

CA Gen

Distributed Processing - WebSphere MQ User Guide

Release 8.5



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 © 2013 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:

- AllFusion® Gen
- CA Gen
- COOL: Gen
- Advantage™ Gen
- Unicenter® Software Delivery

Contact CA Technologies

Contact CA Support

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

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

Providing Feedback About Product Documentation

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

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

Contents

Chapter 1: Introduction	9
CA Gen Distributed Processing Application	9
WebSphere MQ Concepts and Terminology.....	9
Client Types.....	11
Message and Queue Types	12
Supported Application Behaviors.....	13
Supported Processing Options.....	14
Chapter 2: Developing CA Gen WebSphere MQ Applications	15
Distributed Processing Client Implementation	16
Distributed Processing Server Implementation	17
Non-z/OS Environments.....	18
Server-to-Server Processing	19
z/OS WebSphere Support for Distributed Processing Servers	19
z/OS CICS Environments.....	20
z/OS CICS: The TDC.....	20
z/OS IMS Environment	21
Chapter 3: Specifying Properties for WebSphere MQ Applications	23
Packaging and Construction Attributes.....	23
Specify Properties Through the Toolset	24
Open Cooperative Packaging Tool	24
Set Business System Defaults.....	25
Specify the Properties for Individual Server Managers.....	26
Set MQSeries Properties	28
Specify Properties Through the CSE Construction Client	29
Set Server Environment Parameters.....	30
Set MQSeries Properties	32
Chapter 4: Generating and Building Cooperative Code for WebSphere MQ	33
Override Generation Defaults	33
Overriding in the Toolset	33
CSE Construction Client.....	35
Construct Cooperative Applications.....	36
Windows Build Tool Setup	36

UNIX Build Tool Setup	38
z/OS IT	38
Chapter 5: Testing WebSphere MQ Applications	39
Diagram Trace Utility	39
Regenerate for Changes	40
Application Testing	40
Enable Diagram Trace Utility	40
Access and Use Diagram Trace Utility	41
Multi-user Diagram Trace Utility Support	41
Regenerate Remote Files After Testing	42
Chapter 6: Configuring and Deploying WebSphere MQ for CA Gen	43
How to Configure MQSeries for CA Gen Clients	43
Configure Client Connection	43
Configure the Local Queue	44
Configure the Reply-To Queue	45
Configuring WebSphere MQ for CA Gen Servers	45
Configure z/OS Servers	45
Install the TDC	46
Triggering and the TDC	48
Transactional Security with the TDC	49
Initiate Multiple TDCs	49
Define the TDC to MQSeries	49
Configure Windows and UNIX Servers	51
Start Server Manager as a Daemon Task	51
Start a Server Manager through an WebSphere MQ Trigger Monitor	52
Appendix A: User Exit Functions	53
Non-z/OS WebSphere MQ Client Exits	53
C Language Exits	53
Java Language Exit	54
.NET Language Exit	55
Non-z/OS WebSphere MQ Server Exits	55
CI_MQS_DPS_Exit	56
CI_MQS_DynamicQName_Exit	56
CI_MQS_DPC_setReportOptions	56
z/OS WebSphere MQ Server Exit TIRMQTDX	56
TIRMQTDX	57

Appendix B: Flow Diagrams	59
Appendix C: z/OS CICS TDC Messages and Codes	61
Appendix D: Using the Application.ini File	69
Application.ini Contents	69
Index	71

Chapter 1: Introduction

The *Distributed Processing - WebSphere MQ User Guide* explains how to develop, generate, build, and deploy a CA Gen cooperative code application using WebSphere MQ.

This section contains the following topics:

[CA Gen Distributed Processing Application](#) (see page 9)

[WebSphere MQ Concepts and Terminology](#) (see page 9)

CA Gen Distributed Processing Application

CA Gen supports various types of client/server applications, including Distributed Process (DP) client/server applications. The main characteristic of a DP application is its division into two or more application components, where each application component is designed to perform a particular function for the overall application.

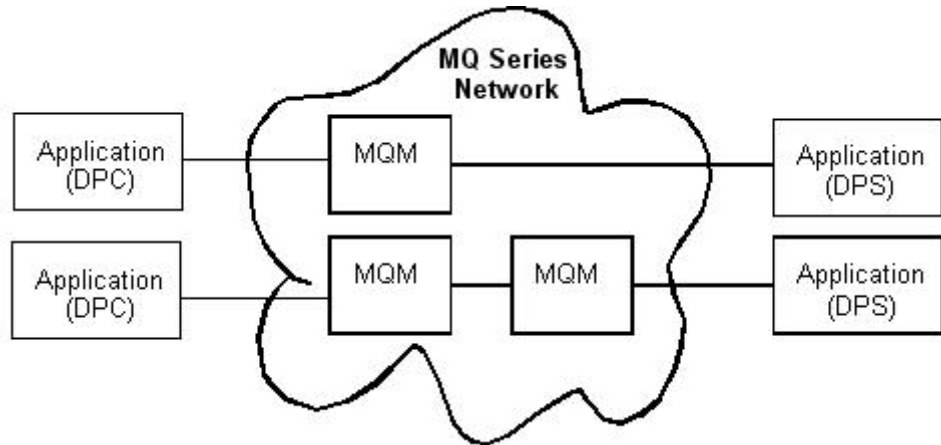
Note: For more information about the characteristics of a DP application, see the *Distributed Processing - Overview Guide*.

WebSphere MQ Concepts and Terminology

Before exploring how to define a CA Gen distributed processing application to use WebSphere MQ, it is important to understand the concepts and terminology of WebSphere MQ. This section presents the WebSphere MQ topics that have a direct influence on the definition and execution of a CA Gen WebSphere MQ DP application.

Note: If you have an understanding of WebSphere MQ, you can skip this section.

WebSphere MQ is an integrated component in the family of communication server products from IBM, and is the Message Oriented Middleware (MOM) of IBM. It simplifies the task of connecting applications across unlike operating environments. WebSphere MQ applications communicate in a loosely coupled manner. That is, rather than being components bound together through some transport protocol, each application component communicates to a WebSphere MQ component known as a queue manager.



A collection of queue managers can be deployed to support the distribution of application components across a network, where each WebSphere MQ queue manager is responsible for forwarding messages to the queue manager associated with the target of a message request. The receiving queue manager can be on the same physical machine, or on a different machine that is connected through a network. The machines and operating environments do not have to be the same type as the queue manager accepting the request, they can be any type that is supported by WebSphere MQ.

The queue manager that accepts a request from an application component is known as the local queue manager (also called the source queue manager). The queue manager associated with the target of a message request is known as the remote queue manager (or target queue manager).

Note: If an application component making a request and the application component that is the target of the request are associated with the same queue manager, then the same queue manager that acts as both local and remote queue manager.

The components of a WebSphere MQ application use a common application programming interface (API) known as the Message Queue Interface, or MQI, to communicate with their associated queue managers. Therefore, before an application component can issue any MQI calls, it must be connected to a queue manager. After the application is connected to a queue manager, the queue manager processes all MQI calls issued by the application. For the application, the queue manager is synonymous with the network, and takes care of all routing and delivery issues.

Each application component defines the data to be exchanged, where the data is handled as a string of bytes that has meaning to the communicating components. The application data, together with a collection of WebSphere MQ control information (a message descriptor) comprise a WebSphere MQ message, which is subsequently placed into one of the various WebSphere MQ queues.

WebSphere MQ queues exist independently of the applications that use them. A queue is a data structure that stores messages, and each queue has queue attributes that determine what happens when an application references the queue through an MQI call. Each queue belongs to, and is maintained by, a given queue manager, and is a local queue to that queue manager. By contrast, a remote queue is a queue that belongs to another queue manager. A given queue manager puts the messages it receives onto the appropriate queue, as directed by the application, through the MQI.

Client Types

WebSphere MQ provides two ways to accept MQI requests from an application program. They are:

- Locally

A WebSphere MQ server is a fully capable queue manager, able to process any request it receives.

- Remotely

The WebSphere MQ server accepts MQI calls that originate from application programs executing on different physical machines. This mechanism requires those applications to use the WebSphere MQ client software.

The WebSphere MQ Client software is provided as part of the WebSphere MQ product and can be installed on a machine without having to install an entire WebSphere MQ queue manager. This client software accepts MQI requests from application programs, and passes the MQI requests to a WebSphere MQ server executing on another processor. When a client application component issues an MQI call, the client formats the parameter values of the call into an MQI request and sends the request to the server. The server receives the request, performs the action specified in the request, and sends a response back to the client.

Note: Whether the WebSphere MQ application is a client or server, the MQI remains the same and the application code does not change. However, it is the library used at link-edit time that determines if the application communicates with a WebSphere MQ server directly or uses the WebSphere MQ client software.

Message and Queue Types

WebSphere MQ defines the following types of messages:

- Datagram
A simple message with no expected reply.
- Request
A message with an expected reply.
- Reply
The answer to a request.
- Report
A message that describes an event such as an error.

Each application component can access one or more queues to process a flow of one or more messages through its queue manager. For example, an application component can have a request message placed in a target queue managed by a remote queue manager. The same application receives the reply to the request from a local queue managed by its associated queue manager. The key thing to note here is that there does not have to be a network connection between the application components, only a connection to their respective queue managers.

As WebSphere MQ supports different types of messages. It also supports different types of queues. Each queue must have a unique name within a given queue manager. Before an application can make use of a queue, the queue must first be created and made known to its owning queue manager. Additionally, an application must open the queue so that it can be accessed for its expected usage (such as input, output, and inquiry). Each queue has many attributes that define the allowed behavior for the queue to its owning queue manager.

The queue type attribute helps establish the intended use of the queue. WebSphere MQ supports the following types of queues:

- Local Queue
A queue owned by the queue manager to which a given application is connected. Application programs can Put messages to, or Get messages from, local queues.
Additionally, each queue manager can have several special purpose local queues that support the processing of application messages by the queue manager.
Note: Applications cannot get messages from remote queues.

- Remote Queue

A queue owned by a queue manager other than the local queue manager. A local queue manager defines a remote queue with only sufficient information to locate the queue. The queue manager that owns the queue manages all attributes for the queue. Remote queues can only be used to Put messages. It is not possible to Get messages from remote queues.

- Alias Queue

An application processes an alias queue as though it were an actual queue. The owning queue manager maps any request for the alias queue to the actual queue.

- Model Queue

A model queue is not an actual queue. It acts as a template containing a set of queue attributes. An application uses a model queue when it needs to create a dynamic local queue.

- Dynamic Queue

A temporary local queue that is created on demand by an application request. The model queue, specified when the dynamic queue is created, provides the collection of queue attributes that are needed by a queue manager before it can create the dynamic queue.

Supported Application Behaviors

WebSphere MQ includes support for the following application behaviors:

- Simple Messaging

Applications that use simple messaging do not expect a reply. The application sends a datagram message to a target queue. This type of application behavior is also known as fire and forget. Simple messaging is often used for event logging.

- Request/Response

Applications using request/response behavior make use of both the request and reply message types. The requesting application adds control information to the request message to identify the name of the queue manager and the queue to which the corresponding response should be sent. The application processing the request uses the reply to control information to return the response to the requesting application.

- **Event Driven**

WebSphere MQ can initiate applications in response to a message arriving on a particular queue. The event is represented by the contents of the message. WebSphere MQ drives the application associated with the queue, and the queue manager defines certain conditions as constituting trigger events. When a queue that is configured for triggering receives a trigger event, the queue manager sends a trigger message to a local queue, which is known as an initiation queue. The presence of a trigger message on the initiation queue indicates that a triggered event occurred.
- **A trigger monitor is an application that monitors an initiation queue associated with a queue manager. When a trigger message arrives on the initiation queue, the trigger monitor retrieves it and, typically, the trigger monitor starts an application that is specified in the message on the initiation queue.**
- **Time Independent**

With message queuing, the exchange of messages between sending and receiving programs is time independent. That is, the sending and receiving applications are decoupled so that the sender can continue processing without having to wait for the receiver to acknowledge the receipt of the message. The target application does not need to be running when the message is sent, as it is able to retrieve the message after it starts.

Supported Processing Options

WebSphere MQ lets an application designate which message it will process from a designated queue. You can select the messages based on:

- MsgId
- Correlation ID
- Both MsgId and Correlation ID
- No criteria

In each case, the message returned is the first message on the queue that satisfies the selection criteria. If the application is using no criteria, the first message on the queue is returned.

Note: For more information about WebSphere MQ, see the appropriate IBM WebSphere MQ documentation.

Chapter 2: Developing CA Gen WebSphere MQ Applications

To understand how CA Gen uses WebSphere MQ, you must understand the following concepts:

- A CA Gen Distributed Processing (DP) application is a client/server application, where the client exchanges a cooperative flow request with the server.
- CA Gen applications make use of the Request/Response type of application behavior, exchanging WebSphere MQ request and reply messages. Client interaction with servers is considered to be time dependent. A CA Gen application is considered tightly coupled, in that the client and server are considered to be logically bound together while the cooperative flow is processed.
- CA Gen applications do not use WebSphere MQ syncpoint coordination. Message Put and Get requests operate outside the WebSphere MQ unit-of-work protocol. Put requests are immediately written to the queue, and Get requests immediately remove messages from the queue. That is, no application syncpoint is necessary for these operations to take effect.
- CA Gen WebSphere MQ applications do not require that their queues be persistent. The persistence attribute for a given message is determined by the default persistence value (DEFPSIST) associated with the target message queue. The queue manager maintains the attributes of the target queue.

Note: For more information about WebSphere MQ, see the appropriate documentation from IBM.

This section contains the following topics:

[Distributed Processing Client Implementation](#) (see page 16)

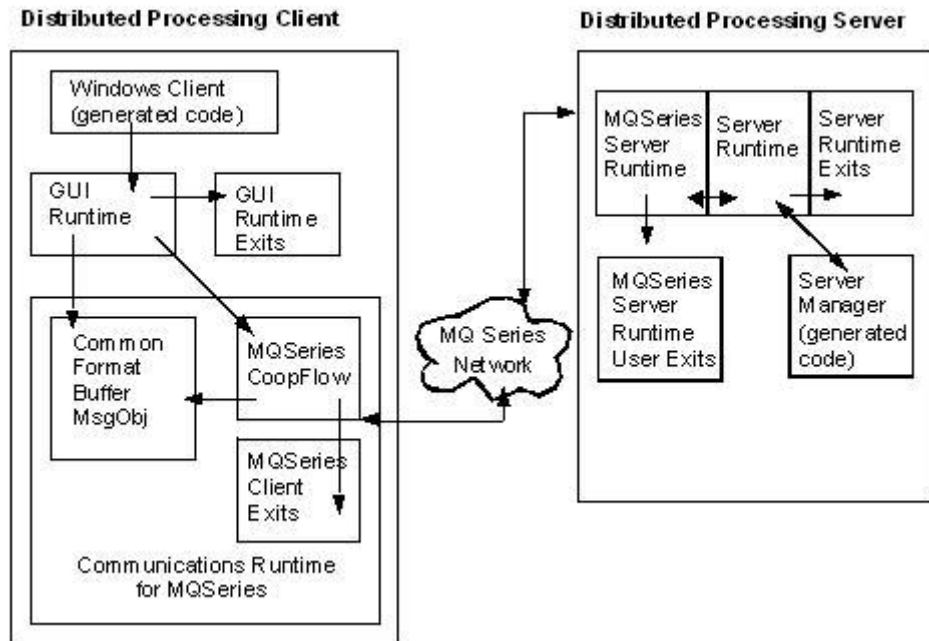
[Distributed Processing Server Implementation](#) (see page 17)

[Server-to-Server Processing](#) (see page 19)

[z/OS WebSphere Support for Distributed Processing Servers](#) (see page 19)

Distributed Processing Client Implementation

The following diagram shows an example of a Windows GUI Client communicating to a CA Gen WebSphere MQ Server. This diagram depicts the relationship of the application components and WebSphere MQ when processing a cooperative flow.



CA Gen DPC applications initiate cooperative flows to DPS applications. When a window manager is generated, the set of server environment parameters associated with the server manager that contains the target Pstep determines the transport type used by the DPC, for a given flow. That is, the communication type associated with the server manager must be set to WebSphere MQ before the window manager is generated. The subset of server environment parameters describing WebSphere MQ values are also important, as they control the interface between the DPC and WebSphere MQ so that the DPC can communicate with the Psteps located within the given server manager.

The set of server environment parameters describing WebSphere MQ attributes define the following:

- WebSphere MQ Client Type

Distributed Processing Client (DPC) applications can dynamically load one of two WebSphere MQ libraries to provide access to their local queue managers. The value of the client type parameter allows the DPC to communicate directly with a WebSphere MQ server, or to make use of the WebSphere MQ client software. The default causes the supporting runtime to dynamically load a CA Gen runtime library that uses the WebSphere MQ client software.

- Queue Manager Name

Defines the name of the local queue manager. The default queue manager is defined within WebSphere MQ configuration.

- Queue Name

Defines the queue name associated with the server manager containing the target Pstep. The default is to use the name of the associated server manager.

- Reply Queue Name

Provides the name of the queue that defines the reply-to queue. By default the reply-to queue parameter defines a dynamic queue, and has the value:

```
SYSTEM.DEFAULT.MODEL.QUEUE
```

If the specified reply queue name provides the name of a model queue, WebSphere MQ creates the reply-to queue as a dynamic reply-to queue. The dynamic reply-to queue takes its attributes from a specified model queue. The name of the resulting dynamic queue defaults to:

```
username.taskid.threadid.*
```

You can override the default value using a CA Gen provided user exit. The user exit is specific to the language in which the MQ client is implemented.

- Clients implemented in C need to use the `CI_MQS_DynamicQName_Exit()` user exit function located in the `cimqplex.c` source file.
- Clients implemented in Java need to use the `getDynamicQName()` method contained within the `MQSDynamicCoopFlowExit` class.

Note: For more information about these user exits, see the *User Exit Reference Guide*.

Optionally, the reply-to queue parameter can specify the name of a local queue. MQSeries configures the attributes of the local queue, which can be defined to its queue manager as being shared or non-shared.

Distributed Processing Server Implementation

You can deploy CA Gen DPS applications built for WebSphere MQ to a variety of target environments that include Windows, UNIX, z/OS CICS, and z/OS IMS, only if the interaction is with a WebSphere MQ server that supports a fully capable queue manager. DPS applications are built using only the WebSphere MQ server library. So, you cannot deploy them to environments that only use the WebSphere MQ client software, as explained in Client Types.

Due to the differences in the target environments, WebSphere MQ server manager behavior, from a CA Gen perspective can be categorized into one of the following:

- UNIX or Windows
- z/OS (CICS and IMS)

The following subsections explain WebSphere MQ server managers within these two categories.

Non-z/OS Environments

The non-z/OS WebSphere MQ environments support a CA Gen DPS application execution in one of two ways. You can invoke a server manager by using one of the following methods:

- Started as a long-running daemon task.
- Invoked by a WebSphere MQ trigger monitor.

The difference between these two methods of execution is in how you invoke the server manager executable, and includes differences in the startup arguments. For information about the commands used to start the daemon, see [Configuring Windows and UNIX Servers](#) (see page 51).

After the server manager starts, the server manager completes the following tasks:

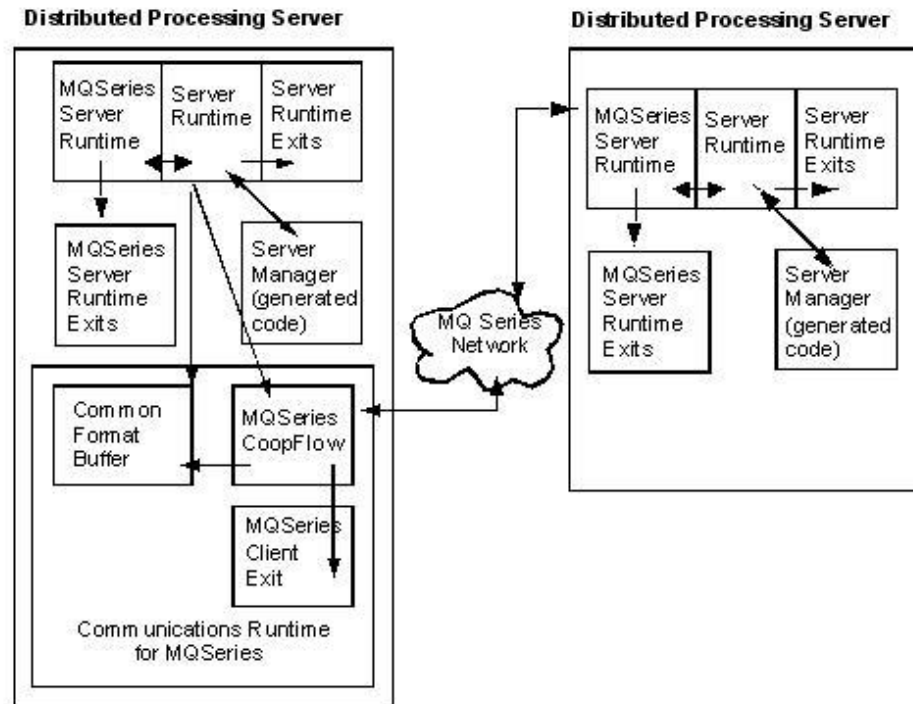
1. Connects to its queue manager
2. Processes a request messages that has been delivered to the queue manager's local queue
3. Extracts the message ID and reply-to queue name from the request message
4. Incorporates the message ID into a response message (to become the correlation ID), and returns a reply message to the reply-to queue name.

The correlation ID in the reply message allows the CA Gen DPC to retrieve the corresponding response to a given request when the requesting DPC application is sharing a reply-to queue.

The non-z/OS CA Gen WebSphere MQ server managers support server-to-server flows using the WebSphere MQ Communication Runtime (the CFB MsgObj and WebSphere MQ CoopFlow). The server manager that initiates the flow takes on the role of the DPC.

Server-to-Server Processing

The following diagram shows the relationship of application components and WebSphere MQ when processing a server-to-server cooperative flow:



The two distinct CA Gen WebSphere MQ runtimes that are involved in processing a server-to-server flow are as follows:

- WebSphere MQ server runtime is linked with the server manager, allowing the server manager to receive message requests and to return reply messages through its connected WebSphere MQ Queue Manager.
- Communications runtime for WebSphere MQ allows a DPS server application to flow to another DPS. This is the same runtime that a DPC application uses (the CFB MsgObj and WebSphere MQ CoopFlow).

Server to server MQ cooperative flows are not supported within z/OS environments.

z/OS WebSphere Support for Distributed Processing Servers

You can build CA Gen WebSphere MQ DPS applications for deployment as z/OS CICS or IMS transactions. The supporting Server Runtimes for each of these z/OS environments determine how incoming messages are obtained, based on how the transaction is started in its respective TP environment.

z/OS CICS Environments

CA Gen supports two methods for invoking a CICS DPS application through WebSphere MQ. They are:

- Using a trigger monitor to dispatch each server manager directly.
- Using a trigger monitor to invoke the Transaction Dispatcher for CICS (TDC).

Dispatching each server manager directly makes use of the trigger monitor for CICS that is provided by IBM. Each server manager has its own local triggered application queue defined to WebSphere MQ, and each of these application queues must be associated with a WebSphere MQ process definition object that describes the application that eventually processes the messages placed upon the queue. For information about this type of DPC-to-DPS flow, see *DPC-to-DPS Flow Using a z/OS CICS Trigger Monitor* in the appendix “Flow Diagrams.”

Additionally, each triggered application queue must define the trigger event rules that cause placement of a trigger message into an initiation queue. A trigger monitor services those trigger messages delivered to one or more initiation queues. When a trigger message arrives on an initiation queue, the trigger monitor retrieves the trigger message and issues a command to activate the defined application. The trigger message contains detailed information, specifically the application queue name passed as data to the invoked DPS application. The invoked application opens the specified application queue and retrieves the request message. For more information about triggering, see the appropriate IBM documentation.

z/OS CICS: The TDC

The Transaction Dispatcher for CICS (TDC) is a CICS application provided by CA Gen, and is the second method of invoking a CICS DPS application. The TDC also makes use of the trigger monitor from IBM, but the TDC is designated as the processing application associated with the local queue.

The TDC acts as a surrogate for inbound messages. When triggered, it reads the WebSphere MQ queue and spawns the appropriate CA Gen server manager, based on the content of the inbound message. The TDC passes the message to the server manager either in CICS temporary storage or as a record in a VSAM file. When the server manager begins processing, the runtime detects initiation through the TDC and reads the message without directly accessing the MQS. The server manager posts the reply message through the runtime. For an illustration, see *DPC-to-DPS Flow Using TDC* in the appendix “Flow Diagrams.”

The TDC allows multiple CICS CA Gen server managers to share the same WebSphere MQ application queue. Additionally, the TDC ensures that a single trigger event causes complete processing of the associated application queue, thus emptying the queue. If you only use a trigger monitor without the TDC (see DPC-to-DPS Flow Using a z/OS CICS Trigger Monitor) it is possible that the trigger rules will not always create a trigger event, which could leave request messages in the queue awaiting processing.

If the TDC detects that the WebSphere MQ queue depth rose above a set level, the TDC spawns additional copies of itself, processing the current WebSphere MQ queue in parallel. These copies are considered child processes of the TDC, inheriting their parameters from the original process. These child processes can not spawn additional instances of themselves. The TIRMQTDX user exit allows you to set the queue depth and number of allowed child processes.

Another option available with the TDC is transaction level security. Without the TDC, the security context of the target DPS application is derived from the user context associated with the trigger monitor. With the TDC, there are alternate implementations for obtaining the user ID, through the user exit TIRMQTDX. Among its tasks, TIRMQTDX controls which security context should be in effect when starting the target DPS application. The exit directs the TDC to start the DPS application in one of the following three ways:

- Use the user ID associated with the started TDC (this is the default behavior, and is the same as in a non-TDC implementation).
- Use the user ID context extracted from the inbound request message. The TDC obtains the inbound request message from the designated application queue prior to invoking the target DPS application. The exposure to the request message allows the TDC to extract the user ID from the message and use it when starting the DPS application.
- Use a user ID value that is provided through the TIRMQTDX user exit.

For more information about the TDC, see the appendix “Flow Diagrams.”

z/OS IMS Environment

CA Gen also supports WebSphere MQ DPS deployment on the z/OS IMS application environment. The IMS DPS WebSphere MQ environment is implemented similarly to the CICS WebSphere MQ runtime, where the DPS is dispatched using the CICS trigger monitor from IBM (the non-TDC implementation). For IMS, the trigger monitor is an IMS transaction from IBM called CSQQTRMN.

Chapter 3: Specifying Properties for WebSphere MQ Applications

This chapter describes how to detail a WebSphere MQ cooperative application using the Packaging tool.

This section contains the following topics:

[Packaging and Construction Attributes](#) (see page 23)

[Specify Properties Through the Toolset](#) (see page 24)

[Set Business System Defaults](#) (see page 25)

[Specify Properties Through the CSE Construction Client](#) (see page 29)

Packaging and Construction Attributes

The process of developing a WebSphere MQ cooperative application is similar to building other types of CA Gen cooperative applications. The CA Gen Design tool provides a set of integrated facilities that allow an application developer to specify different aspects of the CA Gen cooperative application. These tools include the following facilities:

- Setting Business System Defaults
- Dialog Design
- Dialog Flow Browser
- Window Navigation Diagram
- Event Browser
- Action Diagram
- Dialect Definition

In addition to the tools used to design an application, CA Gen provides a set of integrated tools to describe the construction characteristics of the various components that make up the modeled application. These integrated tools allow you to perform the following tasks:

- Set default properties for the application.
- Detail the packaging characteristics of individual application components.
- Generate the packaged application components.

Within a model, each defined component has a set of properties that can be added to the model definition of the application. For a cooperative application, there is a set of properties that apply to a window manager, and a set of properties that apply to a server manager. Each individual property typically has a default value, although the construction tools allow overriding of the various property value settings.

In addition, the target execution environment of the application component can be detailed and added to the model definition of the application. Environment parameters are used when generating a given application component, and each application component can have its own set of environment parameters, or it can make use of a set of default environment parameters defined for the entire application.

When an application component is generated, the values of the environment parameters determine which of the property values of the component are relevant. For example, WebSphere MQ properties are only referenced when generating a WebSphere MQ application component, and TCP/IP properties are only referenced when generating a TCP/IP application component.

The transport mechanism used by a DPC (window manager, or proxy) to flow to a DPS (server manager) is a characteristic of the execution environment of the server manager. The associated environment parameters of the server manager define its communication method and the TP Monitor environment into which it is deployed. That is, the specific transport mechanism generated for a window manager or proxy to use for a given DPC-to-DPS flow, is determined by the environment parameters associated with the server manager containing the target DPS.

Within construction, the component of an application can have its properties and environment parameters detailed using either the packaging or the generation tool.

Specify Properties Through the Toolset

Use the Cooperative Packaging tool to specify both the server manager environment properties, and WebSphere MQ -specific properties.

Open Cooperative Packaging Tool

The Cooperative Packaging tool displays all the defined server and window managers. With this tool, you have the option of setting parameters for the complete business system or for individual server managers. If you set parameters for the complete business system, you can still override these settings for individual servers

Follow these steps:

1. Select Construction, Packaging.

2. Double-click Packaging in the Diagrams panel of the Tree View.

Or

Select Diagram, Open, Cooperative Packaging.

The Cooperative Packaging tool displays all of the defined server and window managers.

Set Business System Defaults

You can set Websphere MQ as the default transport for all server managers in the Business System.

Follow these steps:

1. Highlight a Business System in the Cooperative Packaging tool.
2. Select or right-click Detail, Server Environment to open the Server Environment Parameters dialog box.
3. Select the target Operating System, DBMS, and Language.
4. Set the TP Monitor field. The valid TP Monitor entries based on the operating system you select are as follows:

Z/OS

CICS

IMS

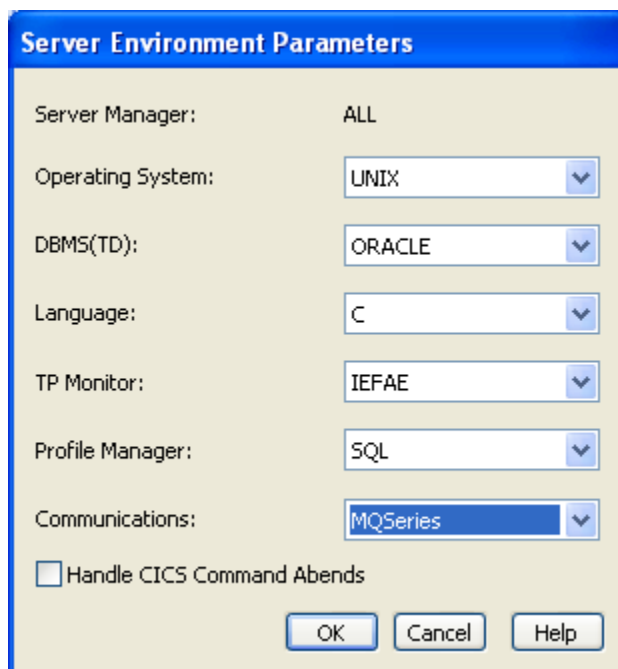
UNIX

IEFAE

Windows

IEFAE

5. In the Communications field, select MQSeries.



Note: The Server Environment Parameters dialog box shows only valid options. For example, if you choose Windows as your operating system, you do not have the option of choosing CICS as your TP Monitor.

Specify the Properties for Individual Server Managers

You can use the Server Environment Parameters dialog to specify the properties for individual server managers.

Follow these steps:

1. Highlight a specific server manager in the Cooperative Packaging tool.
2. Select or right-click Detail, Server Environment to open the Server Environment Parameters dialog box.

3. Select the target Operating System, DBMS, and Language.
4. Set the TP Monitor field. The valid TP Monitor entries based on the operating system you select are as follows:

Z/OS

CICS

IMS

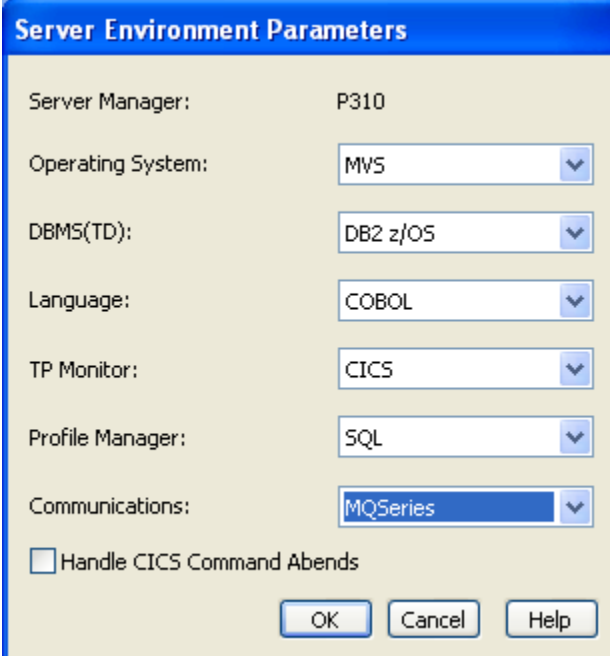
UNIX

IEFAE

Windows

IEFAE

5. In the Communications field, select MQSeries.



Server Environment Parameters

Server Manager:	P310
Operating System:	MVS
DBMS(TD):	DB2 z/OS
Language:	COBOL
TP Monitor:	CICS
Profile Manager:	SQL
Communications:	MQSeries

Handle CICS Command Abends

OK Cancel Help

Note: The Server Manager entry in the above screenshot shows the name of the specific server manager. Compare this with the name *ALL* appearing in the figure Server Environment Parameters Dialog Box for Business System. There, the *ALL* represents all server managers, and individual server managers do not have to be individually detailed.

Set MQSeries Properties

CA Gen makes MQSeries properties available from each server manager through the Server Properties dialog box. You can choose to specify properties, or keep the default values. You can specify the following properties:

Queue Manager Name

Specifies the name of the local queue manager. By default, the name is a null string, causing the client and server to connect to the default queue manager. On a given platform, more than one queue manager can be executing. The default queue manager is defined by WebSphere MQ.

Queue Name

Specifies the name of the local queue. The client uses this queue as its Put queue for requests. The WebSphere MQ server uses this queue as its Get queue to retrieve requests. The default value is the server load module name.

Reply Queue Name

Specifies the name of the reply-to queue. You can specify an MQ model queue or local queue name. By default, the reply-to queue name is the MQ model queue name SYSTEM.DEFAULT.MODEL.QUEUE. If a MQ model queue name is specified, a unique dynamic reply queue name is created for each DPC-DPS flow.

Client Type

Identifies the WebSphere MQ mechanism used for handling MQI requests (see Client Types).

- **MQI-Client**

Indicates that the DPC application will use the MQSeries client software for flows to the target DPS application.

- **MQS**

Indicates that the client will communicate with a local queue manager that is installed on the same machine as the DPC application.

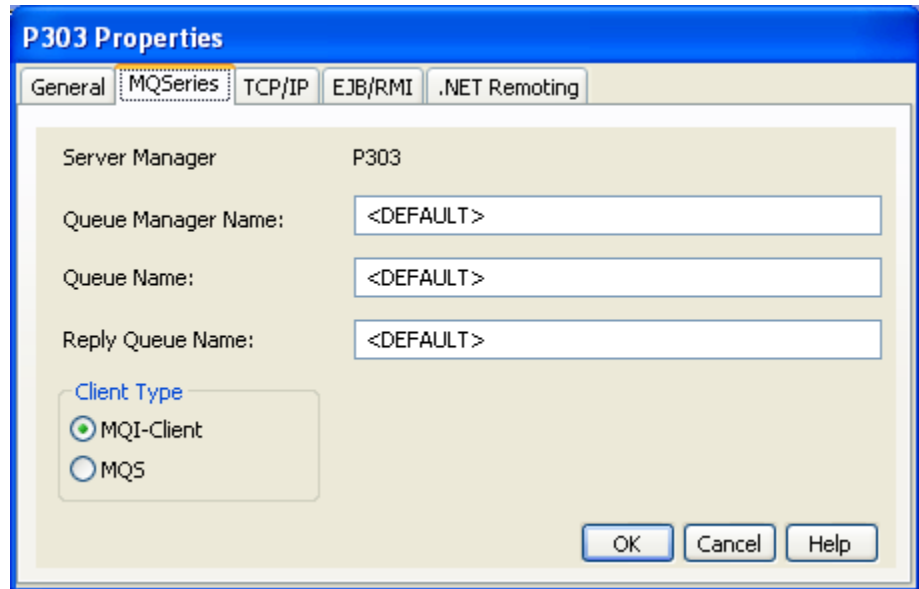
Set WebSphere MQ Properties

You can set the WebSphere MQ properties from the Server Properties dialog during packaging.

Follow these steps:

1. Highlight a specific server manager in the Cooperative Packaging tool.
2. Select or right-click Detail, Properties to open the Server Properties dialog box.

3. Click the MQSeries tab. The Server Properties dialog box displays the default MQSeries property settings for your application.



Note: The above steps show how you set WebSphere MQ properties through packaging. You can also perform the same steps with the generation tool.

Specify Properties Through the CSE Construction Client

You can set server environment properties and WebSphere MQ properties using the CSE Construction Client. However, the Construction Client has different approaches to setting parameters and properties in the model. Unlike the Toolset, the Construction Client does not let you set the communications parameter across an entire Business System; you can define the communications parameter only at the server manager level.

Through the Construction Client, you can set environment parameters through the Business System during packaging or during generation. This section shows you how to set server environment parameters and MQSeries properties through packaging.

Set Server Environment Parameters

From the Construction Client, you can set the WebSphere MQ Server Environment Parameters for an application during packaging.

To configure WebSphere MQ Parameters for an application

Follow these steps:

1. Start from the Construction Model Selection window.
2. Select Code, Package, Cooperative, to start the Cooperative Packaging dialog box.

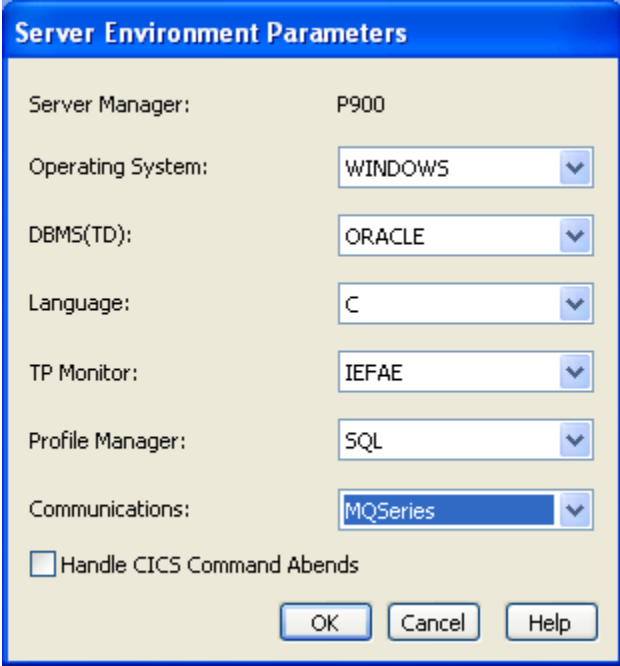
The Cooperative Package dialog box shows Business Systems. It allows you to set the TP Monitor for an entire Business System through Detail, Configuration. However, you only can set the communications parameter to MQSeries by continuing with the following steps.

3. Select a Business System in the Cooperative Packaging dialog box.
4. Choose View, Expand to open the List Load Modules dialog box.
5. Configure the WebSphere MQ server manager environment parameters for each of the server managers in the List Load Modules dialog box.

To configure the WebSphere MQ server manager environment parameters

Follow these steps:

1. Select a server manager.
2. Choose Detail, Configuration. The Server Manager Environment Parameters dialog box opens.



Server Environment Parameters

Server Manager: P900

Operating System: WINDOWS

DBMS(TD): ORACLE

Language: C

TP Monitor: IEFAE

Profile Manager: SQL

Communications: MQSeries

Handle CICS Command Abends

OK Cancel Help

3. Set the TP Monitor to one of the following parameters based on the Operating System you selected:

Z/OS

CICS

IMS

UNIX

IEFAE

Windows

IEFAE

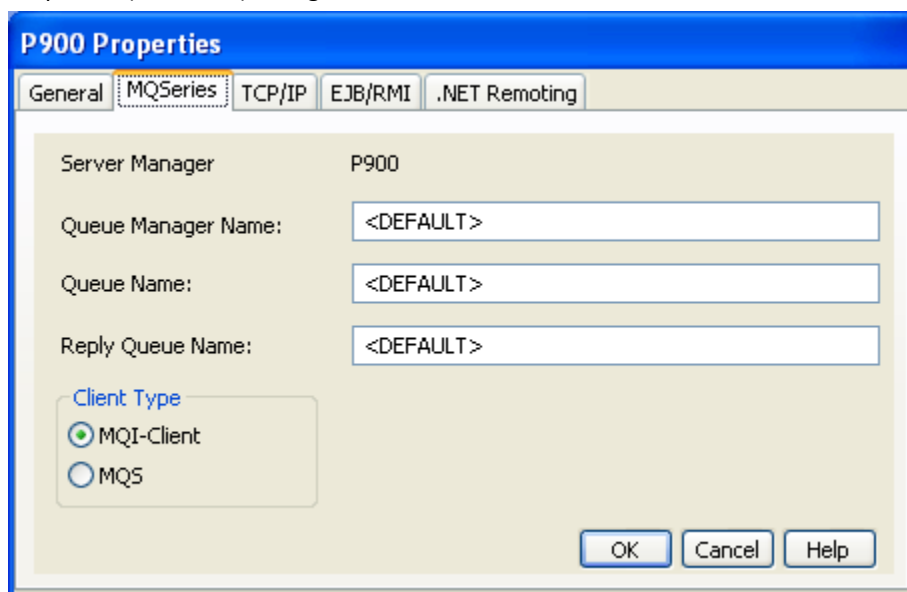
4. Select MQSeries in the Communications field
5. Click OK.

Set MQSeries Properties

From the Construction Client, you can set the MQSeries properties for an application during packaging.

Follow these steps:

1. Start from the List Load Modules dialog box.
2. Select a server manager.
3. Select Detail, Properties. Then click the MQSeries tab to open the Server Manager Properties (MQSeries) dialog box.



4. Provide the MQSeries parameters for your application. You can keep any value designated as <DEFAULT>.
5. Choose the client type for your application.

Note: For information about the default values, see [Set MQSeries Properties](#) (see page 28).

6. Click OK.

Chapter 4: Generating and Building Cooperative Code for WebSphere MQ

This section contains the following topics:

[Override Generation Defaults](#) (see page 33)

[Construct Cooperative Applications](#) (see page 36)

Override Generation Defaults

If you have a model that is already configured to use middleware other than WebSphere MQ, you can override the communications settings in the model, during generation, so that your application builds with WebSphere MQ. Overriding the environment settings for your window and server managers does not change their values in the model. Instead, it applies the change, for those items you have selected, for generation.

Note: You can override environment settings, but you cannot override the specific WebSphere MQ properties for a server manager, such as queue name or client type. The properties for a server manager must be detailed before generation if you do not intend to use default values.

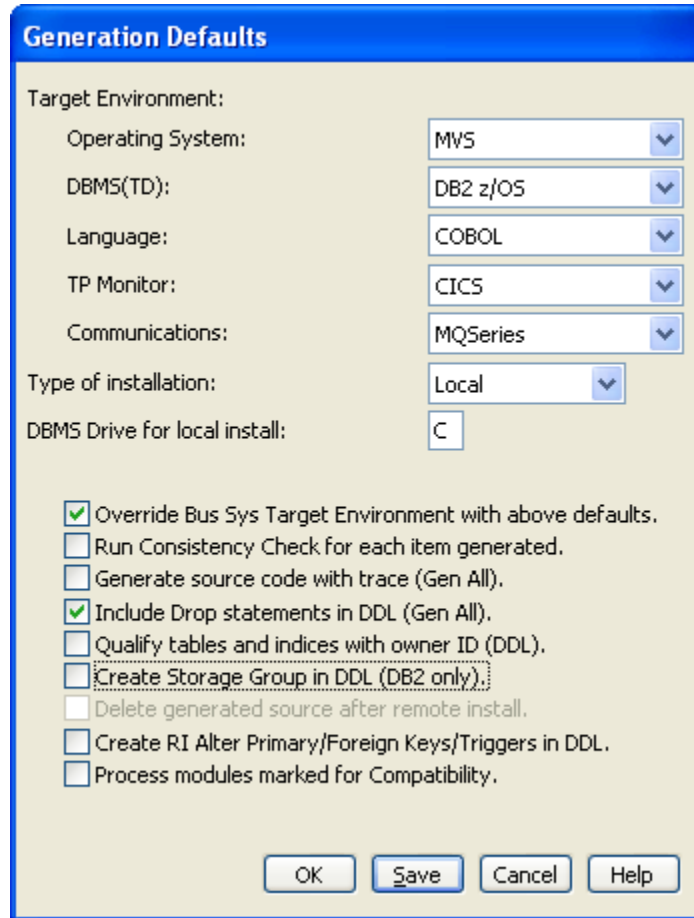
Overriding in the Toolset

You can override the settings for the communication environment parameters for a build in the Toolset.

Follow these steps:

1. Do one of the following:
 - Select Construction, Generation from the Model window of the toolset
 - Double-click Generation in the Diagrams page of the Tree View.

2. Select Options, Generation Defaults, which opens the Generation Defaults dialog
The Generation Defaults dialog appears.



3. Set the TP Monitor field. The valid TP Monitor entries based on the operating system you select are as follows:

Z/OS

CICS

IMS

UNIX

IEFAE

Windows

IEFAE

4. Select MQSeries in the Communications field
5. Select the Override Business System Target Environment check box.
6. Click OK.

CSE Construction Client

You can override the settings for the communication environment parameters for a build in the CSE Construction Client.

Follow these steps:

1. Select Code, Generate, Cooperative from the Construction Model Selection window to open the Cooperative Application Generation dialog.
2. Select Defaults to display the Generation Defaults dialog.

Note: You do not need to select a server manager from the Cooperative Application Generation dialog first. The Defaults selection applies to the complete business system.

3. Set the TP Monitor field. The valid TP Monitor entries based on the operating system you select are as follows:

Z/OS

CICS

IMS

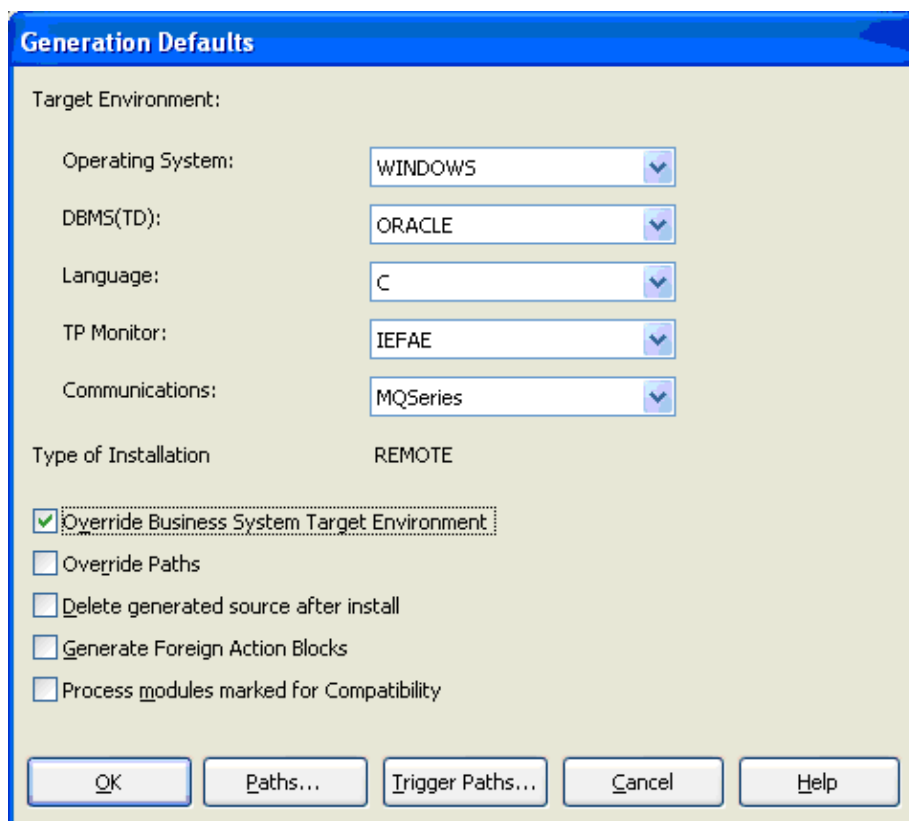
UNIX

IEFAE

Windows

IEFAE

4. Select the Override Business System Target Environment check box.
5. Select MQSeries in the Communications field.



6. Click OK.

Construct Cooperative Applications

After you have set all parameters and variables, you can generate your window and server managers.

Windows Build Tool Setup

When the Windows Build Tool builds server manager applications, it looks for MQS libraries in the following location:

```
C:\PROGRAM FILES\IBM\WEBSHERE MQ\TOOLS\LIB
```

If you have installed your MQS libraries anywhere else, you need to set the LOC.MQSLIB token to point at that new location before starting your server manager build.

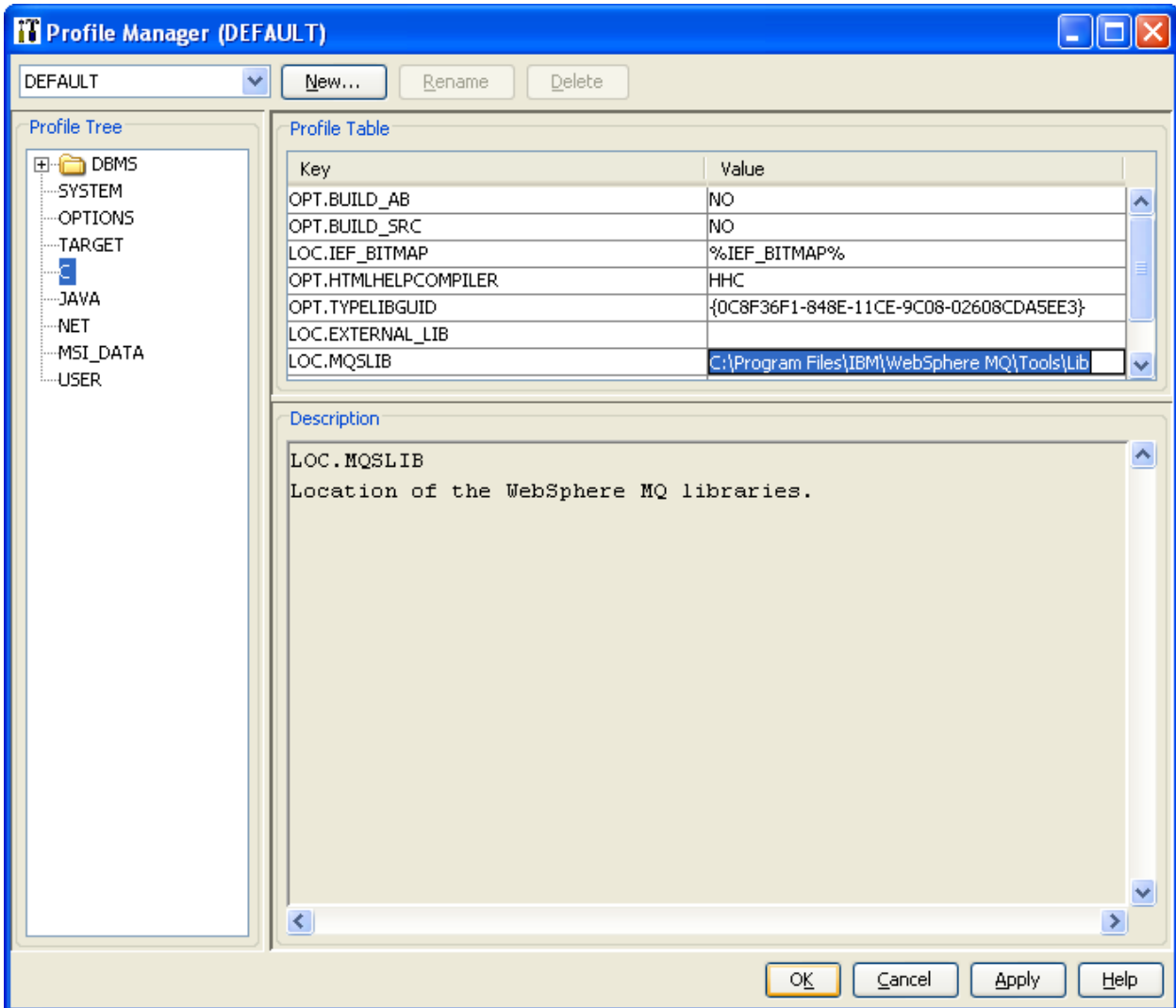
Follow these steps:

1. Click the Profile Manager toolbar icon after selecting the appropriate profile name, and clicking the OK button.

The Profile Manager dialog appears.

2. Select C subtree within the Profile Tree on the left-hand side.

The right-hand side of the panel will be populated with a set of tokens.



3. Find LOC.MQSLIB in the token list. Either select the token and then select the Edit... button, or double click on the token. A popup dialog will appear in order to edit this token.

4. Change the path to match your MQM library path.
5. Select the OK button to update the token and then the Save button to save the file.

You can now build your server manager. For more information about using the Build Tool, see the *Build Tool User Guide*.

UNIX Build Tool Setup

When the UNIX Build Tool builds server manager applications, it looks for MQS libraries using the library environment variables:

- \$SHLIB_PATH for HP systems
- \$LD_LIBRARY_PATH for Solaris systems
- \$LIBPATH for AIX systems

Ensure that the MQS libraries' location is set in the appropriate environment variable before starting the UNIX Build Tool.

After you have built your remote file and transferred it to your UNIX system, you can now build your server manager.

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

z/OS IT

After you have built your remote file and transferred it to your z/OS system, you need to run the IT under ISPF and build the module as you would any other CA Gen module.

Note: For more information about using the Implementation Toolset, see the *z/OS Implementation Toolset User Guide*.

Chapter 5: Testing WebSphere MQ Applications

Although an application is ready for execution after the remote files are processed, it is recommended that you test your application using the Diagram Trace Utility before introducing it into your production environment.

This section contains the following topics:

[Diagram Trace Utility](#) (see page 39)

[Regenerate for Changes](#) (see page 40)

[Application Testing](#) (see page 40)

[Regenerate Remote Files After Testing](#) (see page 42)

Diagram Trace Utility

The Diagram Trace Utility can be used to test a generated application before you move it to the production environment.

Diagram Trace provides a number of special capabilities that allow viewing of the CA Gen model elements such as Action Diagram statements, used to build the application when the program is being executed. To use the special capabilities of Diagram Trace, you need to make certain selections when remote files are generated. These selections cause the generation of the additional code required to implement the special capabilities available through Diagram Trace.

Before you test a complete application, you must build the the following application components using the Build tool:

- The application database
- RI trigger logic
- Any applicable operations libraries
- All load modules

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

Regenerate for Changes

If you need to make changes to the application, you must change the CA Gen model to generate the updated application, not the generated code. This applies whether the change is made to correct a problem or enhance the application. After you change the model, regenerate that portion of the application on the development platform.

If the load module definition has not been updated by the changes, then only the necessary load module components need to be selected for regeneration instead of the complete model. The regenerated components can then be moved to the target system, overlaying the old versions of the components. The load module is then rebuilt. This saves the time and resources required to split a remote file whose definition has not changed.

Note: Ensure that file names and suffixes to file names are carefully verified before overlaying files. If file names differ, change the name of the new file to match the old one so that the MAKE command procedure used during the rebuild will search for the correct file.

If a load module definition has been changed, or if most of its components have been changed, then an entirely new remote file can be generated. The resulting remote file can then be moved to the target system, and the application built again.

After an application is tested, it can be regenerated without the trace code (if required) and installed for production use.

Application Testing

The following sections describe the concept of WebSphere MQ application testing and the procedure required to access and use the Diagram Trace Utility.

Enable Diagram Trace Utility

Follow these steps:

1. Build the test application that has been generated with trace.
2. Invoke the Diagram Trace Utility on any accessible Windows system (this can be the same system):
 - Launch the Diagram Trace Utility by selecting Start, All Programs, CA, Gen <release>, Diagram Trace Utility.
 - The Diagram Trace Utility will *autostart* and listen on port 4567. You can change the default port from within the Diagram Trace Utility.

3. Set the trace environment variables in the application.ini file. For more information on the use of the application.ini file and the trace environment variables, see the appendix "[Using the Application.ini file](#) (see page 69)"
4. Set your AEPATH environment variable to include the location of the WebSphere MQ application that you want to test.

Access and Use Diagram Trace Utility

After you enable the Diagram Trace Utility, invoke your application as you would normally.

When an application is being tested, any of its procedure steps and/or action blocks that have been generated with trace will communicate with the Diagram Trace Utility. No additional information is displayed with the application screens; the application will wait until control is returned from the Diagram Trace Utility.

The Diagram Trace Utility lets you step through the execution of the application by viewing the sequence of action block calls and action diagram statements being executed as the generated program executes.

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

Multi-user Diagram Trace Utility Support

When a remote server application is started, a copy of the application.ini file for that application is available. Setting the trace environment variables within this application.ini file will allow only one Diagram Trace Utility to trace any of the servers that make up this server application.

Test environments involve multiple testers working with the same server application. In this scenario, you need to debug from multiple client workstations and therefore, use multiple Diagram Trace Utilities.

Note: For more information about using the *Application.ini* file, see the appendix, "[Using the Application.ini file](#) (see page 69)."

Override the default trace environment variables

A client transaction can transfer the host and port of the client that is executing the transaction. By providing this information, the server can establish communication with the Diagram Trace Utility running on that client workstation.

Follow these steps:

1. Build the server application that has been generated with trace (as before).
2. Leave the trace environment variables within the application.ini file that is located in the server application's model directory commented out.
3. Start your servers.
4. Invoke the Diagram Trace Utility on your client workstation.
5. Edit the application.ini file residing in your client application's executables directory.
6. Uncomment and set the trace environment variables. You can use 'localhost' for the TRACE_HOST variable.
7. Execute your client application.

The Diagram Trace Utility on the client workstation will be used to trace the server for each transaction initiated from that client.

Note: To provide the server with the client's host and port information, these values are added to the Common Format Buffer (CFB). A CFB is used to exchange view data between the client and server. If the addition of the host and port values exceeds the allowable size of the CFB, the host and port values are not added to the CFB and the statement "*Not enough space in CFB to pass Trace flags to server*" is logged to the client's trace file. The end result is that no override takes place, and the values of the host and the port will be extracted from the server's application.ini file.

Regenerate Remote Files After Testing

The code generated on the CA Gen workstation for testing with trace includes features, such as trace calls, that are only needed during testing. These features increase the size of the load module significantly and impact the application's performance. Even if no changes are required as a result of the tests performed, the load module must be regenerated without trace before it is implemented into production.

Chapter 6: Configuring and Deploying WebSphere MQ for CA Gen

This section contains the following topics:

[How to Configure MQSeries for CA Gen Clients](#) (see page 43)

[Configuring WebSphere MQ for CA Gen Servers](#) (see page 45)

[Configure z/OS Servers](#) (see page 45)

[Configure Windows and UNIX Servers](#) (see page 51)

How to Configure MQSeries for CA Gen Clients

Before you can run your distributed processing client application, you must prepare the WebSphere MQ environment by configuring the queue manager for the application.

To configure the queue manager for the client, you must perform the following tasks::

- Define how the client will connect to its queue manager.
- Create a queue definition for the local queue.
- Define any local reply-to queue, if dynamic reply-to queues are not used.
- Define any references of the model queue in the model, if dynamic reply-to queues are used.

Configure Client Connection

With respect to the use of WebSphere MQ in CA Gen, WebSphere MQ supports two connections types. The type of connection determines how the MQ client application will connect to a MQ queue manager.

- MQI Client connection
- MQI Server connection

The MQI client connection option makes use of an MQ client/server communications protocol to interact between the client application and the queue manager. This option is used when the client application does not execute on the same machine as that of the MQ queue manager.

The client application communicates to the MQ queue manager across an MQI Channel which must be defined at both the MQ client and server ends of the connection. If the MQI client connection option is utilized, a server connection channel must be added to the queue manager configuration. In addition, the queue manager must have an active listener program for each server connection channel. Defining the channel at the client end varies depending on the type of Gen client. For C language clients, MQ Environment variables are used to define the Channel definition. For Java, the MQ Environment settings are provided using the `commcfg.properties` file.

Note: For information about these `commcfg` files, see the *Distributed Processing - Overview Guide*.

The MQI Server connection is used when the MQ application executes on the same machine as the MQ queue manager. The client application connects to the MQ queue manager using a local queue manager handle.

Note: For more information about how client applications connect to WebSphere MQ queue managers, see *IBM WebSphere MQ Clients* book.

Configure the Local Queue

The queue manager for the client needs a queue definition to reach the target (server) application. The queue manager may or may not have a local connection to the target application. If the client shares its queue manager with the server, define the local queue to match the name of the queue associated with the server; this is the name that you specified in the model. If you used the default queue name, this is the same as the server manager load module name, but if you specified a different queue name in the model, this is the name that you must use.

If the server is on a remote queue manager, the queue manager of the client needs to know:

- The remote queue manager name.
- The remote queue name.

Note: For more information about configuring access to remote queue, see the appropriate IBM WebSphere MQ documentation.

Configure the Reply-To Queue

The server uses the reply-to queue to return information to the queue manager of the client application. You must configure the reply-to queue of the client depending on what was detailed in the model, where the reply-to queue is either:

Local reply-to queue

The local reply-to queue name must match the name supplied for the MQSeries properties when the application was detailed.

Note: For more information about MQSeries Properties, see [Setting MQSeries Properties](#) (see page 28).

Model queue

The queue manager must define a model queue from which the dynamic reply-to queue derives all of its attributes.

Configuring WebSphere MQ for CA Gen Servers

With a DPC, you need to prepare the WebSphere MQ environment for your server application. You only need to define the local queue for use by the server application to its queue manager.

CA Gen allows generation of WebSphere MQ server managers for Windows, UNIX, and z/OS (CICS and IMS) platforms. The following sections explain the WebSphere MQ configuration issues specific to each of these platforms.

Configure z/OS Servers

CA Gen supports applications running MQSeries with CICS or IMS serving as the TP monitor. By default, in both z/OS CICS and z/OS IMS, processing requests received by its queue manager triggers a CA Gen DPS, and it sends the processed result back to the DPC through the reply-to queue that is associated with the request. In this default configuration, every DPS is a process object to WebSphere MQ, and each DPS requires its own local queue.

CA Gen server managers deployed to either z/OS CICS or z/OS IMS rely on triggering. With IMS, you can use a trigger type of either FIRST or EVERY, but z/OS CICS users should configure their systems for trigger type EVERY.

With CA Gen, z/OS CICS users have the option of using the CA Gen WebSphere MQ Transaction Dispatcher for CICS. The Transaction Dispatcher for CICS (TDC) acts as a dispatcher between WebSphere MQ and one or more DPS applications. With the TDC, you can perform the following tasks:

- Share the same WebSphere MQ queue between multiple server managers.
- Tune the TDC operation through a user exit routine, the TIRMQTDX.
Note: For more information about the TIRMQTDX user exit, see the *User Exit Reference Guide*.
- Support transaction-level security (see Transactional Security with the TDC).
- Spawn multiple copies of the TDC to process a queue in parallel (see Triggering and the TDC).

The following sections further explain the TDC.

Install the TDC

The base components of the TDC are installed as part of the z/OS Implementation Toolset and Host Encyclopedia installation.

To make use of the base TDC components, you must define them to CICS. You also need to define a VSAM shared data table if you do not intend to use CICS temporary storage queues.

Basic CICS Install

The modules TIRMQTD and TIRMQTDX must be in the DFHRPL concatenation. If the CA Gen LOADLIB is not allocated to DFHRL, copy TIRMQTD from the CA Gen LOADLIB to the DFHRL library of your choice.

Run DFHCSDUP using the following deck, adjusting the language for TIRMQTDX as necessary:

```
DEFINE TRANSACTION(TITD)
DESCRIPTION(CA Gen Transaction Dispatcher)
PROGRAM(TIRMQTDC)
TASKDATALOC(ANY)
GROUP(TDCGROUP)
DEFINE PROGRAM(TIRMQTDC)
DESCRIPTION(CA Gen Transaction Dispatcher)
LANGUAGE(ASSEMBLER)
DATALOCATION(ANY)
GROUP(TDCGROUP)
DEFINE PROGRAM(TIRMQTDX)
DESCRIPTION(CA Gen Transaction Dispatcher control exit)
LANGUAGE(LE370)
DATALOCATION(ANY)
GROUP(TDCGROUP)
```

CICS VSAM Install

Run the following defines if you intend to use a VSAM SDT* instead of CICS temporary storage. Change the dataset name to the DSN you intend to use. You can change the DDNAME if you make the corresponding update in TIRMQTDX. You can change MAXNUMRECS as needed.

Note: Do not use a CICS-maintained table. Use only a USER maintained table.

Run DFHCSDUP using the following deck:

```
DEFINE FILE(TITDTEMP)
DESCRIPTION(CA Gen Transaction Dispatcher temporary SDT)
DSNAME(YOUR.TEMP.DATASET)
STRINGS(5)
ADD(YES) DELETE(YES) READ(YES) BROWSE(YES)
RECORDSIZE(32760) RECORDFORMAT(V) KEYLEN(8)
TABLE(USER) MAXNUMRECS(4096)
GROUP(TDCGROUP)
Run IDCAMS as follows:
//IDCAMS EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//DD1 DD *
00000000
/*
//SYSIN DD *
DEFINE CLUSTER(NAME(YOUR.TEMP.DATASET) -
VOLUME(VVVVV) -
CYLINDERS(5 5)
RECORDSIZE(865 32760) -
KEYS(8 0) -
)
REPRO INFILE(DD1) OUTDATASET(YOUR.TEMP.DATASET)
```

Note: You can also use a standard VSAM KSDS without using the data table feature. Adjust cluster space accordingly, and make certain that the cluster is deleted and redefined on each restart of CICS. Use SDT for better performance.

Triggering and the TDC

When a request initially invokes the TDC, the TDC disables triggering for the queue it is processing. The TDC re-enables triggering when the parent TDC completes processing. Since the TDC remains active (with the queue open) while messages arrive, triggering activity resembles that of using type FIRST.

Note: You must use trigger type EVERY. Whether you use trigger type FIRST or EVERY, triggers usually are generated only after the queue level falls to zero and returns to non-zero. Using trigger type EVERY assures that any new messages empty the queue, whereas trigger type FIRST risks delaying the processing of a request. Trigger type EVERY does not produce any more trigger messages than type FIRST, so there is no added risk of additional overhead.

Transactional Security with the TDC

The TDC makes transactional security available to z/OS CICS Distributed Processing Servers, allowing three possible modes:

Default

DPS transactions start with the user ID of the TDC, which it inherits from the trigger monitor transaction (CKTI) or the transaction definition.

Security Override

Specify a user ID to be used by all Distributed Processing Servers the TDC has started. The external security manager must also know this ID. If CICS uses surrogate checking, you must define the user ID associated with the transaction as surrogates for the override user IDs.

Message ID Security

The TDC extracts the user ID from the inbound message and uses it as the user ID for starting the DPS. Each individual user ID likely to initiate a DPS must be defined to the external security manager, while the user ID associated with the TDC must be authorized as surrogates for them all.

Note: The TDC uses the CICS system programmer interface (SPI) and must be authorized to use SPI commands.

Initiate Multiple TDCs

The TDC is able to monitor queue depth and create copies of itself to process the queue in parallel. The new, child TDC processes inherit their parameters from the parent TDC, but are not allowed to initiate additional child processes on their own. By default, the TDC initiates no more than five new processes. However, you can control this and other aspects of it through the TIRMQTDX user exit.

Note: For more information about the TIRMQTDX user exit, see the *User Exit Reference Guide*.

Define the TDC to MQSeries

To enable the TDC in WebSphere MQ, you must define one or more process definitions with a target application of TITD, application type CICS. The local queues can then point to this definition, which causes the CICS trigger monitor to start the TITD (a TDC application).

Queue

DEFINE NOREPLACE -
QLOCAL('YOUR.QUEUE.NAME') -
STGCLASS('DEFAULT') -
DESCR('CA GEN LOCAL QUEUE') -
PUT(ENABLED) -
DEFPRTY(0) -
DEFPSIST(NO) -
MAXDEPTH(999999999) -
PROCESS('YOUR.PROCESS.NAME') -
TRIGGER -
MAXMSGL(32767) -
BOTHRESH(0) -
BOQNAME(' ') -
INITQ('YOUR.INITQ') -
USAGE(NORMAL) -
SHARE -
DEFSOPT(SHARED) -
MSGDLVSQ(PRIORITY) -
RETINTVL(999999999) -
TRIGTYPE(FIRST) -
TRIGDPTH(1) -
TRIGMPRI(0) -
TRIGDATA(' ') -
NOHARDENBO -
GET(ENABLED) -
QDEPTHHI(80) -
QDEPTHLO(40) -
QDPMAXEV(ENABLED) -
QDPHIEV(DISABLED) -
QDPLOEV(DISABLED) -
QSVCIINT(999999999) -
QSVCIIEV(NONE)

Process

DEFINE NOREPLACE -
PROCESS('YOUR.PROCESS.NAME') -
DESCR('CA GEN TDC PROCESS') -
APPLTYPE('CICS') -
APPLICID('TITD') -
USERDATA(' ') -
ENVRDATA(' ') -

Configure Windows and UNIX Servers

In a Windows or UNIX environment, the server manager can be started in one of two ways:

- As a long-running daemon task
- Invoked by an WebSphere MQ trigger monitor

Start Server Manager as a Daemon Task

Daemon tasks take the following command line syntax to invoke the server manager executable:

```
xxxx[-t n ] [-m|M queue_manager_name ] [-q|Q queue_name ][-r|R number_of_requests ]
```

Where:

xxxx

Specifies the name of the server manager executable (case sensitive on UNIX).

-t n

Identifies the trace level that the executing server manager should use. This value ranges from 0 to 15 with 0 specifying no trace data and 15 specifying the greatest level of detail. The default trace level is 0.

The trace level data writes to a file named *lg-<procname>-<procid>.log*, where *<procname>* is the name of the program and *<procid>* is the process number of the program. For example, in the log file, *lg-p126-163574.log*, *p126* is the name of the program, and *163574* is the name of the process.

On Windows, this file is created in the directory, *%USERPROFILE%\AppData\Local\CA\Gen 8.5\logs\server*. On UNIX, the file is created in the directory where the program was started.

-m|M

Identifies the name of the local queue manager to which the server manager attempts a connection. The default queue manager name is an empty string (" ") meaning that the server manager attempts to connect to the default queue manager defined by WebSphere MQ.

-q|Q

Identifies the name of the local queue that the server manager opens. The default is the server manager load module name.

-r|R

Identifies the number of inbound message requests the server manager processes prior to terminating. The default is -1, indicating that the server manager remains active and servicing its inbound message queue.

Start a Server Manager through an WebSphere MQ Trigger Monitor

When you use a trigger monitor to invoke a server manager, the trigger monitor looks for the WebSphere MQ ENVRDATA attribute of the PROCESS object that is associated with the local queue (which is, in turn, associated with the target server manager). The ENVRDATA attribute contains the same data that you specify from the command line for the daemon process. See the appropriate MQSeries documentation from IBM for information about trigger monitor processing.

It is expected that, at minimum, the `-r|R` command line argument be modified to `-r 1`. The expectation is that when a message arrives in the initiation queue, the trigger message corresponds to one message arriving in the local queue that is serviced by a given DPS server manager.

With trigger monitoring, you should change the `-r|R` command to `1` so that once a message arrives on a queue, the trigger message corresponds to one message arriving in the local queue that a given DPS server manager is servicing.

Appendix A: User Exit Functions

This section contains the following topics:

[Non-z/OS WebSphere MQ Client Exits](#) (see page 53)

[Non-z/OS WebSphere MQ Server Exits](#) (see page 55)

[z/OS WebSphere MQ Server Exit TIRMQTDX](#) (see page 56)

Non-z/OS WebSphere MQ Client Exits

This section explains the Non-z/OS WebSphere MQ Client exits.

Note: For more information about each exit, see the User Exit Reference Guide.

C Language Exits

The following client exit entry points are located in the file named CIMQCLEX.C. This file is installed into the CA Gen installation directory on Windows systems, and for UNIX systems, this source file is installed in the /src subdirectory of the CA Gen installation directory.

CI_MQS_DPC_Exit

This client exit routine allows the user to modify various processing attributes used by WebSphere MQ when processing a cooperative flow request. These attributes include such items as the name of the queue manager, queue name to which the request will be sent, the reply-to-queue name, and the get request timeout values.

Note: For more information about the CI_MQS_DPC_Exit - MQSeries DPC Directory Services Exit, see the *User Exit Reference Guide*.

CI_MQS_DynamicQName_Exit

This client exit routine exposes the name of the queue that the queue manager uses when it opens a dynamic queue. The dynamic queue created by the queue manager obtains the dynamic queues attributes from those attributes associated with the specified WebSphere MQ model queue.

Note: For more information about the CI_MQS_DynamicQName_Exit - Dynamic Queue Name Override Exit, see the *User Exit Reference Guide*.

CI_MQS_DPC_setupComm_Complete

The processing of a given cooperative flow is broken up into two large grain activities. The first is setupComm, which is invoked to insure a connection to the target server is available. This exit is invoked at the completion of the setupComm processing to expose the processing results.

Note: For more information about the CI_MQS_DPC_setupComm_Complete exit, see the *User Exit Reference Guide*.

CI_MQS_DPC_handleComm_Complete

The processing of a given cooperative flow is broken up into two large grain activities. The second is handleComm, which is invoked to send and receive data over an already active connection. This exit is invoked at the completion of the handleComm processing to expose the processing results.

Note: For more information about the CI_MQS_DPC_handleComm_Complete - Handle Comm Retry Count Exit, see the *User Exit Reference Guide*.

CI_MQS_MQShutdownTest

This exit can be used to modify the behavior of the normal put/get queue disposition after successful completion of a cooperative flow. Normal disposition will leave the connection valid with the put and get queues open, ready to handle subsequent flows. This exit can override that behavior and cause the queues and connection to be closed after each flow has completed.

Note: For more information about the CI_MQS_MQShutdownTest - MQSeries Queue Disconnect Exit, see the *User Exit Reference Guide*.

CI_MQS_DPC_setReportOptions

This exit can be used to override the set of report options defined for the MQSeries Put Message Descriptor prior to the issuance of an MQPUT() operation.

Note: For more information about the CI_MQS_DPC_setReportOptions - Override Put Queue Report Options Exit Description, see the *User Exit Reference Guide*.

Java Language Exit

The following client exit class is located in the file named MQSDynamicCoopFlowExit.java. This file is installed the CA Gen installation area, within the CLASSES\COM\CA\GENnn\EXITS\COOPFLOW\MQS subdirectory, where *nn* is the current CA Gen release number.

MQSDynamicCoopFlowExit Class

This client user exit allows the modification of various processing attributes used by WebSphere MQ when processing a Java cooperative flow request. These attributes include such items as the name of the queue manager, queue name to which the request will be sent, the reply-to-queue name, the get timeout value, put/get queue disposition option, MQEnvironment variables and Message Descriptor report options.

Note: For more information about the MQSDynamicCoopFlowExit - Java MQSeries Communications Exit, see the *User Exit Reference Guide*.

.NET Language Exit

The following client exit class is located in the file named MQSDynamicCoopFlowExit.cs. This file is installed the CA Gen installation area, within the .NET\exits\src\coopflow directory.

MQSDynamicCoopFlowExit Class

This client user exit allows the modification of various processing attributes used by WebSphere MQ when processing a .NET cooperative flow request. These attributes include such items as the name of the queue manager, queue name to which the request will be sent, the reply-to-queue name, the get timeout value, put/get queue disposition option, MQEnvironment variables and Message Descriptor report options.

Note: For more information on the MQSDynamicCoopFlowExit - .NET MQSeries Communications Exit, see the *User Exit Reference Guide*.

Non-z/OS WebSphere MQ Server Exits

The following server user exit entry point is located in the file named CIMQSVEX.C. On Windows systems, this source file is installed into the CA Gen installation directory. On UNIX systems, this file is installed in the /src subdirectory of the CA Gen installation directory.

CI_MQS_DPS_Exit

This server user exit routine allows the user to modify various processing attributes used by WebSphere MQ when processing a cooperative flow request. These attributes include such items as the name of the queue manager, queue name from which requests will be processed, the number of requests used to set triggering behavior, and the get request timeout value.

Note: For more information about the CI_MQS_DPS_Exit-MQSeries DPS Directory Services, see the *User Exit Reference Guide*.

CI_MQS_DynamicQName_Exit

The Dynamic Queue Name exit allows override of the queue name that will be used when opening a dynamic queue. The resulting Dynamic Queue will obtain its attributes from the specified MQSeries Model Queue name.

Note: For more information about the CI_MQS_DynamicQName_Exit-Dynamic Queue Name Override Exit, see the *User Exit Reference Guide*.

CI_MQS_DPC_setReportOptions

This exit can be used to override the set of report options defined for the MQSeries Put Message Descriptor prior to the issuance of an MQPUT() operation.

Note: For more information about the CI_MQS_DPC_setReportOptions - Override Put Queue Report Options Exit, see the *User Exit Reference Guide*.

z/OS WebSphere MQ Server Exit TIRMQTDX

This section explains the z/OS WebSphere MQ Server Exit TIRMQTDX.

TIRMQTDX

TIRMQTDX (also known as the TDX) is a standalone CICS-based user exit that controls the operating behavior of the CICS Transaction Dispatcher through a parameter list. For each trigger event that initiates the TDC, the TDC sends a copy of the parameter list to the TDX in the DFHCOMMAREA. The TDX sends the list back to the TDC including any parameter modifications programmed into the TDX. The TDC accepts the parameter list without checking for invalid entries.

Note: For more information about TIRMQTDX-WebSphere MQ Transaction Dispatcher for CICS Parameter Exit, see the *User Exit Reference Guide*.

Appendix B: Flow Diagrams

This information is confidential. Log in to the CA Support site to access this information.

Follow these steps:

1. Go to <https://support.ca.com/irj/portal/DocumentationResults> and login.
2. Scroll down to Find Other Documentation area.
3. Type CA Gen in the product drop-down list.
4. Select Release 8.5 under Release and click go.
5. Click the PDF file for Distributed Processing WebSphere MQ Guide – Flow Diagrams

The PDF file opens on your computer.

Note: Alternatively, you can search for the PDF using the Q002911E document number.

Appendix C: z/OS CICS TDC Messages and Codes

This appendix contains messages and code descriptions for z/OS CICS TDC.

Message Descriptions

TITD001E Invalid MQSeries Trigger Record - Execution Terminated

Reason:

The trigger record retrieved contains invalid data. The task abends with code TDR.

Action:

Verify that the TDC (TITD transaction) was started by the MQS trigger monitor transaction, or another TDC. If it was, save the transaction dump, and contact Technical Support.

TITD201E Unable to Obtain Trigger Record - Execution Terminated

Reason:

The retrieve of the trigger record failed. Message TITD299I follows this message showing the failing command and return codes, and the task abends with code TDTR.

Action:

Verify that the TCD (TITD transaction) was started by the MQS trigger monitor transaction, or another TDC. If it was, note the failing command and return codes, save the transaction dump, and contact Technical Support.

TITD202W User Exit TIRMQTDX Not Found - Continuing With Default Parameters

Reason:

TIRMQTDX was unable to load the user exit TIRMQTDX. Message TITD299I follows this message with the failing command and return codes. Processing continues using the default values.

Action:

Generally, none. If you coded TIRMQTDX, verify that the program is defined to CICS and available in DFHRPL. Use the information in the accompanying message TITD299I to determine the cause of the failure.

TITD203E Unable to Obtain Buffer Storage - Execution Terminated

Reason:

TIRMQTDX was unable to obtain dynamic CICS storage for processing. Message TITD299I follows this message with the failing command and return codes. The task abends with code TDST.

Action:

Verify that sufficient DSA storage is available to the task. If so, note the failing command and return codes, save the transaction dump, and contact Technical Support.

TITD204E Slowdown Delay Value Invalid - Continuing to Process

Reason:

The task has entered slowdown mode, but CICS has rejected the delay value. Processing continues with slowdown mode effectively nullified. Message TITD299I follows this message with the failing command and return codes.

Action:

If you are overriding the slowdown delay with TIRMQTDX, verify that the value supplied is valid. Otherwise, note the failing return codes, and contact Technical Support.

TITD205E Unable to Write Temporary CFB - Server Manager Not Started

Reason:

A failure occurred while attempting to write the message to the selected temporary storage medium. A 619 "Unknown server error" is returned to the requester and processing continues. Message TITD299I follows this message with the failing command and return codes.

Action:

Verify that the selected temporary storage medium is available and has sufficient resources available. Verify that the messages being sent by the requester do not exceed 32KB in length. Otherwise, note the failing command and return codes, and contact Technical Support.

TITD206E Server Manager Start Failed - Unable to Process Message**Reason:**

An attempt to start the server manager indicated in the message failed. A 619 “Unknown server error” is returned to the requester, and processing continues. Message TITD299I follows this message showing the failing command and return codes.

Action:

Use the accompanying TITD299I message to determine the cause of the failure. Ensure that proper authorization was granted to the TDC. Otherwise, note the failing command and return codes, and contact Technical Support.

TITD207W Unable to Issue Inquire - Bypassing System Status Processing**Reason:**

An attempt to obtain CICS system statistics has failed. Processing continues with slowdown mode nullified. Message TITD299I follows this message showing the failing command and return codes.

Action:

Use the accompanying TITD299I message to determine the cause of failure. Ensure that proper authorization was granted to the TDC. Otherwise, note the failing command and return codes, and contact Technical Support.

TITD299I Exec CICS xxxxxxxx Failed EIBRESP = nnnn EIBRESP2= nnnn**Reason:**

This message is produced along with TITD2nnx messages. It indicates the failing CICS command (xxx ...) along with the EIBRESP and EIBRESP2 fields (nnnn) in decimal format.

Action:

See the associated TITD2nnx message.

TITD301W MQSeries Queue Not Shareable - Multi-Processing Disabled**Reason:**

The MQSeries queue being processed is marked non-shareable. Processing continues, but no additional TDC processes will be spawned, regardless of their queue depth.

Action:

If multi-processing is desired for this queue, redefine it with the shareable attribute.

TITD302W MQSeries Trigger Type Not EQ FIRST or EVERY - Multiprocessing Disabled

Reason:

The MQSeries queue being processed is using a trigger type other than FIRST or EVERY. Processing continues, but no additional TDC processes will be spawned, regardless of their queue depth.

Action:

If multi-processing is desired for this queue, redefine it with triggering set to FIRST or EVERY.

TITD303I MQSeries Connect to Queue Manager Failed - Processing Continues

Reason:

An attempt to connect to the MQSeries queue manager has failed. Since CICS is connected to a queue manager by default, processing continues and an open for the queue is attempted. Message TITD399I follows this message showing the failing command and return codes.

TITD304E MQSeries Open Failed for Input Queue - Processing Terminated

Reason:

An attempt to open the MQSeries queue has failed. The task abends with code TDMQ. Message TITD399I follows this message showing the failing command and return codes.

Action:

Verify that MQSeries was active and available at the time of the failure. If so, note the failing command and return codes, save the transaction dump, and contact Technical Support.

TITD305W MQSeries Inquire Failed for Input Queue - Multi-Processing Disabled

Reason:

The TDC was unable to obtain MQSeries status for the queue. Processing continues, but no additional TDC processes will be spawned regardless of the queue depth. Message TITD399I follows this message showing the failing command and return codes.

Action:

Verify that the TDC is properly authorized to perform this function. Otherwise, note the failing command and return codes, save the transaction dump, and contact Technical Support.

TITD306E MQSeries Get Failed for Input Queue - Processing Terminated

Reason:

An attempt to read the MQSeries queue has failed. The task abends with code TDMQ. Message TITD399I follows this message showing the failing command and return codes.

Action:

Verify that MQSeries was active and available at the time of the failure. If so, note the failing command and return codes, save the transaction dump, and contact Technical Support.

TITD307W MQSeries Put Failed for Reply To Queue - Unable to Notify Client of Server Failure

Reason:

The TDC attempted to reply to the requester directly and that reply has failed. Message TITD299I follows this message showing the failing command and return codes. Processing continues, but the requester is notified of the failure that precipitated this action.

Action:

Verify that MQSeries was active and available at the time of the failure. If so, note the failing command and return codes, save the transaction dump, and contact Technical Support.

TITD308W MQSeries Get Received Invalid Message Buffer - Message Bypassed

Reason:

MQSeries reported an invalid message to the TDC. The message is bypassed and processing continues. Message TITD299I follows this message showing the failing command and return codes.

Action:

Ensure that no non-CA Gen messages are posted to this queue, and that no messages exceed a size of 32KB. Otherwise, note the failing command and return codes, and contact Technical Support.

TITD309E MQSeries Get Warning Threshold Exceeded - Processing Terminated

Reason:

MQSeries reported five consecutive invalid messages to the TDC. The task abends with code TDMQ. Message TITD299I follows this message showing the failing command and return codes.

Action:

Ensure that no non-CA Gen messages are posted to this queue, and that no messages exceed a size of 32KB. Otherwise, note the failing command and return codes, save the transaction dump, and contact Technical Support.

TITD301E MQSeries Close Failed For Input Queue - Processing Terminated

Reason:

An attempt to close the queue has failed. The task abends with code TDMQ. Message TITD299I follows this message showing the failing command and return codes.

Action:

Verify that MQSeries was active and available at the time of the failure. If so, note the failing command and return codes, save the transaction dump, and contact Technical Support.

TITD311E Unable to Set Triggering - Verify Security for TITD

Reason:

An attempt to set triggering for the queue has failed. Processing continues. Message TITD399I follows this message showing the failing command and return codes.

Action:

Verify that proper authorization was granted to the TITD transaction. Otherwise contact Technical Support.

TITD399I MQSeries Call xxxx Failed CC= nnnn RC= nnnn

Reason:

This message is produced along with TITD3nnx messages. It indicates the failing MQSeries call (xxx ...), along with the condition and return code fields (nnnn) in decimal format.

Action:

See the associated TITD3nnx message.

TITD998I Transaction Dispatcher Completed Normally - Statistics Follow

Reason:

An instance of the TDC completed normally. A statistical summary is printed for the task under message TITD999I.

TITD999I

This ID represents one of the following messages:

Task: nnnn MQS Queue Name: xxxx

Reason:

The TDC task number nnnn completed using MQSeries queue xxxx .

Task: nnnn Messages Processed: mmmm Server Managers Started: ssss

Reason:

The TDC task number nnnn completed, processing mmmm messages and starting ssss server managers.

Task: nnnn Server Managers Errors: eeee Times Slowdown Entered: dddd

Reason:

The TDC task number nnnn completed, observing eeee server manager errors and entering slowdown mode dddd times.

Task: nnnn TDCS Spawned By This Task: tttt Max Observed Queue Depth: qqqq

Reason:

The TDC task number nnnn completed, and tttt additional TDCs were spawned by this task. The maximum observed queue depth was qqqq. This message appears only for the parent task.

Appendix D: Using the Application.ini File

Starting with AllFusion Gen 7.6, the runtime utilizes a file named application.ini. This file is delivered in the %IEFH% (for Windows) or the \$IEFH (for UNIX) directory. It contains a set of environment variables that are read by the runtime during application execution. The use of this file is to reduce the number of environment variables that must be set outside of the runtime for generated applications.

For providing environment variable uniqueness to each generated application, the application.ini file is copied from either %IEFH% or \$IEFH into the application's build directory. Before the application is executed, this file can be modified by the user to set these environment variables exclusively for the associated application. Once the file is copied into the application's build directory, it does not get overridden.

Application.ini Contents

The application.ini file contains a set of commented environment variables. Before using any of these environment variables, you must remove the comment from the variable (";") and modify the value as necessary.

The application.ini contains the following environment variables:

TRACE_ENABLE

Used to enable communications with the Diagram Trace Utility.

Default: 1

TRACE_HOST

Used to define the Windows machine that will run the Diagram Trace Utility.

Default: localhost for Windows; <none> for UNIX

TRACE_PORT

Used to define the port that the Diagram Trace Utility will listen to.

Default: 4567

Index

A

- accessing and using diagram trace utility • 41
- application • 13
 - behavior, supported • 13
 - constructing cooperative • 36
 - testing • 40

B

- build tool setup, Windows • 36, 38
- business system defaults setting • 25

C

- CA Gen
 - clients, configuring WebSphere MQ for • 43
 - distributed processing application • 9
 - servers, configuring WebSphere MQ for • 45
- CICS
 - basic install • 46
 - VSAM install • 47
- client exits, non-z/OS • 53, 54, 55
- client implementation, distributed processing • 16
- client types • 11
- concepts and terminology, WebSphere MQ • 9
- constructing cooperative applications • 36
- construction attributes, packaging and • 23
- cooperative code for WebSphere MQ • 33
 - configuring the LOC.MQSLIB path • 36
 - constructing cooperative applications • 36
 - CSE construction client • 35
 - override generation defaults • 33
 - overriding environment parameters in toolset • 33
 - Windows build tool setup • 36, 38
- CSE construction client • 35
 - detailing through • 29

D

- daemon task, starting server manager as • 51
- detailing WebSphere MQ applications
 - individual server managers • 26
 - packaging and construction attributes • 23
 - setting server environment parameters • 30
 - setting WebSphere MQ properties • 28
 - through CSE construction client • 29

- through toolset • 24
- developing CA Gen WebSphere MQ applications
 - distributed processing client implementation • 16
 - distributed processing server implementation • 17
 - non-z/OS environments • 18
 - server-to-server processing • 19
 - z/OS WebSphere MQ support for distributed processing servers • 19
 - z/OS/CICS
 - the TDC • 20
 - z/OS/CICS environments • 20
 - z/OS/IMS environments • 21
- diagram trace utility • 39, 41
- distributed processing
 - client implementation • 16
 - server implementation • 17
 - servers, z/OS WebSphere MQ support for • 19

E

- environment
 - non-z/OS • 18
 - z/OS/CICS • 20
 - z/OS/IMS • 21
- environment parameters
 - overriding in toolset • 33
 - setting server • 30

G

- generation defaults, overriding • 33

L

- LOC.MQSLIB path, configuring • 36
- local queue, configuring • 44

M

- message and queue types • 12
- messages and codes, z/OS CICS TDC • 61
- MQSeries for CA Gen
 - basic CICS install • 46
 - configuring z/OS servers • 45
 - properties, setting • 28
 - transactional security with TDC • 49

N

non-z/OS

- client exits • 53, 54, 55
- environments • 18

O

overview, WebSphere MQ

- CA Gen distributed processing application • 9
- client types • 11
- message and queue types • 12
- supported application behavior • 13
- WebSphere MQ concepts and terminology • 9

P

packaging and construction attributes • 23

parameters

- setting server environment • 30

processing, server-to-server • 19

properties, setting WebSphere MQ • 28

Q

queue

- configuring reply-to • 45
- message and types • 12

R

Regenerating remote files after testing • 42

Regeneration after testing • 40

S

server environment parameters, setting • 30

server implementation, distributed processing • 17

server manager

- detailing individual • 26
- starting as daemon task • 51
- starting through WebSphere MQ trigger monitor • 52

server-to-server processing • 19

T

TDC

- defining to WebSphere MQ • 49
- initiating multiple • 49
- installing • 46
- messages and codes, z/OS CICS • 61
- transactional security • 49

triggering • 48

z/OS/CICS • 20

Testing

application • 40

Diagram trace • 39

Regenerating remote files after testing • 42

Regeneration after testing • 40

Testing and running application • 39

toolset

detailing through • 24

overriding environment parameters in • 33

transactional security

with TDC • 49

triggering and TDC • 48

U

user exit functions

- non-z/OS client exits • 53, 54, 55

W

WebSphere MQ for CA Gen • 43

CICS VSAM install • 47

concepts and terminology • 9

configuring for clients • 43

configuring for servers • 45

configuring local queue • 44

configuring reply-to queue • 45

configuring Windows and UNIX servers • 51

defining TDC to WebSphere MQ • 49

initiating multiple TDCs • 49

installing the TDC • 46

starting server manager as daemon task • 51

starting server manager through WebSphere MQ

trigger monitor • 52

trigger monitor, starting server manager via • 52

triggering and TDC • 48

Windows and UNIX servers, configuring • 51

Windows build tool setup • 36, 38

Z

z/OS CICS TDC messages and codes • 61

z/OS servers, configuring • 45

z/OS WebSphere MQ support for distributed processing servers • 19

z/OS/CICS

environments • 20

the TDC • 20

z/OS/IMS environments • 21

