

# iWay

iWay Adapter for CICS User's Guide  
User's Guide  
Version 5 Release 5



**8.1 VALIDATED**

BEA WEBLOGIC PLATFORM

DN3501321.0104

EDA, EDA/SQL, FIDEL, FOCCALC, FOCUS, FOCUS Fusion, FOCUS Vision, Hospital-Trac, Information Builders, the Information Builders logo, Parlay, PC/FOCUS, SmartMart, SmartMode, SNAPpack, TableTalk, WALDO, Web390, WebFOCUS and WorldMART are registered trademarks, and iWay and iWay Software are trademarks of Information Builders, Inc.

Due to the nature of this material, this document refers to numerous hardware and software products by their trademarks. In most, if not all cases, these designations are claimed as trademarks or registered trademarks by their respective companies. It is not this publisher's intent to use any of these names generically. The reader is therefore cautioned to investigate all claimed trademark rights before using any of these names other than to refer to the product described.

Copyright © 2004, by Information Builders, Inc and iWay Software. All rights reserved. Patent Pending. This manual, or parts thereof, may not be reproduced in any form without the written permission of Information Builders, Inc.

---

---

## Preface

This documentation describes how to configure and use the iWay Adapter for CICS.

## How This Manual Is Organized

---

This manual includes the following chapters:

Chapter/Appendix		Contents
<b>1</b>	Introducing the iWay Adapter for CICS	Introduces the adapter environment.
<b>2</b>	Configuring the iWay Adapter for CICS	Describes how to configure a connection to the adapter.
<b>3</b>	Designing the iWay Adapter for CICS	Describes how to create transactions and events for the adapter. It also provides information on how to use the generated schemas to listen for events in CICS and create iWay Business Services, which expose functionality as Web services.
<b>4</b>	Using WebLogic Workshop	Describes how to produce and access a Web service generated from a CICS transaction.
<b>A</b>	Configuring VTAM and CICS	Provides examples of the major VTAM nodes for AnyNet operation and connection and session definitions required for the adapter to connect to CICS.
<b>B</b>	Installing the Sample IWAYIVP and IWAYSAMP Programs in CICS	Describes how to verify correct installation of the adapter.
<b>C</b>	Sample Requests, Schemas, and Cobol File Descriptions	Provides documents and schemas for the sample programs and the Cobol descriptions used as input for the sample CICS transactions.
<b>D</b>	Sample CICS Programs	Includes the source code for the sample CICS programs.
<b>E</b>	Debugging and Troubleshooting	Includes tips and techniques for debugging the iWay Adapter for CICS.

## Documentation Conventions

---

The following conventions apply throughout this manual:

Convention	Description
<code>THIS TYPEFACE</code> or <code>this typeface</code>	Denotes syntax that you must enter exactly as shown.
<i>this typeface</i>	Represents a placeholder (or variable) in syntax for a value that you or the system must supply.
<u>underscore</u>	Indicates a default setting.
<i>this typeface</i>	Represents a placeholder (or variable), a cross-reference, or an important term. It may also indicate a button, menu item, or dialog box option you can click or select.
<b>this typeface</b>	Highlights a file name or command.
Key + Key	Indicates keys that you must press simultaneously.
{   }	Indicates two or three choices; type one of them, not the braces.
[   ]	Indicates a group of optional parameters. None are required, but you may select one of them. Type only the parameter in the brackets, not the brackets.
	Separates mutually exclusive choices in syntax. Type one of them, not the symbol.
...	Indicates that you can enter a parameter multiple times. Type only the parameter, not the ellipsis points (...).
. . . . . .	Indicates that there are (or could be) intervening or additional commands.

## Related Publications

---

To view a current listing of our publications and to place an order, visit our World Wide Web site, <http://www.informationbuilders.com>. You can also contact the Publications Order Department at (800) 969-4636.

## Customer Support

---

Do you have questions about the iWay Adapter for CICS?

Call Information Builders Customer Support Service (CSS) at (800) 736-6130 or (212) 736-6130. Customer Support Consultants are available Monday through Friday between 8:00 A.M. and 8:00 P.M. EST to address all your questions. Information Builders consultants can also give you general guidance regarding product capabilities and documentation. Please be ready to provide your six-digit site code (xxxx.xx) when you call.

You can also access support services electronically, 24 hours a day, with InfoResponse Online. InfoResponse Online is accessible through our World Wide Web site, <http://www.informationbuilders.com>. It connects you to the tracking system and known-problem database at the Information Builders support center. Registered users can open, update, and view the status of cases in the tracking system and read descriptions of reported software issues. New users can register immediately for this service. The technical support section of [www.informationbuilders.com](http://www.informationbuilders.com) also provides usage techniques, diagnostic tips, and answers to frequently asked questions.

To learn about the full range of available support services, ask your Information Builders representative about InfoResponse Online, or call (800) 969-INFO.

## Information You Should Have

---

To help our consultants answer your questions most effectively, be ready to provide the following information when you call:

- Your six-digit site code (xxxx.xx).
- Your iWay Software configuration:
  - The iWay Software version and release. You can find your server version and release using the *Version* option in the Web Console. (**Note:** the MVS and VM servers do not use the Web Console.)
  - The communications protocol (for example, TCP/IP or LU6.2), including vendor and release.
- The stored procedure (preferably with line numbers) or SQL statements being used in server access.
- The database server release level.
- The database name and release level.
- The Master File and Access File.

- The exact nature of the problem:
  - Are the results or the format incorrect? Are the text or calculations missing or misplaced?
  - Is there an error message and return code (if applicable)?
  - Is this related to any other problem?
- Has the procedure or query ever worked in its present form? Has it been changed recently? How often does the problem occur?
- What release of the operating system are you using? Has it, your security system, communications protocol, or front-end software changed?
- Is this problem reproducible? If so, how?
- Have you tried to reproduce your problem in the simplest form possible? For example, if you are having problems joining two data sources, have you tried executing a query containing just the code to access the data source?
- Do you have a trace file?
- How is the problem affecting your business? Is it halting development or production? Do you just have questions about functionality or documentation?

## **User Feedback**

---

In an effort to produce effective documentation, the Documentation Services staff welcomes your opinions regarding this manual. Please use the Reader Comments form at the end of this manual to relay suggestions for improving the publication or to alert us to corrections. You can also use the Documentation Feedback form on our Web site, <http://www.iwaysoftware.com>. Thank you, in advance, for your comments.

## **iWay Software Training and Professional Services**

---

Interested in training? Our Education Department offers a wide variety of training courses for iWay Software and other Information Builders products.

For information on course descriptions, locations, and dates, or to register for classes, visit our World Wide Web site (<http://www.iwaysoftware.com>) or call (800) 969-INFO to speak to an Education Representative.

Interested in technical assistance for your implementation? Our Professional Services department provides expert design, systems architecture, implementation, and project management services for all your business integration projects. For information, visit our World Wide Web site (<http://www.iwaysoftware.com>).

---

---

# Contents

<b>1. Introducing the iWay Adapter for CICS .....</b>	<b>1-1</b>
Overview of the Adapter .....	1-2
The iWay Adapter for CICS .....	1-3
CICS Programs .....	1-4
Software Requirements for the iWay Adapter for CICS .....	1-4
<b>2. Configuring the iWay Adapter for CICS .....</b>	<b>2-1</b>
Accessing a Region .....	2-2
Configuring a Connection to CICS .....	2-2
Connecting to a System From Application Explorer .....	2-6
Disconnecting From or Deleting a Connection to CICS .....	2-7
<b>3. Designing the iWay Adapter for CICS .....</b>	<b>3-1</b>
Creating an Adapter Transaction .....	3-2
Sample Program IWAYSAMP .....	3-3
Creating an Event .....	3-6
Creating an Event Port .....	3-7
Editing and Deleting an Event Port .....	3-11
Creating a Channel .....	3-13
Editing and Deleting a Channel .....	3-20
Understanding iWay Business Services .....	3-20
Creating an iWay Business Service .....	3-20
Testing a Web Service .....	3-22
Accessing a Web Service .....	3-23
<b>4. Using WebLogic Workshop .....</b>	<b>4-1</b>
Using WebLogic Workshop to Create and Access a Web Service .....	4-2
Creating a Schema for a Process Definition .....	4-3
Creating and Inserting a Control for a Web Service .....	4-6
Creating an Input Variable and an Output Variable .....	4-10
Configuring Parameters to Start an Event .....	4-13
Running the Process Definition From WebLogic Workshop .....	4-15
<b>A. Configuring VTAM and CICS .....</b>	<b>A-1</b>
VTAM and AnyNet Configuration .....	A-2
CICS Configuration .....	A-4
<b>B. Installing the Sample IWAYIVP and IWAYSAMP Programs in CICS .....</b>	<b>B-1</b>
Installing and Configuring the Sample CICS Program IWAYIVP in CICS .....	B-2
Installing and Configuring the Sample CICS Program IWAYSAMP in CICS .....	B-5

- C. Sample Requests, Schemas, and Cobol File Descriptions .....C-1**
  - Request Document for the Generic Transaction IWAYIVP .....C-2
  - Request Schema for the Generic Transaction IWAYIVP .....C-2
  - Response Schema for the Generic Transaction IWAYIVP .....C-3
  - Request Document for the Program IWAYSAMP .....C-3
  - Request Schema for the Program IWAYSAMP .....C-4
  - Response Schema for the Program IWAYSAMP .....C-5
  - Sample Cobol File Descriptions .....C-6
- D. Sample CICS Programs ..... D-1**
  - IWAYIVP Program ..... D-2
  - IWAYSAMP Program ..... D-3
  - CICS TCP/IP Sockets Program for Mainframe ..... D-4
  - CICS TCP/IP Sockets Program for TX ..... D-9
- E. Debugging and Troubleshooting .....E-1**
  - Messages .....E-2



---

---

## CHAPTER 1

# Introducing the iWay Adapter for CICS

### Topics:

- Overview of the Adapter
- The iWay Adapter for CICS

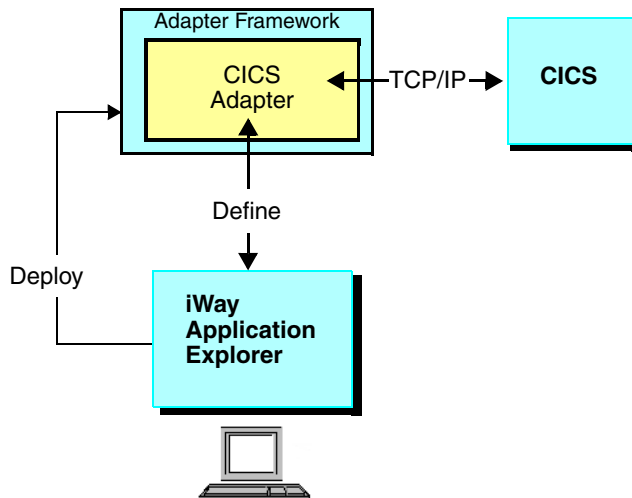
This section describes the iWay Adapter for CICS. The adapter supports automatic transaction invocation, message transformation, and error recovery. The adapter enables applications to call CICS programs and to work with the native features and syntax of CICS.

## Overview of the Adapter

---

The adapter enables you to execute CICS programs. The advantages of the adapter include the following:

- No modification required to existing CICS programs.
- No installation of new code required on CICS.
- Adapter processing performed off of the mainframe.
- Configuration by metadata—no coding required.
- Support for older versions of CICS.
- Support for CICS COMMAREA programs.

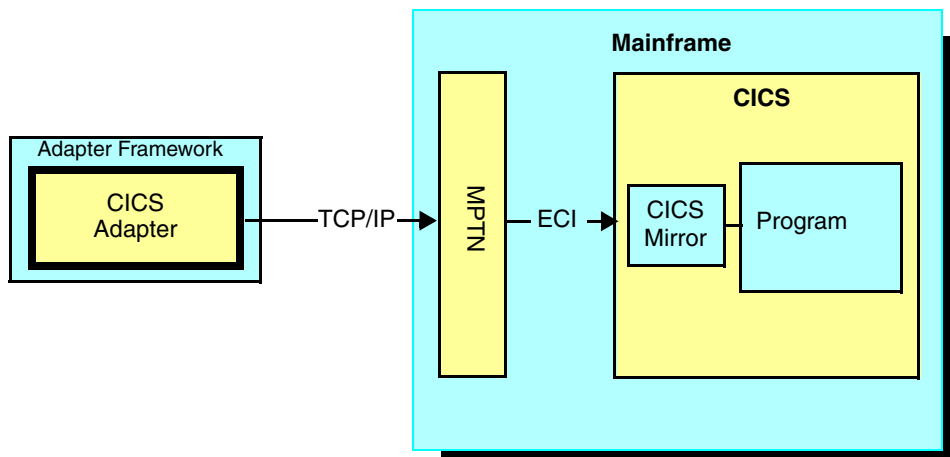


The following bidirectional scenarios are supported by the adapter:

- CICS services
- CICS events

## The iWay Adapter for CICS

The following diagram illustrates the iWay Adapter for CICS:



The iWay Adapter for CICS is the component that connects to CICS. It is hosted in a container that supports events. The adapter enables the following functions:

- Connecting to CICS
- Executing COMMAREA programs
- Mapping XML messages to and from CICS data structures
- Listening for events triggered in CICS

The adapter enables you to invoke a CICS program by sending a request and retrieving the response.

The iWay Adapter for CICS sends the request to execute a transaction over the Multi-Platform Transport Network (MPTN, also known as AnyNet). This enables the adapter to use TCP/IP to send the request although CICS is expecting LU6.2 (also known as APPC).

The iWay Adapter for CICS attaches the CICS Mirror transaction, CPMI, which is the standard External Communication Interface (ECI) transaction for ASCII clients.

At design time, you describe the request and response messages by mapping them to Cobol descriptions.

**Note:**

- Because distributed transactions are not supported, the synchronization level is CONFIRM.
- An extended call (executing several program calls in one unit of work) currently is not supported.

## **CICS Programs**

The two main types of CICS programs are:

- COMMAREA programs that are designed to be called by other CICS programs.
- 3270 programs that read and write terminal screen maps using Basic Mapping Support (BMS).

Because the adapter can execute only COMMAREA programs, this distinction is important.

To execute 3270 programs, you require a screen scraper such as the iWay Adapter for 3270. For many years CICS applications were structured so that the business processing, as opposed to the screen dialogue, was in COMMAREA programs. Therefore, in many cases, executing a COMMAREA program is recommended for application integration.

## **Software Requirements for the iWay Adapter for CICS**

The following are the software requirements for the iWay Adapter for CICS:

- OS/390 v2.6 or higher or z/OS.
- TCP/IP communication available to the adapter.
- VTAM AnyNet option.

The AnyNet option always communicates back to the calling environment on a specific port (usually, 397). Therefore, the server platform must not use any other product that also requires the services of port 397.

On UNIX, the use of port number 397 requires root privileges, therefore, the server must be installed with a user ID that has root privileges. A port number greater than 1000 does not require root privileges.

- One of the following releases of CICS:
  - CICS Transaction Server for z/OS, Version 2 Release 2.
  - CICS Transaction Server for OS/390, Version 1 Release 2 or higher.
  - IBM CICS/ESA, Version 4 Release 1.
  - CICS Transaction Server for VSE/ESA, Version 1.1.0 or higher.
  - CICS for VSE/ESA, Version 2.3.
  - CICS for IBM OS/400, Version 4.4.
  - TXSeries, Version 4.2 (HP-UX); TXSeries, Version 4.3 with PTF 4 (Windows NT, AIX, Sun<sup>SM</sup> Solaris<sup>TM</sup> operating environment); and TXSeries, Version 5.0 (AIX and Windows)



---

---

## CHAPTER 2

# Configuring the iWay Adapter for CICS

### Topics:

- Accessing a Region
- Disconnecting From or Deleting a Connection to CICS

At design time, you use Application Explorer to create the configuration and metadata the adapter requires at run time. This section describes how to configure a connection to CICS.

## Accessing a Region

---

To access a CICS region, you must configure a connection to that region. After the connection is created, it automatically is saved. You must establish a connection to the system every time you start Application Explorer or after disconnecting.

### Configuring a Connection to CICS

You configure a connection to the region by entering values in the Set connection info dialog box.

#### **Procedure** How to Configure a Connection to CICS

To configure a connection to CICS:

1. Expand the *iWay Adapters* node in Application Explorer.
2. Click the *CICS* node.
3. In the right pane, move your pointer over *Operations* and select *Define a new target*.

The Add a new CICS target dialog box opens.

- a. In the Target Name field, type a name for the connection, for example, CICS\_Connection.

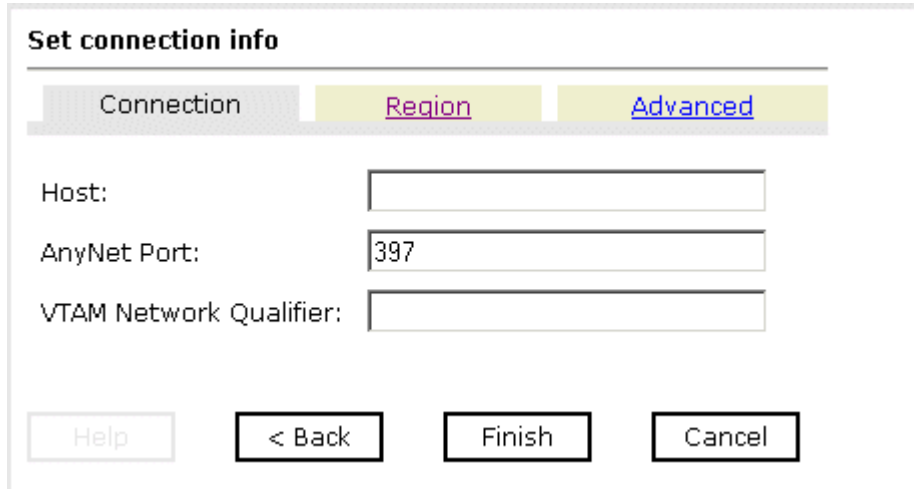
The name is used to build a repository entry as well as to identify the connection.

- b. In the Description field, type a description for the target name you just created, for example, Connection to CICS.
  - c. In the Target Type drop-down list, select *CICS*.
4. Click *Next*.

The connection name is verified for the system. If you entered an invalid instance name, a new dialog box opens and prompts you for an instance name again.



The Set connection info dialog box opens.



The dialog box titled "Set connection info" has three tabs: "Connection", "Region", and "Advanced". The "Region" tab is currently selected and highlighted in yellow. Below the tabs, there are three input fields: "Host:" (empty), "AnyNet Port:" (containing "397"), and "VTAM Network Qualifier:" (empty). At the bottom, there are four buttons: "Help", "< Back", "Finish", and "Cancel".

**5.** Enter the parameters to create a new connection to CICS.

You can obtain this information from the CICS systems programmer. This information should be the same for all transactions and messages in a single CICS system.

For information on the relationship of these parameters to the CICS VTAM definitions, see Appendix A, *Configuring VTAM and CICS*.

- a.** Click the *Connection* tab and type values for the parameters described in the following table:

Parameter	Description
Host	Host name, or IP address, for the computer where CICS is running.
AnyNet Port	Port number on which CICS is listening. The AnyNet port, for example, 397, is the port which AnyNet (the mainframe TCP to SNA bridge) listens on.
VTAM Network Qualifier	The network qualifier for VTAM.

- b. Click *Region* to view the parameters on the Region tab.

**Set connection info**

[Connection](#) | Region | [Advanced](#)

User ID:

Password:

CICS LU:

Local LU:

LogMode:

Trusted:

☐

Codepage:

EBCDIC

Number of Sessions:

2

Help

< Back

Finish

Cancel

- c. Enter values for the following parameters:

Parameter	Description
User ID	Valid user ID for CICS.
Password	Valid password associated with the CICS user ID.
CICS LU	VTAM applid to connect to the CICS system.
Local LU	LU of the SNA access point to which you have access, for example, SNA server.
LogMode	Log mode for the connection to CICS, for example, PARALLEL.
Trusted	Security level of VERIFIED (user ID and password are required) is supported unless Trusted is checked. If Trusted is checked, a security level of IDENTIFY (user ID only) is supported.  Do not specify LOCAL as the user ID is always sent.
Codepage	CICS system's code page that enables the adapter to correctly translate the incoming data to XML, for example, transforming EBCDIC to ASCII and correctly interpreting binary data.
Number of Sessions	Number of sessions for the AnyNet connection.

- d. If you are executing Adabas/Natural programs, click the *Advanced* tab.

**Set connection info**

Connection Region Advanced

Natural Nucleus:

Proxy Program:

Natural Logon Parameters:

Help < Back Finish Cancel

The CICS/Natural Bridge enables Natural programs to be invoked by the adapter through CICS using Software AG's Natural CICS Interface.

- e. On the Advanced tab, enter values for the following parameters:

Parameter	Description
Natural Nucleus	Name of the Natural subsystem on which the Natural program you wish to invoke resides.
Proxy Program	CICS program that calls the Natural CICS Interface. The name of the proxy provided by iWay Software is AASNATC, however, you can use another proxy.
Natural Logon Parameters	String that represents the default logon parameters. It can be modified depending on installation requirements.  <b>Note:</b> Software AG's Natural CICS Interface requires a programmatic "logon" to the Natural System.

6. Click *Finish*.

The newly created connection, CICS\_Connection, appears as a node under the CICS service adapter. The configuration information is stored in the repository for the configuration you defined at installation time.

## Connecting to a System From Application Explorer

You must establish a connection to the system every time you start Application Explorer or after disconnecting.

### **Procedure** How to Connect to CICS\_Connection From Application Explorer

To connect to CICS\_Connection from Application Explorer:

1. Move your pointer over *Operations*, and select *Connect*.

The Connect to CICS\_Connection dialog box opens, populated with values you entered for the connection parameters.

2. Verify your connection parameters and enter a valid password for your system.
3. Click *OK*.

Application Explorer connects and loads the CICS metadata. Now you can create schemas for transactions. The schemas are stored in

`c:\iWay55\config\base\CICS\CICS_Connection`

where:

`CICS_Connection`

Is the descriptive name for the session.

The icon for the node changes indicating that the node is connected.

## Disconnecting From or Deleting a Connection to CICS

---

To manage CICS connections, you can:

- Disconnect from a connection that currently is not in use.  
Although you can maintain multiple open connections to different transaction processing systems, it is recommended to disconnect from connections not in use.
- Delete a connection that is no longer required.

### **Procedure** How to Disconnect From a Connection to CICS

To disconnect a connection:

1. Expand the *CICS* node.
2. Click the connection, for example, *CICS\_Connection*, move your pointer over *Operations* and select *Disconnect* from the pop-up menu.  
Disconnecting from CICS drops the connection with CICS, but the node remains.  
The icon for the node changes indicating that the node is disconnected.
3. To establish the connection again, click the disconnected node, move your pointer over *Operations*, and select *Connect* from the pop-up menu.

### **Procedure** How to Delete a Connection to CICS

To delete a connection from the list of existing connections:

1. Expand the *CICS* node.
2. Click the connection, for example, *CICS\_Connection*, move your pointer over *Operations* and select *Delete* from the pop-up menu.  
A message appears, prompting you to confirm the deletion of the node.
3. Click *OK*.  
The node disappears from the list of available connections.



---

---

## CHAPTER 3

# Designing the iWay Adapter for CICS

### Topics:

- Creating an Adapter Transaction
- Creating an Event
- Creating an Event Port
- Creating a Channel
- Understanding iWay Business Services
- Creating an iWay Business Service

Application Explorer is a Web application running within a servlet container and is accessible through a browser. It enables the iWay Adapter for CICS to explore metadata and create XML schemas.

The following topics describe how to use Application Explorer to create CICS transactions and generate request and response XML schemas for new or existing transactions. These schemas are used to represent a transaction for integration with external systems.

In addition, this section provides information on how to use the generated schemas to listen for events in CICS and create iWay Business Services, which expose functionality as Web services.

## Creating an Adapter Transaction

---

After you create a connection to CICS, you can add CICS adapter transactions using Application Explorer. A single CICS connection may be associated with multiple transactions. Each transaction represents one service offered by CICS and consists of a program and its metadata.

A generic transaction always is added automatically and represents CICS services whose data will not be mapped to XML. You can use a generic transaction for programs that accept no input and for programs that return no output or when it is acceptable to return a non-formatted answer set.

For example, the supplied program IWAYIVP connects to CICS and returns "Congratulations" on successful adapter installation and configuration. Because IWAYIVP requires no input or output, you do not require Cobol descriptions for the input or output. One request and response schema is applicable for this program. The request schema for the generic transaction is in Appendix B, *Sample Requests, Schemas, and Cobol File Descriptions*.

Using the generic transaction, the XML request document that is received must include the name of the program to be called in the <Transaction> element. The payload to be sent as the COMMAREA must be in the <message> tag, which can be a maximum of 32,500 bytes.

The generic response schema is constructed from the data received from CICS. If the <message> element has more than 80 bytes, the received COMMAREA is split into 80 byte messages. Illegal XML characters ('<', '/', and '&') are converted to XML entities. For programs that require input and output and a formatted response, which is usually the case, (IWAYIVP is an exception), you must add your own adapter transactions, as described in the following procedure. XML request messages must specify the transaction to use in the location attribute of the <Transaction> tag. For example, if you create a CICS transaction called IWAYSAMP, the location is "CICS/Transactions/IWAYSAMP".

To view a sample generic request or response schema or for information about specifying a transaction to use in the location attribute of the <Transaction> tag, see Appendix B, *Sample Requests, Schemas, and Cobol File Descriptions*.



## Sample Program IWAYSAMP

iWay Software supplies the IWAYSAMP and IWAYIVP programs with the iWay Adapter for CICS. This document uses the IWAYSAMP program for illustration purposes and as a reference for the adapter. Based on what is passed to the IWAYSAMP program, an answer set is returned from the program with two possible layouts.

- If the value for the field STAT is 'P', the program returns 40 bytes of data.
- If the value for the field STAT is 'F', the program returns 60 bytes of data.

The IWAYSAMP program is an example of a program that returns multiple answer sets. Two different answer sets are returned based on what is passed in the request. The adapter enables you to create a response schema that contains different possible return messages.

Sample request documents are in Appendix C, *Sample Requests, Schemas, and Cobol File Descriptions*, with a sample response schema for the IWAYSAMP program. You specify the output as explained in *Creating an Adapter Transaction* on page 3-2. You must know the field in the Cobol description that can be used as a record type and the value of that field. You specify the value of the field to create the appropriate response schema. This is also true for events to determine what layout is returned from CICS when you configure a CICS event.

### **Procedure** How to Create an Adapter Transaction

To create an adapter transaction:

1. Expand the *CICS* node and connect to a CICS target.
2. Expand the node to which you connected.  
The Transaction node appears under the connected node.
3. Click *Transactions*, move your pointer over *Operations*, and select *Add Service*.  
The Add dialog box opens.

4. To map to the Cobol descriptions, enter the appropriate information for the CICS transaction.

The following table lists the parameters.

Field	Description
Node Name	Describes the adapter transaction you create.  The name, for example, CICS_Transaction appears under the Transactions node for the current connection. The name to use in the <Transaction location="..."> attribute.
Transaction Name	Name of the program to be called in CICS, for example, IWAYSAMP. The IWAYSAMP program appears in Appendix D, <i>Sample CICS Programs</i> .
Cobol File Description for Input	Location of the Cobol description that describes the COMMAREA of the CICS program to execute.  Converted by the adapter to an XML schema that the adapter uses to map from XML to the format required by CICS at run time.
Size of COMMAREA	Size of the COMMAREA (in bytes) for programs that expect a specific size. By default, the adapter passes 32,500 to the program. For best performance, specify a number that can accommodate the larger of the input COMMAREA or output COMMAREA. For example, to run IWAYSAMP, specify 60.
Natural Library	Run-time location of the Natural program to execute.  Use only if the adapter is to execute Adabas/Natural programs.
Cobol Description for Output (outFD)	Path that corresponds to the message you want returned from the CICS program.  If the program can return multiple types of messages, for each output Cobol description, enter the Cobol description field and value to determine the schema to use for a particular message.  Application Explorer creates the schema to use for a particular message based on the contents of a field that is returned. For example, a program called IWAYSAMP_IN populates the COMMAREA field called STAT. Depending on program logic, Application Explorer creates the correct response schema.

Field	Description	
	<p>Path that corresponds to the message you want returned from the CICS program.</p> <p>If the program can return multiple types of messages, for each output Cobol description, enter the Cobol description field and value to determine the schema to use for a particular message.</p> <p>Application Explorer creates the schema to use for a particular message based on the contents of a field that is returned. For example, a program called IWAYSAMP_IN populates the COMMAREA field called STAT. Depending on program logic, Application Explorer creates the correct response schema.</p>	
	Value in STAT Field	Cobol Description
	<p>'P'</p> <p>'F'</p>	<p>IWAYSAMP_OUT_P</p> <p>IWAYSAMP_OUT_F</p>
	<p>The IWAYSAMP_OUT_P and IWAYSAMP_OUT_F Cobol descriptions appear in Appendix C, <i>Sample Requests, Schemas, and Cobol File Descriptions</i>.</p>	

**Note:** You must transfer the Cobol descriptions to a location accessible to Application Explorer. For the correct Cobol descriptions to use for the program, contact your CICS Administrator or application developer.

**5.** Click *Add*.

The new CICS transaction is added, for example, CICS\_Transaction under the Transactions node for the current connection.

The adapter generates the schemas for the selected Cobol descriptions and associates them with the transaction. The schemas generated for the sample Cobol descriptions appear in Appendix B, *Sample Requests, Schemas, and Cobol File Descriptions*.

## Creating an Event

---

Events are generated by the CICS transaction processing system as a result of activity on that system. You can use events to trigger an action in your application. For example, the CICS application program can generate an event when customer information is updated. If your application must perform when this happens, you must provide a mechanism to publish that event to the outside world.

The adapter has the capability of capturing events using protocols such as TCP/IP or a File directory. For other protocols, such as HTTP, contact Customer Support Services.

The following steps describe creating an event and processing it by the adapter:

- Creating the CICS application program.

For example, a program can be triggered by certain criteria and publish information to the specific protocol, such as TCP/IP. The application program that is written for the event passes data to a TCP/IP port.

- Configuring the connection using Application Explorer to create the event schema from a Cobol File Description.

The Cobol File Description describes the format and layout of the data that is passed by the TCP/IP program to the TCP/IP port.

- Configuring an iWay port and channel using Application Explorer.

After the event schema is created, you can configure an iWay port and channel using Application Explorer. The port you specify is the same port to which the CICS application event program is writing the event data.

For information on creating an iWay event port, see *Creating an Event Port* on page 3-7. For information on creating a channel, see *Creating a Channel* on page 3-13.

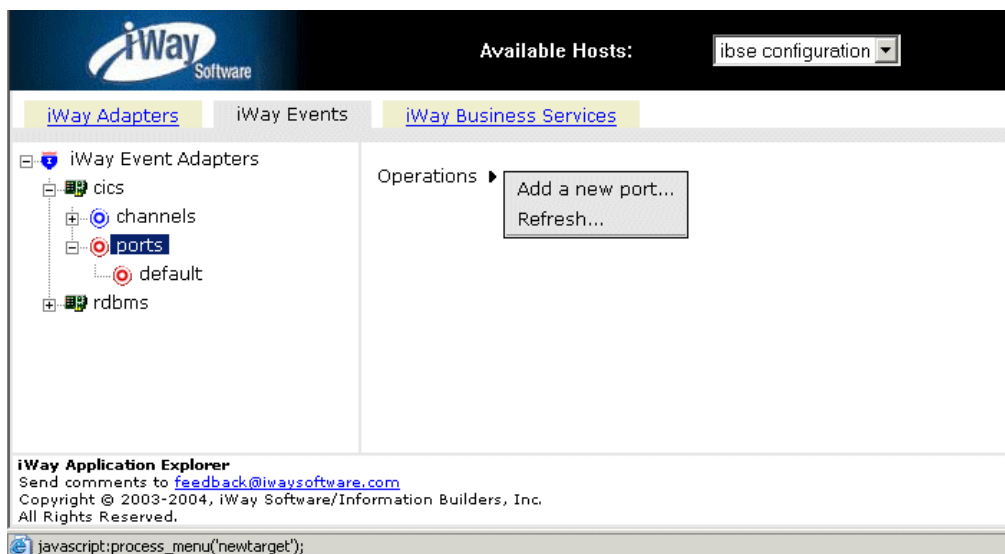
## Creating an Event Port

Creating an iWay port is used to designate the disposition of the event. The event is captured, and the message is sent to a destination. The event can be posted to a file, which is the default disposition for the adapter. This guide illustrates events using a file disposition, which enables the CICS event to be written to a file on disk. For other dispositions, such as HTTP, contact Customer Support Services.

The following procedure describes how to create an event port using Application Explorer.

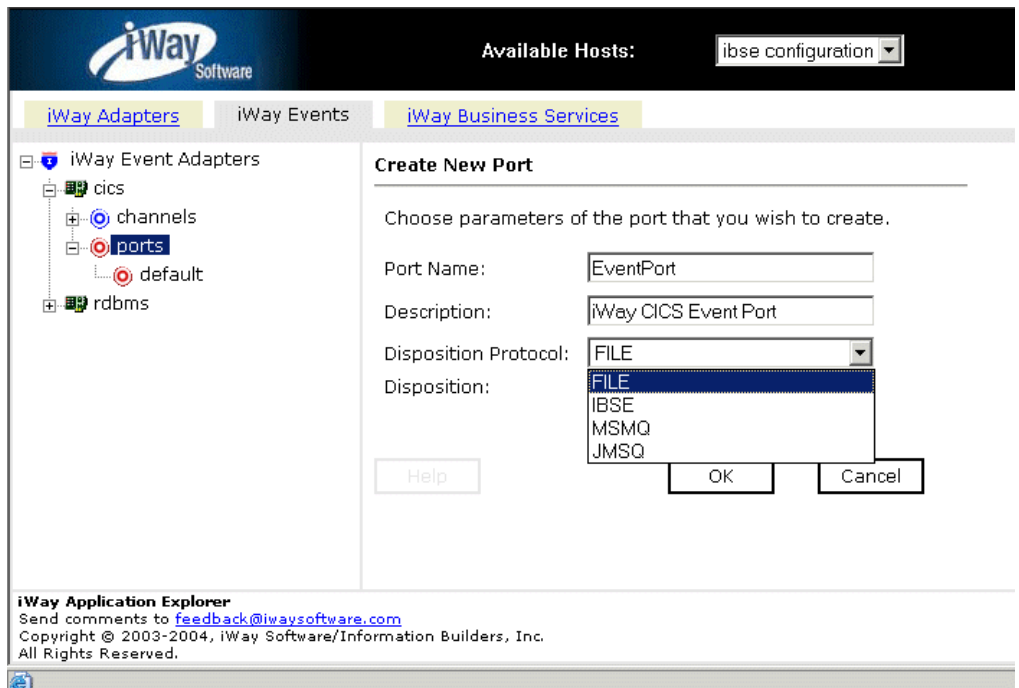
### **Procedure** How to Create an Event Port

To create a specific event port using Application Explorer:



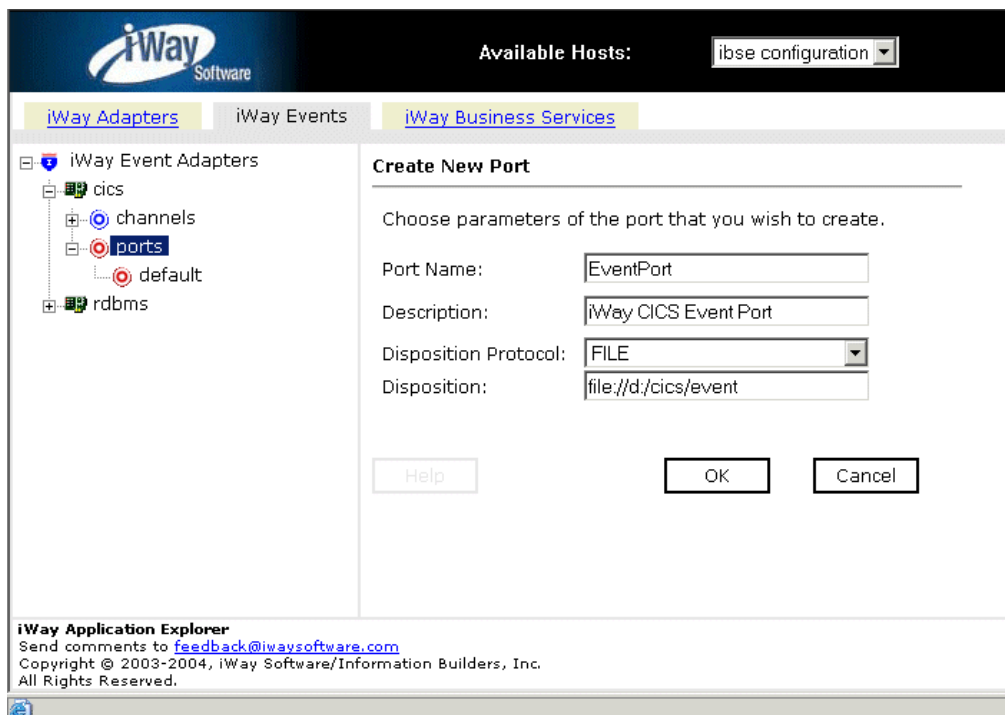
1. In the left pane, click the *ports* node.
2. In the right pane, move your cursor over *Operations* and select *Add a new port* from the pop-up menu.

The Create New Port pane opens.



3. Specify information for the port you are creating:
  - a. In the Port Name field, type a descriptive name for the port, for example, EventPort.
  - b. In the Description field, type a brief description of the port, for example, iWay CICS EventPort.
  - c. In the Disposition Protocol field, select a disposition from the drop-down list, for example, FILE.
4. In the Disposition field, enter a destination for the event data, for example, for FILE:  
`file://c:\temp\CICSEvent`

The following window illustrates a pane with the fields completed.



5. If you want to configure a JMS queue as a disposition, configure the disposition as follows:
  - a. In the Disposition Protocol drop-down list, select *JMSQ*.
  - b. In the Disposition field, enter a JMS destination, using the following format:

```
jmsq:myQueueName@javax.jms.QueueConnectionFactory;jndiurl=t3:
//localhost:7001;jndifactory=weblogic.jndi.WLInitialContextFactory
```

The following table lists and describes each parameter.

Parameter	Description
queue	JNDI name of a queue to which events are emitted.
Connection Factory	Resource that contains information about the JMS Server. The WebLogic connection factory is: <a href="#">javax.jms.QueueConnectionFactory</a>
jndiurl	URL of the WebLogic Server <a href="#">t3://host:port</a> where: <a href="#">host</a> Is the machine name where the Weblogic Server is installed. <a href="#">port</a> Is the port on which Weblogic Server is listening. The default port, if not changed at installation, is 7001.
jndifactory	JNDI context.INITIAL_CONTEXT_FACTORY provided by the JNDI service provider. For WebLogic Server, the WebLogic factory is weblogic.jndi.WLInitialContextFactory.

**6.** Click *OK*.

The newly created port appears under the port section of the CICS event adapter.

You are now ready to associate the event port with a channel. For more information, see *Creating a Channel*.

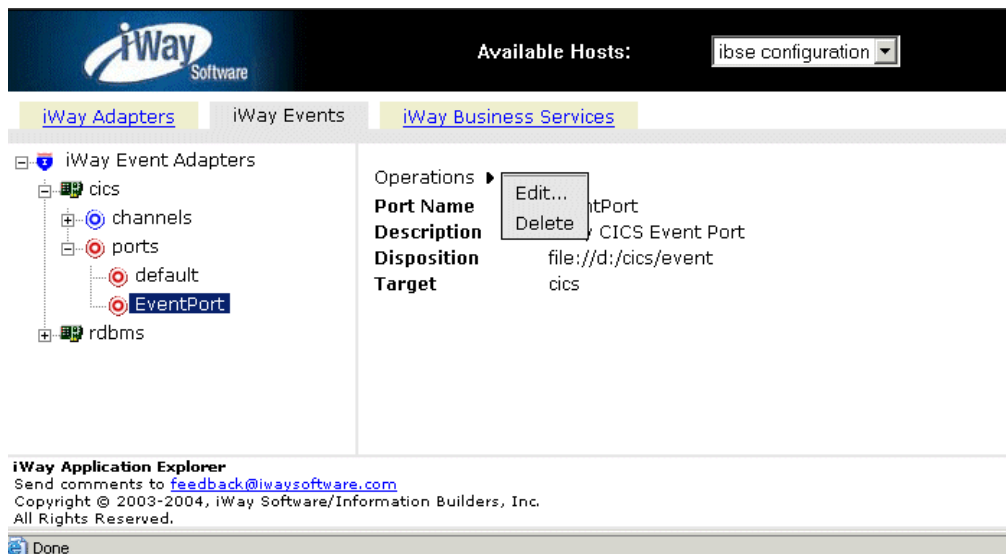


## Editing and Deleting an Event Port

The following procedures describe how to edit and delete an event port

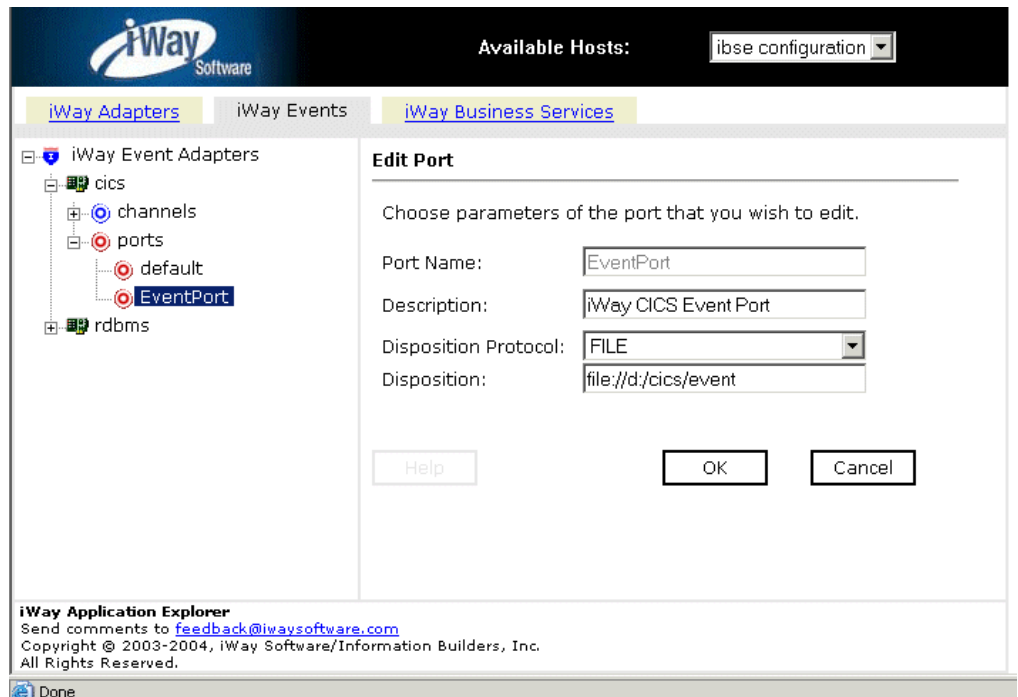
### **Procedure** How to Edit an Event Port

To edit an existing event port:



1. In the left pane, select the event port you want to edit, for example, EventPort.
2. In the right pane, move your cursor over *Operations* and select *Edit* from the pop-up menu.

The Edit Port pane opens.



3. Make any required changes to the event port configuration and click **OK**.

### **Procedure** How to Delete an Event Port

To delete an existing event port:

1. In the left pane, select the event port you want to delete.
2. In the right pane, move your cursor over *Operations* and select *Delete* from the pop-up menu.

A confirmation dialog box opens.

3. To delete the event port you selected, click **OK**.

The selected event port disappears from the list in the left pane.

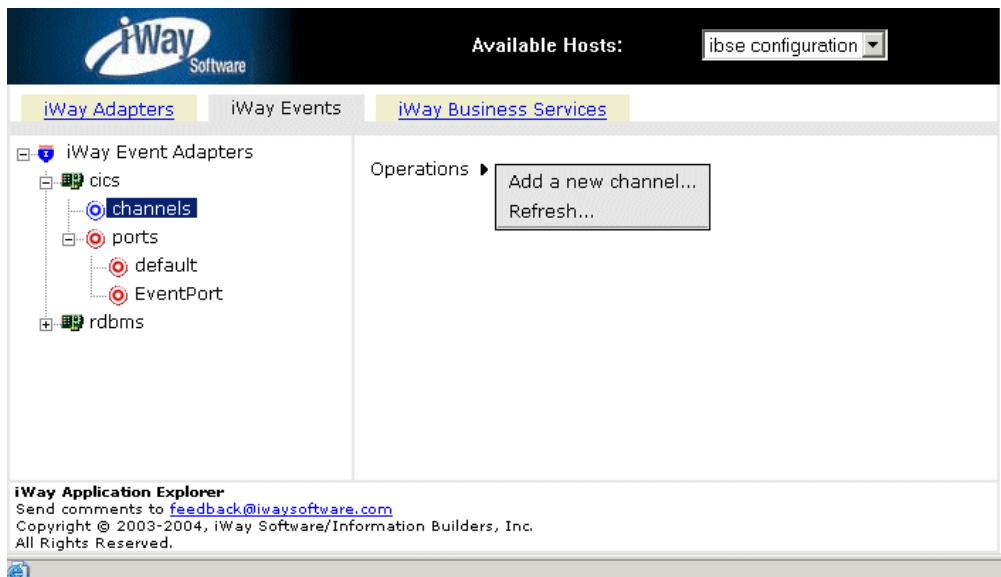
## Creating a Channel

You create a channel to associate an event with an iWay port and also potentially to process the event document. The event that is passed through the channel is processed by preparers or premitters defined in the channel. A premitter is created automatically by assigning a Cobol description to it that describes the layout of the data being passed by CICS to the channel.

The following procedure describes how to create a channel for your event. All defined event ports must be associated with a channel.

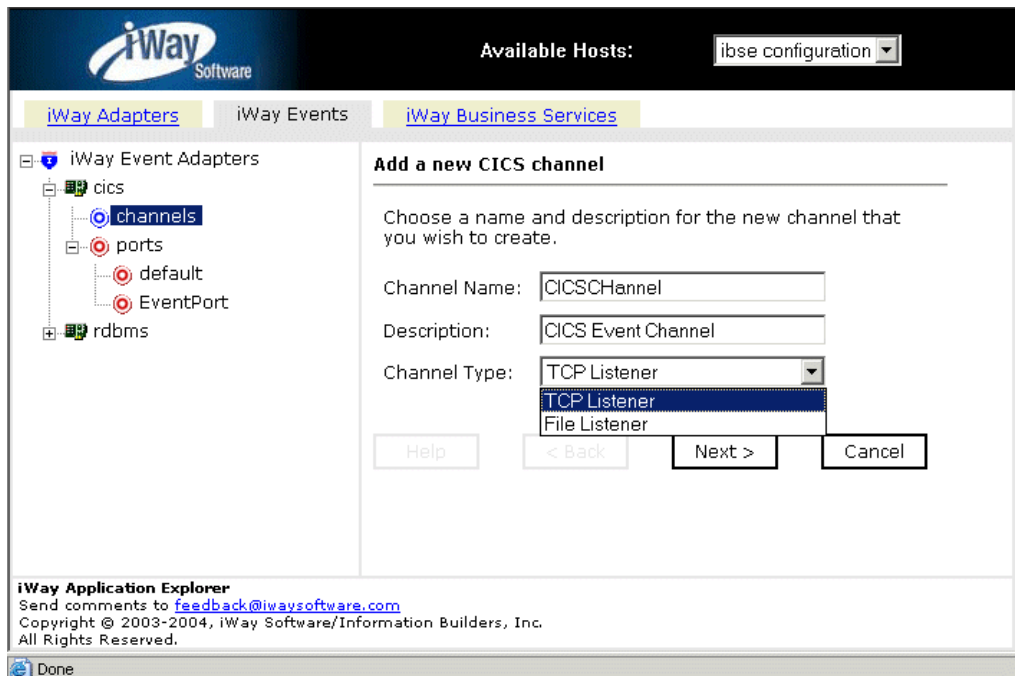
### **Procedure** How to Create a Channel

To create a channel using Application Explorer:



1. Click the *channels* node.
2. In the right pane, move your cursor over *Operations* and select *Add a new channel* from the pop-up menu.

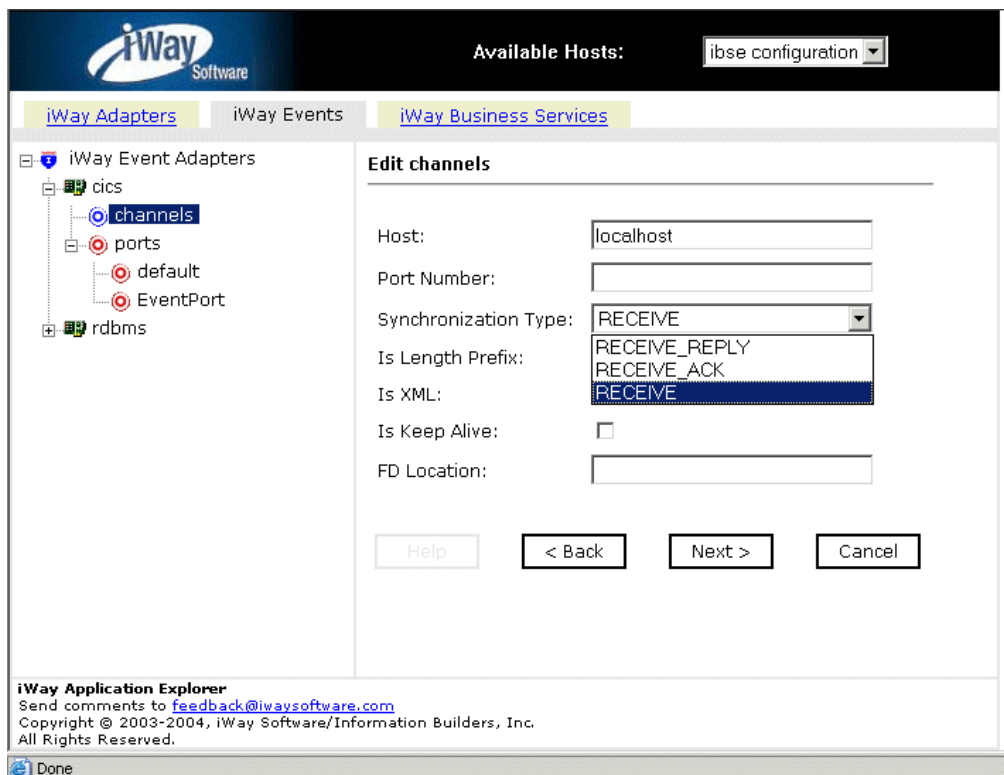
The Add a new channel pane opens.



3. Specify information for the channel you are creating:
  - a. In the Channel Name field, enter a descriptive name for the channel, for example, CICSChannel.
  - b. In the Description field, enter a brief description for the channel, for example, CICS Event Channel.
  - c. In the Channel Type field, select a channel type from the drop-down list, for example, TCP Listener.

4. Click **Next**.

The Edit channels pane opens.



5. Specify the values for the protocol you are using with the channel.

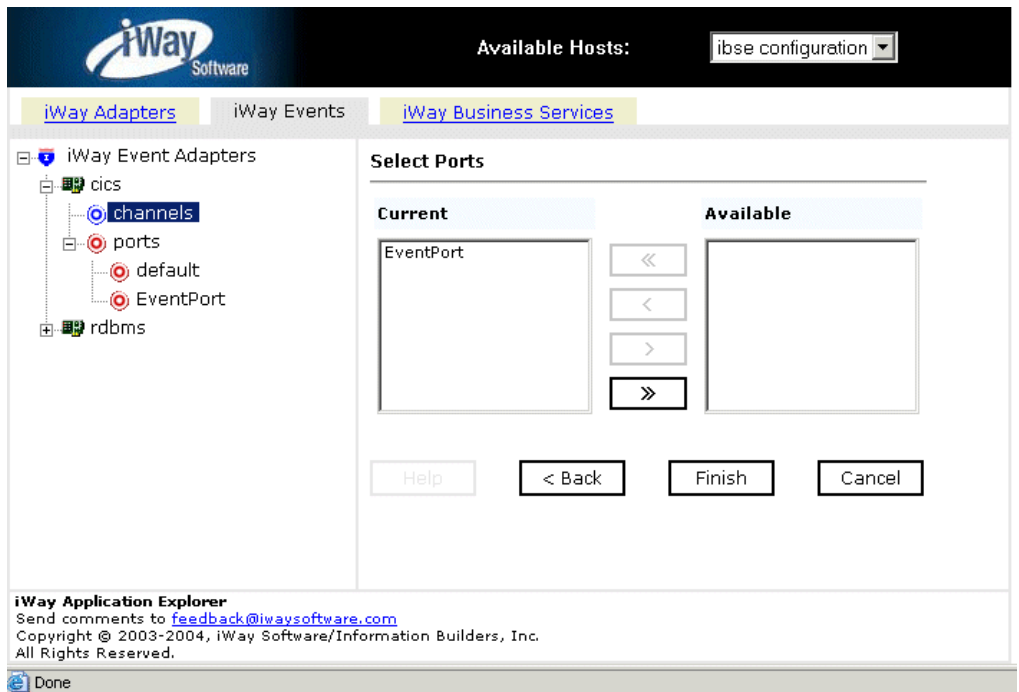
The following table lists and describes each property.

Property	Description
Host	Host name of the application server.
Port Number	For TCP/IP, specify port number.
Synchronization Type	<ul style="list-style-type: none"><li>• Select RECEIVE_REPLY if the event application expects a reply sent back to it. Specify a preemitter.</li><li>• Select RECEIVE_ACK when a TCP/IP acknowledgement (ACK) is sent back to the event application.</li><li>• Select RECEIVE if the event application does not expect a return.</li></ul>
Is Length Prefix	For CICS events that send data back that is not in XML format. The TCP/IP event application must prefix the data with a 4-byte binary length field when writing the data to the TCP/IP port.
Is XML	For CICS events that send data back in XML format. No preparser is required.
Is Keep Alive	Maintains continuous communication between the event transaction and the channel.

Property	Description
FD Location	<p>Path that corresponds to the message you want returned from CICS, based on an event.</p> <p>If the program can return multiple types of messages, for each output Cobol description, enter the Cobol description field and value to determine the schema used for a particular message.</p> <p>Application Explorer creates the schema to use for a particular message based on the contents of a particular field that is returned. For example, a program called IWAYSAMP populates the COMMAREA field called STAT. Depending on program logic, a value of P in the STAT field indicates the IWAYSAMP_OUT_P Cobol description, a value of F in the STAT field, indicates the IWAYSAMP_OUT_F Cobol description.</p> <p>The IWAYSAMP_OUT_F and IWAYSAMP_OUT_P Cobol descriptions appear in Appendix C, <i>Sample Requests, Schemas, and Cobol File Descriptions</i>.</p>

6. Click *Next*.

The Select Ports pane opens.

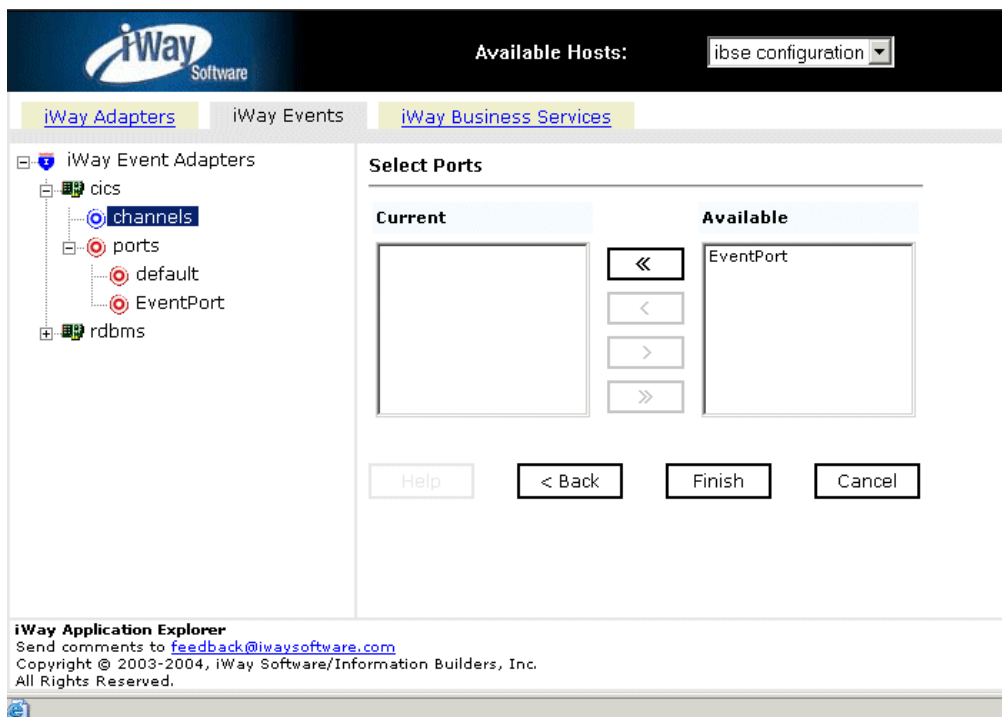


You must associate an existing event port with the channel you are creating.

- a. From the list of current ports, select an event port and click the single right arrow button (>) to transfer it to the list of available ports.
- b. To associate all the event ports when there are more than one, click the double right arrow button (>>).



After you select and move the event port or ports, the port(s) appear(s) in the list of available ports.



**7. Click *Finish*.**

The summary pane opens on the right. The summary provides the channel description, channel status, and available ports. All the information is associated with the channel you created.

The channel also appears under the channels node in the left pane. An X over the icon indicates that the channel is currently disconnected.

You must start the channel to activate your event configuration.

**8. In the right pane, move your cursor over *Operations*.**

- a. Select *Start the channel* from the pop-up menu.**

The channel you created becomes active, and the X over the icon disappears.

- b. If you want to stop the channel at any time, move your cursor over *Operations* and select *Stop the channel* from the pop-up menu.**

## Editing and Deleting a Channel

The following procedures describe how to edit and delete a channel.

### **Procedure** How to Edit a Channel

To edit an existing channel:

1. Select the channel you want to edit.
2. In the right pane, move your cursor over *Operations* and select *Edit* from the pop-up menu.

The Edit Channel pane opens.

3. Make any required changes to the channel configuration and click *OK*.

### **Procedure** How to Delete a Channel

To delete an existing channel:

1. Select the channel you want to delete.
2. In the right pane, move your cursor over *Operations* and select *Delete*.

The selected channel disappears.

## Understanding iWay Business Services

---

Application Explorer provides Web developers with a simple, consistent mechanism for extending the capabilities of the adapter. The iWay Business Services Engine exposes functionality as Web services. It serves as a gateway to heterogeneous back-end applications and databases.

A Web service is a self-contained, modularized function that can be published and accessed across a network using open standards. It is the implementation of an interface by a component and is an executable entity. For the caller or sender, a Web service can be considered as a “black box” that may require input and delivers a result. A Web service integrates within an enterprise as well as across enterprises on any communication technology stack, whether asynchronous or synchronous, in any format.

## Creating an iWay Business Service

After you connect to your application system and create an XML schema for a transaction, you can create a Web service. The following procedure describes how to create iWay Business Services using Application Explorer.

**Procedure How to Create an iWay Business Service**

To create an iWay business service:

1. In the left pane, select the transaction for which you want to create a Web service.
2. In the right pane, move your cursor over *Operations* and select *Create iWay Business Service* from the pop-up menu.

The Create Web Service pane opens.
3. Select the *Create a new service* option button and click *Next*.
4. Enter information that is specific to the Web service you are defining.
  - a. In the Service Name field, type a descriptive name for the Web service.
  - b. In the Description field, type a brief description for the Web service (optional).
  - c. In the License field, select one of more license codes to assign to the Web Service.
5. Click *Next*.
6. Enter information that is specific to the method you are defining.
  - a. In the Method Name field, type a descriptive name for the method.
  - b. In the Description field, type a brief description for the method.
7. Click *Finish*.

The iWay Business Services engine tab opens. The Web service is created and published to the iWay Business Services engine. Application Explorer displays the newly created Web Service under the iWay Business Services folder.

## Testing a Web Service

When the Web service has been published, a window opens enabling you to test the newly created Web service.

### **Procedure** How to Test the Web Service

To test the Web service:

1. In the input xml field, either enter a sample XML document that queries the service, for example,

```
<?xml version="1.0" encoding="UTF-8" ?>
<CICS>
  <Transaction location="/CICS/Transaction/IWAYSAMP">
    <FIELD1>P will pass down 40 bytes</FIELD1>
  </Transaction>
</CICS>

<?xml version="1.0" encoding="UTF-8" ?>
<CICS>
  <Transaction location="/CICS/Transaction/IWAYSAMP">
    <FIELD1>F will pass down 60 bytes</FIELD1>
  </Transaction>
</CICS>
```

or browse to the location of an XML instance and click *Upload*.

2. Click *Invoke*.

The result appears in the right pane.

## Accessing a Web Service

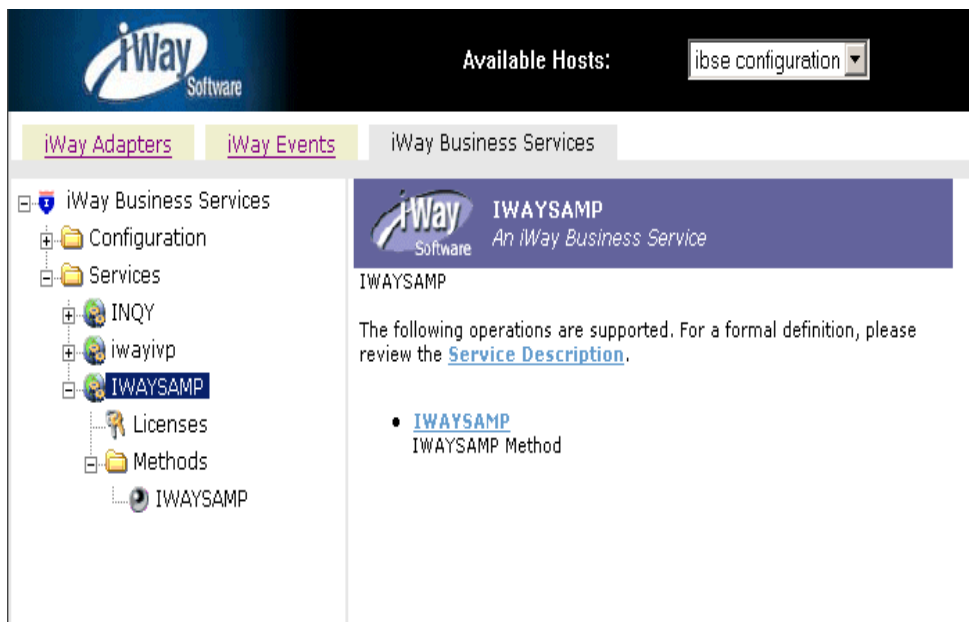
The Web service metadata is stored in a serialized object called webservicename.ibs. The WebLogic Workshop requires a WSDL file at design time.

### **Procedure** How to Create the WSDL Used to Access the Web Service

This procedure creates a WSDL file from the IWAYSAMP.ibs serialized object.

1. Click the *iWay Business Services* tab.

The following window opens.

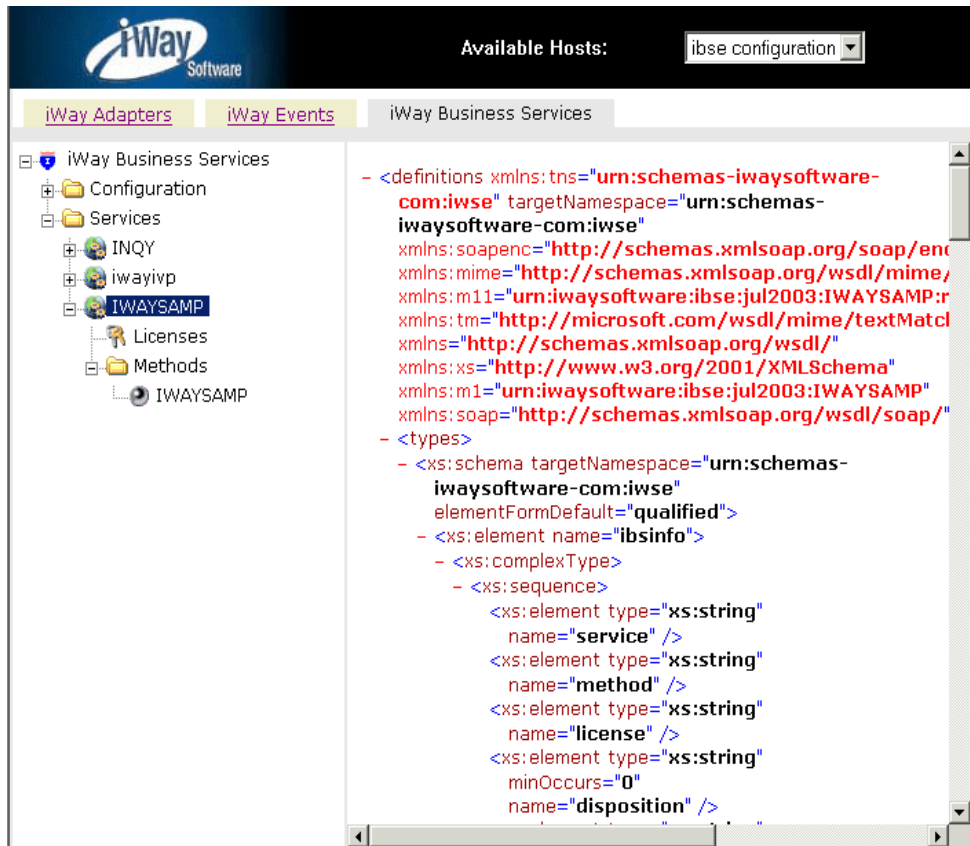


2. In the left pane, expand the newly created Web service, *IWAYSAMP*.
3. In the right pane, right-click *Service Description* and select *Save Target as*.

4. Choose a location to save the IWAYSAMP.wsdl file, for example, C:\IWASRV\IWAYSAMP.wsdl.

**Note:** The file extension must be wsdl.

The IWAYSAMP.wsdl file looks similar to the following:



---

---

## CHAPTER 4

# Using WebLogic Workshop

### Topics:

- Using WebLogic Workshop to Create and Access a Web Service
- Running the Process Definition From WebLogic Workshop

This section describes how to create and access a Web service using WebLogic Workshop.

## Using WebLogic Workshop to Create and Access a Web Service

WebLogic Workshop provides a framework for building Web services that are enterprise-class services. It provides simple controls for connecting to your enterprise resources.

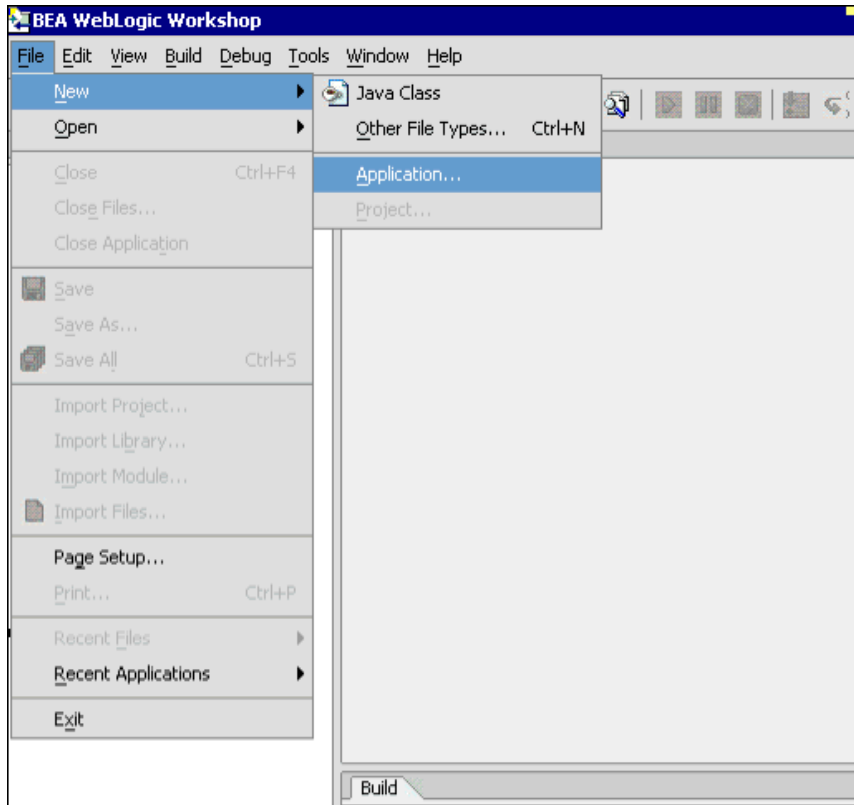
At the same time, WebLogic Workshop simplifies the process of creating Web services by insulating developers from the low-level implementation details that traditionally made Web service development the domain of sophisticated J2EE™ developers. With WebLogic Workshop, you can build powerful Web services whether you are an application developer or a J2EE expert.

### Procedure How to Create an Application

To create an application:

1. From the Start menu, choose *Programs, BEA WebLogic Platform 8.1*, and then *WebLogic Workshop 8.1*.

WebLogic Workshop opens.





2. To create a new application, select *New* from the File menu and then, *Application*.

The New Application dialog box opens.

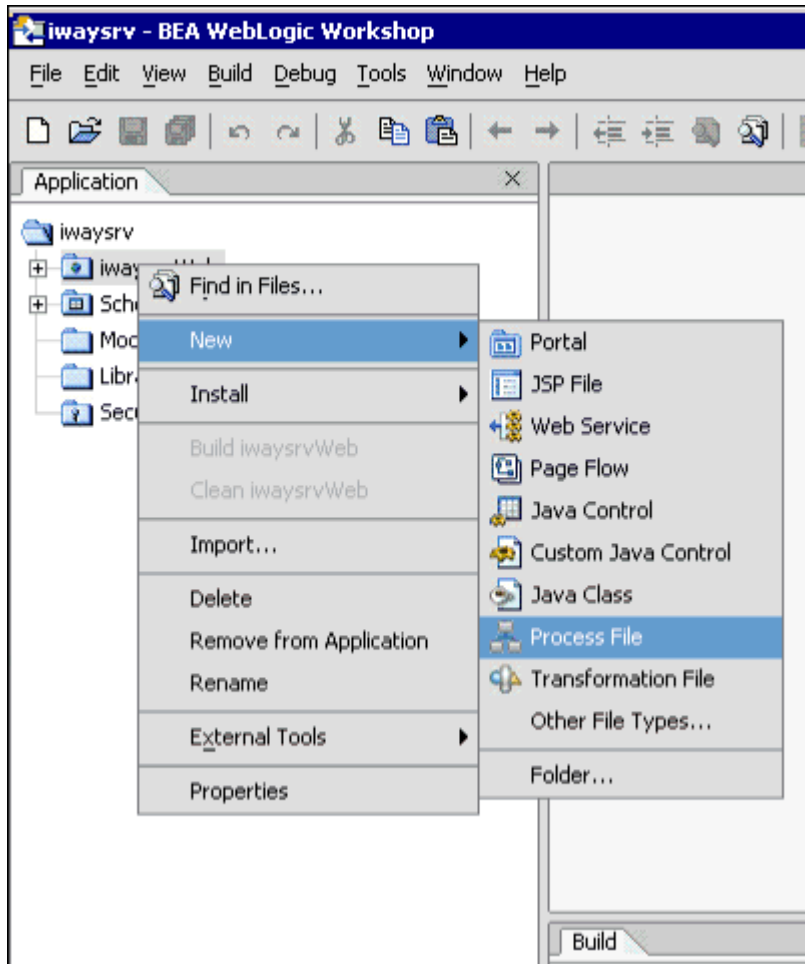
- a. In the upper-left pane, select *All*.
  - b. In the upper-right pane, select *Default Application*.
  - c. In the Directory field, type *C:\WAYSrv*.
3. Click *Create*.
- A new Workshop application with a Web project and schema project is created. You can add additional projects later.

## Creating a Schema for a Process Definition

The code for a process file resides within a Java™ Process Definition (JPD) file. A JPD file is considered to be a JAVA file in that it contains code for a Java class. However, because a file with a JPD extension contains the implementation code intended specifically for a process definition class, the extension gives it special meaning in the context of WebLogic Server.

## Procedure How to Create a Schema

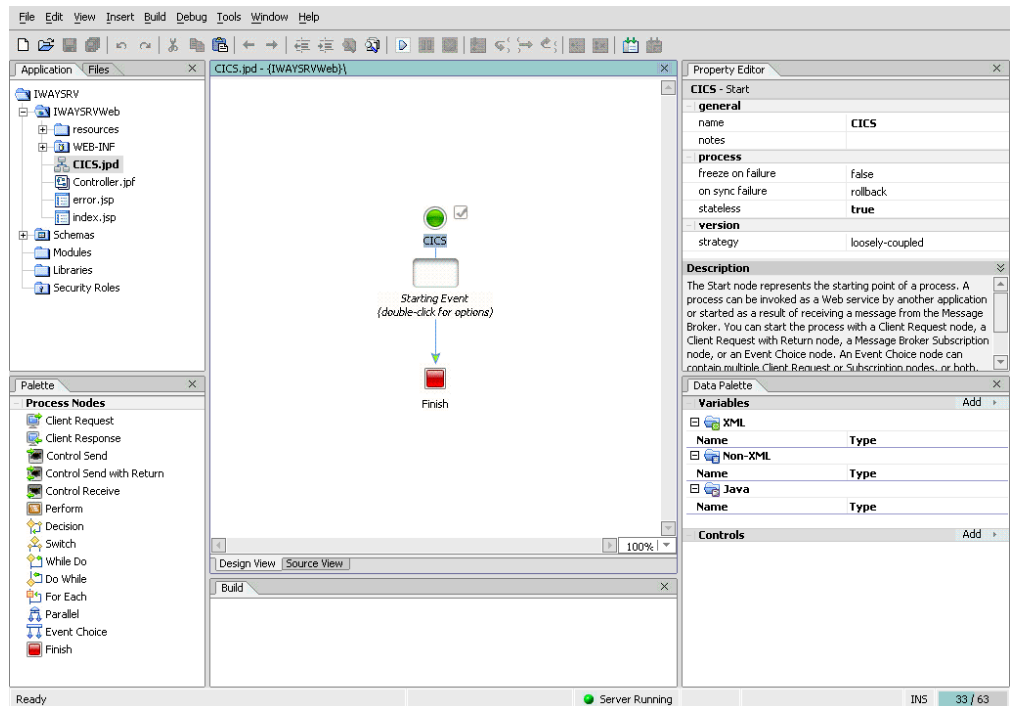
To create a schema to be used by the process definition:



1. In the Application tab, right-click the *iwaysrvWeb* folder.
  - a. Select *New*.
  - b. Then, select *Process File*.
2. In the File name field, type *CICS.jpd*.

3. Click *Create*.

A window similar to the following opens.

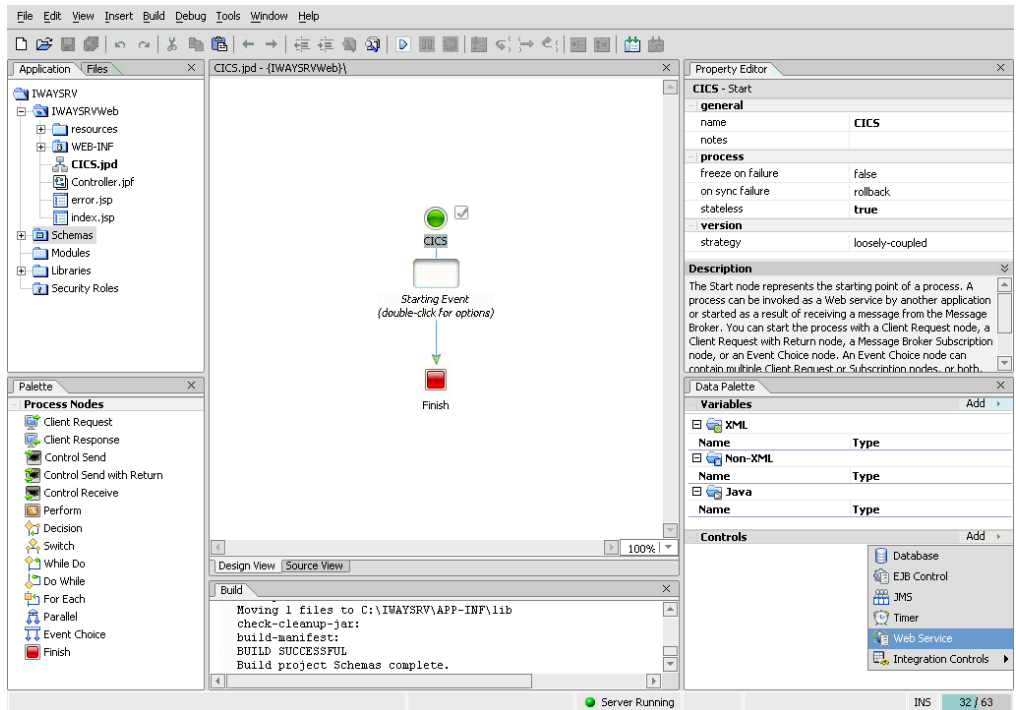


- a. If not already selected, click the *Application* tab.
- b. To import the *IWAYSAMP.wsdl* file previously saved, right-click the *Schemas* folder and select *Import*.

The Import Files window opens.

4. In the Import Files window, navigate to the location where you have saved the *IWAYSAMP.wsdl* file, for example, *C:\IWAYSRV*.
  - a. Highlight the *IWAYSAMP.wsdl* file.
  - b. Click *Import*.

WebLogic Workshop creates schemas to be used by the process definition as shown in the following illustration.



## Creating and Inserting a Control for a Web Service

To access a Web service, you must first create and insert a control. A Web service control makes is easy to access an external Web service from a WebLogic Workshop application.

### Procedure How to Create a Control for a Web Service

To create and insert a control for a Web service:

1. In the Controls pane, click *Add*.

The Control menu opens.

## 2. Select *Web Service*.

The Insert Control – Web Service window opens.

**Insert Control - Web Service**

**STEP 1** Variable name for this control:

---

**STEP 2** I would like to :

☐ Use a Web Service control already defined by a JCX file

JCX file:

☒ Create a new Web Service control to use.

New JCX name:

☐ Make this a control factory that can create multiple instances at runtime

---

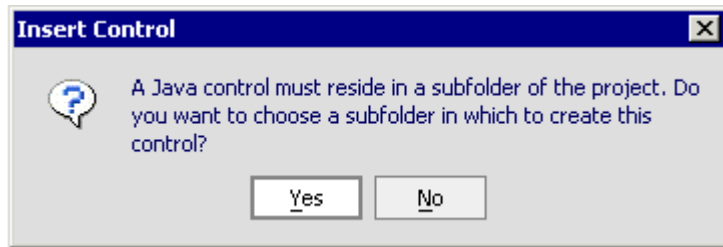
**STEP 3** WSDL definition from which new Service Control will be created:

File or URL:

- a. Type a name for the control in the *Variable name* field, for example, message.
- b. Select the *Create a new Web Service control to use* option button.
- c. Type a name for the JCX, for example, IWAYSAMP.
- d. Enter (or browse to) the location of the saved wsdl, for example, C:\IWAYSrv\IWAYSAMP.wsdl.

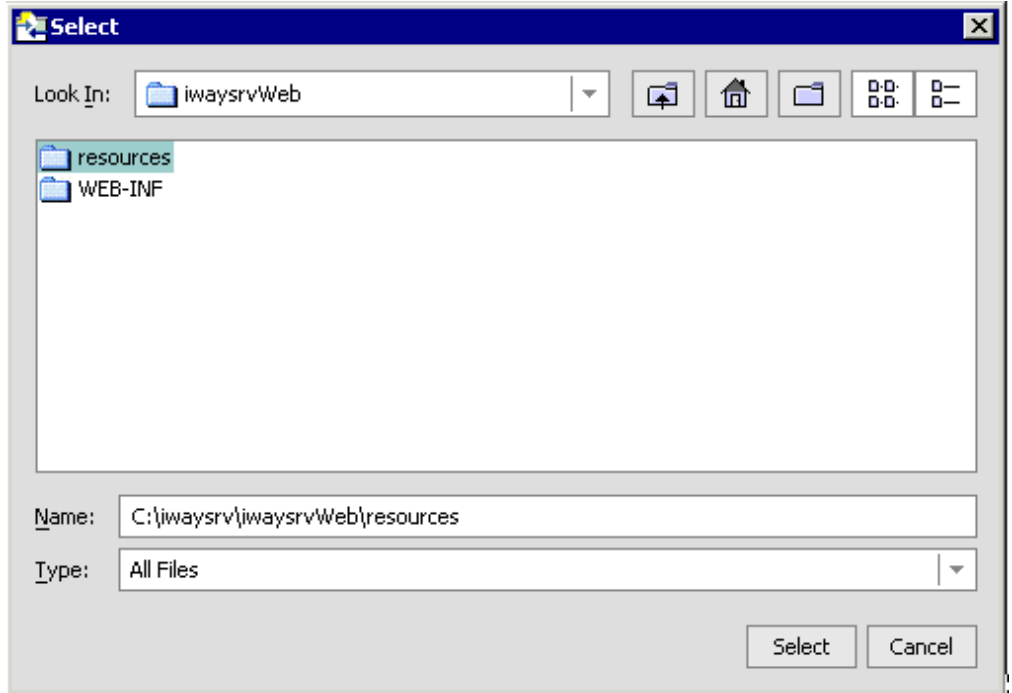
3. Click *Create*.

An Insert Control message appears.



4. To choose a subfolder for the control, click *Yes*.

The system prompts you to choose a subfolder for the Web service.



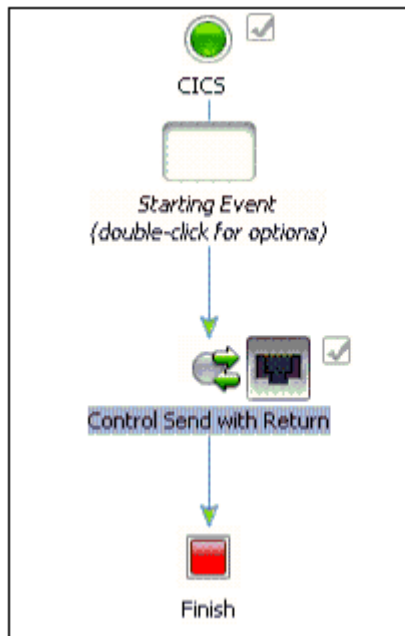
5. Highlight the *resources* subfolder and click *Select*.

The Web service is created in the resources subfolder.

**Procedure How to Insert a Control**

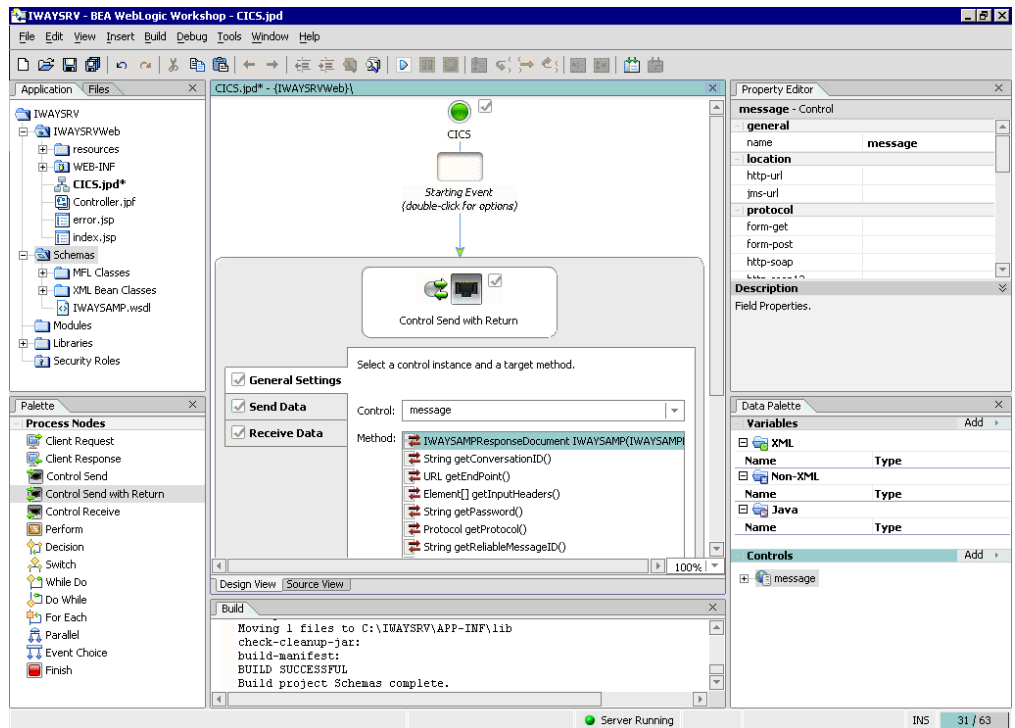
To insert a control:

1. Select the Palette pane.
2. Drag and drop the *Control Send with Return* node onto the process as follows:



3. Double-click the *Control Send with Return* node.

The following window opens.



a. From the list of available controls in the middle pane, select *message*.

b. For the method, select *IWAYSAMPResponseDocument*.

4. Click *Apply* and then, *Close*.

**Note:** This control was created in the previous procedure.

## Creating an Input Variable and an Output Variable

You must create an input and output variable for your business process.

### Procedure How to Create a Variable

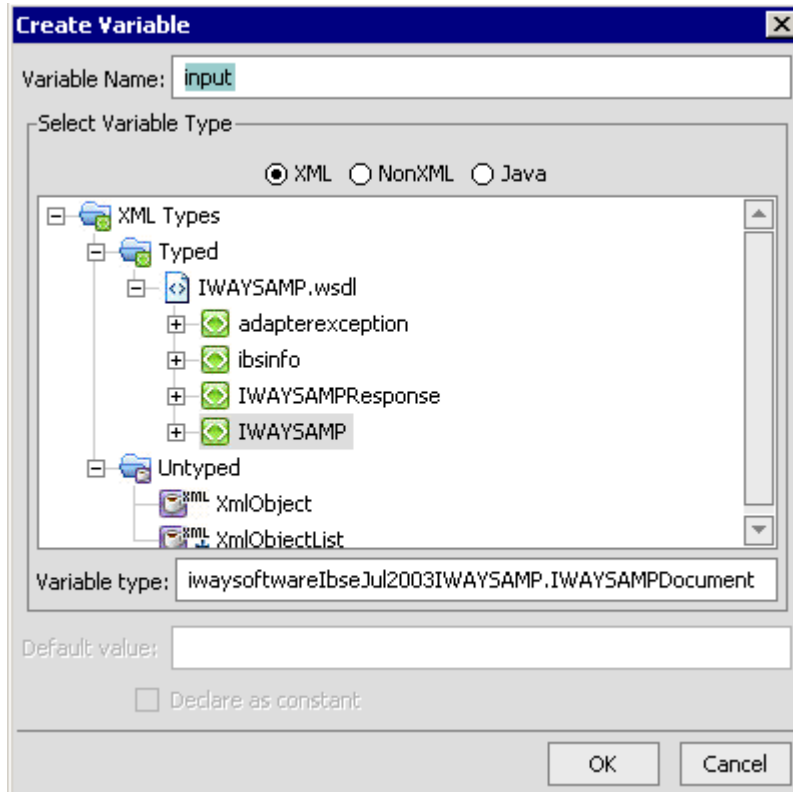
To create a variable:

1. Double-click the *Control Send with Return* node.
2. Select *Send Data*.



3. In the *Select variables to assign* area, select *Create new variable*.

The Create Variable dialog box opens.



- a. For the Send data, type a name, *input*, in the *Variable Name* field.
- b. Select variable type, *xml*, and click *OK*.

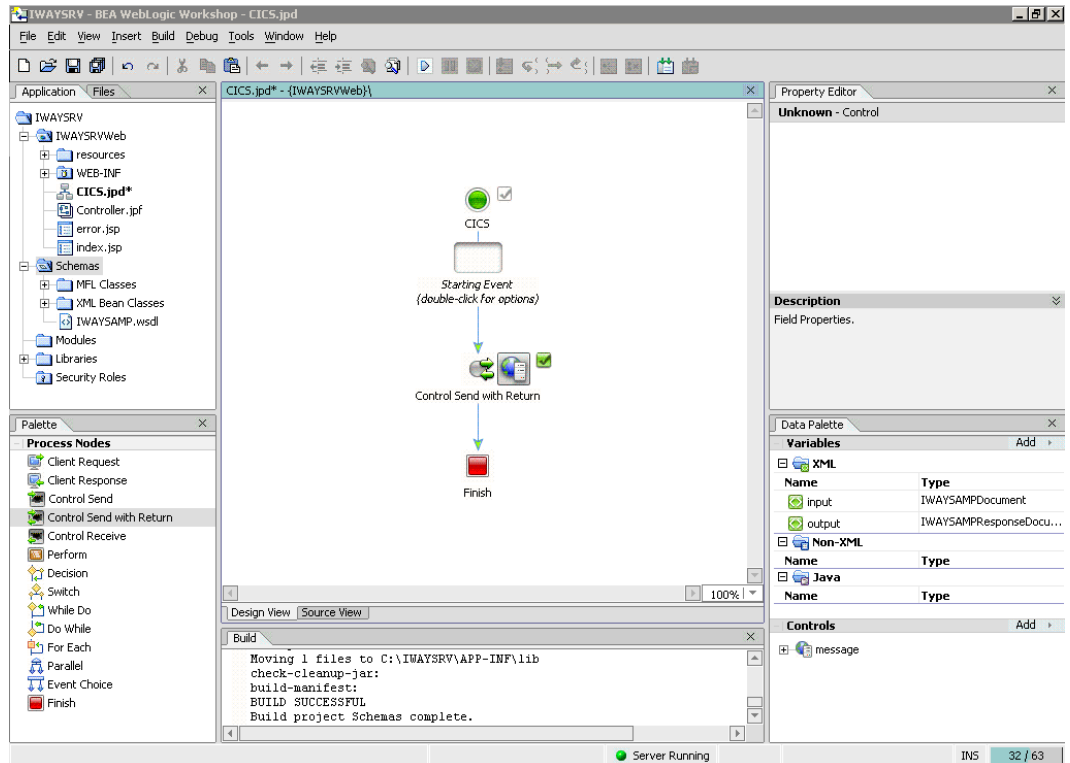
The dialog box closes.

4. In the middle pane, select *Receive Data*.
5. From the *Select variables to assign* area, select *Create new variable*.

The Create Variable dialog box opens.

6. For the Receive data, create a new output variable.
  - a. Type a name, *output*, in the *Variable Name* field.
  - b. Select variable type, *xml*, and click *OK*.
  - c. Click *Apply*, and then, click *Close*.

If you successfully configured the input and output variables, green arrows appear next to the *Control Receive* node as shown in the following illustration.



## Configuring Parameters to Start an Event

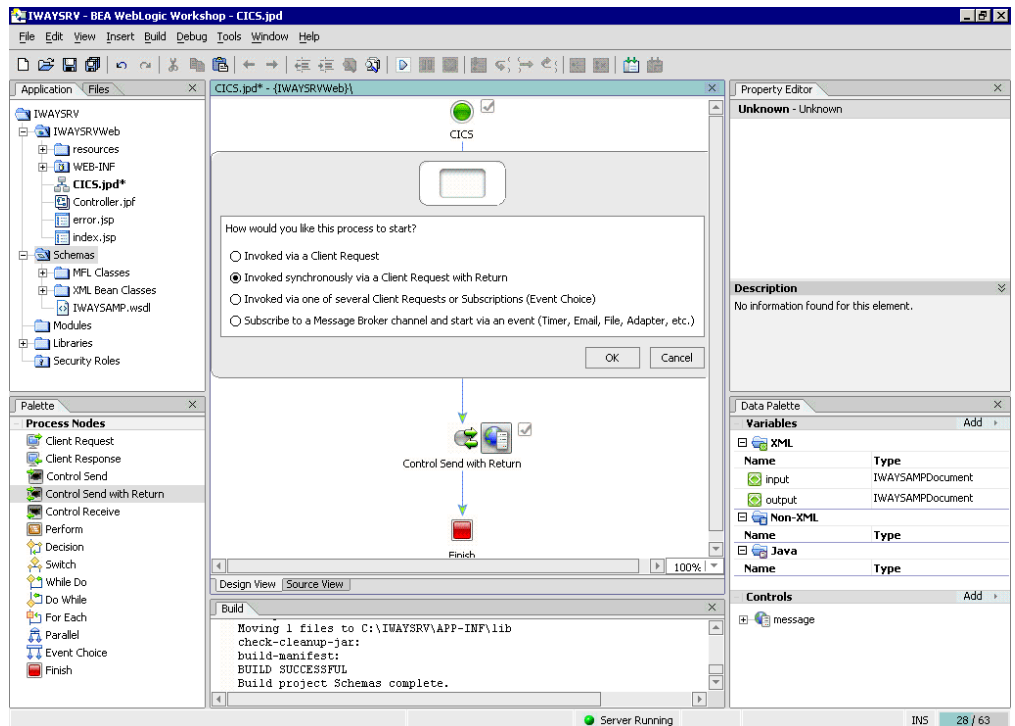
The first action in the business process is specified at the Start node. You specify how the business process is started at runtime by defining a Starting Event.

### Procedure How to Configure Parameters to Start an Event

To configure parameters to start an event:

1. In the middle pane, double-click the *Starting Event* node.

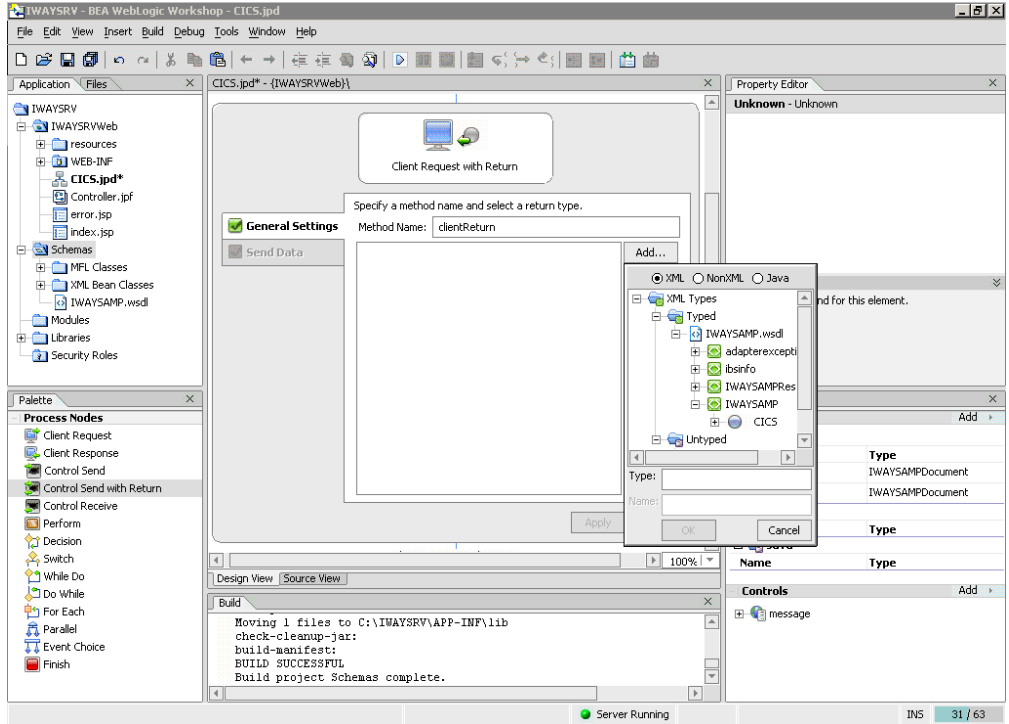
A pane opens and asks how you would like the process to start.



2. Select the *Invoked synchronously via a Client Request with Return* option button and click **OK**.
3. Double-click the *Client Request with Return* node.

4. Click **Add**.

A pane opens where you can specify a method name and select a return type. An additional pane opens where you can view the hierarchy.



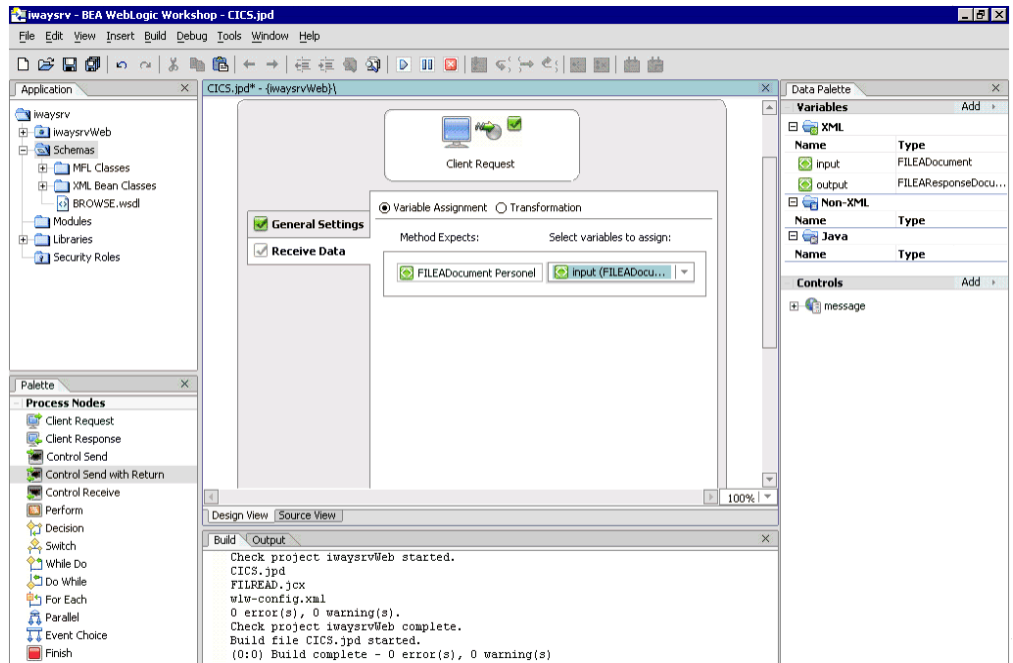
- a. Expand the *IWAYSAMP.wsdl* tree.
- b. Select *IWAYSAMP* from the tree.

You can provide or modify the name of the variable to use.

The default value is x0.

5. Click **OK**.

You can now map the input variable to the output variable.



6. In the middle pane, select *Receive Data*.

7. From the *Select variables to assign* drop-down list, select the input variable.

8. To activate these changes, click *Apply* and then, click *Close*.

## Running the Process Definition From WebLogic Workshop

After you create a new process definition, you must ensure that WebLogic Server is running while you build your Web service. You can see whether or not WebLogic Server is running by viewing the status bar in WebLogic Workshop.

- If WebLogic Server is running, a green ball appears with the message, *Server Running*.
- If WebLogic Server is not running, a red ball appears with the message, *Server Stopped*.

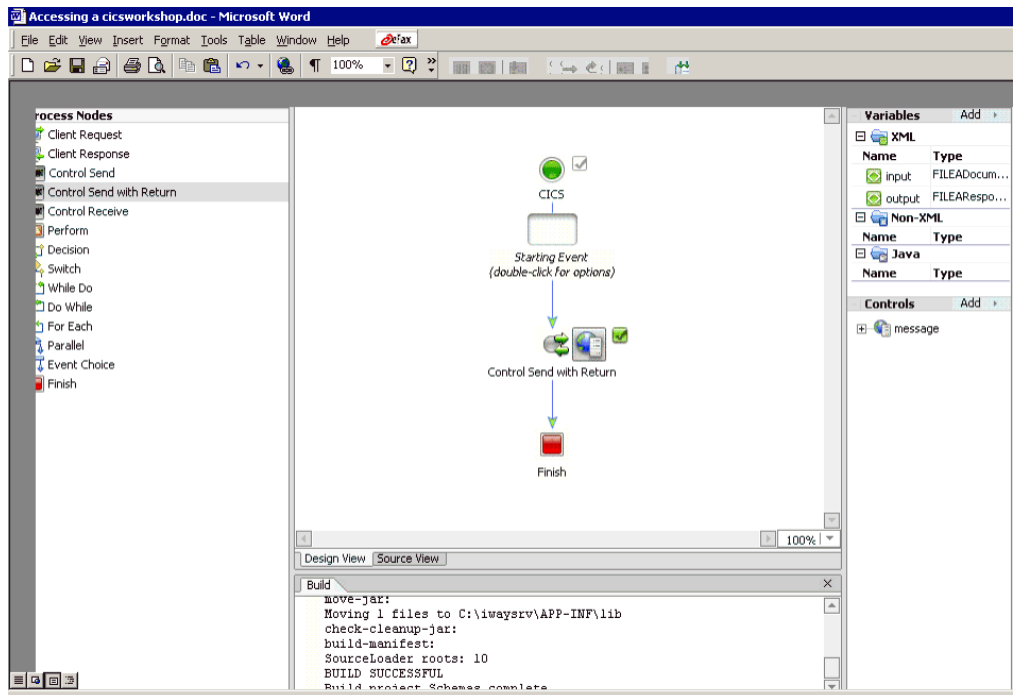
If you see the red ball in the status bar, follow the instructions in the following procedure to start WebLogic Server.

## Procedure How to Start WebLogic Server and Deploy the Application to WebLogic

To start WebLogic Server:

1. From the Tools menu, choose *WebLogic Server* and then, *Start WebLogic Server*.
2. To deploy the application to WebLogic, select *Tools* and then, *Deploy Application*.
3. To start the application, click the *Start* button in the toolbar.

The following test window opens.



4. Click the *Test SOAP* tab to test the XML stream to pass to the Web service.
5. Replace the following:

`location="string"`

with

`location="/CICS/Transaction/IWAYSAMP"`

**6. Modify the following:**

```
<urn:NUMB>anyType</urn:NUMB>
```

to

```
<urn:NUMB>000100</urn:NUMB>
```

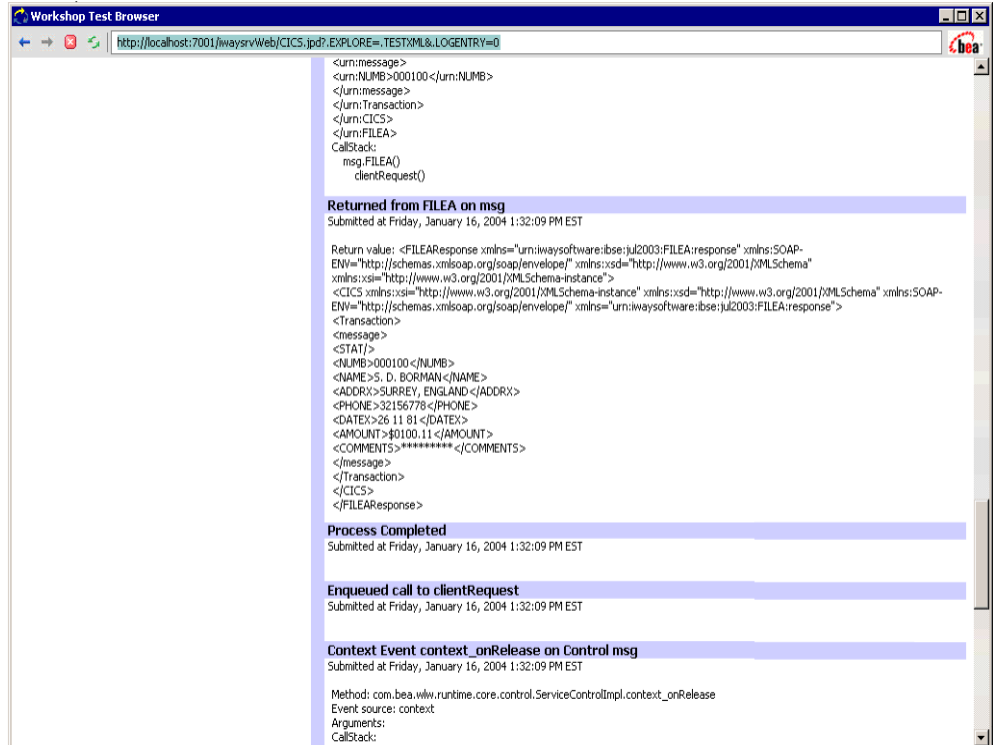
**7. To submit the request, click *clientResponse*.**

Your input should look similar to the following:

```
<clientRequest xmlns="http://www.openuri.org/"
xmlns:urn="urn:iwaysoftware:ibse:jul2003:IWAYSAMP">
  <urn:IWAYSAMP>
    <urn:CICS>
      <urn:Transaction location="/CICS/Transaction/TEST">
        <urn:message>
          <urn:NUMB>000100</urn:NUMB>
        </urn:message>
      </urn:Transaction>
    </urn:CICS>
  </urn:IWAYSAMP>
</clientRequest>
```

8. To view the output, scroll to the middle of the screen.

The output should look similar to the following.



The output appears under the *Returned from IWAYSAMP under msg* heading.

**Note:** It may take a few seconds to display the output. You can click the refresh button while waiting for the response.



---

---

## APPENDIX A

# Configuring VTAM and CICS

### Topics:

- VTAM and AnyNet Configuration
- CICS Configuration

Although the iWay Adapter for CICS requires no code to be installed on the mainframe, VTAM and CICS must be configured correctly. This appendix provides the configuration and communications information the adapter requires to communicate with CICS.

## VTAM and AnyNet Configuration

---

For its operation, AnyNet requires the following VTAM nodes:

- Major Node for the AnyNet Listener
- Standard VTAM LU Definitions for APPC
- Major Node for Cross Domain Resources

**Note:** For specific information for your site, see your CICS Systems Administrator and VTAM Administrator.

In the following examples, the parameters in boldface relate to the configuration entries in the logon dialog box described in Chapter 2, *Configuring the iWay Adapter for CICS*.

The following illustrates the CICS/AnyNet Connection dialog box where you configure the attach header for the connection to CICS.

**CICS/AnyNet Connection**

Host\*

Port AnyNet is listening on

User ID\*

Password\*

VTAM Id (VTAM network qualifier\*)

Partner LU (CICS LU) \*

Local LU\*

logMode\*

☐ Trusted

Remote Codepage

Fields marked with \* are required.

**Example Major Node for the AnyNet Listener**

```

*****      TCP62 MAJOR NODE DEFINITION FOR ANYNET
TCP62      VBUILD TYPE=TCP,DNSUFFIX=IBI.COM,PORT=397
TCP62G     GROUP  ISTATUS=ACTIVE
TCP62L     LINE   ISTATUS=ACTIVE
TCP62P     PU     ISTATUS=ACTIVE,NETID=vtam_id

```

**Example Standard VTAM LU Definitions for APPC**

The LU names that are defined in the following node also must be defined in the Cross Domain Resources node.

```

          VBUILD TYPE=SWNET
T29DPBPU  PU      ADDR=01,IDBLK=05D,IDNUM=10101,          X
          MAXPATH=3,                                     X
          PUTYPE=2,                                       X
          MODETAB=MTOS2EE,                               X
          DLOGMOD=logmode,                               X
          ISTATUS=ACTIVE
local_lu  LU      LOCADDR=2
T29DPB02  LU      LOCADDR=3
T29DPB03  LU      LOCADDR=4
T29DPB04  LU      LOCADDR=5

```

**Example Major Node for Cross Domain Resources**

The LU names defined in the major node for the AnyNet listener also must be defined in the following node:

```

          VBUILD TYPE=CDRSC
CDRSC62   NETWORK NETID=vtam_id
CDRSC62   GROUP
T29DPB01  CDRSC   ALSLIST=TCP62P
T29DPB02  CDRSC   ALSLIST=TCP62P
T29DPB03  CDRSC   ALSLIST=TCP62P
T29DPB04  CDRSC   ALSLIST=TCP62P

```

After these definitions are completed, the major nodes they describe must be started in the VTAM system.

# CICS Configuration

---

For the adapter to connect to CICS, connection and session definitions are required. The definitions depend upon the platform on which the adapter runs. For specific information for your site, see your VTAM Administrator.

**Note:** All users must have access to run the transaction CPMI (the mirror transaction).

## Example Connection Definition

Connection	:	PB01	
Netname	:	local_lu	
ACcessmethod	:	Vtam	Vtam   IRc   INdirect   Xm
PRotocol	:	Appc	Appc   Lu61   Exci
SInglesess	:	No	No   Yes
DAstream	:	User	User   3270   SCs   STRfield   Lms
INService	:	Yes	Yes   No
ATTachsec	:	Verify	Local   Identify   Verify   Persistent

## Example Sessions Definition

Sessions	:	local_lu	
Connection	:	PB01	
MOdename	:	logmode	
Protocol	:	Appc	Appc   Lu61   Exci
MAximum	:	008 , 000	0-999

The Maximum parameter determines the number of concurrent sessions available to process requests. Its first value should be in the range of 2 - 255, and the second value must always be zero.

---

---

## APPENDIX B

# Installing the Sample IWAYIVP and IWAYSAMP Programs in CICS

### Topics:

- Installing and Configuring the Sample CICS Program IWAYIVP in CICS
- Installing and Configuring the Sample CICS Program IWAYSAMP in CICS

This section describes how to verify that you have correctly installed the adapter.

## **Installing and Configuring the Sample CICS Program IWAYIVP in CICS**

---

The following procedure demonstrates how to install and configure the sample CICS program, IWAYIVP. Sample source code for the Cobol program IWAYIVP is provided in Appendix D, *Sample CICS Programs*. For specific information for your site, see your CICS Systems Administrator.

### ***Procedure* How to Install and Configure the Sample CICS Program IWAYIVP**

To install and configure the sample CICS program, IWAYIVP:

- 1.** Use the source code provided in Appendix D, *Sample CICS Programs*, compile the program, and make it available to CICS.
- 2.** Define the Cobol program to the CICS region.
  - a.** Log onto the CICS region.
  - b.** Issue the following command:

```
CEDA DEF PROG(IWAYIVP) GROUP(IWAY)
```

The following screen appears after you issue the previous command.

The screenshot shows a CICS terminal window titled "Session A - [24 x 80]". The window contains the following text:

```

DEF PROG(IWAYIVP)  GROUP(IWAY)
OVERTYPE TO MODIFY                                CICS RELEASE = 0530
CEDA  DEFine PROGram( IWAYIVP )
  PROGRAM          : IWAYIVP
  Group            : IWAY
  Description      ==> -
  Language         ==> -                CObol | Assembler | Le370 | C | PlI
  RELoad           ==> No                No | Yes
  RESident         ==> No                No | Yes
  USAge            ==> Normal            Normal | Transient
  USElpacopy       ==> No                No | Yes
  Status           ==> Enabled            Enabled | Disabled
  RSl              : 00                  0-24 | Public
  CEdf             ==> Yes                Yes | No
  DATalocation     ==> Below              Below | Any
  EXECKey          ==> User                User | Cics
  COncurrency      ==> Quasirent          Quasirent | Threadsafe
  REMOTE ATTRIBUTES
    DYnamic         ==> No                No | Yes
+  REMOTESystem    ==>
I New group IWAY created.

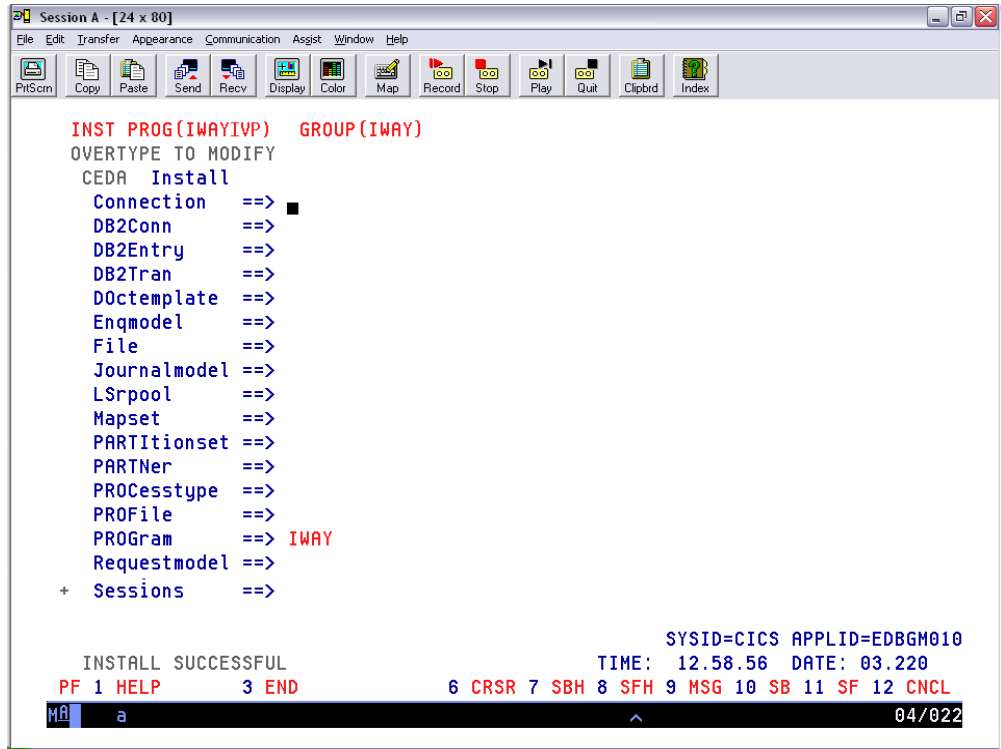
                                SYSID=CICS  APPLID=EDBGH010
DEFINE SUCCESSFUL                TIME: 12.58.17  DATE: 03.220
PF 1 HELP 2 COM 3 END            6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
MA a                                06/022

```

3. Install the program in the CICS region.
  - a. Issue the following command:

```
CEDA INST PROG(IWAYIVP) GROUP(IWAY)
```

The system displays a screen similar to the following:



The screenshot shows a CICS terminal window titled "Session A - [24 x 80]". The menu bar includes File, Edit, Transfer, Appearance, Communication, Assist, Window, and Help. The toolbar contains icons for PrintScreen, Copy, Paste, Send, Recv, Display, Color, Map, Record, Stop, Play, Quit, Clipboard, and Index. The main display area shows the following text:

```
INST PROG(IWAYIVP)  GROUP(IWAY)
OVERTYPE TO MODIFY
CEDA  Install
Connection ==>
DB2Conn ==>
DB2Entry ==>
DB2Tran ==>
DOctemplate ==>
Enqmodel ==>
File ==>
Journalmodel ==>
LSrpool ==>
Mapset ==>
PARTitionset ==>
PARTner ==>
PROcesstype ==>
PROfile ==>
PROGRAM ==> IWAY
Requestmodel ==>
+ Sessions ==>
```

At the bottom of the screen, the following status information is displayed:

```
INSTALL SUCCESSFUL                                SYSID=CICS APPLID=EDBGM010
TIME: 12.58.56 DATE: 03.220
PF 1 HELP      3 END      6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

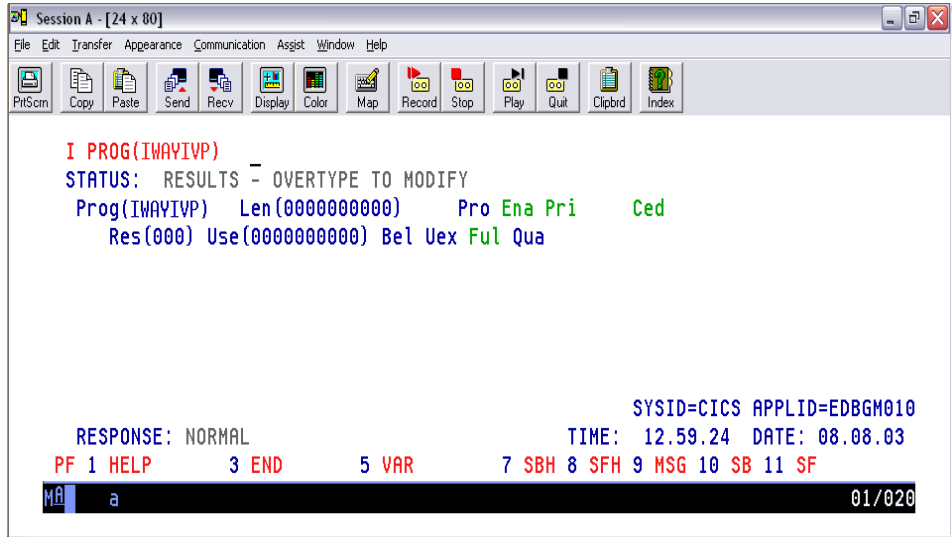
The bottom status bar shows "Mâ a" on the left and "04/022" on the right.

- b. Display the program by entering the following command:

```
CEMT I PROG(IWAYIVP)
```



The result is a screen similar to the following:



## Installing and Configuring the Sample CICS Program IWAYSAMP in CICS

The following procedure demonstrates how to install and configure the sample CICS program, IWAYSAMP. Sample source code for the Cobol program IWAYSAMP is provided in Appendix D, *Sample CICS Programs*. For specific information for your site, see your CICS Systems Administrator.

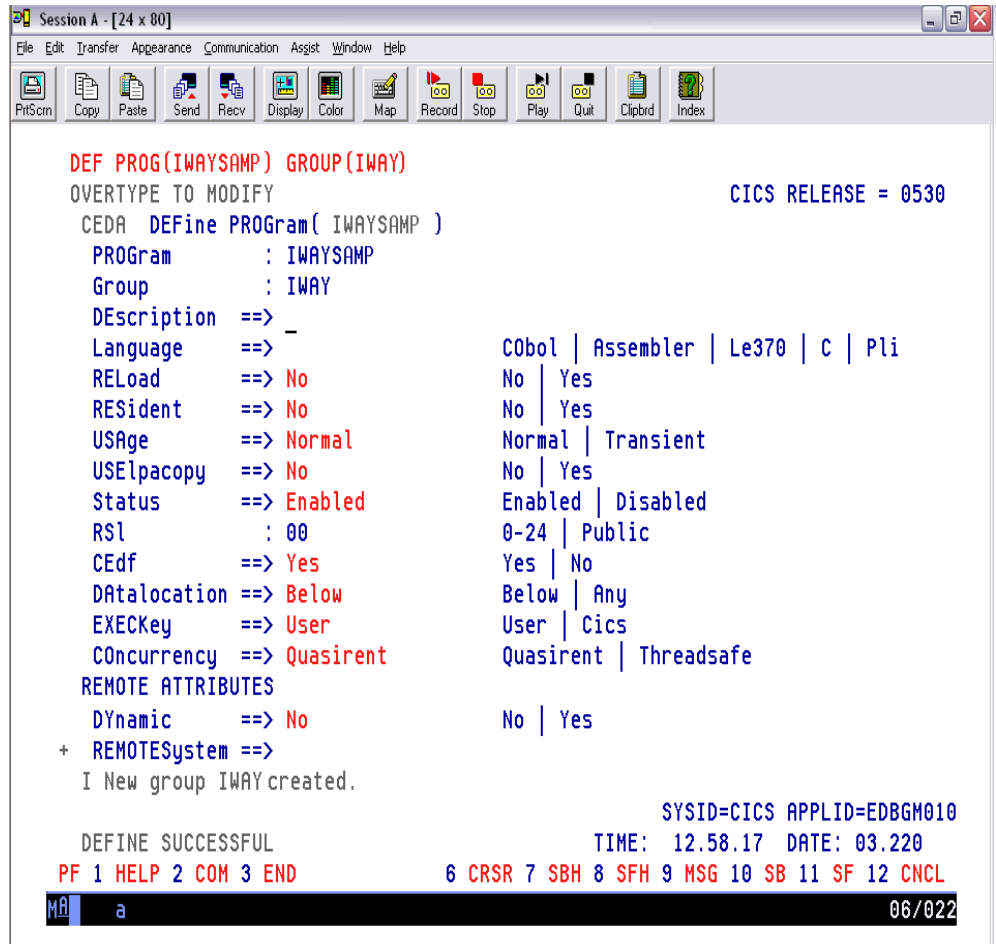
### **Procedure** How to Install and Configure the Sample CICS Program IWAYSAMP

To install and configure the sample CICS program, IWAYSAMP:

1. Use the source code provided in Appendix D, *Sample CICS Programs*, compile the program, and make it available to CICS.
2. Define the Cobol program to the CICS region.
  - a. Log onto the CICS region.
  - b. Issue the following command:

```
CEDA DEF PROG(IWAYSAMP) GROUP(IWAY)
```

The following screen appears after you issue the previous command.



The screenshot shows a CICS terminal window titled "Session A - [24 x 80]". The window contains the following text:

```
DEF PROG(IWAYSAMP) GROUP(IWAY)
OVERTYPE TO MODIFY                                CICS RELEASE = 0530
CEDA DEFINE PROGRAM( IWAYSAMP )
  PROGRAM      : IWAYSAMP
  GROUP        : IWAY
  DESCRIPTION  ==>
  LANGUAGE     ==> -                               COBOL | Assembler | LE370 | C | PLI
  RELOAD       ==> No                               No | Yes
  RESIDENT     ==> No                               No | Yes
  USAGE        ==> Normal                           Normal | Transient
  USEELPACOPY  ==> No                               No | Yes
  STATUS       ==> Enabled                           Enabled | Disabled
  RSL          : 00                                0-24 | Public
  CEDF         ==> Yes                               Yes | No
  DATALOCATION ==> Below                             Below | Any
  EXECKEY      ==> User                               User | CICS
  CONCURRENCY  ==> Quasirent                         Quasirent | Threadsafe
  REMOTE ATTRIBUTES
  DYNAMIC      ==> No                               No | Yes
+ REMOTESYSTEM ==>
I New group IWAY created.

DEFINE SUCCESSFUL                                SYSID=CICS APPLID=EDBGM010
                                           TIME: 12.58.17 DATE: 03.220
PF 1 HELP 2 COM 3 END                        6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
MA a                                           06/022
```

3. Install the program in the CICS region.

a. Issue the following command:

```
CEDA INST PROG(IWAYSAMP) GROUP(IWAY)
```

The system displays a screen similar to the following:

```

Session A - [24 x 80]
File Edit Transfer Appearance Communication Assist Window Help

PrintScreen Copy Paste Send Recv Display Color Map Record Stop Play Quit Clipboard Index

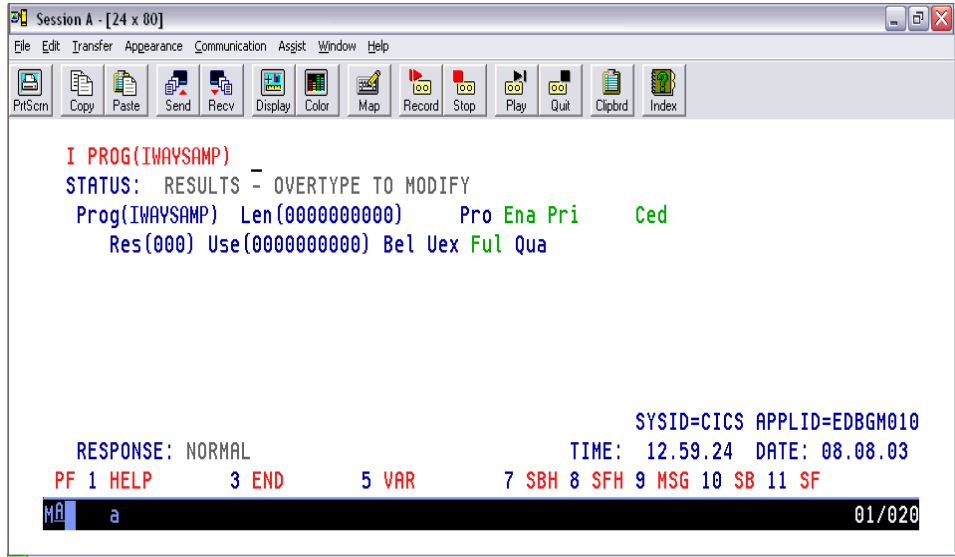
INST PROG(IWAYSAMP) GROUP(IWAY)
OVERTYPE TO MODIFY
CEDA Install
Connection ==>
DB2Conn ==>
DB2Entry ==>
DB2Tran ==>
DOctemplate ==>
Enqmodel ==>
File ==>
Journalmodel ==>
LSrpool ==>
Mapset ==>
PARTitionset ==>
PARTner ==>
PROCesstype ==>
PROFile ==>
Program ==> IWAY
Requestmodel ==>
+ Sessions ==>

SYSID=CICS APPLID=EDBGH010
TIME: 12.58.56 DATE: 03.220
PF 1 HELP 3 END 6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
MA a 04/022
    
```

b. Display the program by entering the following command:

```
CEMT I PROG(IWAYSAMP)
```

The result is a screen similar to the following:



The screenshot shows a CICS terminal window titled "Session A - [24 x 80]". The window has a menu bar with "File", "Edit", "Transfer", "Appearance", "Communication", "Assist", "Window", and "Help". Below the menu bar is a toolbar with icons for "PrintScreen", "Copy", "Paste", "Send", "Recv", "Display", "Color", "Map", "Record", "Stop", "Play", "Quit", "Clipboard", and "Index". The main display area contains the following text:

```
I PROG(IWAYSAMP)
STATUS: RESULTS - OVERTYPE TO MODIFY
  Prog(IWAYSAMP) Len(0000000000)    Pro Ena Pri    Ced
    Res(000) Use(0000000000) Bel Uex Ful Qua

                                SYSID=CICS APPLID=EDBGM010
RESPONSE: NORMAL                                TIME: 12.59.24 DATE: 08.08.03
PF 1 HELP      3 END      5 VAR      7 SBH 8 SFH 9 MSG 10 SB 11 SF

MA a 01/020
```

---

---

## APPENDIX C

### Sample Requests, Schemas, and Cobol File Descriptions

#### Topics:

- Request Document for the Generic Transaction IWAYIVP
- Request Schema for the Generic Transaction IWAYIVP
- Response Schema for the Generic Transaction IWAYIVP
- Request Document for the Program IWAYSAMP
- Request Schema for the Program IWAYSAMP
- Response Schema for the Program IWAYSAMP
- Sample Cobol File Descriptions

After you create a connection to CICS, you can add CICS transactions using Application Explorer. The generic transaction always is added automatically and represents CICS services whose data will not be mapped to XML.

The documents and schemas for the sample programs are shown in the following topics. Also, the Cobol descriptions that were used as input for the sample CICS transactions are shown.

## Request Document for the Generic Transaction IWAYIVP

---

The following is a sample XML request document to run the IWAYIVP program:

```
<?xml version="1.0" encoding="UTF-8" ?>
<CICS>
<Transaction tpname="IWAYIVP" noreply="NO">
<message></message>
</Transaction>
</CICS>
```

where:

*tpname*

Specifies the program to be called in CICS. This is provided at run time.

## Request Schema for the Generic Transaction IWAYIVP

---

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified"
xml:lang="en">
  <xs:element name="CICS">
    <xs:complexType>
      <xs:all>
        <xs:element name="Transaction">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="message">
                <xs:simpleType>
                  <xs:restriction base="xs:string">
                    <xs:length value="80"/>
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
            </xs:sequence>
            <xs:attribute name="tpname" type="xs:string"
              use="required" fixed="Generic_Transaction"/>
            <xs:attribute name="casize" type="xs:int"
              use="optional"/>
          </xs:complexType>
        </xs:element>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Although the message element is restricted to 80 bytes in the schema, this is not enforced at run time. The message can be up to 32,500 bytes long and is sent as the COMMAREA.

## Response Schema for the Generic Transaction IWAYIVP

---

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified"
xml:lang="en">
  <xs:element name="CICS">
    <xs:complexType>
      <xs:all>
        <xs:element name="Transaction">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="message">
                <xs:simpleType>
                  <xs:restriction base="xs:string">
                    <xs:length value="80"/>
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

The returned data is split into 80 byte parts, each encoded in <message> tags. If there are illegal XML characters in the data returned from CICS (<', '/', or '&'), they are turned into XML entities as part of the encoding.

## Request Document for the Program IWAYSAMP

---

The following is a sample XML request document to run the IWAYSAMP program:

```
<?xml version="1.0" encoding="UTF-8" ?>
<CICS>
  <Transaction location="/CICS/Transaction/IWAYSAMP">
    <FIELD1>P will pass down 40 bytes</FIELD1>
  </Transaction>
</CICS>

<?xml version="1.0" encoding="UTF-8" ?>
<CICS>
  <Transaction location="/CICS/Transaction/IWAYSAMP">
    <FIELD1>F will pass down 60 bytes</FIELD1>
  </Transaction>
</CICS>
```

## Request Schema for the Program IWAYSAMP

---

```
<xs:schema xml:lang="en" attributeFormDefault="unqualified"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  "elementFormDefault="qualified"
  <xs:element name="CICS">
    <xs:complexType>
      <xs:all>
        <xs:element name="Transaction">
          <xs:complexType>
            <xs:all>
              <xs:element name="message">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element minOccurs="1" name="DATA" />
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:all>
            <xs:attribute type="xs:string" use="required"
              fixed="CICS/Transactions/IWAYSAMP" name="location" />
          </xs:complexType>
        </xs:element>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```



## Response Schema for the Program IWAYSAMP

---

```

<xs:schema xml:lang="en" attributeFormDefault="unqualified"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <xs:element name="CICS">
    <xs:complexType>
      <xs:all>
        <xs:element name="Transaction">
          <xs:complexType>
            <xs:choice>
              <xs:element name="message1">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element minOccurs="1" name="STAT" maxOccurs="1" />
                    <xs:element minOccurs="1" name="FILLER" maxOccurs="1" />
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="message2">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element minOccurs="1" name="STAT" maxOccurs="1" />
                    <xs:element minOccurs="1" name="FILLER" maxOccurs="1" />
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:choice>
          </xs:complexType>
        </xs:element>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>

```

The returned data is split into 80 byte parts, each encoded in <message> tags. If there are illegal XML characters in the data returned from CICS ('<', '/', or '&'), they are turned into XML entities as part of the encoding.

## Sample Cobol File Descriptions

---

The following sample Cobol File Descriptions are used as input for the CICS transactions in Chapter 3, *Designing the iWay Adapter for CICS*.

### IWAYSAMP\_in.cbl

```
01 DFHCOMMAREA.  
   05 DATA          PIC X(100) .
```

### IWAYSAMP\_out\_P.cbl

```
01 DFHCOMMAREA.  
   05 STAT           PIC X(1) .  
   05 FILLER         PIC X(39)
```

### IWAYSAMP\_out\_F.cbl

```
01 DFHCOMMAREA.  
   05 STAT           PIC X(1) .  
   05 FILLER         PIC X(59) .
```

---

---

## APPENDIX D

### Sample CICS Programs

**Topics:**

- IWAYIVP Program
- IWAYSAMP Program
- CICS TCP/IP Sockets Program for Mainframe
- CICS TCP/IP Sockets Program for TX

This appendix includes the source code for the sample CICS programs.

## IWAYIVP Program

---

The following sample program called IWAYIVP tests the adapter installation. This program takes no input and returns a fixed string as output. Therefore, the generic transaction is suitable.

```
*-----+-----*
* IWAYIVP      |
*-----*
* IWAY SOFTWARE INSTALLATION VERIFICATION PROGRAM (IVP)
*
* THIS PROGRAM WILL RETURN CONGRATULATIONS SIGNIFYING
* SUCCESSFUL CONNECTION AND EXECUTION.
*
*-----*
* TAILORING:                                         *
*-----+-----+-----+-----*
IDENTIFICATION DIVISION.
PROGRAM-ID. IWAYIVP.
ENVIRONMENT DIVISION.
DATA DIVISION.

WORKING-STORAGE SECTION.

01 WS-AREA          PIC X(50) VALUE
   'CONGRATULATIONS!!!'.

LINKAGE SECTION.
01 DFHCOMMAREA      PIC X(50) .

PROCEDURE DIVISION.
    EXEC CICS HANDLE ABEND
        LABEL (030-ABEND)
    END-EXEC .

    MOVE WS-AREA TO DFHCOMMAREA.

    EXEC CICS RETURN
    END-EXEC .
    GOBACK.
030-ABEND.

    MOVE 'PROGRAM ERROR HAS OCCURRED!' TO WS-AREA.
    MOVE WS-AREA TO DFHCOMMAREA.

    EXEC CICS RETURN
    END-EXEC .
    GOBACK.
```

## IWAYSAMP Program

---

The following sample program called IWAYSAMP processes multiple record type outputs.

```

*-----+-----*
* IWAYSAMP
*-----
* 100 BYTE COMMAREA
*      'P' IN FIRST POSITION RETURNS 40 BYTES
*      'F' IN FIRST POSITION RETURNS 60 BYTES
*      ELSE
*          RETURN ERROR MESSAGE UNFORMATTED STRING
*
*-----*
* TAILORING:
*-----+-----+-----+-----*
IDENTIFICATION DIVISION.
PROGRAM-ID. IWAYSAMP.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01 WS-DATA-OUT-P          PIC X(40).
01 WS-DATA-OUT-F          PIC X(60).

LINKAGE SECTION.

01 DFHCOMMAREA.
   05 CA-DATA.
      10 CA-FIRST-1          PIC X(1).
      10 FILLER              PIC X(99).

PROCEDURE DIVISION.
   EXEC CICS HANDLE ABEND
      LABEL (030-ABEND)
   END-EXEC

   IF EIBCALEN = 0
      MOVE ' EIBCALEN IS = ZERO ' TO CA-DATA
      PERFORM 020-RETURN
   END-IF.

   IF CA-FIRST-1 = 'P'
      MOVE 'VALUE OF P IN FIRST BYTE OF COMMAREA'
         TO WS-DATA-OUT-P
      MOVE WS-DATA-OUT-P TO CA-DATA
      GO TO 020-RETURN
   END-IF.

```

```
IF CA-FIRST-1 = 'F'
  MOVE 'VALUE OF F IN FIRST BYTE OF COMMAREA'
    TO WS-DATA-OUT-F
  MOVE WS-DATA-OUT-F TO CA-DATA
  GO TO 020-RETURN
END-IF.

MOVE 'SUPPLY P OR F IN FIRST BYTE OF INPUT...PLEASE'
  TO CA-DATA.

020-RETURN.

EXEC CICS RETURN
END-EXEC.
GOBACK.

030-ABEND.
GO TO 020-RETURN.
```

## CICS TCP/IP Sockets Program for Mainframe

---

This program uses the MVS sockets interface EZASOCKET. For other platforms, use the socket interface appropriate for that platform.

```
000100 CBL TRUNC(BIN)
000200 ID DIVISION.
000300 PROGRAM-ID. TCPIPPOC.
000400*****
000500* CICS TCP/IP SOCKET INTERFACE PROGRAM--SEND TO SOUND-LISTENER
000600*****
000700 ENVIRONMENT DIVISION.
000800 DATA DIVISION.
000900 WORKING-STORAGE SECTION.
001000 01 SOCKET-GROUP.
001100     05 SOC-FUNCTION          PIC X(16) VALUE SPACES.
001200     05 ERRNO                PIC 9(8) BINARY VALUE ZEROES.
001300     05 RETCODE              PIC S9(8) BINARY VALUE ZEROES.
001400     05 AF                   PIC 9(8) BINARY VALUE 2.
001500     05 SOCTYPE              PIC 9(8) BINARY VALUE 1.
001600     05 PROTO                PIC 9(8) BINARY VALUE 0.
001700     05 NAMELEN              PIC 9(8) BINARY.
001800     05 HOSTNAME             PIC X(255).
001900     05 HOSTENT              PIC 9(8) BINARY.
002000     05 NAME.
002100     10 FAMILY               PIC 9(4) BINARY VALUE 2.
002200     10 PORT                 PIC 9(4) BINARY VALUE 4772.
002300     10 IP-ADDRESS           PIC 9(8) BINARY.
002400     10 IP-ADDRESS-ALPHA REDEFINES IP-ADDRESS PIC X(4).
002500     10 RESERVED            PIC X(8) VALUE LOW-VALUES.
```

```

002600      05  FLAGS                      PIC 9(8) BINARY VALUE 0.
002700      05  SOCKET                     PIC 9(4) BINARY.
002800      05  NBYTE                        PIC 9(8) BINARY.
002900      05  CMD                          PIC 9(8) BINARY.
003000      05  REQARG                      PIC 9(8) BINARY.
003100
003200 01  WORKAREA.
003300      05  RESP                        PIC S9(9) BINARY VALUE 0.
003400      05  DATAPTR                     POINTER VALUE NULL.
003500      05  DATAPTR-L REDEFINES DATAPTR PIC 9(9) BINARY.
003600      05  DATAKEY                     PIC S9(9) BINARY VALUE 0.
003700      05  DATALEN                     PIC S9(4) BINARY VALUE 0.
003800      05  STARTTIME                     PIC S9(15) PACKED-DECIMAL.
003900      05  ENDTIME                       PIC S9(15) PACKED-DECIMAL.
004000      05  DURATION                     PIC 99999.
004100      05  LLEN                         PIC 9(8) BINARY VALUE 4.
004200      05  TMSG                        PIC X(25)
004300      VALUE 'ELASPED TIME = '.
004400      05  ERRMSG                        PIC X(41)
004500      VALUE 'ERROR ENCOUNTERED DURING '.
004600      05  TRCMSG                        PIC X(41)
004700      VALUE 'RC = 9999999999 FOR '.
004800
004900 LINKAGE SECTION.
005000 PROCEDURE DIVISION.
005100 MAINLINE.
005200      PERFORM GETSOCK
005300*     PERFORM GETHOSTBYNAME
005400      PERFORM SETBLOCK
005500      PERFORM CONNECTTOHOST
005600      PERFORM SENDDATA
005700      PERFORM CLOSESOCK
005800      EXEC CICS RETURN END-EXEC
005900      GOBACK
006000      .
006100
006200006300 GETSOCK.
006400      MOVE 'SOCKET'                      ' TO SOC-FUNCTION
006500      CALL 'EZASOCKET' USING SOC-FUNCTION,
006600          AF,
006700          SOCTYPE,
006800          PROTO,
006900          ERRNO,
007000          RETCODE
007100

```

```
007200      MOVE RETCODE TO SOCKET
007300      IF RETCODE < 0
007400          PERFORM WRITERR-EXIT
007500      END-IF
007600      .
007700
007800 SETBLOCK.
007900      MOVE 'FCNTL          ' TO SOC-FUNCTION
008000      MOVE 4 TO CMD
008100      MOVE 0 TO REQARG
008200      CALL 'EZASOKET' USING SOC-FUNCTION, SOCKET, CMD, REQARG,
008300                      ERRNO, RETCODE
008400      .
008500
008600 CONNECTTOHOST.
008700      MOVE X'AC1EA611' TO IP-ADDRESS-ALPHA
008800      MOVE 'CONNECT      ' TO SOC-FUNCTION
008900      CALL 'EZASOKET' USING SOC-FUNCTION,
009000          SOCKET,
009100          NAME,
009200          ERRNO,
009300          RETCODE,
009400      IF RETCODE = 0
009500          CONTINUE
009600      ELSE
009700          PERFORM WRITERR-EXIT
009800      END-IF
009900      .
010000
010100 SENDDATA.
010200      MOVE 'SEND          ' TO SOC-FUNCTION
010300      EXEC CICS STARTBR FILE('TESTFILE') RIDFLD(DATAKEY) RBA
010400          RESP(RESP)
010500      END-EXEC
010600
010700      EXEC CICS ASKTIME ABSTIME(STARTTIME) END-EXEC
010800
010900      PERFORM 300 TIMES
010910*      PERFORM UNTIL RESP > 0
011000          EXEC CICS READNEXT FILE('TESTFILE')
011100              RIDFLD(DATAKEY)
011200              RBA
011300              LENGTH(DATALEN)
011400              RESP(RESP)
011500              SET(DATAPTR)
011600      END-EXEC
```



```

011700      IF RESP = 0
011710          MOVE DATALEN TO NBYTE
011800*      ADD 0 TO DATALEN GIVING NBYTE
011900      MOVE 4 TO LLEN
012000      MOVE 0 TO RETCODE
012100*      PERFORM UNTIL RETCODE = NBYTE
012200*          OR RETCODE < 0
012300
012400          CALL 'EZASOKET' USING SOC-FUNCTION,
012500              SOCKET,
012600              FLAGS,
012700              LLEN,
012800              NBYTE,
012900              ERRNO,
013000              RETCODE
013100
013200      IF RETCODE = -1
013300          MOVE 1 TO RESP
013400          PERFORM WRITERR-EXIT
013500      END-IF
013600*      MOVE RETCODE TO TRCMSG(6:10)
013700*      MOVE NBYTE TO TRCMSG(22:10)
013800*      EXEC CICS WRITEQ TD QUEUE('CSSL')
013900*          FROM(TRCMSG) LENGTH(41)
014000*      END-EXEC
014100
014200      MOVE DATALEN TO NBYTE
014300
014400      CALL 'EZASOKET' USING SOC-FUNCTION,
014500          SOCKET,
014600          FLAGS,
014700          NBYTE,
014800          BY VALUE DATAPTR,
014900          BY REFERENCE ERRNO,
015000          RETCODE
015100
015200**      MOVE RETCODE TO TRCMSG(6:10)
015300**      MOVE NBYTE TO TRCMSG(22:10)
015400**      EXEC CICS WRITEQ TD QUEUE('CSSL')
015500**          FROM(TRCMSG) LENGTH(41)
015600*      END-EXEC
015700*      IF RETCODE NOT = NBYTE
015800*          SUBTRACT RETCODE FROM NBYTE
015900*          ADD RETCODE TO DATAPTR-L
016000*      END-IF
016100*      END-PERFORM

```

```
016200             IF RETCODE = -1
016300                 MOVE 1 TO RESP
016400                 PERFORM WRITERR-EXIT
016500             END-IF
016600         END-IF
016700     END-PERFORM
016800
016900     IF RESP = 1
017000         PERFORM WRITERR-EXIT
017100     END-IF
017200
017300     EXEC CICS ASKTIME ABSTIME(ENDTIME) END-EXEC
017400
017500     SUBTRACT STARTTIME FROM ENDTIME GIVING DURATION
017600
017700     MOVE DURATION TO TMSG(17:8)
017800
017900     EXEC CICS SEND TEXT FROM(TMSG)
018000         LENGTH(LENGTH OF TMSG)
018100     END-EXEC
018200
018300     .
018400
018500 CLOSESOCK.
018600     MOVE ZEROES TO RETCODE ERRNO
018700     MOVE 'CLOSE' ' TO SOC-FUNCTION
018800     CALL 'EZASOKET' USING SOC-FUNCTION
018900         SOCKET
019000         ERRNO
019100         RETCODE
019200     IF RETCODE < 0
019300         PERFORM WRITERR-EXIT
019400     END-IF
019500     .019600 GETHOSTBYNAME.
019700     MOVE 'GETHOSTBYNAME' ' TO SOC-FUNCTION
019800     CALL 'EZASOKET' USING SOC-FUNCTION NAMELEN NAME
019900         HOSTENT RETCODE
020000     .
020100
020200 WRITERR-EXIT.
020300     MOVE SOC-FUNCTION TO ERRMSG(26:15)
020400     EXEC CICS SEND TEXT FROM(ERRMSG)
020500         LENGTH(LENGTH OF ERRMSG)
020600     END-EXEC
020700
020800     EXEC CICS RETURN END-EXEC.
```

## CICS TCP/IP Sockets Program for TX

---

The following sample C program uses a TX sockets interface.

```
#include <cics_api.h>
#include <cics_eib.h>

/*****
/* C Sample Program
/*
/* This is a sample program for a cics client to the iWay adapter.
/*
/* Author John Schlesinger
/* Version 1.0
*****/

/*****
/* #includes
*****/

#include <cicstype.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>

#include <ctype.h>
#include <errno.h>
/* Include the right sockets headers */
#ifdef _MSC_VER
#include <winsock2.h>
#else
#include <arpa/inet.h>
#include <sys/socket.h>
#endif

/*****
/* #defines
*****/

#define InsertCursor '\x13' /* 3270 Insert cursor */
#define ebcdic_ProtectAttribute '\xe4' /* 3270 Protected Attr */
#define ascii_ProtectAttribute '\x55' /* 3270 Protected Attr */

#define MSG_SIZE 32500 /*
```

```

/*****
/* Procedure Declarations
*****/

void Output_Text (char*);
void Log_Text (char*);
void cics_time (cics_char_t []);
void Process_Startup_Parameters (void);
void return_to_cics (void);

void SendData(char *, short);
int  writen(int, const void *, size_t);
int  readn(int, void *, size_t);

/*****
/* Structures
*****/

struct screen_struct
{
    cics_char_t sf;
    cics_char_t attr;
    cics_char_t display [160];
};

struct log_struct
{
    cics_char_t program [8];
    cics_char_t filler0;
    cics_char_t applid [8];
    cics_char_t filler1;
    cics_char_t msg [80];
};

/*****
/* Global Variables
*****/
cics_char_t Term_Code [2]  = "00";    /* Terminal Code
cics_char_t sba [4];          /* 3270 set buffer address
cics_char_t ProtectAttribute ;    /* 3270 Protect Attribute
cics_sshort_t scrnwd = 80;      /* Width of the screen
struct screen_struct output_screen = {'\x1d', ' ', ""};
struct log_struct CSMT_log = {"CICSDEXI", ' ', "", ' ', ""};

```

```

/*****
/* Procedure      : Main                                     */
/* Function       : To call all sub-procedures              */
/* Input          : None                                     */
/* Returns        : Nothing                                  */
*****/

void main( void )
{ \
    int cics_api_temp_var = cics_api_edf_init_c_extended(0, &CicsArgs);

    Process_Startup_Parameters();

    Output_Text("Starting the test");

    SendData("168.135.180.16", 4773);
//    SendData("168.135.63.14", 4773);

    Output_Text("Test completed");

    return_to_cics();

} /* End of main */

/*****
/* Procedure      : Process_Startup_Parameters              */
/* Function       : To Determine how the transaction was initiated */
/*                and process any parameters supplied       */
/* Input          : None                                     */
/* Returns        : Nothing                                  */
*****/

void Process_Startup_Parameters()
{
    cics_sshort_t len_out;
    cics_char_t ascii_sba [4] = {'\x11', '\x20', '\x41', '\x13'};
    cics_char_t ebcdic_sba [4] = {'\x11', '\x40', '\xC1', '\x13'};
    cics_char_t Start_Code [2] = "00"; /* Start Code */
    cics_sshort_t Text_Length = 80;
    cics_char_t Text_Buffer [80];
    cics_ubyte_t This_OP SYS = '\0'; /* local OPSYS */
    cics_char_t This_RELEASE [5] = "" ; /* local cics release */
    char * pch;

    /* initialise storage */
    memset(Text_Buffer, '\0', 79);
    Text_Buffer [79] = '\0';
    /* EXEC CICS ADDRESS EIB(dfheiptr); */ \

```

```

    { \
    CicsArgs.ArgCode[0] = 77; \
    CicsArgs.ArgData[0].PointerRef = &dfheiptr; \
    CicsArgs.ArgMask = 0x20000000; \
    CicsArgs.ArgCount = 1; \
    CicsArgs.FnCode = 2; \
    CicsArgs.DebugLine = -1; \
    cics_api_exec_c_extended(&CicsArgs); \
    dfheiptr = (dfheiptr); \
    }

    /* EXEC CICS INQUIRE SYSTEM
        OPSYS(&This_OPSYS)
        RELEASE(This_RELEASE); */ \

    { \
    CicsArgs.ArgCode[0] = 325; \
    CicsArgs.ArgData[0].StringArea = This_RELEASE; \
    CicsArgs.ArgCode[1] = 252; \
    CicsArgs.ArgData[1].ByteArea = &This_OPSYS; \
    CicsArgs.ArgMask = 0x20000000; \
    CicsArgs.ArgCount = 2; \
    CicsArgs.FnCode = 42; \
    CicsArgs.DebugLine = -1; \
    cics_api_exec_c_extended(&CicsArgs); \
    }

    /* if ASCII opsys use ASCII 3270 cursor positioning*/
    if ((This_OPSYS == 'P') || (This_OPSYS == 'A') ||
        (This_OPSYS == 'H') || (This_OPSYS == 'O') ||
        (This_OPSYS == 'S') || (This_OPSYS == 'L') ||
        (This_OPSYS == 'N'))
    {
        strncpy(sba, ascii_sba, 4);
        output_screen.attr = '\x48';
        ProtectAttribute = ascii_ProtectAttribute;
    }
    else
    {
        strncpy(sba, ebcdic_sba, 4);
        output_screen.attr = '\xc8';
        ProtectAttribute = ebcdic_ProtectAttribute;
    }

    /* EXEC CICS ASSIGN
        APPLID(CSMT_log.applid)
        SCRNRD(scrnwd)
        STARTCODE(Start_Code)
        TERMCODE(Term_Code); */ \

```

```

    { \
    CicsArgs.ArgCode[0] = 18; \
    CicsArgs.ArgData[0].StringArea = CSMT_log.applid; \
    CicsArgs.ArgCode[1] = 351; \
    CicsArgs.ArgData[1].ShortArea = &scrnwd; \
    CicsArgs.ArgCode[2] = 373; \
    CicsArgs.ArgData[2].StringArea = Start_Code; \
    CicsArgs.ArgCode[3] = 397; \
    CicsArgs.ArgData[3].StringArea = Term_Code; \
    CicsArgs.ArgMask = 0x20000000; \
    CicsArgs.ArgCount = 4; \
    CicsArgs.FnCode = 5; \
    CicsArgs.DebugLine = -1; \
    cics_api_exec_c_extended(&CicsArgs); \
    }
    /* Handle input from terminal or from a START */
    if (Start_Code [0] == 'S')
    {
        /* EXEC CICS RETRIEVE INTO(Text_Buffer) LENGTH(Text_Length);
*/ \
        { \
        CicsArgs.ArgData[8].DataArea = Text_Buffer; \
        CicsArgs.ArgData[2].ShortArea = &Text_Length; \
        CicsArgs.ArgMask = 0x20000104; \
        CicsArgs.FnCode = 71; \
        CicsArgs.DebugLine = -1; \
        cics_api_exec_c_extended(&CicsArgs); \
        }
        }
        else
        {
            /* EXEC CICS RECEIVE INTO(Text_Buffer) LENGTH(Text_Length); */
\
            { \
            CicsArgs.ArgData[8].DataArea = Text_Buffer; \
            CicsArgs.ArgData[2].ShortArea = &Text_Length; \
            CicsArgs.ArgMask = 0x20000104; \
            CicsArgs.FnCode = 67; \
            CicsArgs.DebugLine = -1; \
            cics_api_exec_c_extended(&CicsArgs); \
            }
        }
    }

```

```

/* Check if screen input is new or modified */
/* If modified need to jump 3 bytes to skip sba */

    if (!(memcmp(Text_Buffer, sba, 3)))
    {
        memcpy(Text_Buffer, Text_Buffer+3, (size_t)Text_Length);
    }

/* if a 3270 terminal resend hostid back to the screen */
/* and position InsertCursor for new line */
if ((Term_Code [0] > '\x90') &&
    (Term_Code [0] < '\xa0'))
{
    strncpy(output_screen.display,
Text_Buffer, (size_t)Text_Length);
    len_out = (cics_sshort_t) (scrnwd + 3);

    output_screen.display [scrnwd - 2] = InsertCursor;
    /* EXEC CICS SEND FROM(&output_screen) LENGTH(len_out) ERASE;
*/ \
    { \
        CicsArgs.ArgData[4].DataArea = &output_screen; \
        CicsArgs.ArgData[2].ShortValue = len_out; \
        CicsArgs.ArgMask = 0x20000414; \
        CicsArgs.FnCode = 74; \
        CicsArgs.DebugLine = -1; \
        cics_api_exec_c_extended(&CicsArgs); \
    }
}

} /* End of Process_Startup_Parameters */

```



```

/*****
/* Procedure      : Output_Text                               */
/* Function       : To Display text to Termianl (if appropriate) */
/* Input          : Text to be displayed                       */
/* Returns        : Nothing                                    */
*****/

void Output_Text(char* Text)
{
    cics_sshort_t len_out;
    char Temp_Text [92];          /* text buffer */
    cics_char_t time_msg [9];

    cics_time(time_msg);
    sprintf (Temp_Text, "%s %s",time_msg,Text);
    len_out = (cics_sshort_t) (scrnwd + 3);

    if ((Term_Code [0] > '\x90') &&
        (Term_Code [0] < '\xa0'))
    {
        strncpy(output_screen.display, Temp_Text, 92);
        memset(output_screen.display+92, ' ',
                sizeof(output_screen.display)-92);
        output_screen.display [scrnwd - 1] = InsertCursor;
        output_screen.attr = ProtectAttribute;

        /* EXEC CICS SEND FROM(&output_screen) LENGTH(len_out) WAIT; */
\
    { \
        CicsArgs.ArgData[4].DataArea = &output_screen; \
        CicsArgs.ArgData[2].ShortValue = len_out; \
        CicsArgs.ArgMask = 0x20800014; \
        CicsArgs.FnCode = 74; \
        CicsArgs.DebugLine = -1; \
        cics_api_exec_c_extended(&CicsArgs); \
    }
    }

    /* clear screen structure */

    memset(output_screen.display, ' ', 159);
    output_screen.display [159]='\0';

} /* End of Output_Text */

```

```

/*****
/* Procedure      : Log_Text                                     */
/* Function       : To send text to CICS CSMT log               */
/* Input          : Text to be displayed                       */
/* Returns        : Nothing                                     */
*****/
void Log_Text(char* Text)
{
    cics_sshort_t Text_Length = 105;
    cics_char_t time_msg [9];

    cics_time(time_msg);
    sprintf (CSMT_log.msg, "%s %s",time_msg,Text);

    /* EXEC CICS WRITEQ TD QUEUE("CSMT") FROM(&CSMT_log)
LENGTH(Text_Length); */ \
    { \
        cics_api_strncpy(CicsArgs.ArgData[9].StringValue, "CSMT", (short)4); \
        CicsArgs.ArgData[4].DataArea = &CSMT_log; \
        CicsArgs.ArgData[2].ShortValue = Text_Length; \
        CicsArgs.ArgMask = 0x20000214; \
        CicsArgs.FnCode = 106; \
        CicsArgs.DebugLine = -1; \
        cics_api_exec_c_extended(&CicsArgs); \
    }

    /* clear log structures */
    memset(CSMT_log.msg, ' ', 79);
    CSMT_log.msg [79]='\0';

} /* End of Output_Text */

```

```

/*****
 * Function name: return_to_cics
 * Description:   Return to CICS and exit program
 * Parameters:    None
 * Returns:       None
 *****/

void return_to_cics(void)
{
    cics_char_t blank_log [80] = "" ;

    /* write blank line to log */
    /* EXEC CICS WRITEQ TD QUEUE("CSMT") FROM(blank_log) LENGTH(80); */
\
    { \
        cics_api_strncpy(CicsArgs.ArgData[9].StringValue, "CSMT", (short)4); \
        CicsArgs.ArgData[4].DataArea = blank_log; \
        CicsArgs.ArgData[2].ShortValue = 80; \
        CicsArgs.ArgMask = 0x20000214; \
        CicsArgs.FnCode = 106; \
        CicsArgs.DebugLine = -1; \
        cics_api_exec_c_extended(&CicsArgs); \
    }

    /* reposition 3270 cursor
    if ((Term_Code [0] > '\x90') &&
        (Term_Code [0] < '\xa0'))
    {
        EXEC CICS SEND FROM(sba) LENGTH(4);
    } */

    /* EXEC CICS RETURN; */ \
    { \
        CicsArgs.ArgMask = 0x20000000; \
        CicsArgs.FnCode = 72; \
        CicsArgs.DebugLine = -1; \
        cics_api_exec_c_extended(&CicsArgs); \
    }
}

```

```

/*****
 * Function name: SendData
 * Description:   Sends the data to the host and port
 * Inputs:       Host id and port number
 * Returns:      None
 *****/

void SendData(char * Host_Id, short Port_Num)
{
    char * pch;
    static char Temp_Text [80];           /* text buffer */
    int connfd, crc, i, BytesRead;
    struct sockaddr_in servaddr;
    char RcvBuf[MSG_SIZE+1];
    char MsgBuf[] = \
"<?xml version='1.0' encoding='UTF-8'?'> \
<ZPBT_IN> \
    <in> \
        <ORDER_NUM/> \
        <ITEM_NUM>00000</ITEM_NUM> \
        <BANK_CODE/> \
        <BANK_ACCT/> \
    </in> \
    <in> \
        <ORDER_NUM>2015689292</ORDER_NUM> \
        <ITEM_NUM>00001</ITEM_NUM> \
        <BANK_CODE>123456789</BANK_CODE> \
        <BANK_ACCT>9753124680</BANK_ACCT> \
    </in> \
    <in> \
        <ORDER_NUM>2025689393</ORDER_NUM> \
        <ITEM_NUM>00001</ITEM_NUM> \
        <BANK_CODE/> \
        <BANK_ACCT>1345678920</BANK_ACCT> \
    </in> \
    <in> \
        <ORDER_NUM>2025689393</ORDER_NUM> \
        <ITEM_NUM>00006</ITEM_NUM> \
        <BANK_CODE/> \
        <BANK_ACCT/> \
    </in> \
    <in> \
        <ORDER_NUM>2025689393</ORDER_NUM> \
        <ITEM_NUM>00000</ITEM_NUM> \
        <BANK_CODE/> \
        <BANK_ACCT/> \
    </in> \
</ZPBT_IN>" ;

```

```

connfd = socket(AF_INET, SOCK_STREAM, 0);

if (connfd < 0) {
    Output_Text("Error creating socket");
    return_to_cics();
}

sprintf (Temp_Text, "Trying to connect to host %s and port %d",
Host_Id, Port_Num);
Output_Text(Temp_Text);
servaddr.sin_family = AF_INET;
servaddr.sin_port = htons(Port_Num);
servaddr.sin_addr.s_addr = inet_addr(Host_Id); /* Use quad 4 notation
*/
    if ((crc = connect(connfd, (struct sockaddr *) &servaddr,
sizeof(servaddr))) < 0) {
        Output_Text("Connect Error");
        return_to_cics();
    }
    sprintf (Temp_Text, "Number of bytes to send is %d bytes",
strlen(MsgBuf));
    Output_Text(Temp_Text);

    writen(connfd, MsgBuf, strlen(MsgBuf));

    BytesRead = readn(connfd, RcvBuf, sizeof(RcvBuf));

    sprintf (Temp_Text, "Number of bytes read is %d bytes", BytesRead);
    Output_Text(Temp_Text);
    Output_Text("First ten lines of payload are:");
    for (i=519;i<1220;i+=70) {
        strncpy(Temp_Text,&RcvBuf[i],70);
        Output_Text(Temp_Text);
    }
    for (i=0;i<BytesRead;i+=78) {
        pch = RcvBuf + i;
        strncpy(Temp_Text, pch, 78);
        Log_Text(Temp_Text);
    }

#ifdef _MSC_VER
    closesocket(connfd);
#else
    close(connfd);
#endif

    return;
}
/* End send data
*/

```

```

/*****
 * Function name: cics_time
 * Description:  gets the time
 * Returns:     time_msg as 8 cics_char_t string
 *****/

void cics_time (cics_char_t time_msg[])
{
    cics_char_t abstime [8];

    /* EXEC CICS ASKTIME ABTIME(abstime); */ \
    { \
    CicsArgs.ArgData[0].DataArea = abstime; \
    CicsArgs.ArgMask = 0x20000001; \
    CicsArgs.FnCode = 4; \
    CicsArgs.DebugLine = -1; \
    cics_api_exec_c_extended(&CicsArgs); \
    }
    /* EXEC CICS FORMATTIME ABTIME(abstime) TIME(time_msg) TIMESEP; */ \
    { \
    CicsArgs.ArgData[0].DataArea = abstime; \
    CicsArgs.ArgData[15].StringArea = time_msg; \
    CicsArgs.ArgMask = 0x20018001; \
    CicsArgs.FnCode = 22; \
    CicsArgs.DebugLine = -1; \
    cics_api_exec_c_extended(&CicsArgs); \
    }
    time_msg [8]='\0';
}

```

```

/*****
 * Function name: readn
 * Description:   Utility function to read a socket
 * Inputs:       fd - socket number,
 *               vptr - buffer to read into
 *               n - length of the data read
 * Returns:      int number of bytes read
 *****/

int readn(int fd, void *vptr, size_t n)
{
    size_t nleft;
    size_t nread;
    char *ptr;

    ptr = vptr;
    nleft = n;
    while (nleft > 0)
    {
        if ( (nread = recv(fd, ptr, nleft, 0)) < 0)
        {
            if (errno == EINTR)
                nread = 0;
            else
                return (-1);
        }
        else if (nread == 0)
            break;

        nleft -= nread;
        ptr += nread;
    }
    return (n - nleft);
}

```

```

/*****
 * Function name: writen
 * Description:   Utility function to write a socket
 * Inputs:       fd - socket number,
 *               vptr - buffer to write from
 *               n - length of the data to write
 * Returns:      int number of bytes not written
 *****/

int writen(int fd, const void *vptr, size_t n)
{
    size_t nleft;
    size_t nwritten;
    const char *ptr;

    ptr = vptr;
    nleft = n;
    while (nleft > 0)
    {
        if ( (nwritten = send(fd, ptr, nleft, 0)) <= 0)
        {
            if (errno == EINTR)
                nwritten = 0;
            else
                return (-1);
        }
        nleft -= nwritten;
        ptr += nwritten;
    }
    return (nleft);
}

```



---

---

## APPENDIX E

# Debugging and Troubleshooting

**Topic:**

- Messages

This appendix includes tips and techniques for debugging the iWay Adapter for CICS.

## Messages

---

The following messages may appear:

Unable to execute Transaction {program name}

VTAM and CICS error message:

VTAM RETURN CODE 1001 SENSE CODE 8004 0000

**Possible Cause:** Network DNS problem: old connection lingering in DNS, flush DNS.

**Solution:** Verify TCP/IP DNS is working correctly.

At the command prompt:

1. Ping the host address for the machine where the iWay Adapter for CICS is running (for example, PMSNJC) to obtain the IP address that you require for the following step.
2. Use the IP address (for example, 172.30.166.90) and enter the following command:

```
nbtstat -A 172.30.166.90
```

```
H:\>
H:\>ping pmsnjc

Pinging pmsnjc.ibi.com [172.30.166.90] with 32 bytes of data:

Reply from 172.30.166.90: bytes=32 time<10ms TTL=127
Reply from 172.30.166.90: bytes=32 time<10ms TTL=127
Reply from 172.30.166.90: bytes=32 time<10ms TTL=127
Reply from 172.30.166.90: bytes=32 time<10ms TTL=127

Ping statistics for 172.30.166.90:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

H:\>nbtstat -A 172.30.166.90

Local Area Connection:
Node IpAddress: [172.30.242.94] Scope Id: []

        NetBIOS Remote Machine Name Table

    Name                 Type                     Status
    ----                 -
    PGMMSGY               <00>    UNIQUE              Registered
    EDA                   <00>    GROUP                Registered
    PGMMSGY               <20>    UNIQUE              Registered
    EDA                   <1E>    GROUP                Registered
    PGMMSGY               <03>    UNIQUE              Registered

    MAC Address = 00-50-DA-BA-15-9E

H:\>
```

3. If the host address (PGMMGY) as shown in the previous screen is not the same as in the previous command (PMSNJC), issue the command:

```
ipconfig /flushdns
```

**Note:** If the problem is still not resolved, contact your Network Administrator.



---

---

## Reader Comments

In an ongoing effort to produce effective documentation, the Documentation Services staff at Information Builders welcomes any opinion you can offer regarding this manual.

Please use this form to relay suggestions for improving this publication or to alert us to corrections. Identify specific pages where applicable. You can contact us through the following methods:

**Mail:** Documentation Services - Customer Support  
Information Builders, Inc.  
Two Penn Plaza  
New York, NY 10121-2898

**Fax:** (212) 967-0460

**E-mail:** [books\\_info@ibi.com](mailto:books_info@ibi.com)

**Web form:** <http://www.informationbuilders.com/bookstore/derf.html>

Name: \_\_\_\_\_

Company: \_\_\_\_\_

Address: \_\_\_\_\_

Telephone: \_\_\_\_\_ Date: \_\_\_\_\_

E-mail: \_\_\_\_\_

Comments:

---

---

## Reader Comments