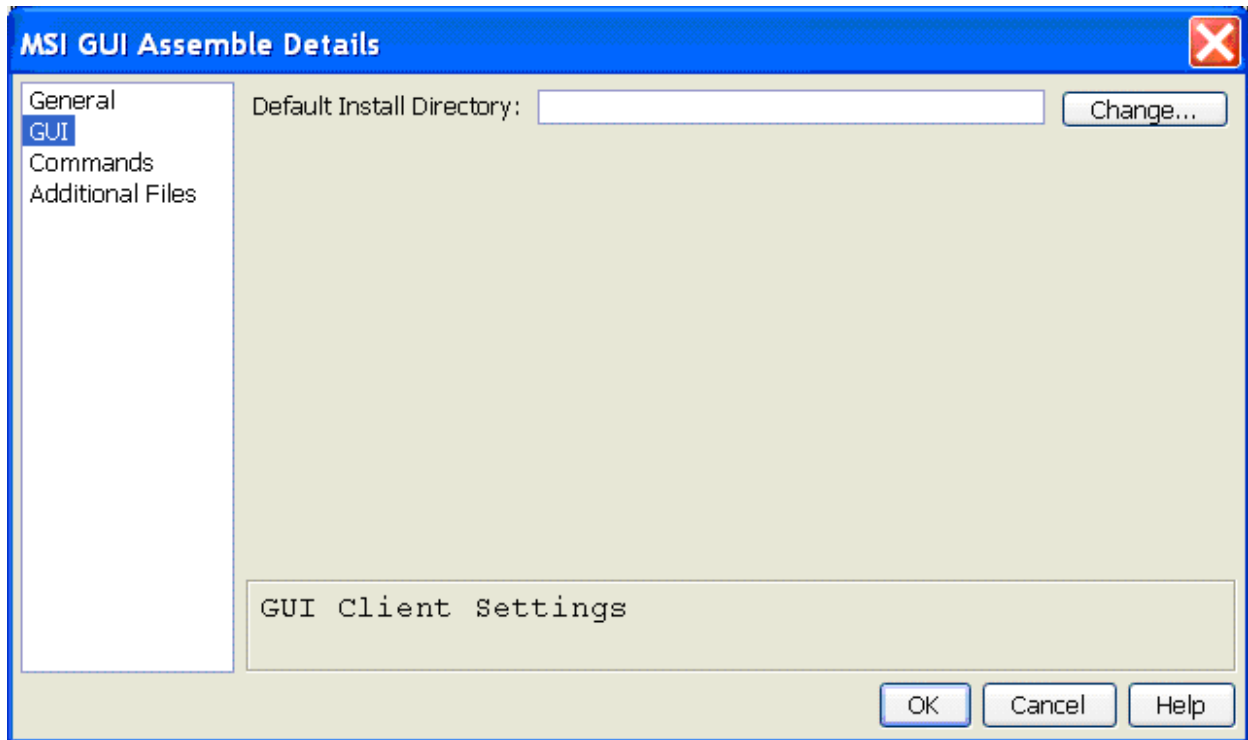
Removes the currently selected item. If the item is a file, then the file is removed from its folder. If the item is a folder, then the folder and all of its contents (files and sub-folders) is removed.

Note: This option makes it possible to specify additional files to be included in the MSI package.

Important! This feature is provided for your convenience, however, CA has no way of knowing whether these files are properly licensed on your deployment environment. Therefore, it is your responsibility to ensure proper licensing on the target machine.

GUI (GUI only)

When you select GUI from the MSI GUI Assembly Details panel, the following dialog appears:

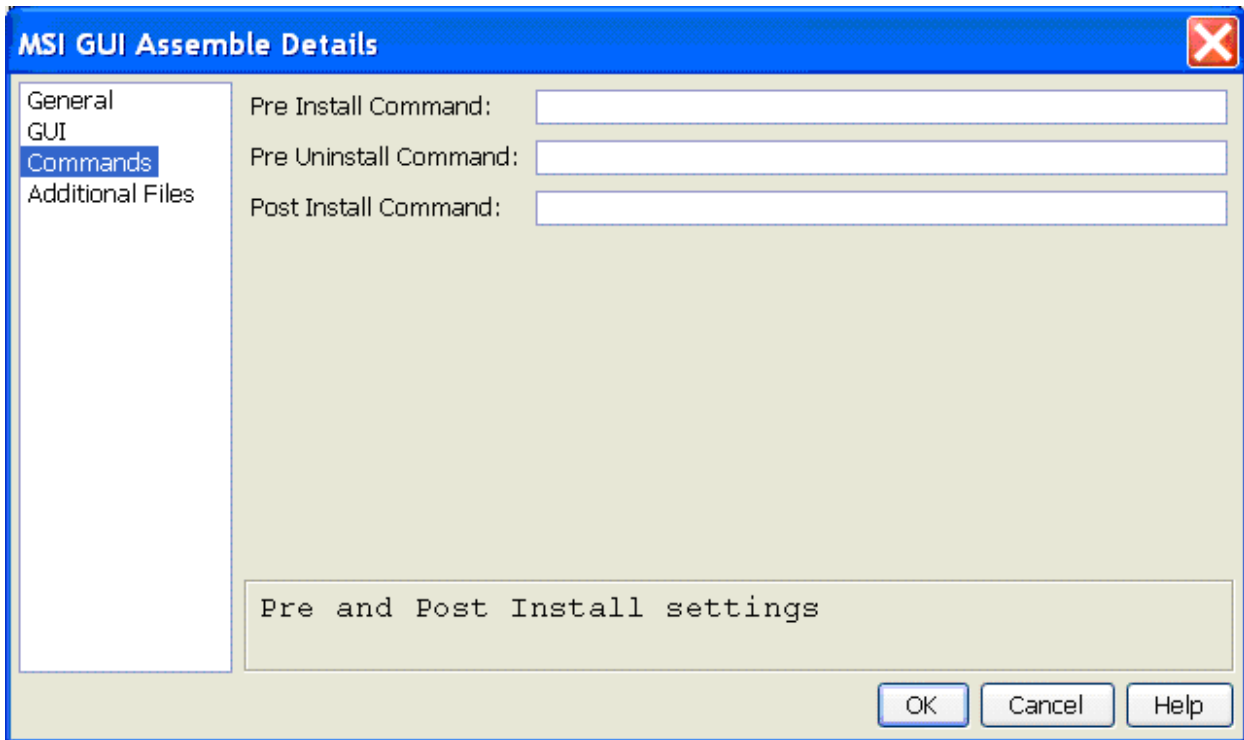


The default install directory is initialized to a combination of the ProgramFilesFolder property expanded by MSI during installation and the Manufacturer and Product Name fields from the General selection details. Any MSI standard property can be substituted for ProgramFilesFolder. Case is significant for MSI properties. See MSI SDK for details. During a *Custom* installation, you can edit this location. This location is used for the application load modules and referential integrity triggers.

During the installation process, MSI will recognize if the same version of the CA Gen runtime libraries is already installed and make that location the default location for only the runtime libraries. The first time the CA Gen runtime libraries are installed on the target host the Default Install Directory field will also become the default location for the runtime libraries for all MSI packaged GUI applications. During a *Custom* installation the installing user will have the opportunity to edit this location separately from the application directory. For runtimes installed into the same location, the Windows operating system keeps a reference count of the number of applications using the libraries.

Commands (GUI only)

When you select Commands from the MSI GUI Assembly Details panel, the following dialog appears:



The fields in this dialog specify optional arbitrary commands to be executed as part of the MSI installation process. The commands are passed to a modal command window opened in addition to the MSI progress dialog. The command processor that is used is specified by the COMSPEC environment variable of the target host, set to cmd.exe. The command must return 0 to indicate success for the MSI to continue. Non-zero returns causes the installation to abort and leave the target host unmodified. The command window is closed when the command is executed. If the window must stay open until the user dismisses it, you can use a *bat* file that uses the *pause* command. All output to the command window is lost when the window closes. The execution start and exit value is recorded in a log file that is saved to the desktop of the user using the basename of the MSI file and the *.log* extension. The same log file is used if an optional DDL feature is installed on the target host. It is possible to change the behavior of these commands to *hide* the command window and automatically log stdout and stderr to the logfile. Edit the word *show* to *hide* on the appropriate line in Buildtool script `bt\scripts\deploy_msi_gui.scr`.

The Pre Install command is run before files are transferred. The command must therefore already exist on the target host. The current directory is the Windows system32 directory.

The Post Install command is run after the assembled files, including any *Additional Files*, have been transferred to the target. The current directory is the application install directory.

The Pre Uninstall command is run before any files are removed. The current directory is the application install directory.

Chapter 6: Build Scripts

Scripts are tokenized command procedures that contain all the specific commands necessary to perform the compile and link steps for a module. During a module build, the Build Tool interprets the script and replaces the tokens with information about the local or remote file from the setup information (found in the ICM file), along with profile information that has been utilized or customized by the user, and produces a complete command procedure. The Build Tool then submits the command procedure for execution.

Tokens are generic placeholders that are used for substitution in a script. When a script is being interpreted, each token is resolved to its representative value so that the resulting command procedure is specific to a module and its set-up environment. When the script is processed, unresolved tokens are resolved as a blank value.

Note: Structures of tokens within tokens are not supported.

The information that appears as tokens in a script includes:

- Information about the elements of the module itself. This information is taken from the Install Control Module (ICM) portion of the file.
- Locations and target configuration information defined previously using the Build Tool profiling utility.
- Option (OPT) tokens. Option tokens are user defined in the Build Tool profile utility.

Locating Scripts

The Build Tool contains a set of valid scripts for a specific operating environment. You can use them as they are or customize them for a particular application. These scripts are located in the CA Gen bt/scripts directory.

Script Names

A set of build scripts exists for each supported platform. These scripts are listed in the following table:

Script Name	Description	Platform
build.scr	Main build operation entry point script. Invokes appropriate build script.	Windows
build_ddl.scr	DDL install script	Windows

Script Name	Description	Platform
build_lm_c.src	C language module build script	Windows
build_ri_c.scr	C language RI build script	Windows
build_lm_cs.scr	C# language build script	Windows
build_ri_cs.scr	C# language Referential Integrity build script	Windows
build_interfaces_net.scr	C# load module interfaces build script	Windows
build_lm_java.scr	Java language module build script	Windows
build_ri_java.scr	Java language RI build script	Windows
build_proxy_com.scr	COM Proxy build script	Windows
build_proxy_java.scr	Java Proxy build script - Classic style	Windows
build_proxy_java_2.scr	Java Proxy build script	Windows
build_proxy_net.scr	.NET proxy build script	Windows
build_wsd_java.scr	Web Services build script	Windows
callsvvars.scr	VS version setting script	Windows
deploy.scr	Main deploy operation entry point script. Invokes appropriate deploy script.	Windows
deploy_ear.scr	EAR file deployment script for J2EE component load modules (Web Generation, EJB).	Windows
deploy_msi_gui.scr	MSI file deployment script for GUI components	Windows
deploy_msi_net.scr	MSI file deployment script for .NET components	Windows
deploy_ear_jboss_ejb.scr	Java deployment script for jboss	Windows
deploy_ear_jboss_web.scr	Java deployment script for Jboss	Windows
deploy_ear_jboss_wsd.scr	Java deployment script for Jboss	Windows
deploy_ear_weblogic_ejb.scr	Java deployment script for Weblogic	Windows
deploy_ear_weblogic_web.scr	Java deployment script for Weblogic	Windows

Script Name	Description	Platform
deploy_ear_weblogic_wsd.scr	Java deployment script for Weblogic	Windows
deploy_ear_websphere_ejb.scr	Java deployment script for WebSphere	Windows
deploy_ear_websphere_ejbws.scr	Java deployment script for WebSphere	Windows
deploy_ear_websphere_web.scr	Java deployment script for WebSphere	Windows
deploy_ear_websphere_wsd.scr	Java deployment script for WebSphere	Windows
deploy_msi_additionalfiles.scr	MSI file deployment script for additional files	Windows
deploy_msi_bat.scr	Drives .net or gui MSI file deployment script	Windows
deploy_msi_net_connectionstrings.scr	MSI file deployment script for .net ConnectionString	Windows
deploy_msi_wix_preamble.scr	Drives .net or gui MSI file deployment script	Windows
genbits.scr	bit setting script	Windows
genbits_check.scr	bit checking script	Windows
genver.scr	VS version checking script	Windows
test.scr	Script used when testing	Windows
qualifyassembly.scr	C# language script containing <qualifyassemblies> which map partial names to full four part names. Included by build_lm_sc.scr and deploy_msi_net.scr.	Windows
build_unix.scr	Main build operation entry point script. Invokes appropriate build script.	UNIX/Linux
build_unix_ddl.scr	DDL install script	UNIX/Linux
build_unix_lm_c.scr	C language module build script	UNIX/Linux
build_unix_ri_c.scr	C language RI build script	UNIX/Linux
genver_unix.scr	version setting script	UNIX/Linux
build_lm_c_nsk_oss.scr	C language module build script	NonStop
build_ri_c_nsk_oss.scr	C language RI build script	NonStop

Tokens

You can reference three types of tokens in a script. They are

- Global

This information is global to the build and deploy processes and usually contains information created during the processing by the script itself.

- Scoped Global

This information is also global but is grouped (scoped) together in pieces to be more easily understood. This is usually location or option information.

- Context Specific

This information is usually referred to inside a FOREACH loop. It usually reflects the data about the Modules from within the ICM files.

The contents of each scoped token prefix type are as follows:

- LOC

These are location tokens. The values for these tokens are defined in the system profile, and overwritten by the user's profile.

- OPT

These are option tokens. The values for these tokens are also defined in the system profile, and overwritten by the user's profile.

Context-Specific Tokens

Information in the ICM files is stored in Generalized Markup Language (GML). This language is very similar to HTML and XML but with a different syntax. It is a method of specifying a hierarchical set of items or nodes. Each node can also contain data (attributes) that describe the node.

As the data is generic and hierarchical, a special token syntax is required to access the data. You must first isolate the script engine down to one specific node using the FOREACH statement.

One aspect of a FOREACH statement is the ability to specify a token variable name to reference the current node in the for loop. You can do this by using the *AS name* clause. After the variable name is defined, it can be referenced in a token to get the node's attributes. For example, if the following FOREACH is executed, the NAME attribute on the EXECUNIT can be referred to as {data.NAME}:

```
{[FOREACH]} EXECUNIT AS data
```

You can use the Nested FOREACH loops to drill down through the GML hierarchy.

Token Delimiters

The following table describes the token delimiters:

Description	Starting Delimiter	Ending Delimiter
Script Token Delimiter	{	}
Script Comment Delimiter	{*	*}
Script Directive Delimiter	{[]}
Script Line Continuation Character	{\}	

Note: An additional category of tokens can be added to scripts to provide additional capabilities not available using standard tokens.

Customizing Scripts

You can modify the CA Gen scripts using any standard ASCII text editor to fulfill the requirements of a particular target system. Modifications can include compiler and linker options, make logic, and the addition of new option tokens.

If adding comments to scripts, you can place a comment as a single line, multiple lines or at the end of the line. Do *not* place comments in between a token or after the continuation syntax.

Note: Editing scripts is not generally recommended. The provided scripts are tested for supported targets and should be sufficient for general use. However, specific user needs can vary or newer target versions can require that existing scripts be modified. In this case, any changes to a script must only be made by a system administrator familiar with any possible consequences that might result from making the changes.

Script Statements

A number of different script statements are used to control the activity within a script. Some script statements allow the use of a looping structure to perform an action on every occurrence of a specified type, while others let you create a module token or control the flow of logic from a compare operation. It is important to ensure that the script still works properly if the option token values are not found and tokens are resolved to blank. The following script statements are supported in this release:

- BLANKLINE
- ERROR
- FOREACH...ENDFOR

- BREAK
- CONTINUE
- IF...ELSEIF...ENDIF
- AND
- CONTAINS
- EQUAL
- EXISTS
- FALSE
- NAND
- NOT_CONTAINS
- NOT_EQUAL
- NOT_EXISTS
- NOR
- OR
- TRUE
- INCLUDE
- OPENFILE...CLOSEFILE
- SET
- VERSION

Chapter 7: Profile Tokens

The Build Tool provides settings to tailor application builds to your specifications in the form of tokens consisting of a key and a value assigned to the key. Tokens are used by the build scripts and substituted with the value assigned when the .MAK file is generated at build time.

You can edit the value of any defined token record (for example, OPT.DBUSER) to use a customer-supplied value (for example, USERID).

Note: You can use environment variables as token values. Environment variables must be formatted as appropriate for their target system.

For example, the following variable can be used for Windows:

```
%MY_ENVIRONMENT_VARIABLE%
```

The following variable can be used for UNIX, Linux, and NonStop:

```
$MY_ENVIRONMENT_VARIABLE
```

The environment variable is evaluated by the Build Tool when the token is processed.

You can take advantage of many profile tokens. The following sections list groupings of token as they are listed in the Profile Manager. Additional information is provided to describe the sets of tokens that may be applicable to your particular application needs.

Token Tables

This section contains tables describing the complete set of profile tokens.

The following profile sections (categories) are described:

- DBMS
- SYSTEM
- OPTIONS
- TARGET
- C
- JAVA
- NET
- MSI_DATA

Values shown are defaults. Values in brackets (< and >) provide a list of available choices for that key. Under the Platform column, All refers to UNIX, Linux, Windows, and NonStop.

The following table describes **DBMS** profile tokens.

Tree-Sub-section	Key	Value	Platform	Description
ORACLE	OPT.DBUSER		Windows, UNIX/Linux	DBUSER should be the value to match a user which has been DBA granted. Example: GRANT CONNECT,RESOURCE TO <i>user</i> IDENTIFIED BY <i>password</i> ; GRANT CREATE TABLESPACE TO <i>user</i> ; GRANT DROP TABLESPACE TO <i>user</i> ;
	OPT.DBPSWD		Windows, UNIX/Linux	DBPSWD should be the value to match a user's password which has been DBA granted. Example: GRANT CONNECT,RESOURCE TO <i>user</i> IDENTIFIED BY <i>password</i> ;
	OPT.DBCONNECT		Windows, UNIX/Linux	DBCONNECT overrides the database name to connect. If blank, the database name stored in the model is used. If set to LOCAL, the connection will be to the default of the DBMS if it is defined by the DBMS.
	LOC.DBLIB	%ORACLE_HOME%\precomp\lib	Windows	The database library location.
	OPT.DBLIB	ORASQL11.LIB	Windows	The database library name.
	OPT.DBSQLLIB		Windows	The database SQL library name; normally blank.
	LOC.DBPATH	%ORACLE_HOME%\bin	Windows	The database binaries location.
	LOC.DBINCLUDE	%ORACLE_HOME%\precomp\public;%ORACLE_HOME%\oci\include	Windows	The database include location.
	OPT.DBPCC	PROC.EXE	Windows	The database precompiler name.

Tree-Sub-section	Key	Value	Platform	Description
	OPT. DBPCCFLAGS	SQLCHECK=SYNTAX MODE=ANSI IRECLLEN=255 ORECLLEN=255 LTYPE=NONE (Windows) IRECLLEN=511 ORECLLEN=511 CODE=ANSI_C LTYPE=NONE MODE=ANSI DBMS=V7 SQLCHECK=SYNTAX PARSE=NONE RELEASE_CURSOR= NO HOLD_CURSOR=YES (UNIX)	Windows, UNIX/Linux	The database C precompiler flags. This is a <i>keyword=value set of flags</i> , each separated by a space. This set of flags can be added to as appropriate, or existing values can be modified. Note: In the case of Oracle, if the value of SQLCHECK is changed from SYNTAX to SEMANTICS, you need to add USER=userid/passwd@dbname if the database is remote, or USER=userid/passwd for a local database.
DB2	OPT.DBUSER		Windows, UNIX/Linux	DBUSER should be the value to match a user which has been DBA granted.
	OPT.DBPSWD		Windows, UNIX/Linux	DBPSWD should be the value to match a user's password which has been DBA granted.
	OPT. DBCONNECT		Windows, UNIX/Linux	DBCONNECT overrides the database name to connect. If blank, the database name stored in the model is used. If set to LOCAL, the connection will be to the default of the DBMS if it is defined by the DBMS.
	LOC.DBLIB	%DB2PATH%\LIB	Windows	The database library location.
	OPT.DBLIB	DB2API.LIB	Windows	The database library name.
	OPT. DBSQLLIB		Windows	The database SQL library name; normally blank.
	LOC.DBPATH	%DB2PATH%\BIN	Windows	The database binaries location.
	LOC. DBINCLUDE	%DB2PATH%\INCLUDE	Windows	The database include location.
	OPT.DBPCC	TIDB2PRP.EXE	Windows	The database precompiler name.
	OPT. DBPCCFLAGS	bindfile datetime iso	Windows, UNIX/Linux	The database C precompiler flags. This is a set of flags, each separated by a space. This set of flags can be added to as appropriate, or existing values can be modified.
MSSQL	OPT.DBUSER		Windows	DBUSER should be the value to match a user which has been DBA granted.

Tree-Sub-section	Key	Value	Platform	Description
	OPT.DBPSWD		Windows	DBPSWD should be the value to match a user's password which has been DBA granted.
	OPT.DBCONNECT		Windows	<p>DBCONNECT overrides the database name to connect.</p> <p>If blank, the database name stored in the model is used.</p> <p>If set to LOCAL, the connection will be to the default of the DBMS if it is defined by the DBMS.</p>
ODBC	OPT.DBUSER		Windows	DBUSER should be the value to match a user which has been DBA granted.
	OPT.DBPSWD		Windows	DBPSWD should be the value to match a user's password which has been DBA granted.
	OPT.DBCONNECT		Windows	<p>DBCONNECT overrides the database name to connect.</p> <p>If blank, the database name stored in the model is used.</p> <p>If set to LOCAL, the connection will be to the default of the DBMS if it is defined by the DBMS.</p>
	LOC.DBLIB		Windows	The database library location.
	OPT.DBLIB	ODBC32.LIB	Windows	The database library name.
	OPT.DBSQLLIB	TIODBC.LIB	Windows	The database SQL library name; normally blank.
	LOC.DBPATH		Windows	The database binaries location.
	LOC.DBINCLUDE		Windows	The database include location.
JDBC	OPT.DBUSER		Windows	DBUSER should be the value to match a user which has been DBA granted.
	OPT.DBPSWD		Windows	DBPSWD should be the value to match a user's password which has been DBA granted.

Tree-Sub-section	Key	Value	Platform	Description
	OPT. DBCONNECT		Windows	DBCONNECT overrides the database name to connect. If blank, the database name stored in the model is used. If set to LOCAL, the connection will be to the default of the DBMS if it is defined by the DBMS.
	LOC. JDBCDRIVERS		Windows	JDBC drivers directory locations.
	LOC. JDBCDRIVERSCL ASSPATH		Windows	Class library path used to locate JDBC drivers.

The following table describes **SYSTEM** profile tokens.

Key	Value	Platform	Description
SCRIPTER_VERSION	2.0	All	The version of Build Tool script language to be used by this version of the Build Tool.
LOC.IEFH	\$IEFH (UNIX/Linux/NonStop) or %GENxx%gen (Windows) Note: xx refers to the current release of CA Gen. For the current release number, see the <i>Release Notes</i> .	All	CA directory location.
LOC.PLATFORM	\$IEFPLATFORM	UNIX/ Linux NonStop	Name of Operating System. Valid values are: HPia64, IEF_AIX, IEF_SOL ,IEF_LINUX, or IEF_NONSTOP.
LOC.SCRIPT	\$IEFH/bt/scripts (UNIX/Linux/NonStop) or %GENxx%gen\bt\scripts (Windows) Note: xx refers to the current release of CA Gen. For the current release number, see the <i>Release Notes</i> .	All	The directory containing the build script files.
OPT.C_SCRIPT_NAME	build_unix_lm_c.scr (UNIX/Linux) or build_lm_c_nsk_oss.scr (NonStop)	UNIX/ Linux NonStop	The C Load Module script filename.

Key	Value	Platform	Description
OPT.C_RI_SCRIPT_NAME	build_unix_ri_c.scr (UNIX/Linux) or build_ri_c_nsk_oss.scr (NonStop)	UNIX/ Linux NonStop	The C RI Triggers script filename.
OPT.DB_SCRIPT_NAME	build_unix_ddl.scr	UNIX/ Linux	The Database script filename.

The following table describes **OPTIONS** profile tokens.

Key	Value	Platform	Description
OPT.DEBUG	NO <YES/NO>	All	Setting this to YES will generate the application with debug.
OPT.HAS_SQL	NO <YES/NO>	Windows	To force the use of the selected DBMS, chosen during generation, to be linked into the Blockmode or Server load modules, regardless of whether or not the generated code contains any embedded SQL statements.
OPT.CBDLIST		All	For component consumptions, set this token to a fully qualified file name which contains a list of Operations Libraries to be included in the linking of the current Load Module or Operations Library.
OPT.INSTALL_TRANCODS	YES <YES/NO>	Windows, UNIX/Linux	Token used to add the transaction codes to the generated AEENV file during the build.
OPT.IEFC		All	Use this token to provide additional compiler and optimization flags.
OPT.IEFLINK		All	Use this token to provide additional linker flags.

Key	Value	Platform	Description
OPT.CMD_FORM	NO (UNIX), YES (Windows) <YES/NO>	Windows, UNIX/Linux	<p>This token specifies the type of application to build, and where to place the executables.</p> <p>For Windows:</p> <p>Setting this to YES causes the application to be built as a console application, leaving all executables and the trigger dll in the model's \c directory. Set this value to NO for Server or Blockmode interactive applications, which will move all built executables and the trigger dll into the model's \c\inqload directory.</p> <p>For UNIX and Linux:</p> <p>Setting this to YES causes the application to be built as a console application, moving all executables and the trigger shared library into the model's /bin subdirectory. Set this value to NO for Server or Blockmode interactive applications, which will move all built executables and the trigger shared library into the model's /inqload subdirectory.</p> <p>Note: This token is not applicable to GUI applications.</p>
OPT.VSVERSION	VS2010	Windows	Represents the Visual Studio version used to build generated applications.
OPT.BITS	32 <32/ 64>	Windows	Represents the compiler bit setting used when building applications.
OPT.LIBTYPE	SHARED <SHARED/ ARCHIVE>	UNIX/ Linux	Applications use either CA Runtime shared libraries or CA Runtime archived libraries (static) when linking. This token is used to determine which style to link to the application. This also impacts which type of RI Triggers library to build (shared or archived).
OPT.COMPRESS_STATICCONTENT	YES <YES/NO>	Windows	When the static content is requested by the browser, the runtime requests the .gz version of the file from the Application Server. This content is then streamed to the browser and automatically de-compressed by the browser.
OPT.COMPRESS_DYNAMICCONTENT	YES <YES/NO>	Windows	All content generated at runtime is compressed using the GZIP compress algorithm. This content is streamed to the browser and automatically de-compressed by the browser.
OPT.INCREMENT_VERSION	YES <YES/NO>	Windows	When building applications, the version number (which can be initially set in application.h) is incremented for each build. This feature can be prevented by setting this token to NO.

The following table describes **TARGET** profile tokens.

Key	Value	Platform	Description
LOC.INSTALL	model_directory	UNIX/ Linux	Directory to place application modules. This directory must contain a trailing delimiter.
LOC.MAKE_DRIVE	model_drive	Windows	Drive which contains the model directory.
LOC.MAKE_DIR	model_directory	Windows	Directory where the make files are located, and the build takes place. This directory must contain a trailing delimiter.
LOC.MAKE_OUTPUT		All	Directory where the model's build output files are located. This directory must contain a trailing delimiter.
LOC.CODE_SRC	model_directory	All	Directory where the model's source and remote files are located. This directory must contain a trailing delimiter.
LOC.CODE_EXE	model_directory	All	Directory where the model's execution files must be created. For server module, it is the Directory which is used as a staging directory for the build of the model's execution files and which are moved to LOC.CODE_SRC\inqload. This directory must contain a trailing delimiter.
LOC.CODE_OBJ	model_directory	All	Directory where the model's object files are located. This directory must contain a trailing delimiter.
LOC.CODE_LIB	model_directory	All	Directory where the model's libraries should be located. This directory must contain a trailing delimiter.
LOC.DDL	model_directory	All	Directory where the model's database files are located. This directory must contain a trailing delimiter.
LOC.RI_TRIG_SRC	model_directory	All	Directory where the model's RI trigger source files are located. This directory must contain a trailing delimiter.
LOC.RI_TRIG_LIB	model_directory	All	Directory where the model's RI trigger libraries should be located. This directory must contain a trailing delimiter.
LOC.RI_TRIG_OBJ	model_directory	All	Directory where the model's RI trigger object files are located. This directory must contain a trailing delimiter.

Key	Value	Platform	Description
LOC.PROD_SRC	model_directory	UNIX/ Linux NonStop	Directory where the production model's source files are located. This directory must contain a trailing delimiter.
LOC.PROD_OBJ	model_directory	UNIX/ Linux NonStop	Directory where the productions model's object files are located. This directory must contain a trailing delimiter.
LOC.OPLIB	model_directory	All	For component consumption, this is the location of any Operations Library(s) that need to be linked into the current Load Module or Operations Library being linked. Defaults to same directory as the Load Module. This directory must contain a trailing delimiter.

The following table describes **C** profile tokens.

Key	Value	Platform	Description
OPT.TYPELIBGUID	{0C8F36F1-848E-11 CE-9C08-02608 CDA5EE3}	Windows	GUID for Windows application library.
OPT.BUILD_AB	NO <YES/NO>	Windows	If set to YES, the build process will separate all action blocks from the {LoadModule}.DLL and create an A_{LoadModule}.DLL that is linked to the {LoadModule}.DLL. If the size of the {LoadModule}.DLL is an issue, this option can be set to YES to reduce the size of the Load Module.
OPT.BUILD_SRC	NO <YES/NO>	Windows	<p>GUI applications require DBMS-specific startup stubs, including a DBMS choice of NONE. If this token is set to YES, the build process will copy the DBMS-specific stub source to the build directory and precompile, compile and link the stub. Otherwise, the DBMS-specific stub executable will be copied into the build directory and merely renamed to {LoadModule}.EXE. Also, the generated Window Manager resource file is bound with the {LoadModule}.EXE in order to provide the executable with the application icons.</p> <p>Note: If both OPT.DEBUG is YES and OPT.BUILD_SRC is YES, the stubs will not build debuggable, since the delivered GUI Runtime is not debuggable. Runtime crashes occur when mixing debug stubs with nondebug runtime libraries.</p>

Key	Value	Platform	Description
OPT.HTMLHELPCOMPI LER	HHC	Windows	Name of the Help compiler used for building .NET applications.
LOC.IEF_BITMAP	%IEF_BITMAP%	Windows	Directory containing the bitmap files for the application. If not set, then the ..\bitmap subdirectory existence is checked; otherwise, the model's source directory is used.
LOC.MQSLIB	C:\Program Files\IBM\ WebSphere MQ\Tools\Lib	Windows	Location of the Websphere MQ libraries.
LOC.EXTERNAL_ LIB		All	<p>This token specifies one or more fully qualified external libraries that are linked into the application.</p> <p>For Windows:</p> <p>When working with pathnames which contain spaces, or linking to multiple external libraries, you need to quote each library and separate the library names with a single space. Also, do not include extrnc.lib in the set of libraries. The Build Tool will automatically search for extrnc.lib in the following locations:</p> <p>{LOC.CODE_SRC}..\extrn {LOC.CODE_SRC}extrn {LOC.CODE_LIB}extrn {LOC.MAKE_DIR}extrn</p> <p>For UNIX and Linux:</p> <p>When linking to multiple external libraries, you need to separate the library names with a single space. Also, do not include extrnc.a in this set of libraries. The Build Tool will automatically search for extrnc.a in the following locations:</p> <p>{LOC.CODE_SRC}../extrn {LOC.CODE_SRC}extrn {LOC.CODE_LIB}extrn {LOC.MAKE_DIR}extrn</p>
OPT.APPMANIFEST	EMBEDDED <EMBEDDED/ EXTERNAL>	Windows	Applications use either an embedded or external manifest. This token is used to determine the location of the manifest for the application.

Key	Value	Platform	Description
OPT.REGISTERAPP	NO <YES/NO>	Windows	On Windows 7, executing REGEDIT.EXE to register a generated application requires elevated privileges and the authorization process suspends build processing. This token is used to determine whether to execute REGEDIT.EXE during the build process.

The following table describes **JAVA** profile tokens.

Key	Value	Platform	Description
LOC.IEFJREDIR	%GENxxJRE% Note: xx refers to the current release of CA Gen. For the current release number, see the <i>Release Notes</i> .	Windows	Location of the Java Runtime.
LOC.JDK_HOME		Windows	Location of the Java Development Kit.
LOC.JAVAEE_HOME		Windows	Location of the Java Enterprise Edition development kit.
OPT.USECLASSPATH	NO <YES/NO>	Windows	Token used to determine whether or not to append %CLASSPATH% to the local classpath for the current build.
OPT.JAVAFLAGS	-Xmx256m	Windows	This token contains the Java compile flags.

The following table describes **NET** profile tokens.

Key	Value	Platform	Description
LOC.NET_EXTERNAL_LIB_DIRECTORIES		Windows	This token specifies additional directories to be used during the compile/link for .NET applications. Locations of external assemblies/modules may be specified here. The locations of external operation library assemblies <i>must</i> be specified here. When working with multiple directories, separate them with a semicolon, as in the following example: c:\user1;c:\user2

Key	Value	Platform	Description
LOC.NET_EXTERNAL_ASSEMBLIES		Windows	<p>This token specifies the external assemblies to be referenced in the .NET generated assemblies. The assembly name may be fully qualified, or just the assembly name. If only the assembly name is given, then the LOC.NET_EXTERNAL_LIB_DIRECTORIES token should be set to indicate the location of the external assembly.</p> <p>When working with multiple external assemblies, separate the assembly names with a semicolon, as in the following example: extern1.dll;c:\user\extern2.dll</p>
LOC.NET_EXTERNAL_MODULES		Windows	<p>This token specifies the external modules to be added into the .NET generated assemblies. The module name may be fully qualified, or just the module name. If only a module name is given, then the LOC.NET_EXTERNAL_LIB_DIRECTORIES token should be set to indicate the location of the external module.</p> <p>When working with multiple external modules, separate the module names with a semicolon, as in the following example: c:\user\extern1.netmodule;c:\user\extern2.netmodule</p>

The following table describes **MSI_DATA** profile tokens.

Key	Value	Platform	Description
OPT.DEFAULTRUNTIMEDIR		Windows	Directory to install the runtime libraries contained in the MSI package. The default is to install the runtime libraries into the same directory as the loadmodule. Preempted at install time if the runtime libraries are already installed.
OPT.PRODUCTLANGUAGE	1033	Windows	The default language of the deployment.
OPT.READMEFILE		Windows	The fully qualified path of an optional Readme file for the deployment (in RTF format).
OPT.EULAFILE		Windows	The fully qualified path of an optional End User License Agreement for the deployment (in RTF format).
OPT.PRODUCTICONFILE		Windows	The fully qualified path of a product icon file displayed in the Add or Remove Programs dialog.
OPT.SPLASHBITMAPFILE		Windows	The fully qualified path of a main splash bitmap for the deployment (480x320 pixels BMP format).
OPT.BANNERBITMAPFILE		Windows	The fully qualified path of a banner at the top of each dialog of the deployment (493x58 pixels BMP format).
OPT.DIALOGBITMAPFILE		Windows	The fully qualified path of a background bitmap used on Welcome and Install-complete dialogs of the deployment (493x312 pixels BMP format).

Key	Value	Platform	Description
OPT.COPYRIGHTWARNING		Windows	Text of a copyright warning message for the deployment displayed on the Welcome dialog.
OPT.WELCOMETEXT		Windows	Text of a welcome message for the deployment displayed on the Welcome dialog.
OPT.UPDATETEXT		Windows	Text displayed on the Exit dialog during deployment.
OPT.SUPPORTURL		Windows	Support URL displayed on the Support Information pop-up window in the Add or Remove Programs dialog.
OPT.SUPPORTPHONE		Windows	Support Phone number displayed on the Support Information pop-up window in the Add or Remove Programs dialog.
OPT.MANUFACTUREURL		Windows	Manufacturer URL displayed on the Support Information pop-up window in the Add or Remove Programs dialog.
OPT.PRODUCTDESCRIPTION		Windows	Text description of the deployed application displayed on the Support Information pop-up window in the Add or Remove Programs dialog.
OPT.PRODUCTCODEPAGE		Windows	The default code page for the deployment.

The following table describes **NonStop** profile tokens.

Key	Value	Description
OPT.NS_TCP_AUTORESTART	2	Number of times that the PATHMON process attempts to restart the TCP. Valid value from 0 through 32,767.
OPT.NS_TCP_CHECKDIRECTORY	OFF <ON/OFF>	Specifies whether the TCP check the SCREEN COBOL TCLPROG directory file (for example, POBJDIR) for the latest version of a called program when executing a SCREEN COBOL CALL statement.
OPT.NS_TCP_CPUS		Primary and backup processors on which the TCPs run. You must set TCP NONSTOP to 1 to have a backup process created. Value is formatted as primary:backup.
OPT.NS_TCP_GUARDIAN_LIB		TCP Library if not using the default (\$SYSTEM.SYSTEM.PATHTCPL).
OPT.NS_TCP_HOMETERM		Name of the terminal that receives the TCP debugging information.

Key	Value	Description
OPT.NS_TCP_INSPECT	OFF <ON/OFF>	Specifies whether you can use the Inspect program to examine the SCREEN COBOL programs running on the terminals controlled by the TCP.
OPT.NS_TCP_MAXINPUTMSGLEN	133	Maximum length in bytes of any unsolicited message that the TCP accepts. Valid value from 0 through 6000.
OPT.NS_TCP_MAXINPUTMSG	0	Maximum number of unsolicited messages that the TCP queues at any one time for all its requesters. Valid value from 0 through 2045.
OPT.NS_TCP_MAXPATHWAYS	2	Maximum number of external PATHMON processes that the TCP can communicate with at one time. Valid value from 0 through 4095.
OPT.NS_TCP_MAXREPLY	32000	Maximum number of bytes permitted for an outgoing SEND message or a server reply message. Valid value from 0 through 4095.
OPT.NS_TCP_MAXTERMDATA	32000	Number of bytes that the TCP allocates for context data for each terminal. Valid value from 2804 to 2,147,483,647.
OPT.NS_TCP_MAXTERMS	2	Maximum number of terminals that the TCP can have open at the same time. Valid value from 0 through 4095.
OPT.NS_TCP_NONSTOP	0 <0/1>	Specifies whether a TCP runs with a backup process and performs normal checkpoint operations.
OPT.NS_TCP_PRIORITY	50	Specifies the priority at which a TCP runs. Valid value from 1 through 199.
OPT.NS_TCP_PROGRAM_NAME	\$SYSTEM.SYSTEM.PATHTCP2	Specifies the TCP object file name.
OPT.NS_TCP_SERVERPOOL	20000	Number of bytes that the TCP allocates for I/O requests and replies between the SCREEN COBOL programs and server processes. Valid value from 10,000 to 30,000.
OPT.NS_TCP_SWAP		Specifies the disk volume name for the temporary file formerly created and managed by the OS for memory swaps of the TCP extended data segment.

Key	Value	Description
OPT.NS_TCP_TERMBUF	1012	Maximum number of bytes that the TCP allocates from the TERMPOOL area for its terminal output buffers. Valid value from 256 through 4095.
OPT.NS_TCP_TERMPOOL	10000	Number of bytes that the TCP allocates in its data area for all terminal I/O buffers.
LOC.NS_PATHWAY_GEN_INSTALL		Guardian location of CA Gen installation (\$volume.subvolume).
OPT.NS_PATHWAY_MAXASSIGNS	20	Maximum number of ASSIGN definitions that you can specify across all the server classes in the Pathway environment. Valid value from 0 through 8191.
OPT.NS_PATHWAY_MAXDEFINES Formula: This value should be set to the number of load modules * 10.	30	Maximum number of DEFINE definitions that you can specify for all server classes in a Pathway environment. Valid value from 0 through 4095.
OPT.NS_PATHWAY_MAXLINKMONS	5	Maximum number of LINKMON processes that can communicate with the PATHMON process at the same time. Valid value from 0 through 255.
OPT.NS_PATHWAY_MAXPARAMS Formula: This value should be set to the number of load modules * 10.	30	Maximum number of PARAM messages that you can specify across all server classes in a given Pathway environment. Valid value from 0 through 4095.
OPT.NS_PATHWAY_MAXPATHCOMS	5	Maximum number of PATHCOM processes that can run simultaneously within Pathway. Valid value from 1 through 100.
OPT.NS_PATHWAY_MAXPROGRAMS	2	Maximum number of PROGRAM descriptions that you can add to the PATHMON configuration file. Valid value from 0 through 4095.
OPT.NS_PATHWAY_MAXSERVERCLASSES Formula: This value should be set to the number of load modules + 5.	8	Maximum number of server class descriptions that you can add to the PATHMON configuration file. Valid value from 0 through 4095.
OPT.NS_PATHWAY_MAXSERVERPROCESSES Formula: This value should be set to the (number of load modules * number of servers) + 10. The number of servers is OPT.NS_SERVER_MAXSERVERS.	16	Maximum number of server processes that you can define for all server classes using the MAXSERVERS option of the SET SERVER command. Valid value from 0 through 4095.

Key	Value	Description
OPT.NS_PATHWAY_MAXSTARTUPS Formula: This value should be set to the number of load modules * 10.	30	Maximum number of server classes that can have STARTUP messages. Valid value from 0 through 4095.
OPT.NS_PATHWAY_MAXTCPS	1	Maximum number of TCP objects that you can add to the PATHMON configuration file. Valid value from 0 through 800.
OPT.NS_PATHWAY_MAXTERMS	2	Maximum number of TERM objects that you can add to the PATHMON configuration file. Valid value from 0 through 4095.
OPT.NS_PATHWAY_NAME		Pathway Name
OPT.NS_PATHWAY_SECURITY	N	Security for the PATHMON environment. Values are the same as Guardian security values (A,G,O,-,N,C,U).
OPT.NS_SERVER_AUTORESTART	1	Number of times that the PATHMON process attempts to restart a server process within a fixed 10-minute interval after an abnormal termination. Valid values from 0 through 32,767.
OPT.NS_SERVER_CPUS		List of processors in which the server processes in this server class run.
OPT.NS_SERVER_CREATEDELAY	20	Maximum amount of time a link manager waits to use an established link to a server class before requesting a new link from the PATHMON process that controls the server class. Value valid from 0 through 16,383 (Seconds).
OPT.NS_SERVER_DEBUG	OFF <ON/OFF>	Specifies whether the server processes in this server class enter debug mode when starting.
OPT.NS_SERVER_DELETEDELAY	10	Maximum amount of time a link between a link manager and a dynamic server process in a server class can remain idle before the link manager automatically returns the link to the dynamic server. Value valid from 0 through 1092 (Minutes).
OPT.NS_SERVER_HOMETERM	\$ZHOME	Name of the home terminal for servers in this server class.
OPT.NS_SERVER_LINKDEPTH	10	Maximum number of links that any one link manager can have to an individual server process in a server class. Server Link Depth. Valid value from 0 through 4095.

Key	Value	Description
OPT.NS_SERVER_MAXLINKS	100	Maximum number of links to an individual server process from all link managers, such as a LINKMON process. Valid value from 0 through 4095.
OPT.NS_SERVER_MAXSERVERS	2	Maximum number of server processes in a server class that can run at the same time. Value valid from 0 through 4095.
OPT.NS_SERVER_NUMSTATIC	1	Maximum number of static servers within this server class. Valid value from 0 through 4095.
OPT.NS_SERVER_OUT	\$0	Specifies the name of the OUT file in the startup message.
OPT.NS_SERVER_PRI	150	Priority at which the servers of this server class run. Valid value from 1 through 199.
OPT.NS_SERVER_SECURITY	N	Specifies the users, in relation to the OWNER attribute, who can access a server class from a Pathsend requester. Values are the same as Guardian file security attributes (A,G,O,-,N,C,U).
OPT.NS_SERVER_SHORT_MODEL_NAME		Server Target SubVolume
LOC.NS_SERVER_TARGET_VOLUME		Server Target Volume
OPT.NS_SERVER_TMF	ON <ON/OFF>	Specifies whether servers in this server class can lock and update data files audited by the TMF subsystem.
OPT.NS_SERVER_TMTSERVER_IN	\$ZHOME	Name of the input file passed to the TMT-SERVER.
OPT.NS_SERVER_TMTSERVER_OUT	\$0	Name of the OUT file in the startup message of the TMT-SERVER.
OPT.MODULE_CATALOG		SQL/MX Module Catalog Name
OPT.MODULE_SCHEMA		SQL/MX Module Schema Name
OPT.MODULE_LOCATION	moduleLocal	SQL/MX Module Location

Make sure that you have set values for the following tokens, since these cannot be defaulted:

- OPT.NS_PATHWAY_NAME
- LOC.NS_PATHWAY_GEN_INSTALL
- LOC.NS_SERVER_TARGET_VOLUME
- OPT.NS_SERVER_SHORT_MODEL_NAME
- OPT.MODULE_CATALOG
- OPT.MODULE_SCHEMA

Database-Specific Tokens

Most applications require a database. The following tokens support the DBMS that your application will use:

- OPT.DBUSER
- OPT.DBPSWD
- OPT.DBCONNECT
- LOC.DBLIB
- LOC.DBPATH
- LOC.DBINCLUDE
- OPT.DBLIB
- OPT.DBSQLLIB
- OPT.DBPCC
- OPT.DBPCCFLAGS

In most cases, the tokens OPT.DBUSER, OPT.DBPSWD, and OPT.DBCONNECT need to be set to connect to and log on to the running database instances. These tokens have no defaults, so it is advisable to set these values and store the profile for future use. In most cases, OPT.DBCONNECT is set using the generated DBNAME provided in the databases ICM file.

For Windows systems only

If OPT.DBCONNECT is set to LOCAL, no database name is used for connections. In this case, it is expected that the user's local environment is set up with a default location to connect to. The method of indicating a default location varies by DBMS.

For UNIX/Linux systems

Only OPT.DBUSER, OPT.DBPSWD, OPT.DBCONNECT, and OPT.DBPCCFLAGS are used. Since database library and binary paths are set by the system administrator, the remaining LOC and OPT tokens above are not used in the UNIX build scripts (UNIX users use the PATH and LIBPATH, SHLIB_PATH, or LD_LIBRARY_PATH environment variables instead. Linux users use the LD_LIBRARY_PATH instead).

The token OPT.DBPCCFLAGS (C) contains a default set of precompiler flags for each particular database. These flags can be modified, added to, or removed.

For Windows systems, the LOC tokens that represent the DBLIB, DBPATH, and DBINCLUDE must be set for each DBMS that is used. These tokens represent the locations of the database binaries, libraries, and include files. The OPT tokens that represent the DBLIB, DBSQLLIB and DBPCC are set to the database library, database SQL library, and database precompiler, if applicable.

For installing DDL using JDBC, the DBCONNECT string needs to be the data source name for the JDBC Driver (that is, for DB2 'jdbc:db2:mydb'; for Oracle 'jdbc:oracle:thin:@myhost:1521:mydb').

For Web Generation using DataSource Objects and EJBs, the DBCONNECT string needs to be the data source name for the defined resource adapter in the Application server (that is, for J2EE 'java:jdbc/MYDB').

Installation-Specific Tokens

The following installation-specific tokens are set based on the installation of the Build Tool and the associated Runtime:

- LOC.IEFH
- LOC.PLATFORM

The token LOC.IEFH defaults to the location of the currently installed CA Gen Runtime, and is used to reference files that have been installed during build processing. The token LOC.PLATFORM is UNIX, Linux, and NonStop specific, and is used to determine which platform the Build Tool is running on (HPUX, AIX, or Solaris for UNIX, and SuSE and Redhat for Linux and NonStop).

Script Name-Specific Tokens

You can specify the build scripts in an alternative location, which provides users the ability to have their own copy of these scripts. The following tokens are script name-specific:

- LOC.SCRIPT
- OPT.C_SCRIPT_NAME

- OPT.C_RI_SCRIPT_NAME
- OPT.DB_SCRIPT_NAME

LOC.SCRIPT defaults to the current location of the Build Tool scripts on Windows, UNIX, and Linux installations. The remaining tokens are for UNIX and Linux systems only, and provide the user the capability of renaming the build scripts. Windows systems do not provide these tokens since there are many more build scripts on Windows systems than on UNIX and Linux systems. For the list of the Windows build scripts, see the appendix "[Build Scripts](#) (see page 79)."

Optional Tokens

Most builds do not require modifications to optional tokens. Standard defaults are overridden when tokens in this category are changed. Avoid modifying optional tokens unless necessary to prevent unexpected side effects. The following optional tokens are available:

- OPT.CMD_FORM
- OPT.INSTALL_TRANCODES
- OPT.CBDLIST
- OPT.VSVERSION
- OPT.BITS
- OPT.LIBTYPE
- OPT.DEBUG
- OPT.HAS_SQL
- OPT.IEFC
- OPT.IEFLINK
- OPT.COMPRESS_STATICCONTENT
- OPT.COMPRESS_DYNAMICCONTENT

The following information applies to optional tokens:

- **OPT.CMD_FORM**

This token specifies the type of application to build, and where to place the executables.

For Windows systems

Setting this token to YES will cause the Blockmode application to be built as a console application, leaving all executables and the trigger dll in the model's \c directory. Setting this token to NO for Server or BlockMode Transactional applications, will move all built executables and the trigger dll into the model's \c\inqlod directory.

For UNIX and Linux systems

Setting this token to YES will cause the Blockmode application to be built as a console application, moving all executables and the trigger shared library into the model's /bin directory. Setting this token to NO for Server or BlockMode Transactional applications will move all built executables and the trigger shared library into the model's /inqlod directory.

This token is not applicable for GUI applications.

- **OPT.CBDLIST**

This token is used to identify the name of a text file that lists component libraries to use at link time for other components or load modules.

For Windows systems

When working with pathnames which contain spaces, you need to quote each file name.

- **OPT.HAS_SQL**

This token is used to force the use of the selected DBMS, chosen during generation, to be linked into the BlockMode or Server load module, regardless of whether or not the generated code contains any embedded SQL statements.

- **OPT.BITS**

With the introduction of Visual Studio 2010 compiler support, the customer can create both 32-bit and 64-bit BlockMode and Server applications. This token controls the bit setting of the compiler and which CA Gen libraries (bitwise) will be linked into the application being built.

Location Tokens

As with optional tokens, most builds do not require modifications to location tokens. Standard defaults are overridden when tokens in this category are changed. A modified target (location) token needs to be terminated with a trailing delimiter (back slash) or the files will be placed in the wrong directory. The following location tokens are available:

- LOC.INSTALL
- LOC.MAKE_DRIVE
- LOC.MAKE_DIR
- LOC.MAKE_OUTPUT
- LOC.CODE_SRC
- LOC.CODE_EXE
- LOC.CODE_OBJ
- LOC.CODE_LIB
- LOC.DDL
- LOC.RI_TRIG_SRC
- LOC.RI_TRIG_LIB
- LOC.RI_TRIG_OBJ
- LOC.PROD_SRC
- LOC.PROD_OBJ
- LOC.OPLIB

By default, all tokens except LOC.MAKE_DRIVE and LOC.OPLIB are set to the current model directory. LOC.MAKE_DRIVE defaults to the current model : drive, while LOC.OPLIB is left blank (also representing the current model directory)

Important! When building a generated application with Dialects, you should not modify the default profile manager target location tokens. They must remain set to *MODELDIR*.

GUI Application Tokens for C Applications

Several tokens are available for GUI applications (C language only). The following tokens are available:

- OPT.TYPELIBGUID
- OPT.BUILD_AB
- OPT.BUILD_SRC

- OPT.HTMLHELPCOMPILER
- OPT.APPMANIFEST
- LOC.IEF_BITMAP
- OPT.REGISTERAPP

The following information applies to some tokens:

- OPT.BUILD_AB

This token is used to separate all action blocks from the LoadModule.DLL. If YES, all action blocks within the .ICM are processed separately, creating a A_LoadModule.DLL that is linked to the LoadModule.DLL. This is used to reduce the size of the LoadModule.DLL.

- OPT.BUILD_SRC

This token is used to rebuild the DBMS-specific stub source during the build process. If YES, the DBMS-specific stub module (STUB*N.*), along with STUBMAIN.CPP, are copied to the model directory, where the DBMS-specific stub is rebuilt and then renamed to the LoadModule.EXE. If NO, the DBMS-specific stub executable in the %GENxx%Gen\VSabc directory is copied and renamed to the LoadModule.EXE.

Note: VSabc refers to the supported version of Visual Studio. Replace VSabc with VS100 for Visual Studio 2010 and VS110 for Visual Studio 2012. xx refers to the current release of CA Gen. For the current release number, see the *Release Notes*.

- OPT.BUILD_SRC

This token is used to indicate whether or not to execute REGEDIT.EXE during a generated application build. On Windows 7, executing REGEDIT.EXE to register a generated application requires elevated privileges, and as a result, suspends the build process.

Note: If OPT.DEBUG is YES and OPT.BUILD_SRC is YES, the stubs do not build debuggable, as the delivered GUI runtime is not debuggable.

Additional Tokens for C Applications

Two additional tokens are available for C language applications. These tokens are:

- LOC.MQSLIB

The token LOC.MQSLIB is self-explanatory.

- LOC.EXTERNAL_LIB

The token LOC.EXTERNAL_LIB specifies one or more fully qualified external libraries that are linked into the application.

For Windows systems

When working with pathnames which contain spaces, or linking to multiple external libraries, you need to quote each library and separate the library names with a single space. Also, do not include extrnc.lib in this set of libraries. The Build Tool will automatically search for extrnc.lib in the following locations:

```
{LOC.CODE_SRC}..\extrn\  
{LOC.CODE_SRC}extrn\  
{LOC.CODE_LIB}extrn\  
{LOC.MAKE_PROC}extrn\
```

For UNIX, Linux, and NonStop systems

When linking to multiple external libraries, you need to separate the library names with a single space. Also, do not include extrnc.a in this set of libraries. The Build Tool will automatically search for extrnc.a in the following locations:

```
{LOC.CODE_SRC}../extrn/  
{LOC.CODE_SRC}extrn/  
{LOC.CODE_LIB}extrn/  
{LOC.MAKE_PROC}extrn/
```

Java Tokens

The following tokens are available for Java applications:

- LOC.IEFJREDIR
- LOC.JDK_HOME
- LOC.JAVAEE_HOME
- OPT.USECLASSPATH
- OPT.JAVAFLAGS

Java Options

For proper compilation of the Java generated code (proxies and Web Generation), you must set the value of the LOC.JDK_HOME key to indicate where the JDK is installed.

Example**Key**

LOC.JDK_HOME

Value

c:\jdk

For Java version 1.2 and later, you do not need to use class path to find installation common jar files. You should place all installation common jar files in %LOC.JDK_HOME%\jre\lib\ext instead of having a CLASSPATH entry for each.

If you still want to use the CLASSPATH, set the value of OPT.USECLASSPATH to YES.

.NET Tokens

The following tokens are available for .NET applications:

- LOC.NET_EXTERNAL_LIB_DIRECTORIES
- LOC.NET_EXTERNAL_ASSEMBLIES
- LOC.NET_EXTERNAL_MODULES

These tokens allow C# builds to reference any external built assemblies and .NET modules. They are typically are used when working with external action blocks or sub-transactional component-based design (CBD) operations libraries.

MSI Data Tokens

The following tokens are available in the MSI_DATA branch for assembling MSI files after an application is built:

- OPT.DEFAULTRUNTIMEDIR
- OPT.PRODUCTLANGUAGE
- OPT.EULAFILE
- OPT.READMEFILE
- OPT.PRODUCTICONFILE
- OPT.SPLASHBITMAPFILE
- OPT.BANNERBITMAPFILE
- OPT.DIALOGBITMAPFILE
- OPT.COPYRIGHTWARNING
- OPT.WELCOMETEXT
- OPT.UPDATETEXT
- OPT.SUPPORTURL
- OPT.SUPPORTPHONE
- OPT.MANUFACTUREURL

- OPT.PRODUCTDESCRIPTION
- OPT.PRODUCTCODEPAGE

These tokens allow the assembled MSI installation data/behaviors to be customized. These tokens apply to both the .NET and GUI MSI installation files, unless specifically indicated within the token descriptions.

The token OPT.PRODUCTCODEPAGE is default to blank, and it will obtain the platform codepage. For example, if MSI is assembled on Arabic Windows and OPT.PRODUCTCODEPAGE is empty, the Build Tool script obtains the Arabic default codepage 1256 from the system.

Note: During processing, the specified codepage is verified against some string data in WiX. If the WiX source contains characters different from what the codepage specifies, the script generates an error message and stops further processing.

Chapter 8: Troubleshooting

As troubleshooting information can vary based on your environment, this appendix contains troubleshooting information for each Build Tool environment.

Problems and Resolutions for all Supported Systems

The following problems might occur when using the Build Tool on all supported systems. Follow the recommendations provided in the action sections to recover normal operation.

Connection Timed Out

Symptom:

A connection timeout occurs when processing a remote build from a Build Tool Client. This is usually indicated in the Build Tool Client message panel with a “Building” statement that is not followed up by a “Build complete” statement.

Reason:

If your PC running the Build Tool Client is outside the network where your server is running the Build Tool Server, the listener socket established by the Build Tool Server cannot be reached by the Build Tool Client. There could be several issues related to being outside the network that can prevent the Build Tool client from reaching the listener socket within the Build Tool server, including firewalls and network access restrictions.

Action:

Ensure that your PC running the Build Tool client is within the network where your server is running the Build Tool Server.

Problems and Resolutions for Windows

The following problems might occur when using the Build Tool on Windows systems. Follow the recommendations provided in the Action sections to recover normal operation.

Stream of Warnings

A stream of warnings appears whenever a model is built.

Reason:

Typically, only errors or lines preceded by multiple asterisks (******* ...) indicate a valid problem when the results of a build are reviewed. Watching the session during a build might provide indications of any errors that are encountered.

Warning levels in the Review output file are enabled so that you can return to the list in case any problems are seen when running the CA Gen application.

The following list contains typical warnings that can be expected at the onset of Review. These warnings are **not** significant.

...warning C4018:'initializing': signed/un-signed mismatch

...warning C4035:'x':no return value

...warning C4101:'x':unreferenced local variable

...warning C4761: integral size mismatch in argument; conversion supplied

...warning C4013:'x':undefined;assuming extern returning int

...warning C4051: type conversion; possible loss of data

Action:

If you prefer that the compiler not display warning messages, make the following modifications to the BUILD_RI_C.SCR and BUILD_LM_C.SCR scripts:

1. Make a backup copy of both scripts.
2. Edit both scripts to change *-W3* to *-W0*.

Error - L1093: ... object not found

Reason:

Reviewing an RI or LM module build reveals the following error:

L1093: ... object not found.

Action:

Perform the following steps:

1. Verify that the SQL precompiler has enough memory and that the referenced object file exists.
2. Rebuild the module.

Compile Step Failed #1

Reason:

The compile step fails and yields an error such as “Bad command or filename.”

Action:

Use a Windows Command Prompt and perform the following checks:

1. Enter *CL* at the prompt to check compiler availability.
2. Enter *LINK* at the prompt to check Linker availability.
3. Enter *RC* at the prompt to check GUI Resource Compiler availability.

Models Will Not Build

Reason:

Various causes might be involved.

Action:

Perform the following steps:

1. Verify your system configuration and ensure that you are using the correct version of the Microsoft Visual Studio compiler.

For more information about the version of the Microsoft Visual Studio compiler, see the Technical requirements document at <http://ca.com/support>.

2. Close all other Windows applications or restart Windows to optimize available resources for the current environment.

Note: A network connection is not required during local builds.

Load Module Failed to Generate

Symptom:

Load module fails to generate, displays the following message:

```
Precompile on P6052435.SQC failed for [model_name].
```

Reason:

This is an indication that certain Oracle precompilation parameters for the current user might not have been properly set.

Action:

1. In the Build Tool Client, bring up the profile that you used to build your application.
2. Modify the OPT.DBPCCFLAGS, changing the IRECLen= and ORECLen= values within the profile token to be large enough for the longest input line. We recommend a value of 260.
3. Save your changes and rebuild.

Compile Step Failed #2

Symptom:

The compile step fails during the building of a module.

Reason:

This is normally an indication that your workstation is not properly configured.

It is likely that a support application is unavailable or not properly configured in the PATH.

Action:

See the software requirements in the *Technical Requirements* sheet and verify that all the necessary software components have been properly built. If it becomes necessary to rebuild one or more of the items on the list, see the *Installation Guide for Distributed Systems*.

Close all other Windows applications or restart Windows, then rebuild the module.

Error - Can't Find MSVCRXX.LL

Reason:

You receive the following error when running a CA Gen application on a production workstation:

Can't Find MSVCRXX.LL

Action:

When an application is constructed with any version of Microsoft Visual Studio, the appropriate MSVCRxx.DLL must also be distributed as the application is distributed to other production workstations.

Exception caught: No such file or directory

Reason:

You receive the following similar missing directory error when building a CA Gen application:

```
Exception caught: com.ca.genxx.bt.scripiter.ScriptException:
[Function: Scripiter.processOpenFile] ScriptException "
[Function: Environment.openFile]IOException opening file:
"%GENxx%\Gen\sample.ief\make\CASCADE.MAK",
(%GENxx%\Gen\sample.ief\make\CASCADE.MAK (The system cannot find the path
specified))" at: file: %GENxx%\Gen\Gen\bt\scripts\build_ri_c.scr #: 213

"{{[OPENFILE]}} {LOC.MAKE_DIR}{RI_NAME}.MAK"
```

Compose Failed

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

Action:

Ensure that all subdirectories referenced through the TARGET profile tokens are created prior to processing any builds. The Build Tool is not responsible for the creation of directories. Create the missing directory and rebuild.

Error - Test FAILED statement in Message Panel

Reason:

You receive the following statement in the Message Panel after attempting to test the application through the Test Button:

Load Module Test: Test FAILED returned errorlevel=-XXXXXXXXXX

Action:

Outside the Build Tool, in a Command Window, cd to the model directory and execute the same Load Module manually. More than likely you will receive an ERROR message pop-up indicating that you have failed to start the application because a DLL was not found. Please ensure that you have the path of the missing DLL in your PATH environment variable.

Error - ODBC BlockMode Application fails to start

Reason:

If the ODBC fails to connect, the error message flashes by too quickly to be read. Connection failures are often the result of improper connection settings in the user's profile during the application build.

Action:

1. Use the Window's ODBC Data Source Administrator to verify the *Name*, *User ID*, and *Password* necessary to connect to the DBMS.
2. These values should be set in the Build Tool profile under DBMS, ODBC:
3. OPT.DBCONNECT, OPT.DBUSER, and OPT.DBPSWD.
4. Rebuild the application after correcting the profile values.

Build Scheduling Status for All Modules Being Processed

Symptom:

All modules that have been selected to build are marked with status *Build Scheduling* but nothing is building.

Reason:

The Build Tool client is sharing the same port as the Build Tool server.

Action:

Either restart the Build Tool client with a different port, or kill the Build Tool server and rebuild the modules previously selected.

Remote Host Not Available Error Message

Symptom:

When attempting to connect to a remote server from the Build Tool client, you receive a message that states "Remote Host Not Available".

Reason:

The port being used to connect to the remote Build Tool server does not match that of the remote Build Tool server.

Action:

Use the matching port number when connecting to remote Build Tool server.

Duplicate Filenames Ignored

Symptom:

When transferring several files from multiple directories, there is a chance that a file with the same name could be transferred, that is, when working with dialects. In this case, the duplicate file is not transferred to the target directory.

Reason:

The duplicate is not copied to prevent writing over the same-named filename in the target directory.

Action:

Be more selective in the files being transferred.

Problems and Resolutions for UNIX, Linux, and NonStop

The following problems might occur when using the Build Tool on UNIX, Linux, and NonStop systems. Follow the recommendations provided in the action sections to recover normal operation.

Unresolved Symbol

Symptom:

An unresolved symbol is received at application link time.

Reason:

Typically, an application link is generated with a well-known list of libraries that are defined in the Build Tool's profile. Even if all of the libraries are found during the link, if the library is incorrect or out-of-date, a symbol can end up unresolved.

Also, if there are External Action Blocks linked into the application, the EAB could use a function that does not exist in any of the well-known list of libraries.

Action:

Perform any of the following steps:

- Check the MAKE file for the correct list of libraries.
- Make sure that the libraries exist.
- Use the LOC.EXTERNAL_LIB token if necessary.

Command-Line Application

Symptom:

Application hangs when executed from the command line.

Reason:

The default style of block mode application is a transactional application, which is executed through the AEF utility. A token will need to be set to flag the Build Tool to create a command-line application.

Action:

Set the token OPT.CMD_FORM to YES.

Rebuild your application.

Output File Not Updated

Symptom:

Generated compile output file <module>.out is not being updated when builds are repeated.

Reason:

The *noclobber* shell variable controls whether or not to overwrite existing files when you are redirecting output to that file. The Build Tool scripts redirect output to <module>.out during the build process.

By default, the build scripts *unset noclobber* in the generated MAKE files to enable overwriting.

Action:

Ensure that *unset noclobber* is defined in the generated <module>.mak file.

Compile/Link Failed #1

Reason:

An aliased variable causes a shell script command in the generated MAKE file behavior incorrectly, which results in a failed build.

By default, the build scripts *unalias ** in the generated MAKE files to prevent aliased variables from impacting the generated MAKE file.

Action:

Ensure that *unalias ** is defined in the generated <module>.mak file.

Make: line too long

Symptom:

Generated compile output file <module>.out contains the following statement:

```
make: line too long
```

The target (library or application executable) is not created.

Reason:

When numerous modules are being built into an RI trigger library, a CBD library, or an application executable, and explicit paths are used for these modules, the length of the make statement generated to create the library or executable exceeds the system's default line limit.

Action:

Shorten the module paths.

Or

Modify the tokens that point to the location of the modules being used (LOC.CODE_OBJ or LOC.RI_TRIG_OBJ) for the library creation or executable creation, and add a cd {LOC.CODE_OBJ} or cd {LOC.RI_TRIG_OBJ} to the appropriate build script prior to the library creation command or execute link command.

Make: don't know how to make <member-name>.pc

Symptom:

Generated compile output file <module>.out contains the following statement:

```
make: don't know how to make <member-name>.pc
```

The target (library or application executable) is not created.

Reason:

Use of CABs (Command Action Blocks) can cause build timing issues, since multiple load modules are referencing a common generated Action Block. If the load modules are built out-of-order, then the CAB may not be present at the time of the first load module's build.

Action:

Determine whether the source files are all there. If so, check whether a required CAB might not be available yet because it is physically stored in a remote file that has not been split yet. If that is not the answer, determine whether the code was generated correctly.

Exception caught: No such file or directory

Reason:

You receive the following similar missing directory error when building a CA Gen application:

```
Exception caught: com.ca.genxx.bt.scripiter.ScriptException:
[Function: Scripiter.processOpenFile] ScriptException "
[Function: Environment.openFile]IOException opening file:
"/home/ptauto/gen/oracle/cbd05/make/CASCADE.scrpt",
(/home/ptauto/gen/oracle/cbd05/make/CASCADE.scrpt (No such file or directory))" at:
file: /home/ptauto/it/genxx/runtime/bt/scripts/build_unix_ri_c.scr line #: 52
```

```
"{{[OPENFILE]}} {LOC.MAKE_DIR}{RI_NAME}.scrpt"
```

Compose Failed

Action:

Ensure that all subdirectories referenced through the TARGET profile tokens are created prior to processing any builds. The Build Tool is not responsible for the creation of directories. Create the missing directory and rebuild.

Build Scheduling Status for All Modules Being Processed

Symptom:

All modules that have been selected to build are marked with status *Build Scheduling* but nothing is building.

Reason:

The Build Tool client is sharing the same port as the Build Tool server.

Action:

Either restart the Build Tool client with a different port, or kill the Build Tool server and rebuild the modules previously selected.

Index

.

.NET tokens • 109

A

Additional Files

 EAR File Assemble Details • 60

 MSI .NET Assemble Details • 73

Additional Files, dialog • 60, 73

Application Server dialog • 52

ASP.NET dialog • 65

Assemble option • 49

assembly

 EAR file • 49

 MSI and GUI • 62

assembly function

 packaging applications • 9

 Windows-only availability • 15

B

build function • 10

Build Log • 14

Build Tool

 assembly function • 10

 build function • 10

 building applications • 9

 clean function • 10

 Command Line Client • 23

 compiling source code • 9

 configuration management • 10

 executing • 9

 file transfer function • 10

 functionality • 10

 GUI Client menu bar • 34

 GUI Client toolbar • 34

 module panel fields • 37

 profile management • 10

 review function • 10

 running server as task on Windows • 21

 security considerations • 20

 split function • 10

Build Tool Client • 14

Build Tool GUI Client, terminology • 10

Build Tool Server • 14

building applications, prerequisites • 17

business application, building • 14

C

C applications, additional tokens • 107

clean function • 10

Command Line Client

 benefits • 23

 commands • 26

 invoking in UNIX and Linux • 24

 invoking in Windows • 24

 options • 24

Command Line Client commands

 BUILD • 27

 CLEAN • 28

 EXIT • 29

 HELP • 29

 QUIT • 29

 SHUTDOWN • 31

 SPLIT • 29

 STATUS • 30

commands, Command Line Client • 26

configuration

 module panel • 42

 Options dialog • 42

configuration management • 10

context-specific tokens • 82

D

database-specific tokens • 102

DataSources dialog • 70

dialogs

 Additional Files • 60, 73

 Application Server • 52

 ASP.NET • 65

 DataSources • 70

 GAC • 69

 General in EAR File Assemble Details panel • 50

 General in MSI .NET Assemble Details panel • 64

 Options • 42

 Tracing in EAR File Assemble Details • 59, 60

directory tree, Graphical User Interface (GUI) Client
 • 37

E

EAR File Assemble Details

- Additional Files • 60
- Application Server dialog • 52
- Cross-Context Flows • 58
- General dialog • 50
- Tracing • 59, 60
- WAR dialog • 54
- Web Gen Client • 56

EAR File Assemble Details panel • 50

EAR file assembly • 49

Enterprise Archive (EAR) file assembly • 49

F

file transfer function • 10

G

GAC dialog • 69

General dialog, EAR File Assemble Details • 50

Graphical User Interface (GUI) Client

- directory tree • 37
- main panel entry fields • 37
- module fields • 37
- module panel • 41

GUI application for C tokens • 106

GUI assembly • 62

I

ICM file, processing • 14

Install Control Module (ICM) file • 12, 17

installation-specific tokens • 103

J

Java options • 108

Java tokens • 108

L

location tokens • 106

M

module panel

- configuring • 42
- displaying full screen • 41
- moving fields • 43
- replacing with search panel • 41

module panel fields, Graphical User Interface (GUI) Client • 37

MSI .NET Assemble Details

- ASP.NET dialog • 65
- DataSources dialog • 70
- GAC dialog • 69
- General dialog • 64

MSI assembly • 62

MSI data tokens • 109

O

optional tokens • 104

P

panels

- EAR File Assemble Details • 50
- MSI .NET Assemble Details • 62

prerequisites

- building applications • 17
- establishing target directory structure • 18
- generating source code • 17
- installing third-party products • 17

profile management • 10

profile tokens table • 85

Profile Tree • 43

profiles, selecting categories • 43

R

Remote (RMT) file, unzipping • 10

review function • 10

RMT file • 10, 12, 17

S

scriptname-specific tokens • 103

scripts

- customizing • 83
- directives • 83
- locating • 79
- names • 79
- statements • 83
- tokens • 79

security considerations, securing commands or paths • 20

server mode

- starting in UNIX or Linux • 19
- starting manually • 19

source code, generating • 17

split function • 10, 19

statements, scripts • 83

T

target directory structure, establishing • 18

terminology, Build Tool GUI Client • 10

third-party products, installation required • 17

tokens

- .NET • 109

- additional for C applications • 107

- context-specific • 82

- database-specific • 102

- GUI application for C • 106

- installation-specific • 103

- Java • 108

- location • 106

- MSI data • 109

- optional • 104

- profile • 85

- scriptname-specific • 103

tokens (Windows)

- categories • 82

- delimiters • 83

Tracing dialog • 59, 60, 72

troubleshooting

- UNIX and Linux • 117

- Windows • 111

U

UNIX or Linux

- starting server mode • 19

- troubleshooting • 117

W

WAR dialog, EAR File Assemble Details • 54

web archive (WAR) dialog • 54

Web Gen Client dialog • 56

Windows

- running server as a task on • 21

- troubleshooting • 111