

# iWay

iWay Transaction Adapter  
for CICS User's Guide

Version 7.0.x and Higher

Active Technologies, EDA, EDA/SQL, FIDEL, FOCUS, Information Builders, the Information Builders logo, iWay, iWay Software, Parlay, PC/FOCUS, RStat, Table Talk, Web390, WebFOCUS, WebFOCUS Active Technologies, and WebFOCUS Magnify are registered trademarks, and DataMigrator and Hyperstage are trademarks of Information Builders, Inc.

Adobe, the Adobe logo, Acrobat, Adobe Reader, Flash, Adobe Flash Builder, Flex, and PostScript are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

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 © 2018, 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.

# Contents

---

<b>Preface</b> .....	<b>7</b>
Documentation Conventions .....	8
Related Publications .....	9
Customer Support .....	9
Help Us to Serve You Better .....	9
User Feedback .....	12
Information Builders Consulting and Training .....	12
<b>1. Introducing the iWay Transaction Adapter for CICS</b> .....	<b>13</b>
iWay Transaction Adapter for CICS Overview .....	13
The iWay Transaction Adapter for CICS .....	15
CICS Programs .....	16
Software Requirements for the Adapter .....	16
Deployment Information for Your iWay Adapter .....	17
iWay Service Manager .....	17
iWay Explorer .....	17
iWay Business Services Provider (iBSP) .....	17
iWay Transaction Adapter for CICS Information Roadmap .....	18
<b>2. Configuring the iWay Transaction Adapter for CICS</b> .....	<b>19</b>
Starting iWay Explorer (Java Servlet) .....	19
Configuring a Connection to CICS .....	20
Managing a Connection to CICS .....	28
<b>3. Creating XML Schemas and iWay Business Services</b> .....	<b>31</b>
Creating an Adapter Transaction .....	31
Sample Program IWAYSrv0 .....	32
Side File Support .....	38
COBOL Descriptions for Input and Output Communications .....	40
Modifying COBOL DD Field Definitions .....	40
Creating Schemas for an Adapter Transaction .....	43
Understanding iWay Business Services .....	44
Creating a Web Service .....	45
Testing the Web Service .....	47

Generating WSDL From a Web Service.....	48
Identity Propagation.....	50
<b>4. Event Processing With the CICS Adapter .....</b>	<b>51</b>
Understanding CICS Events .....	51
Message Format.....	53
Supported Environments.....	53
Configuring CICS Events .....	53
Creating and Modifying an Event Port .....	57
Creating and Modifying a Channel .....	71
Testing CICS Events .....	77
Running CICS Events .....	78
<b>A. Configuring the Transaction Adapter for CICS in an iWay Environment .....</b>	<b>81</b>
Configuring the Transaction Adapter for CICS in Service Manager .....	81
<b>B. Running the Adapter Using LU6.2 Communication .....</b>	<b>85</b>
MVS OS/390 APPC Communication .....	85
LU6.2 Set up on MVS.....	85
LU6.2 Set up on CICS.....	86
Microsoft SNA Server Communication .....	87
LU6.2 Setup on a Windows SNA Server.....	87
Application Run-Time Requirements .....	89
<b>C. Running the Adapter Using TCP/IP Communication .....</b>	<b>91</b>
MVS OS/390 TCP/IP Communication .....	91
TCP/IP Requirements.....	91
<b>D. Using Adabas/Natural Programs .....</b>	<b>93</b>
Adabas/Natural Programs Overview .....	93
Installing the Adabas/Natural Interface .....	94
Writing and Configuring a Natural Program .....	96
<b>E. Installing the Sample IWAYIVP and IWAYSrv0 Programs in CICS .....</b>	<b>103</b>
Installing and Configuring IWAYIVP .....	103
Installing and Configuring IWAYSrv0 .....	106
<b>F. Sample Requests, Schemas, and COBOL File Descriptions .....</b>	<b>111</b>

Request Document for the Generic Transaction, IWAYIVP .....	111
Request Schema for IWAYIVP .....	112
Response Schema for IWAYIVP .....	112
Request Documents for IWAYSРV0 .....	113
Request Schema for IWAYSРV0 .....	114
Response Schema for IWAYSРV0 .....	114
Request Document for AASNATN .....	116
Request Schema for AASNATN .....	116
Response Schema for the Program AASNATN .....	116
Sample COBOL File Descriptions .....	117
<b>G. Sample CICS Programs .....</b>	<b>119</b>
IWAYIVP Program .....	119
IWAYSРV0 Program .....	121
IWAYEVT0 Program .....	124
IWAYEVT1 Program .....	129
Natural Program .....	132
<b>H. Transaction Adapter for CICS Debugging and Troubleshooting .....</b>	<b>137</b>
Transaction Adapter for CICS Troubleshooting .....	137
CICS Data Type Conversions .....	139



# Preface

---

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

**Note:** This Release 7.0.x content is currently being updated to support iWay Release 8.0.x software. In the meantime, it can serve as a reference for your use of iWay Release 8. If you have any questions, please contact [Customer\\_Success@ibi.com](mailto:Customer_Success@ibi.com).

---

## How This Manual Is Organized

This manual includes the following chapters:

	<b>Chapter/Appendix</b>	<b>Contents</b>
1	Introducing the iWay Transaction Adapter for CICS	Introduces the adapter environment.
2	Configuring the iWay Transaction Adapter for CICS	Describes how to configure a connection to the adapter.
3	Creating XML Schemas and iWay Business Services	Describes how to create transactions for the adapter. It also provides information on how to use the generated schemas to create iWay Business Services, which expose functionality as web services.
4	Event Processing With the CICS Adapter	Describes how to configure and use CICS Events.
A	Configuring the Transaction Adapter for CICS in an iWay Environment	Describes how to configure the adapter in the Service Manager.
B	Running the Adapter Using LU6.2 Communication	Contains technical information that you can use as a guide to ensure LU6.2 communication to the CICS region.
C	Running the Adapter Using TCP/IP Communication	Contains technical information that you can use as a guide to ensure TCP/IP communication to the CICS region.
D	Using Adabas/Natural Programs	Describes how to use Adabas/Natural programs with the iWay Transaction Adapter for CICS.

Chapter/Appendix		Contents
E	Installing the Sample IWAYIVP and IWAYSIVO Programs in CICS	Describes how to verify correct installation of the adapter.
F	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.
G	Sample CICS Programs	Describes sample CICS programs provided with the installation.
H	Transaction Adapter for CICS Debugging and Troubleshooting	Includes tips and techniques for debugging the adapter.

## Documentation Conventions

The following table describes the documentation conventions that are used in this manual.

Convention	Description
THIS TYPEFACE or this typeface	Denotes syntax that you must enter exactly as shown.
<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 that you can click or select.
<u>underscore</u>	Indicates a default setting.
Key + Key	Indicates keys that you must press simultaneously.
{ }	Indicates two or three choices. Type one of them, not the braces.
	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 (...).

Convention	Description
.	Indicates that there are (or could be) intervening or additional commands.

## Related Publications

Visit our Technical Content Library at <http://documentation.informationbuilders.com>. You can also contact the Publications Order Department at (800) 969-4636.

## Customer Support

Do you have questions about this product?

Join the Focal Point community. Focal Point is our online developer center and more than a message board. It is an interactive network of more than 3,000 developers from almost every profession and industry, collaborating on solutions and sharing tips and techniques. Access Focal Point at <http://forums.informationbuilders.com/eve/forums>.

You can also access support services electronically, 24 hours a day, with InfoResponse Online. InfoResponse Online is accessible through our website, <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 <http://www.informationbuilders.com> also provides usage techniques, diagnostic tips, and answers to frequently asked questions.

Call Information Builders Customer Support Services (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. Please be ready to provide your six-digit site code number (xxxx.xx) when you call.

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

## Help Us to Serve You Better

To help our consultants answer your questions effectively, be prepared to provide specifications and sample files and to answer questions about errors and problems.

The following tables list the environment information our consultants require.

<b>Platform</b>	
<b>Operating System</b>	
<b>OS Version</b>	
<b>JVM Vendor</b>	
<b>JVM Version</b>	

The following table lists the deployment information our consultants require.

<b>Adapter Deployment</b>	For example, iWay Business Services Provider, iWay Service Manager
<b>Container</b>	For example, WebSphere
<b>Version</b>	
<b>Enterprise Information System (EIS) - if any</b>	
<b>EIS Release Level</b>	
<b>EIS Service Pack</b>	
<b>EIS Platform</b>	

The following table lists iWay-related information needed by our consultants.

<b>iWay Adapter</b>	
<b>iWay Release Level</b>	
<b>iWay Patch</b>	

The following table lists additional questions to help us serve you better.

Request/Question	Error/Problem Details or Information
Did the problem arise through a service or event?	
Provide usage scenarios or summarize the application that produces the problem.	
When did the problem start?	
Can you reproduce this problem consistently?	
Describe the problem.	
Describe the steps to reproduce the problem.	
Specify the error message(s).	
Any change in the application environment: software configuration, EIS/database configuration, application, and so forth?	
Under what circumstance does the problem <i>not</i> occur?	

The following is a list of error/problem files that might be applicable.

- Input documents (XML instance, XML schema, non-XML documents)
- Transformation files
- Error screen shots
- Error output files
- Trace files
- Service Manager package to reproduce problem

- Custom functions and agents in use
- Diagnostic Zip
- Transaction log

For information on tracing, see the *iWay Service Manager User's Guide*.

## User Feedback

In an effort to produce effective documentation, the Technical Content Management staff welcomes your opinions regarding this document. You can contact us through our website, <http://documentation.informationbuilders.com/connections.asp>.

Thank you, in advance, for your comments.

## Information Builders Consulting and Training

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

For information on course descriptions, locations, and dates, or to register for classes, visit our website (<http://education.informationbuilders.com>) or call (800) 969-INFO to speak to an Education Representative.

# Introducing the iWay Transaction Adapter for CICS

---

This section describes the iWay Transaction 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 .

**In this chapter:**

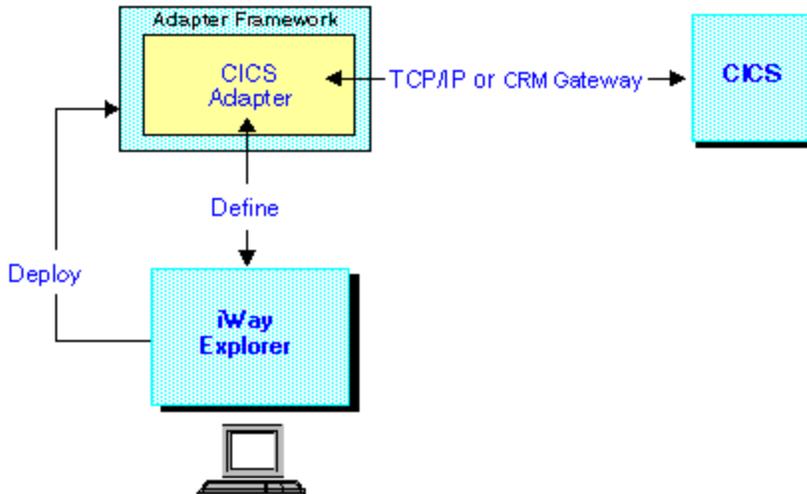
- [iWay Transaction Adapter for CICS Overview](#)
  - [The iWay Transaction Adapter for CICS](#)
  - [Deployment Information for Your iWay Adapter](#)
  - [iWay Transaction Adapter for CICS Information Roadmap](#)
- 

## iWay Transaction Adapter for CICS Overview

The iWay Transaction Adapter for CICS enables you to execute Customer Information Control System (CICS) programs. The advantages of using this adapter include:

- 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 diagram illustrates the framework for executing CICS programs with iWay Explorer and the iWay Transaction Adapter for CICS.



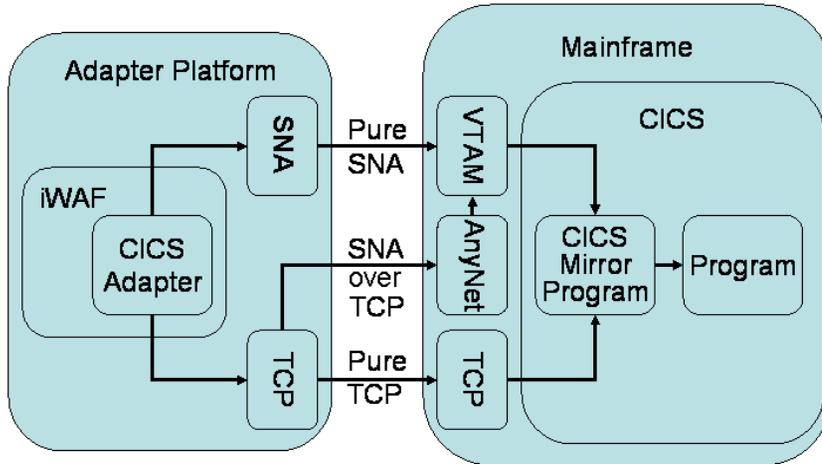
The following bidirectional scenarios are supported by the adapter:

- CICS services
- CICS events

At design-time, you describe the request and response messages by mapping them to COBOL File Descriptions. You can communicate with CICS through either TCP/IP or CRM gateway.

## The iWay Transaction Adapter for CICS

The following diagram shows how a connection is made to the CICS region.



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

The adapter 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 File Descriptions. You can communicate with CICS through either TCP/IP or CRM gateway.

The iWay Transaction Adapter for CICS also supports more complex transactional scenarios involving multiple service calls, commits, and rollbacks, when connecting to IMS via a CRM gateway.

## 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 Terminal Emulation 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 Adapter

The following are the software requirements for the adapter:

- z/OS Version 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, and 1.9.
- TCP/IP communication available to the adapter.
- One of the following releases of CICS:
  - CICS Transaction Server for z/OS, Version 3 Release 1.
  - CICS Transaction Server for z/OS, Version 2 Release 2.
  - CICS Transaction Server for z/OS, Version 2 Release 3.
  - CICS Transaction Server for z/OS, Version 4 Release 1.
  - CICS Transaction Server for z/OS, Version 5 Release 1.
  - CICS Transaction Server for OS/390, Version 1 Release 2.
  - IBM CICS/ESA, Version 4 Release 1.
  - CICS Transaction Server for VSE/ESA, Version 1.1.0.
  - 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); or TXSeries, Version 5.0 (AIX and Windows).

- ❑ For Adabas/Natural execution, Adabas and Natural must be installed and configured within the CICS region.

## Deployment Information for Your iWay Adapter

Your iWay adapter works in conjunction with one of the following components:

- ❑ iWay Service Manager
- ❑ iWay Business Services Provider (iBSP)

When hosted in an iWay environment, the adapter is configured through iWay Service Manager and iWay Explorer. iWay Explorer is used to configure system connections, create web services, and configure event capabilities. Service Manager can access this configuration information through the iWay7 repository to create a robust integration solution.

When the adapter is hosted in a third-party application server environment, you can configure iWay Explorer to work in a web services environment.

### iWay Service Manager

iWay Service Manager is the heart of the Universal Adapter Framework and is an open transport service bus. Service Manager uses graphical tools to create sophisticated integration services without writing custom integration code by:

- ❑ Creating metadata from target applications.
- ❑ Transforming and mapping interfaces.
- ❑ Managing stateless processes.

Its capability to manage complex adapter interactions makes it ideally suited to be the foundation of a service-oriented architecture.

### iWay Explorer

iWay Explorer uses a tree metaphor to introspect a system for metadata. The explorer enables you to create XML schemas and web services for the associated object. In addition, you can create ports and channels to listen for events in a system. External applications that access a system through the adapter use either XML schemas or web services to pass data between the external application and the adapter.

### iWay Business Services Provider (iBSP)

The iWay Business Services Provider (iBSP) exposes, as web services, enterprise assets that are accessible from adapters regardless of the programming language or the particular operating system.

iBSP simplifies the creation and execution of web services when running:

- Custom and legacy applications.
- Database queries and stored procedures.
- Packaged applications.
- Terminal emulation and screen-based systems.
- Transactional systems.

Coupled with a platform and language independent messaging protocol called SOAP (Simple Object Access Protocol), XML enables application development and integration by assembling previously built components from multiple web services.

## iWay Transaction Adapter for CICS Information Roadmap

The following table lists the location of deployment and user information for components of the iWay Transaction Adapter for CICS.

<b>Deployment Option</b>	<b>Chapter/Manual</b>
iWay Service Manager	Appendix A of this guide <i>iWay Service Manager User's Guide</i>
iWay Explorer	Chapters 4 and 5 of this guide <i>iWay Installation and Configuration manual</i>
iWay Business Services Provider (iBSP)	<i>iWay Installation and Configuration manual</i>

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

**In this chapter:**

- [Starting iWay Explorer \(Java Servlet\)](#)
  - [Configuring a Connection to CICS](#)
  - [Managing a Connection to CICS](#)
- 

## Starting iWay Explorer (Java Servlet)

iWay Explorer (Java Servlet) is a GUI tool that works in conjunction with adapters to create schemas and web services for use with iWay components or other XML or web services based programs. iWay Explorer is a web application accessible through a web browser. It must be deployed through an application server or servlet container. For more information on configuring iWay Explorer, see the *iWay Installation and Configuration* documentation. Before you can use iWay Explorer, you must start iWay Service Manager.

### **Procedure:** How to Start iWay Explorer (Java Servlet)

To start iWay Explorer (Java Servlet):

1. Ensure the server is started where iWay Explorer is running.
2. Type the following URL in your browser window

`http://hostname:port/iwae/index.html`

where:

`hostname`

Is the name of the server where Service Manager is installed.

`port`

Is the SOAP port number for the server. The default SOAP port is 9000.

iWay Explorer opens.

The Available Hosts drop-down list appears in the upper-right corner. Three tabs appear near the top of the iWay Explorer window. From left to right they are:

**iWay Adapters**, where you create and manage connections to CICS.

**iWay Events**, where you configure CICS event listening.

**iWay Business Services**, where you create and view business services.

The left pane of the window contains an expandable list of adapter nodes (based on the adapters installed), events, or business services, depending on the tab that is selected. The right pane provides the details of the selected adapter, event, or service and is the work area where you define and modify adapter functions and services.

The Available Hosts drop-down list specifies to which Servlet iBSP instance.

You are now ready to define a target to CICS.

### Configuring a Connection to CICS

To access CICS, you must configure a connection through the adapter, known as a target. After the connection is created, it is automatically saved. You must establish a connection to CICS every time you start iWay Explorer or after disconnecting from a target.

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

To configure a connection to CICS:

1. In the left pane of iWay Explorer, expand the *iWay Adapters* node.
2. Select 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 in the right pane, as shown in the following image.

### Add a new CICS target

Targets represent configured connections to instances of backend systems. Choose a name and description for the new target that you wish to create.

Target Name:

Description:

Target Type:

- a. In the Target Name field, type a name for the new target, for example, CICS\_Connection.
  - b. In the Description field, type a brief description (optional).
  - c. From the Target Type drop-down list, select the type of target, for example, TCP/IP Communication.
4. Click Next.

The Set connection info pane opens for the selected target.

**Note:** The CICS connection parameters are consistent with those found in your CICS system. For more information on parameter values that are specific to your CICS configuration, consult your CICS system administrator. This information should be the same for all transactions and messages in a single CICS system.

- a. If you selected **TCP/IP** as the type of target, proceed to [How to Set Connection Parameters for TCP/IP](#) on page 22.
- b. If you selected **CRM Gateway** as the type of target, proceed to [How to Set Connection Parameters for CRM Gateway](#) on page 24.

**Procedure: How to Set Connection Parameters for TCP/IP**

If you selected CICS via **TCP/IP**, the Set connection info pane opens, with the Connection tab active, as shown in the following image. The Connection tab contains five fields for entry and three active buttons (Back, Finish, and Cancel).

**Set connection info**

Connection      **Advanced**

Host:

Port:

User ID:

Password:

Codepage: Cp500

**Note:** TCP/IP access was introduced with CICS Transaction Server Version 2 Release 2. For more information, see [Running the Adapter Using TCP/IP Communication](#) on page 91.

1. Type values for the connection parameters.

The following table lists and describes the TCP/IP parameters.

Parameter	Description
Host	Host name, or IP address, for the computer where CICS is running.
Port	TCP port that CICS is listening on for ECI or DPL connections.

Parameter	Description
User ID	<p>Valid user ID for CICS.</p> <p>The user ID and password fields correspond to the values provided on the CICS TCP/IP service resource definition, using the ECI option, as one of the following:</p> <p>If you specify the user ID and the password, then the CICS TCP/IP service resource definition must be set to ATTACHSEC(VERIFY).</p> <p>If you specify neither the user ID nor the password, then the CICS TCP/IP resource service definition must be set to ATTACHSEC(LOCAL).</p>
Password	Valid password associated with the CICS user ID. For additional information, see the User ID parameter.
Codepage	Select the codepage from the drop-down menu. Cp500 is the default value.

- For advanced parameters, including parameters for executing Adabas/Natural Programs, see [How to Set Advanced Parameters](#) on page 23. Otherwise, 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 during installation.

### **Procedure:** How to Set Advanced Parameters

- Click the *Advanced* tab.
- Enter values for the Advanced parameters.

The following table lists and defines the Advanced parameters.

Parameter	Description
Connection Time Limit (ms)	Amount of time in milliseconds that the adapter waits for a completed response from CICS. 10,000 milliseconds is the default value.
Natural Nucleus	Name of the Natural subsystem on which the Natural program you wish to invoke resides. For example, for Natural Version 3.14, the nucleus is N314re.

Parameter	Description
Proxy Program	CICS program that calls the Natural CICS Interface. The name of the proxy provided by iWay Software is AASNATC.
Natural Logon Parameters	String that represents the default logon parameters. It can be modified depending on installation requirements.  The CICS/Natural Bridge enables Natural programs to be invoked by the adapter through CICS using the Software AG Natural CICS Interface.  <b>Note:</b> The Software AG Natural CICS Interface requires a programmatic "logon" to the Natural System.
CICS Mirror	An alternative transaction ID to the CPMI (ECI) mirror Transaction ID. This ID must be defined to CICS and point to the CICS mirror program DFHMIRS.

For additional information on executing Adabas/Natural programs, see [Using Adabas/Natural Programs](#) on page 93

3. 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 during installation.

**Procedure: How to Set Connection Parameters for CRM Gateway**

To configure a CRM gateway connection to CICS:

1. In the left pane of iWay Explorer, expand the *iWay Adapters* node.
2. Select 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 in the right pane, as shown in the following image, where you assign a name and description to the new target.

### Add a new CICS target

---

Targets represent configured connections to instances of backend systems. Choose a name and description for the new target that you wish to create.

Target Name:

Description:

Target Type:

4. In the Target Name field, type a name for the connection (for example, *CRM1*).
5. In the Description field, type an optional description for the target name you just created (for example, *CRM on IBIMVS*).
6. From the Target Type drop-down list, select *CRM Gateway*.
7. Click *Next*.

The Set connection info dialog box opens, as shown in the following image.

**Set connection info**

CRM Host:

CRM Port:

Log Mode:

Remote LU:

Max Sync Level:

Maximum Sessions:

Min Wins:

Buttons: Help, < Back, Finish, Cancel

- Enter the CRM gateway parameters to configure a new connection to CICS.

The following table lists and describes the CRM gateway parameters:

Parameter	Description
CRM Host	Host name, or IP address, for the computer where the CRM gateway is running.
CRM Port	Port number on which the CRM gateway is listening.
Log Mode	<p>Defines the characteristics of the APPC sessions that the CRM gateway establishes across a CRM Link.</p> <p>The characteristics of different logon modes are tailored to support different types of applications. For example long-running batch applications might use different logon modes than short-lived online applications. Logon modes can define different logical unit protocols, classes of service, packet sizes, and pacing algorithms.</p> <p>Use the same logon mode for the CRM gateway logical unit and the back-end application logical unit.</p>

Parameter	Description
Remote LU	LU of CICS.
Max Sync Level	Enter 1 to obtain non-transactional connections. Enter 2 for transactional connections.
Maximum Sessions	The number of maximum sessions that are allowed in the allocated pool. This value determines the number of concurrent requests that can be active at any given time. The CRM and the back end system must configure their maximum sessions to the same value.
Min Wins	<p>The CRM gateway and the back-end system are pre-allocated with a number of sessions for their use. This value is referred to as <i>minimum winner</i> sessions. The owner of these pre-allocated sessions has priority for its winner sessions; however, they may be reassigned depending on system load.</p> <p>The total of the minimum winner sessions for both sides must be less than or equal to the maximum session value. If the total is higher than the maximum session value, you will be unable to activate the link.</p> <p>For best results in determining minimum sessions, evaluate the number of sessions that are required for the CRM and for the back-end system to support concurrent requests.</p>

**Note:** The following restrictions apply when using CRM gateway connections with the iWay Transaction Adapter for CICS:

- The referenced CRM gateway must be configured to identify itself as CRM1. This is specified in the JCL that launches the CRM gateway.
  - Each adapter CRM gateway connection Host and Port must be unique. As a result, it is not possible to have two adapter connections to the same CRM gateway instance.
9. Click *Finish*.

The newly created connection, CRM1, appears as a node under the CICS service adapter. The configuration information is stored in the repository for the configuration you defined during installation.

**Procedure: How to Connect to a Defined CICS Target**

To connect to a defined CICS target:

1. In the left pane, expand the *iWay Adapters* node.
2. Expand the *CICS* node.
3. Click the target name under the CICS node, for example, *CICS\_Connection*.
4. Move your pointer over *Operations* and select *Connect*.

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

5. Verify your connection parameters. If required, provide the password.
6. Click *OK*.

The x icon disappears, indicating that the node is connected, as shown in the following image.



**Managing a Connection to CICS**

To manage CICS connections, you can:

- Disconnect from a connection that is not currently in use.

Although you can maintain multiple open connections to different transaction processing systems, it is recommended to disconnect from connections not in use.

- Edit a connection to change its properties.
- Delete a connection that is no longer required.

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

To disconnect from a connection to CICS:

1. In the left pane, expand the *iWay Adapters* node.
2. Expand the *CICS* node.
3. Click the connection, for example, *CICS\_Connection*, move your pointer over *Operations*, and select *Disconnect*.

Disconnecting from CICS drops the connection with CICS, but the node remains.

The x icon appears, indicating that the node is disconnected, as shown in the following image.



### **Procedure:** How to Edit a Connection to CICS

To edit a connection to CICS:

1. In the left pane of iWay Explorer, expand the *iWay Adapters* node.
2. Expand the *CICS* node and select the defined target you want to edit, for example, *CICS\_Connection*.
3. In the right pane, move the pointer over *Operations* and select *Edit*.

The following image shows the Edit pane that opens on the right containing three fields (Target Name, Description, and Target Type) and two active buttons (Next and Cancel).

**Edit CICS target CICS\_Connection**

---

Targets represent configured connections to instances of backend systems. Choose a name and description for the new target that you wish to create.

Target Name:

Description:

Target Type:

4. Modify the target information as required and then click *Next*.

The Set connection info pane opens in the right pane containing the Connection, Region, and Advanced tabs.

5. Modify the information as required and then, click *Finish*.

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

To delete a connection to CICS:

1. In the left pane, expand the *iWay Adapters* node.
2. Expand the *CICS* node.
3. Click the connection, for example, *CICS\_Connection*, move your pointer over *Operations*, and select *Delete*.

A message appears, prompting you to confirm the deletion of the node.

4. Click *OK*.

The node disappears from the list of available connections.

The following topics describe how to use iWay Explorer (Java Servlet) 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 explains how to use the generated schemas to create iWay Business Services, which expose functionality as web services.

**In this chapter:**

- [Creating an Adapter Transaction](#)
  - [Creating Schemas for an Adapter Transaction](#)
  - [Understanding iWay Business Services](#)
- 

### Creating an Adapter Transaction

After you create a connection to CICS, you can add adapter transactions using iWay Explorer (Java Servlet). 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 is automatically added 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 [Sample Requests, Schemas, and COBOL File Descriptions](#) on page 111.

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 <CommArea> tag, which can be a maximum of 32,500 bytes.

The generic response schema is constructed from the data received from CICS. If the <CommArea> 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 [How to Create an Adapter Transaction](#) on page 33. 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 IWAYSRV0, the location is "CICS/Transactions/IWAYSRV0".

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 [Sample Requests, Schemas, and COBOL File Descriptions](#) on page 111.

### Sample Program IWAYSRV0

iWay Software supplies the IWAYSRV0 and IWAYIVP programs with the adapter. This document uses the IWAYSRV0 program for illustration purposes and as a reference for the adapter. IWAYSRV0 is an example of a program that returns one of two possible record layouts depending on what is passed in the request.

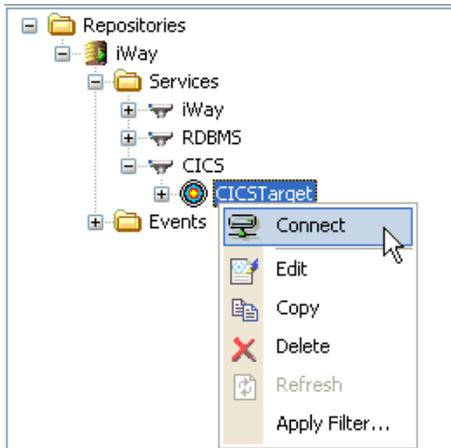
- ❑ If the value for the field COMMAND is 'SHORT', the program returns 40 bytes of data.
- ❑ If the value for the field COMMAND is 'LONG', the program returns 60 bytes of data.

Sample request documents are in [Sample Requests, Schemas, and COBOL File Descriptions](#) on page 111, with a sample response schema for the IWAYSRV0 program. You specify the output as explained in [Creating an Adapter Transaction](#) on page 31. 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. For more information, see [Event Processing With the CICS Adapter](#) on page 51.

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

Perform the following steps to create an adapter transaction.

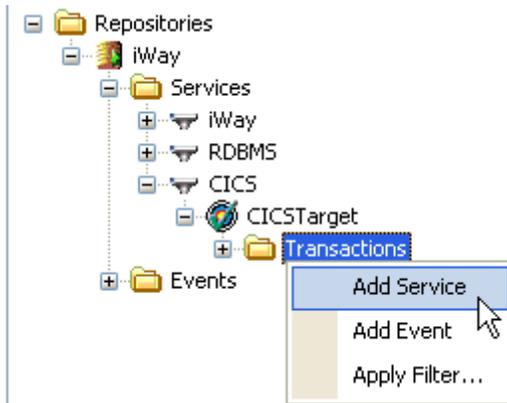
1. Expand the CICS adapter node and connect to an available CICS target (for example, CICSTarget).



The Transaction node appears under the connected target, as shown in the following image.



2. Right-click the *Transactions* node and select *Add Service* from the context menu.



The Add Service dialog box opens and displays two tabs (General and Output COBOL Data Description). The following image shows the parameters in the General tab that will enable you to map the COBOL descriptions for the CICS transaction.

**Interaction - Add Service**

**Run interaction**  
Enter all information on each step for running the interaction. Fields marked with an asterisk are required.

General | Output COBOL Data Description

Property	Value
*Node Name	IWAYSRV0
*Program Name	IWAYSRV0
Input COBOL Data Description	C:\Program Files\iway7\etc\samples
Size of COMMAREA	60
Natural Library	
Use data structure information from	
Accept multiple records in COMMARI	true

Update Cancel

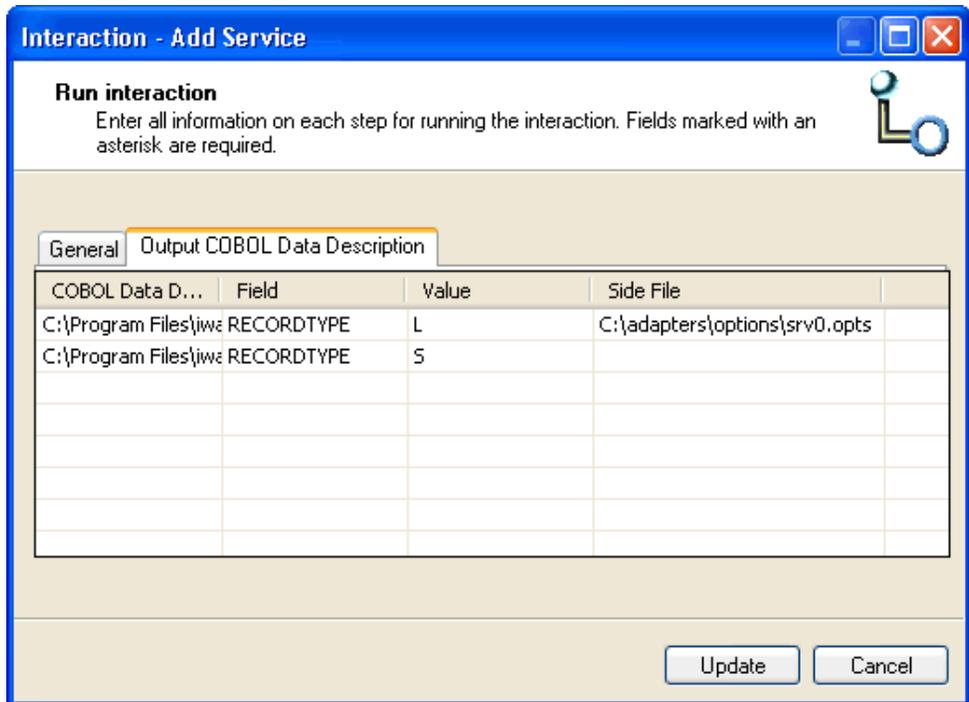
- To map the COBOL descriptions for the CICS transaction, type values for the parameters, type values for the parameters in the General tab, as defined in the following table.

Field	Description
Node Name	Name of the adapter transaction you are creating, for example, IWAYSRV0. Use this name in the <Transaction location="..."> attribute.
Program Name	Name of the program to be called in CICS, for example, IWAYSRV0. The IWAYSRV0 program appears in <a href="#">Sample CICS Programs</a> on page 119.

Field	Description
Input COBOL Data Description	<p>Location of the COBOL description that describes the COMMAREA of the CICS program to execute.</p> <p>It is 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.</p>
Size of COMMAREA	<p>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 IWAYSRV0, specify 60.</p>
Natural Library	<p>Run-time location of the Natural program to execute.</p> <p>Specify a value for this field only if the adapter is expected to execute Adabas/Natural programs.</p>
Use data structure information from COBOL	<p>When this parameter is selected, the adapter creates request and response schemas that reflect COBOL group levels (for example, 05, 10, 20, and so on). The COBOL grouping is reflected in the XML request and response schemas.</p> <p>You must select this parameter when COBOL input or output descriptions contain the COBOL OCCURS or REDEFINES statement.</p> <p><b>Note:</b> When this parameter is selected and there is an OCCURS COBOL statement, you cannot test run an adapter transaction. iWay Explorer returns an "OCCURS in COBOL Data Description" error message.</p>
Accept multiple records in COMMAREA	<p>When this parameter is selected, multiple COMMAREA records are read by the PreParser. Otherwise, any data in excess of the COBOL definition is truncated.</p>

**Important:** When connecting to a remote server, the location path of the COBOL description must match the operating system path of the machine on which the CICS adapter has been installed. For example, d:\iWay7\Cobol\ is a Windows path, whereas /iWay7/Cobol/ is a UNIX path.

4. Click the *Output COBOL Data Description* tab, as shown in the following image.



The Output COBOL Data Description tab allows you to specify the path that corresponds to the message you want returned from the CICS program. This tab contains the following columns:

- COBOL Data Description
- DD Field
- Value
- Side File

For more information on side files, see [Side File Support](#) on page 38.

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.

iWay 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 IWAYSRVO\_IN.CBL populates the COMMAREA field called COMMAND. Depending on program logic, iWay Explorer creates the correct response schema.

- Inbound COBOL - IWAYSRVO\_IN.CBL
- Outbound COBOL (1) - IWAYSRVO\_OUT\_L.CBL
  - Field:** RECORDTYPE
  - Value:** L
- Outbound COBOL (2) - IWAYSRVO\_OUT\_S.CBL
  - Field:** RECORDTYPE
  - Value:** S

The IWAYSRVO\_OUT\_L and IWAYSRVO\_OUT\_S COBOL descriptions appear in [Sample Requests, Schemas, and COBOL File Descriptions](#) on page 111.

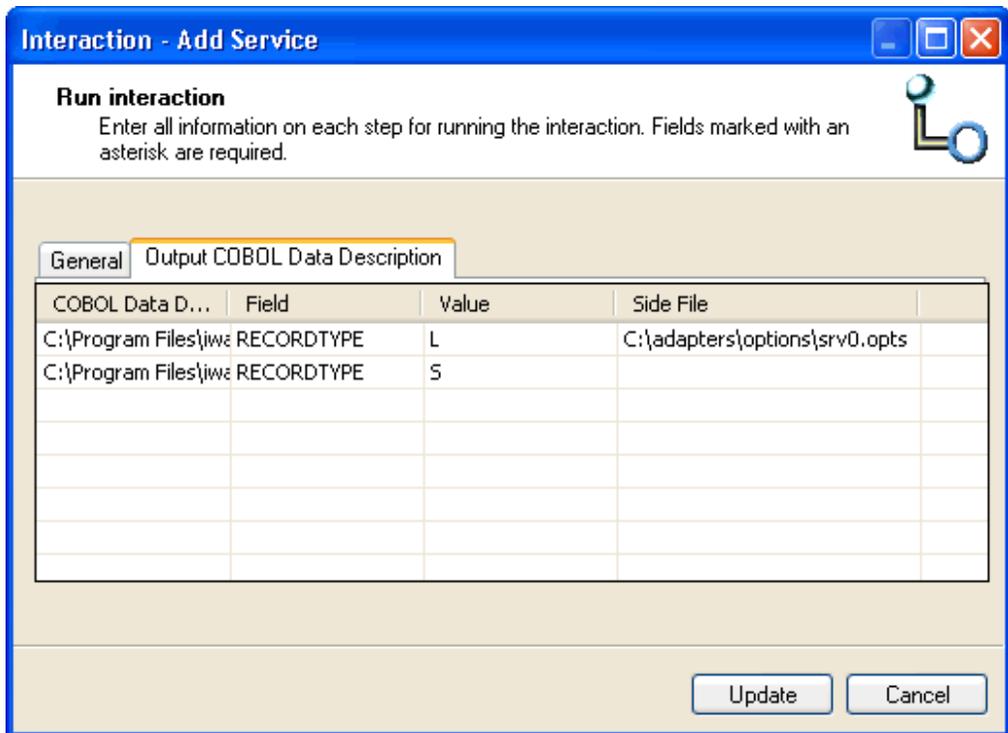
5. Click *Update* when you have finished configuring all the parameters.

The new CICS transaction, for example, IWAYSRVO, is added under the Transactions node for the current target.

### Side File Support

A side file is an XML file that contains data handling options for COBOL copybook fields. The format of the side file is described in the *iWay Service Manager Component Reference Guide*. Refer to the *Legacy Record to XML Converter Service (com.ibi.agents.XDLegacyRecordToXMLAgent)* section in that guide.

The following image shows the Output COBOL Data Description tab in the Add Service dialog for the iWay Transaction Adapter for CICS. A side file is specified in the Side File column.



You can specify a side file by:

- Explicit specification.** One side file may be specified for each output COBOL copybook. If specified, the contents of the side file are persisted along with the rest of the transaction information.
- Implicit specification.** If the directory containing the COBOL copybook also contains a file with the same name as the copybook, and *\_option* is appended, then that file is used as the side file and persisted. For example, if the COBOL copybook is *x.cob*, and the directory also contains a file *x.cob\_option*, then *x.cob\_option* is assumed to be the side file for the *x.cob* copybook, and will be persisted.
- No specification.** If a side file is not specified either explicitly or implicitly, then no side file is used.

## COBOL Descriptions for Input and Output Communications

The following are consideration iBSPs for COBOL descriptions for input and output communications. You must use the following syntax for binary, packed, and float fields for the COBOL descriptions for the adapter transaction input and output formats.

For a binary field:

```
05 BINARY-FIELD          PIC S9(n) COMP.
```

For a packed-decimal field:

```
05 PACKED-FIELD        PIC S9(n) COMP-3.
```

**Note:** Underscores are not supported in COBOL descriptions.

## Modifying COBOL DD Field Definitions

Using iWay Explorer, you can indicate that alpha type fields derived from COBOL DD input will be represented in XML by hex character strings. Specifically, this feature allows you to change the mapping for PIC X fields from XML type “string” to type “hexBinary”. As a result, arbitrary binary data can be transmitted in an XML supported format to CICS applications. This feature is useful when sending flag bits.

The following is an input example:

```
FLAG1 PIC X
```

```
<FLAG>02</FLAG> to transmit byte X'02'
```

The following is an output example:

```
CHARS PIC X 4
```

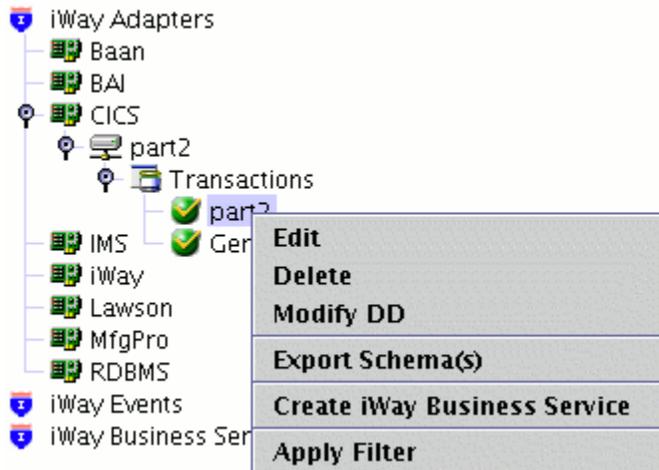
```
AAAA is returned as <CHARS>C1C1C1C1</CHARS>
```

### **Procedure:** How to Modify COBOL DD Field Definitions

To modify COBOL DD field definitions:

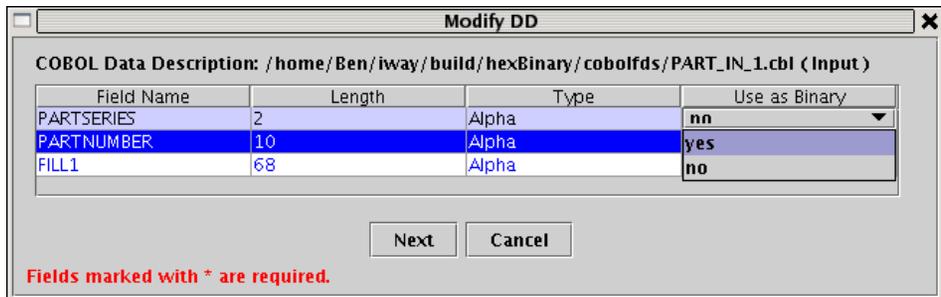
1. Open iWay Explorer and connect to your CICS target.

- Expand the *Transactions* node and select a defined transaction.



- Right-click the transaction and select *Modify DD* from the menu.

A sequence of Modify DD dialog boxes open representing the input and output record layouts for the transaction. Each layout includes the Use as Binary column, which allows you to modify an Alpha field.



- Select the Alpha field in the table that you want to use as Binary.
- Click the *Use as Binary* column and select *yes* or *no* from the drop-down list.
- Click *Next* to browse through the remaining input and output record layouts for the transaction.
- Click *Done* when you are finished to save the Alpha field modifications you made.

Field Name	Length	Type	Decimal Pl...	Use as Bin...
PARTSERIES	2	Alpha		0 yes
PARTNUM...	10	Alpha		0 no
FILL1	68	Alpha		0 no

HexBinary support is available for input and output transaction records. In particular, input RecType fields can also be defined with binary 'Value' arguments. In this case, the 'Value' data must be specified as the hexadecimal representation of the expected field.

8. Right-click the transaction in the left pane and select *Edit* from the menu.

The Edit dialog box opens.

**Edit**

Node Name\*

Program Name\*

Cobol DD for Input

Cobol DD for Output

COBOL Data Description	DD Field	Value
/home/Ben/iway/build/hexBina...	PARTSERIES	FOF1
/home/Ben/iway/build/hexBina...	PARTSERIES	FOF3

Convert Binary Zeros to:

Transaction has no reply

No Reply Wait time (TCP/IP only)

Maximum buffer size for retrieval

LTERM

Use data structure information from COBOL

**Fields marked with \* are required.**

**Note:** Hexadecimal values must be specified before any modifications to Alpha fields are performed. This is required because the Edit dialog box only saves dialog input when you click Update. As a result, the internal COBOL DD representation to be recreated and any prior Alpha field modifications would be lost.

## Creating Schemas for an Adapter Transaction

iWay Explorer stores the schemas it creates in subdirectories under the iWay home directory of the machine where it is installed. The exact location of the schemas differs depending on whether you deploy iWay Explorer with an iBSP.

When the adapter is used with an iBSP configuration, iWay Explorer stores the schemas under a \schemas subdirectory of the iWay home directory, for example,

```
C:\Program Files\iway7\config\base\wsdl\schemas\service\CICS
\CICS_Connection
```

where:

*CICS\_Connection*

Is the name of the connection to the CICS system as defined in iWay Explorer. Under this directory, iWay Explorer creates subdirectories containing schemas.

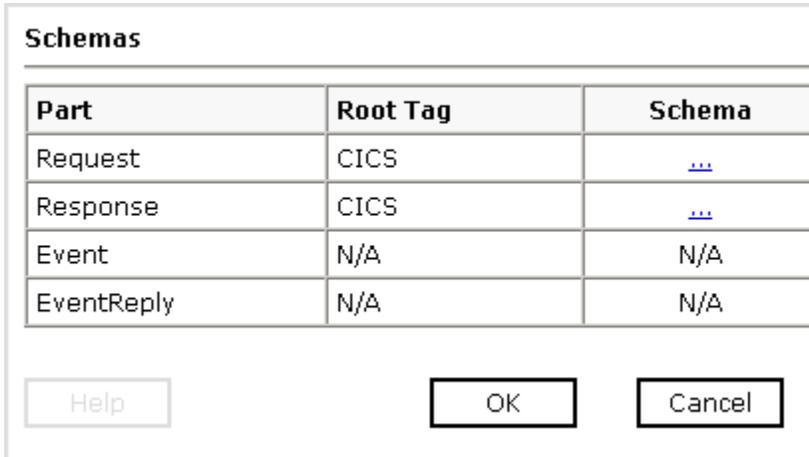
### **Procedure:** How to Create Schemas for an Adapter Transaction

To create schemas for an adapter transaction:

1. In the left pane, select the transaction for which you want to generate schemas.
2. In the right pane, move the pointer over *Operations* and select *Generate Schema*.

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 [Sample Requests, Schemas, and COBOL File Descriptions](#) on page 111.

The Schemas table appears in the right pane. The following image shows the Schemas pane, which consists of four rows (Request, Response, Event, EventReply) and three columns (Part, Root Tag and Schema). Only the Request and Response rows are applicable.



Part	Root Tag	Schema
Request	CICS	...
Response	CICS	...
Event	N/A	N/A
EventReply	N/A	N/A

Help      OK      Cancel

- a. To view the request schema, click the ellipsis symbol that is located in the Schema column of the Request row.
- b. To view the response schema, click the ellipsis symbol that is located in the Schema column of the Response row.
3. To exit the Schemas pane, click OK.

## Understanding iWay Business Services

iWay Explorer provides developers with a simple, consistent mechanism for extending the capabilities of the adapter. The iWay Business Services Provider (iBSP) 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 a Web 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 a web service using iWay Explorer.

### **Procedure:** How to Create a Web Service

To create a web service:

1. Click the *Iway Adapters* tab.

The iWay Adapters window opens.

- a. In the left pane, expand the *CICS* node.
- b. Connect to a CICS target, for example, *CICS\_Connection*.
- c. Expand the node to which you connected.

The Transaction node appears under the connected node.

2. Click *Transactions* and then select the transaction for which you want to create a web service.
3. In the right pane, move your cursor over *Operations* and select *Create iWay Business Services*.

The Create Web Service pane opens on the right, where you enter information that is specific to the web service you are defining, as shown in the following image.

**Create Web Service for Generic\_Transaction**

Service Name:

Description:

License: 

- production
- test

Help   < Back   Next >   Cancel

- 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 or more license codes to assign to the web service. To select more than one, hold down the *Ctrl* key and click the licenses.
4. Click *Next*.

Another pane with the Method Name and Description fields opens, where you enter information that is specific to the method you are defining, as shown in the following image.

**Create Web Service for Generic\_Transaction**

---

Method Name:

Description:

- 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.
5. Click *Finish*.

The iWay Business Services Provider tab opens. The web service is created and published to the iWay Business Services Provider. iWay Explorer displays the newly created web service under the iWay Business Services folder.

## Testing the Web Service

After you create a business service, you can test it to ensure that it functions properly. iWay provides a test tool for testing the business service.

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

To test the web service:

1. If you are not on the iWay Business Services tab of iWay Explorer, click the tab to access business services.
  - a. If it is not expanded, expand the list of business services under iWay Business Services.
  - b. Expand the *Services* node.

2. Select the name of the business service you want to test.

The business service name appears as a link in the right pane.

3. In the right pane, click the named business services link.

The test option appears in the right pane.

4. In the input xml field, either type a sample XML document that queries the service, or browse to the location of an XML instance and click *Upload*.

The following is an example of an XML document that queries the service.

```
<?xml version="1.0" encoding="UTF-8" ?>
<CICS>
  <Transaction location="/CICS/Transaction/IWAYSAMP">
    <CommArea>
      <COMMAND>SHORT</COMMAND>
    </CommArea>
  </Transaction>
</CICS>

<?xml version="1.0" encoding="UTF-8" ?>
<CICS>
  <Transaction location="/CICS/Transaction/IWAYSAMP">
    <CommArea>
      <COMMAND>LONG</COMMAND>
    </CommArea>
  </Transaction>
</CICS>
```

5. Click *Invoke*.

The result appears in the right pane.

### Generating WSDL From a Web Service

The Web Service Description Language (WSDL) file is an XML file that describes the web service documents and provides access to the service, such as iWay run-time environment.

#### ***Procedure:*** How to Generate WSDL From a Web Service

To generate WSDL from a web service:

1. If you are not already on the iWay Business Services tab, click the tab to access business services.
2. In the left pane, expand the list of services to display the iWay Business Services for which you want to generate WSDL.
3. Select the business service.

The link for the service appears in the right pane.

- a. Right-click the *Service Description* link and choose *Save Target As*.
- b. Choose a location for the file and specify *.wsdl* for the extension.

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

4. Click *Save*.

### **Example:** Viewing WSDL Generated from a Web Service

After generating a WSDL file from the *IWAYSrv0.ibs* serialized object, the *IWAYSrv0.wsdl* file looks similar to the following image.

```
- <definitions xmlns:tns="urn:schemas-iwaysoftware-
com:iwse" targetNamespace="urn:schemas-
iwaysoftware-com:iwse"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/enc"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:m11="urn:iwaysoftware:ibse:jul2003:IWAYSrv0 :r
xmlns:tm="http://microsoft.com/wsdl/mime/textMatch
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:m1="urn:iwaysoftware:ibse:jul2003:IWAYSrv0 "
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
- <types>
- <xs:schema targetNamespace="urn:schemas-
iwaysoftware-com:iwse"
elementFormDefault="qualified">
- <xs:element name="ibsinfo">
- <xs:complexType>
- <xs:sequence>
<xs:element type="xs:string"
name="service" />
<xs:element type="xs:string"
name="method" />
<xs:element type="xs:string"
name="license" />
<xs:element type="xs:string"
minOccurs="0"
name="disposition" />
```

## Identity Propagation

If you test or execute a web service using a third party XML editor, for example XMLSPY, the Username and Password values that you specify in the SOAP header must be valid and are used to connect to CICS. The user name and password values that you provided for CICS during target creation using iWay Explorer are overwritten for this web service request. The following is a sample SOAP header that is included in the WSDL file for a web service:

```
<SOAP-ENV:Header>
  <m:ibsinfos xmlns:m="urn:schemas-iwaysoftware-com:iwse">
    <m:service>String</m:service>
    <m:method>String</m:method>
    <m:license>String</m:license>
    <m:disposition>String</m:disposition>
    <m:Username>String</m:Username>
    <m>Password>String</m>Password>
    <m:language>String</m:language>
  </m:ibsinfos>
</SOAP-ENV:Header>
```

**Note:** You can remove the following tags from the SOAP header, as they are not required.

```
<m:disposition>String</m:disposition>
<m:language>String</m:language>
```

## Event Processing With the CICS Adapter

---

The following section describes how the iWay Transaction Adapter for CICS can be used to configure and use Events.

### In this chapter:

- [Understanding CICS Events](#)
  - [Configuring CICS Events](#)
  - [Creating and Modifying an Event Port](#)
  - [Creating and Modifying a Channel](#)
  - [Testing CICS Events](#)
  - [Running CICS Events](#)
- 

### Understanding CICS Events

A CICS Event is the processing defined on a particular message received from CICS. A CICS program sends a message to communicate that a specific event occurs, for example, when an inventory level crosses a threshold. In the most common use case, an EBCDIC COBOL Copybook formatted record is received, converted to an XML document, and delivered to a queue or program.

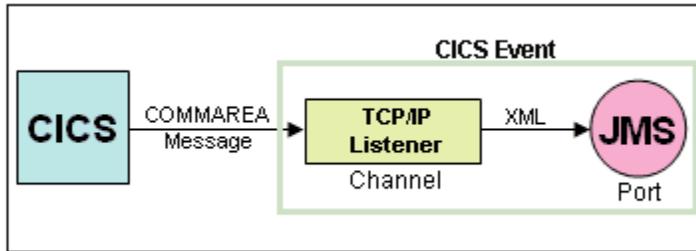
All processing details for a message are configured under a **channel**, which include:

- Communications parameters with CICS, for example, TCP port
- Message layouts defined in COBOL Copybooks
- Assigned **ports** to which the message is routed as an XML document

A CICS Event can be configured as REQUEST only, in which the Event consumes a message, or REQUEST/RESPONSE, in which the Event consumes a message and returns a reply message.

The following diagrams and sequences illustrate REQUEST and REQUEST/RESPONSE scenarios:

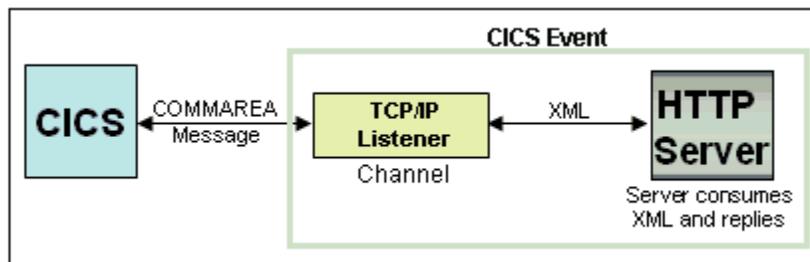
## REQUEST



The following sequence outlines how request messages are processed.

1. A message is created by a CICS user program.
2. The message is sent via TCP to a channel (TCP port).
3. The channel receives the message.
4. If the channel is configured to work with Copybook formatted messages, it converts the message to XML.
5. The channel routes the message to one or more destination ports for further processing, for example, to a JMS queue.

## REQUEST/RESPONSE



The following sequence outlines how request/response messages are processed.

1. A message is created by a CICS user program.
2. The message is sent via TCP to a channel (TCP port).
3. The channel receives the message.
4. If the channel is configured to work with Copybook formatted messages, it converts the message to XML.
5. The channel routes a message to a destination port for further processing, for example, an HTTP request.

6. The port takes an XML document as input and returns an XML document as output to the channel. For channels configured with COBOL Copybook formatted messages, these documents conform to the XML schemas generated from these Copybooks using iWay Explorer.
7. If the channel is configured to work with Copybook formatted messages, it converts the XML into Copybook format.
8. The channel returns the reply message to the CICS program.

## Message Format

A message can be a Copybook formatted record or an XML document.

Copybook formatted messages have the following properties:

- The request message must be prefixed by a 4-byte message length.
- The Reply messages (if returned) will be prefixed by 4-byte message length.
- Messages correspond to Copybooks configured as a PreParser and PreEmitter on the channel.

XML formatted messages have the following properties:

- Request and Reply messages (when returned) are XML documents, conforming to the input and output document types of the backend process.

The following sample event programs are included with the product:

- iwayevt0 - COBOL, copybook described data, request only
- iwayevt1 - C, XML, request/response
- iwayevt2 - COBOL, copybook described data, request/response

For more information, see [Sample CICS Programs](#) on page 119.

## Supported Environments

CICS Event handling is supported under Servlet iBSP.

iWay Service Manager (iSM) does not support CICS Events. However, similar event handling capabilities can be achieved by configuring a CICS PreParser and PreEmitter. For more information, see the *iWay Service Manager User's Guide*.

## Configuring CICS Events

CICS Events are configured using a design time tool such as iWay Explorer. This section will use the iwayevt0 sample program with a File port as an example.

To configure a CICS Event, you must:

1. Create metadata in the form of an XML schema if you are using COBOL Copybooks.
2. Create one or more ports for use as message endpoints.
3. Create a channel to receive messages and associate it with ports and COBOL Copybooks.

**Procedure:** **How to Create XML Schemas from COBOL Copybook Metadata**

This section describes how to create XML schemas for Copybook formatted messages.

**Note:** You can skip this section If your message is in XML format.

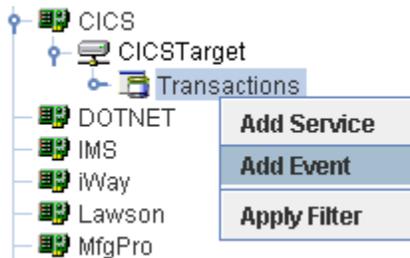
1. Connect to your iBSP.
2. Create and connect to a CICS target.



For more information on creating and connecting to targets, see [Configuring the iWay Transaction Adapter for CICS](#) on page 19.

**Note:** The connectivity being used here is outbound connectivity (CICS adapter), and not strictly necessary for Events.

3. Right-click the Transactions node, and select *Add Event*.

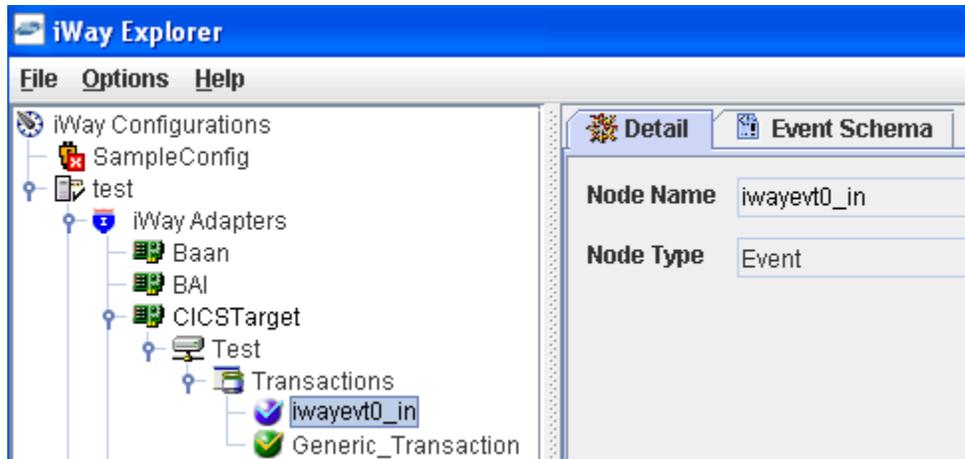


The Add Event dialog box opens.

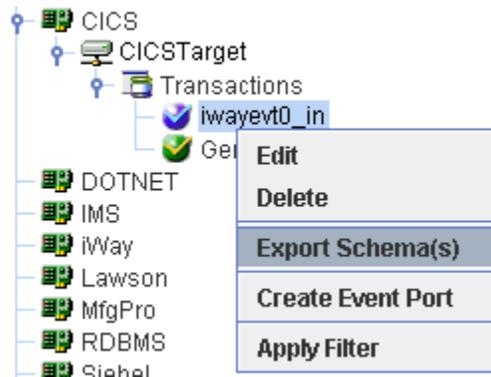
4. Provide the appropriate configuration information as described in the following table and click *Update*.

Field	Description
Node Name	An arbitrary descriptive name for the CICS Event.
Input COBOL Data Description	The location of the COBOL Copybook that describes the input event record.
Use data structure information from COBOL	Select this option if you want to include group names in the schema.
Codepage	The code page used by the input data.

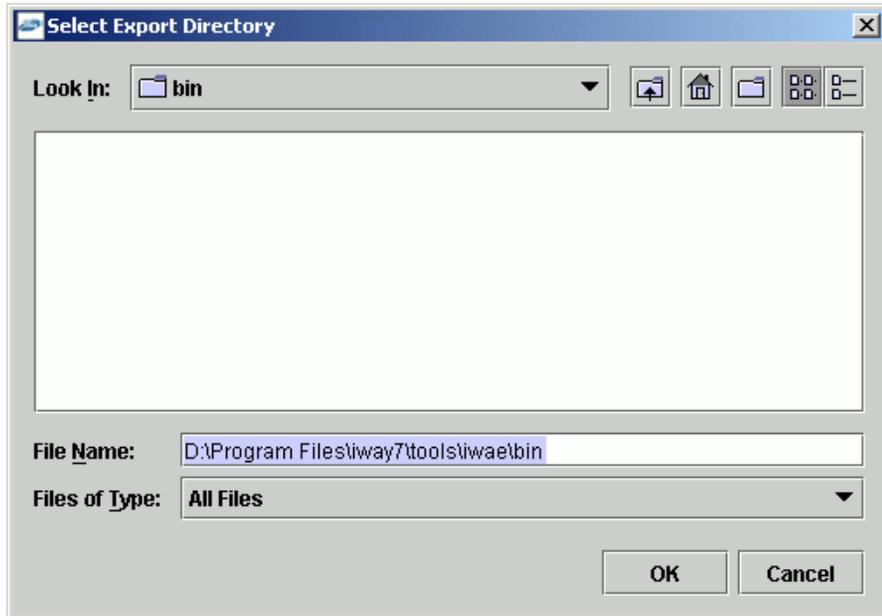
A new Event node is added in the left pane.



5. Right-click the Event node, for example, CICSEvent, and select *Export Schema(s)*.



The Select Export Directory dialog box opens.



6. Provide a file name for the schema and select a destination for future use.  
To view a sample XML schema document that is provided for your reference, see [XML Schema Document](#) on page 78.
7. Click *OK*.
8. Repeat steps 2 through 7 for the outbound message if the request returns a reply.

## Creating and Modifying an Event Port

After you have created the schema(s) for any COBOL Copybook metadata, you must create a port. The port represents the processing disposition of the message from CICS. A CICS Event must have at least one port. The types of port formats that are available depends on the configuration you are connected to in iWay Explorer.

For documents derived from Copybooks, ports should accept as input (and deliver as output) documents with the appropriate schemas.

The following dispositions are available when using iWay Explorer in conjunction with an iBSP deployment.

- File

- iBSP
- MSMQ
- JMSQ
- SOAP
- HTTP
- MQ Series

**Procedure: How to Create an Event Port for File**

To create an Event port for File:

1. In the left pane of iWay Explorer, select the *iWay Events* tab.
2. Expand the *CICS* node.
3. Select the *ports* node.

The following image shows the ports node highlighted in the left pane and the Operations menu options (Add a new port and Refresh) in the right pane.



4. Move the pointer over *Operations* and select *Add a new port*.

The Create New Port pane opens on the right, as shown in the following image.

**Create New Port**

---

Choose parameters of the port that you wish to create.

Port Name:

Description:

Disposition Protocol:

Disposition:

- a. In the Port Name field, type a name for the Event port.
- b. In the Description field, type a brief description for the port (optional).
- c. From the Disposition Protocol drop-down list, select *FILE*.
- d. In the Disposition field, specify a destination file to which the Event data is written.

When pointing iWay Explorer to an iBSP deployment, specify the destination file using the following format:

```
ifile://[location];errorTo=[pre-defined port name or another disposition url]
```

The following table lists and describes the parameters for the File disposition:

Parameter	Description
Location	Destination and file name of the document where Event data is written, for example, D:\in\x.txt
ErrorTo	Location to which error logs are sent. Optional. Predefined port name or another disposition URL. The URL must be complete, including the protocol.

5. Click OK.

The port appears under the ports node in the left pane. In the right pane, a table appears that summarizes the information associated with the Event port you created.

**Procedure: How to Create an Event Port for iBSP**

To create an Event port for iBSP:

1. In the left pane of iWay Explorer, select the *iWay Events* tab.
2. Expand the *CICS* node.
3. Select the *ports* node.
4. Move the pointer over *Operations* and select *Add a new port*.

The Create New Port pane opens on the right, as shown in the following image.

**Create New Port**

Choose parameters of the port that you wish to create.

Port Name:

Description:

Disposition Protocol:

Disposition:

- a. In the Port Name field, type a name for the Event port.
- b. In the Description field, type a brief description for the port (optional).
- c. From the Disposition Protocol drop-down list, select *iBSE*.
- d. In the Disposition field, enter an iBSP destination in the following format:

```
ibse:[svcName].[mthName];responseTo=[pre-defined port name or  
another disposition url];errorTo=[pre-defined port name or another  
disposition url]
```

The following table lists and describes the parameters for the iBSP disposition:

Parameter	Description
svcName	Name of the service created with iBSP.
mthName	Name of the method created for the web service.
responseTo	Location to which responses are posted. Optional. Predefined port name or another disposition URL. The URL must be complete, including the protocol.
errorTo	Location to which error logs are sent. Optional. Predefined port name or another disposition URL. The URL must be complete, including the protocol.

5. Click *OK*.

The port appears under the ports node in the left pane. In the right pane, a table appears that summarizes the information associated with the Event port you created.

### ***Procedure:* How to Create an Event Port for MSMQ**

To create an Event port for MSMQ:

1. In the left pane of iWay Explorer, select the *iWay Events* tab.
2. Expand the *CICS* node.
3. Select the *ports* node.
4. Move the pointer over *Operations* and select *Add a new port*.

The Create New Port pane opens on the right, as shown in the following image.

- a. In the Port Name field, type a name for the Event port.
- b. In the Description field, type a brief description for the port (optional).
- c. From the Disposition Protocol drop-down list, select *MSMQ*.
- d. In the Disposition field, enter an MSMQ destination in the following format:

`msmq://[machineName]/private$/qName;errorTo=[pre-defined port name or another disposition url]`

**Note:** This syntax is for a private queue. Private queues are queues that are not published in Active Directory. They appear only on the local computer that contains them. Private queues are accessible only by Message Queuing applications that recognize the full path name or format name of the queue.

The following table lists and describes the parameters for the MSMQ disposition.

Parameter	Description
machineName	Machine name where the Microsoft Queuing system is running.
qName	Name of the private queue where messages are placed.

Parameter	Description
errorTo	Location to which error logs are sent. Optional. Predefined port name or another disposition URL. The URL must be complete, including the protocol.

- Click *OK*.

The port appears under the ports node in the left pane. In the right pane, a table appears that summarizes the information associated with the Event port you created.

You are now ready to associate the Event port with a channel. For more information, see [How to Create a Channel](#) on page 71.

### **Procedure:** How to Create an Event Port for JMSQ

To create an Event port for JMSQ:

- In the left pane of iWay Explorer, select the *iWay Events* tab.
- Expand the *CICS* node.
- Select the *ports* node.
- Move the pointer over *Operations* and select *Add a new port*.

The Create New Port pane opens on the right, as shown in the following image.

**Create New Port**

Choose parameters of the port that you wish to create.

Port Name:

Description:

Disposition Protocol:

Disposition:

- In the Port Name field, type a name for the Event port.

- b. In the Description field, type a brief description for the port (optional).
- c. From the Disposition Protocol drop-down list, select *JMSQ*.
- d. In the Disposition field, enter a JMS destination.

When pointing iWay Explorer to an iBSP deployment, use the following format.

```
jmsq:[myQueueName]@[myQueueFac];jndiurl=[myurl];
jndifactory=[myfactory];user=[user];password=[xxx];
errorTo=[pre-defined port name or another disposition url]
```

The following table lists and describes the parameters for the JMSQ disposition.

Parameter	Description
myQueueName or jmsqueue	JNDI name of a queue to which Events are emitted.
myQueueFac or jmsfactory	Resource that contains information about the JMS Server.
jndiurl	URL to use to contact the JNDI provider. The syntax of this URL depends on the JNDI provider being used. This value corresponds to the standard JNDI property, <code>java.naming.provider.url</code>
jndifactory	Is JNDI context.INITIAL_CONTEXT_FACTORY and is provided by the JNDI service provider.
user	Valid user name required to access a JMS server.
password	Valid password required to access a JMS server.
errorTo	Location to which error logs are sent. Optional.  Predefined port name or another disposition URL. The URL must be complete, including the protocol.

- 5. Click *OK*.

The port appears under the ports node in the left pane. In the right pane, a table appears that summarizes the information associated with the Event port you created.

You are now ready to associate the Event port with a channel. For more information, see [How to Create a Channel](#) on page 71.

**Procedure: How to Create a Port for SOAP**

To create an Event port for SOAP:

1. In the left pane of iWay Explorer, select the *iWay Events* tab.
2. Expand the *CICS* node.
3. Select the *ports* node.
4. Move the pointer over *Operations* and select *Add a new port*.

The Create New Port pane opens on the right, as shown in the following image.

**Create New Port**

Choose parameters of the port that you wish to create.

Port Name:

Description:

Disposition Protocol:

Disposition:

- a. In the Port Name field, type a name for the Event port.
- b. In the Description field, type a brief description for the port (optional).
- c. From the Disposition Protocol drop-down list, select *SOAP*.
- d. In the Disposition field, enter a SOAP destination in the following format.

```
soap:[wsdl-url];soapaction=[myaction];
method=[web service method];namespace=[namespace];
responseTo=[pre-defined port name or another disposition URL];
errorTo=[pre-defined port name or another disposition url]
```

The following table lists and describes the parameters for the SOAP disposition.

<b>Parameter</b>	<b>Description</b>
wsdl-url	<p>URL to the WSDL file that is required to create the SOAP message, for example:</p> <p><a href="http://localhost:7001/ibse/IBSEServlet/test/webservice.ibs?wsdl">http://localhost:7001/ibse/IBSEServlet/test/webservice.ibs?wsdl</a></p> <p>where:</p> <p><a href="#">webservice</a></p> <p>Is the name of the web service you created using iWay Explorer.</p> <p>This value can be found by navigating to the iWay Business Services tab and opening the Service Description link in a new window. The WSDL URL appears in the Address field.</p> <p>You can also open the WSDL file in a third party XML editor (for example, XMLSPY) and view the SOAP request settings to find this value.</p>
soap action	<p>Method that is called by the SOAP disposition. For example:</p> <p><a href="#">webservice.method@test@</a></p> <p>where:</p> <p><a href="#">webservice</a></p> <p>Is the name of the web service you created using iWay Explorer.</p> <p><a href="#">method</a></p> <p>Is the method being used.</p> <p><a href="#">test</a></p> <p>Is the license that is being used by the web service.</p> <p>This value can be found by navigating to the iWay Business Services tab, opening the Service Description link in a new window, and performing a search for soapAction.</p> <p>You can also open the WSDL file in a third party XML editor (for example, XMLSPY) and view the SOAP request settings to find this value.</p>

Parameter	Description
method	Web service method you are using. This value can be found in the WSDL file.
namespace	XML namespace you are using. This value can be found in the WSDL file.
response	Location to which responses are posted. Optional. Predefined port name or another disposition URL. The URL must be complete, including the protocol.
error	Location to which error logs are sent. Optional. Predefined port name or another disposition URL. The URL must be complete, including the protocol.

5. Click *OK*.

The port appears under the ports node in the left pane. In the right pane, a table appears that summarizes the information associated with the port you created.

You are now ready to associate the Event port with a channel. For more information, see [How to Create a Channel](#) on page 71.

### **Procedure:** How to Create an Event Port for HTTP

To create an Event port for HTTP:

1. In the left pane of iWay Explorer, select the *iWay Events* tab.
2. Expand the *CICS* node.
3. Select the *ports* node.
4. Move the pointer over *Operations* and select *Add a new port*.

The Create New Port pane opens on the right, as shown in the following image.

- a. In the Port Name field, type a name for the Event port.
- b. In the Description field, type a brief description for the port (optional).
- c. From the Disposition Protocol drop-down list, select *HTTP*.
- d. In the Disposition field, enter an HTTP destination.

When pointing iWay Explorer to an iBSP deployment, use the following format.

```
http://[myurl];responseTo=[pre-defined port name or another disposition url];
```

The following table lists and describes the parameters for the HTTP disposition when using an iBSP deployment.

Parameter	Description
myurl	URL target for the post operation, for example, <a href="http://myhost:1234/docroot">http://myhost:1234/docroot</a>
responseTo	Location to which responses are posted. Optional. Predefined port name or another disposition URL. The URL must be complete, including the protocol.

5. Click *OK*.

The port appears under the ports node in the left pane. In the right pane, a table appears that summarizes the information associated with the Event port you created.

You are now ready to associate the Event port with a channel. For more information, see [How to Create a Channel](#) on page 71.

### **Procedure: How to Create an Event Port for MQSeries**

To create an Event port for MQSeries:

1. In the left pane of iWay Explorer, select the *iWay Events* tab.
2. Expand the *CICS* node.
3. Select the *ports* node.
4. Move the pointer over *Operations* and select *Add a new port*.

The Create New Port pane opens on the right, as shown in the following image.

**Create New Port**

Choose parameters of the port that you wish to create.

Port Name:

Description:

Disposition Protocol:

Disposition:

- a. In the Port Name field, type a name for the Event port.
- b. In the Description field, type a brief description for the port (optional).
- c. From the Disposition Protocol drop-down list, select *MQ Series*.
- d. In the Disposition field, enter an MQSeries destination using the following format.

```
mqseries:[qManager]/[qName];host=[hostname];port=[port];
channel=[channelname];errorTo=[pre-defined port name or another
disposition url]
```

The following table lists and describes the parameters for the MQ Series disposition.

Parameter	Description
qManager	Name of the queue manager to which the server must connect.
qName or respqueue	Name of the queue where messages are placed.
host	Host on which the MQ server is located (MQ Client only).
port	Number to connect to an MQ server queue manager (MQ client only).
channel	Case-sensitive name of the channel that connects with the remote MQ server queue manager (MQ client only). SYSTEM.DEF.SVRCONN is the default channel name for MQSeries.
errorTo	Location to which error logs are sent. Optional.  Predefined port name or another disposition URL. The URL must be complete, including the protocol.

5. Click *OK*.

The port appears under the ports node in the left pane. In the right pane, a table appears that summarizes the information associated with the Event port you created.

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

To edit an Event port:

1. In the left pane of iWay Explorer, select the *iWay Events* tab.
2. Expand the *CICS* node.
3. Select the *ports* node.
4. Select the Event port you want to edit.
5. In the right pane, move the pointer over *Operations* and select *Edit*.

The Edit Port dialog box opens.

6. Make the required changes and click *OK*.

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

To delete an Event port:

1. In the left pane of iWay Explorer, select the *iWay Events* tab.
2. Expand the *CICS* node.
3. Select the *ports* node.
4. Select the Event port you want to delete.
5. In the right pane, move the pointer over *Operations* and select *Delete*.

A confirmation dialog box opens.

6. To delete the Event port you selected, click *OK*.

The Event port disappears from the list in the left pane.

**Creating and Modifying a Channel**

A channel is created to listen for the CICS Event and process the event document. A channel that is configured using the REQUEST or REQUEST\_ACK (Acknowledgment) synchronization type does not return data to the CICS application, and may be associated with one or more ports. Specifying multiple ports means that the document will be written to each port. A channel that uses the REQUEST\_RESPONSE synchronization type may only be associated with a single port. If the channel converts Copybook formatted request messages to XML, a PreParser must be configured. If the channel converts XML to Copybook formatted response messages, then a PreEmitter must be configured.

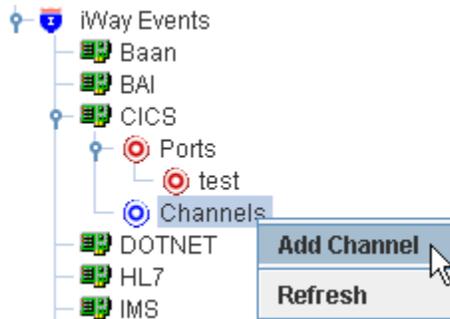
When using a REQUEST/RESPONSE configuration, input and output data must be either both Copybook or XML formatted.

**Procedure: How to Create a Channel**

To create a channel:

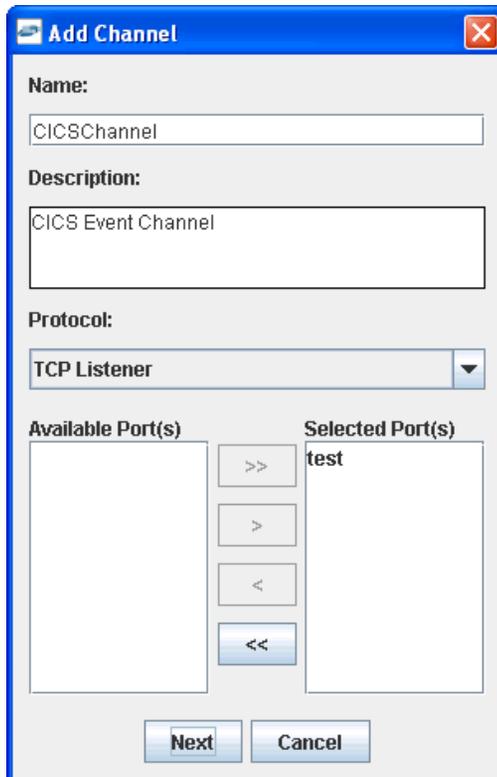
1. In the left pane of iWay Explorer, expand the *iWay Events* node.
2. Expand the *CICS* node.
3. Select the *Channels* node.

The following image shows the Channels node highlighted in the left pane.



4. Right-click the *Channels* node and select *Add Channel*.

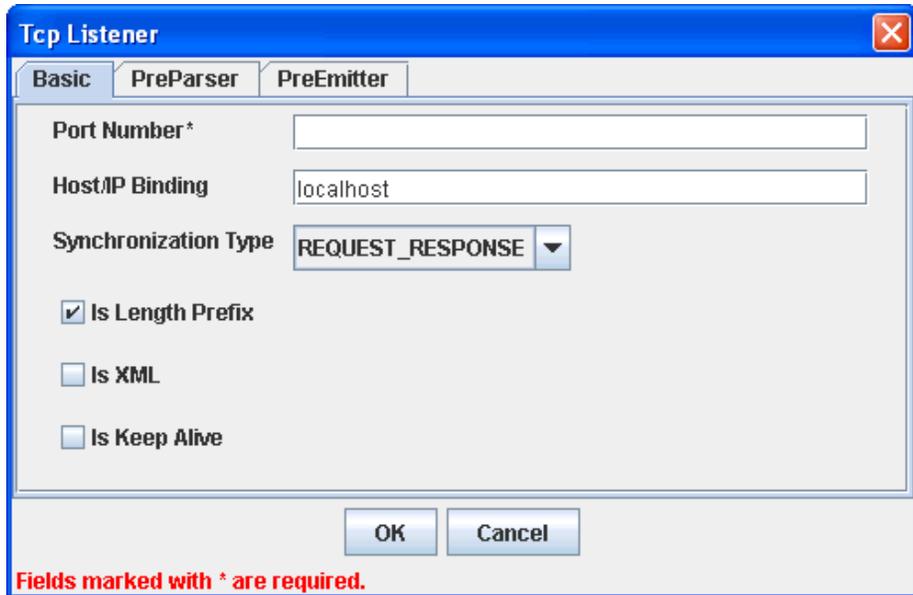
The Add Channel dialog box opens.



- a. In the Name field, type a descriptive name for the channel, for example, CICSChannel.
- b. In the Description field (optional), type a brief description for the channel, for example, CICS Event Channel.

- c. From the Protocol drop-down list, select a channel type. TCP Listener is the only type supported by the CICS adapter.
  - d. Select from the available ports the ones you want to associate with the channel.
5. Click *Next*.

The TCP Listener dialog box opens where you provide parameters to specify values for the protocol you will use with the channel, as shown in the following image.



**Tcp Listener**

Basic PreParser PreEmitter

Port Number\*

Host/IP Binding

Synchronization Type  ▼

Is Length Prefix

Is XML

Is Keep Alive

OK Cancel

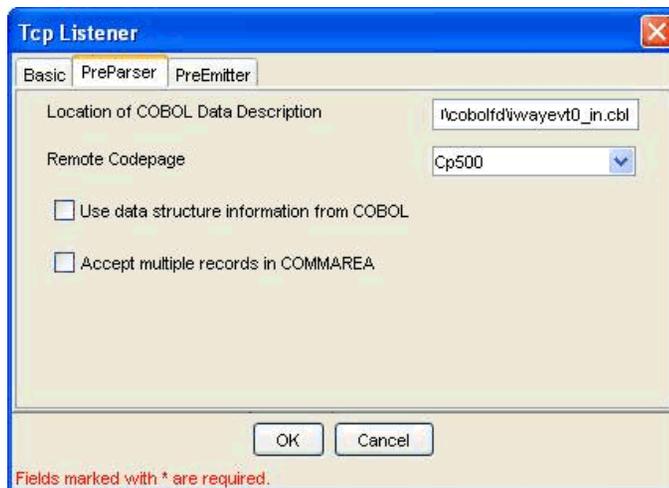
Fields marked with \* are required.

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

The following table lists and describes the properties in the Basic tab.

Property	Description
<b>Basic Tab</b>	
Port Number	Port number where your CICS connection node listens for Events generated by CICS.
Host/IP Binding	Name or IP address of the host computer where the CICS application region is running.

Property	Description
Synchronization Type	<p>Choose one of the following:</p> <ul style="list-style-type: none"> <li><input type="checkbox"/> If the Event application expects a reply sent back to it, select REQUEST_RESPONSE. Specify a PreEmitter.</li> <li><input type="checkbox"/> If the Event application expects a TCP/IP Acknowledgment (ACK), select REQUEST_ACK.</li> <li><input type="checkbox"/> If the Event application does not expect a reply, select REQUEST.</li> </ul>
Is Length Prefix	If the message is prefixed by a 4 byte binary length. Required for COBOL Copybook formatted messages. Optional for XML messages.
Is XML	For CICS Events that send back data in XML format. Do not specify a PreParser or PreEmitter.
Is Keep Alive	Maintains continuous communication between the Event transaction and the channel.



The following table lists and describes the properties in the PreParser tab.

Property	Description
Location of COBOL Data Description	Path to the COBOL description of the layout sent by CICS. For more information, see <a href="#">Sample Requests, Schemas, and COBOL File Descriptions</a> on page 111.
Remote Codepage	Select the code page from the drop-down menu. Cp500 is the default value.
Use data structure information from COBOL	When this parameter is selected, the Event creates request and response documents including COBOL group levels (for example, 05, 10, 20, and so on).  You must select this parameter when COBOL input or output descriptions contain the COBOL OCCURS or REDEFINES statement.
Accept multiple records in COMMAREA	When this parameter is selected, multiple COMMAREA records are read by the PreParser. Otherwise, any data in excess of the COBOL definition is truncated.

The screenshot shows the 'Tcp Listener' dialog box with the 'PreEmitter' tab selected. The 'Location of COBOL Data Description' field contains the path 'i\cobolfd\iwayevt0\_out.cbl'. The 'Remote Codepage' dropdown menu is set to 'Cp500'. The checkbox for 'Use data structure information from COBOL' is unchecked. The dialog includes 'OK' and 'Cancel' buttons and a red note at the bottom left: 'Fields marked with \* are required.'

The following table lists and describes the properties in the PreEmitter tab.

**Note:** Specify a PreEmitter only when using the REQUEST\_RESPONSE synchronization type.

Property	Description
Location of COBOL Data Description	Path to the COBOL description of the layout returned to CICS. For more information, see <a href="#">Sample Requests, Schemas, and COBOL File Descriptions</a> on page 111.
Remote Codepage	Select the code page from the drop-down menu. Cp500 is the default value.
Use data structure information from COBOL	When this parameter is selected, the Event creates request and response documents including COBOL group levels (for example, 05, 10, 20, and so on).  You must select this parameter when COBOL input or output descriptions contain the COBOL OCCURS or REDEFINES statement.

7. Click *OK*.

The following image shows the newly created CICSChannel 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.

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

To edit a channel:

1. In the left pane of iWay Explorer, select the *iWay Events* tab.
2. Expand the *CICS* node.
3. Select the *Channels* node.
4. Select the channel you want to edit.
5. Right-click the channel and select *Edit*.

The Edit Channel dialog box opens.

6. Make any required changes to the channel configuration.

**Procedure: How to Delete a Channel**

To delete a channel:

1. In the left pane of iWay Explorer, select the *iWay Events* tab.
2. Expand the *CICS* node.
3. Select the *Channels* node.
4. Select the channel you want to delete.
5. Right-click the channel and select *Delete*.

The channel disappears from the list in the left pane.

**Testing CICS Events**

After you have created a channel, you may test it by starting and stopping the channel.

**Procedure: How to Start a Channel**

To start a channel:

1. In the left pane of iWay Explorer, select the *iWay Events* tab.
2. Expand the *CICS* node.
3. Select the *Channels* node.
4. Select the channel you want to start.
5. Right-click the channel and select *Start*.

The channel becomes active and the X over the icon disappears.

**Procedure: How to Test a Channel**

To test a channel:

1. Send your test data, for example, by running the IWAYEVTO COBOL sample.
2. Look for the results in the destination specified by the File port.

**Procedure: How to Stop a Channel**

To stop a channel:

1. In the left pane of iWay Explorer, select the *iWay Events* tab.
2. Expand the *CICS* node.
3. Select the *Channels* node.

4. Select the channel you want to stop.
5. Right-click the channel and select *Stop*.

The channel becomes inactive and the X appears over the icon.

## Running CICS Events

After a channel is configured and tested, it may be started in a runtime server.

The channel can continue to be stopped and restarted from these tools. When a server is restarted, the channel will remain in the state it was last left in.

### **Reference:** XML Schema Document

This section provides the sample XML schema document that is generated for the iwayevt0 sample program using iWay Explorer.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated by the iBSE 2007-04-05T20:48:06Z -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:ns1="urn:iwaysoftware:CICS/C:/iWay7/etc/samples/cics/iwayevt0/cobolfd/
iwayevt0_in.cbl_Event"
```

```

targetNamespace="urn:iwaysoftware:CICS/C:/iWay7/etc/samples/cics/iwayevt0/cobolfd/
iwayevt0_in.cbl_Event"
  elementFormDefault="qualified" attributeFormDefault="unqualified" xml:lang="en">
    <xsd:element name="CICSEvent">
      <xsd:complexType>
        <xsd:all>
          <xsd:element name="EventData">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="CommArea" maxOccurs="unbounded">
                  <xsd:complexType>
                    <xsd:sequence>
                      <xsd:element name="ALPHA01" minOccurs="0">
                        <xsd:simpleType>
                          <xsd:restriction base="xsd:string">
                            <xsd:maxLength value="8"/>
                          </xsd:restriction>
                        </xsd:simpleType>
                      </xsd:element>
                      <xsd:element name="INT01" minOccurs="0">
                        <xsd:simpleType>
                          <xsd:restriction base="xsd:integer"/>
                        </xsd:simpleType>
                      </xsd:element>
                      <xsd:element name="PACK01" minOccurs="0">
                        <xsd:simpleType>
                          <xsd:restriction base="xsd:integer"/>
                        </xsd:simpleType>
                      </xsd:element>
                      <xsd:element name="ZONE01" minOccurs="0">
                        <xsd:simpleType>
                          <xsd:restriction base="xsd:integer"/>
                        </xsd:simpleType>
                      </xsd:element>
                    </xsd:sequence>
                  </xsd:complexType>
                </xsd:element>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
        </xsd:all>
      </xsd:complexType>
    </xsd:element>
  </xsd:schema>

```

### **Reference:** XML Output Document

This section provides the sample XML output document.

```
<CICSEvent xmlns="urn:iwaysoftware:CICS/c:/iway7/etc/samples/cics/iwayevt0/
cobolfd/iwayevt0_in.cbl_Event">
<EventData>
  <CommArea>
    <ALPHA01>ABCDEFGH</ALPHA01>
    <INT01>25</INT01>
    <PACK01>50</PACK01>
    <ZONE01>75</ZONE01>
  </CommArea>
</EventData>
</CICSEvent>
```

## Configuring the Transaction Adapter for CICS in an iWay Environment

After you successfully configure the adapter to represent a particular adapter target, the adapter can be assigned to an iWay Service Manager channel.

### In this appendix:

- ❑ [Configuring the Transaction Adapter for CICS in Service Manager](#)

### Configuring the Transaction Adapter for CICS in Service Manager

Before configuring the adapter in iWay Service Manager, you must first create a target, which represents a connection to a backend system, using iWay Explorer. For more information on configuring targets and connections using iWay Explorer, see [Creating XML Schemas and iWay Business Services](#) on page 31.

You configure the adapter in the iWay Service Manager console. The configuration process creates run-time connection and persistent data files within Service Manager. The configuration process interrogates the Service Manager repository entries that were built when the target and connection were created using iWay Explorer. The define adapter process creates the run-time repository based on the design-time repository.

#### **Procedure:** How to Define the Adapter

To define the adapter:

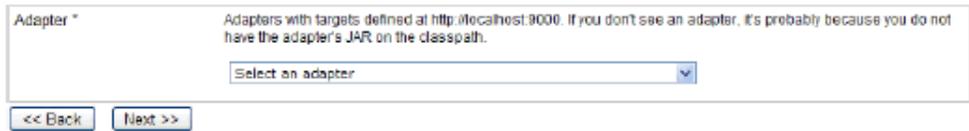
1. In the Service Manager console, select *Registry*, then *Adapters*.
2. Click *Add*.

The iBSP URL pane opens, as shown in the following image.

Provide Repository URL for the new Adapter	
iBSP URL *	Repository of available adapters with user defined targets
	<input type="text" value="http://localhost:9000"/>
<< Back	Next >>

3. Enter your iBSP URL, which is the location of the Service Manager repository, for example, <http://localhost:9000>. This field is required.
4. Click *Next*.

An adapter selection pane opens, as shown in the following image.



5. From the Adapter drop-down list, select the Adapter, then click *Next*.
6. From the Target drop-down list, select a target you configured for the adapter in iWay Explorer, then click *Next*.

The connection information associated with the target selected is displayed.

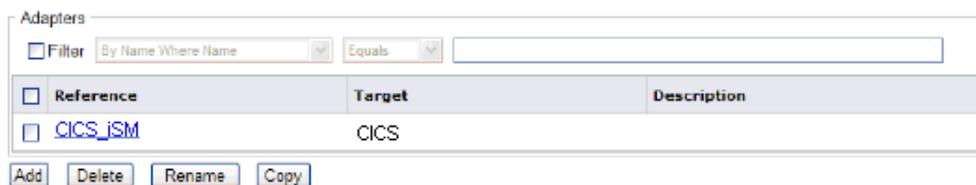
**Adapters**

iWay Service Manager implements an adapter container to configure/invoke iWay Adapters. The adapter container uses the iWay Business Services Provider to access configurational metadata on behalf of its adapters. Listed below are references to adapters defined in the registry.

Set properties of the new Adapter	
Adapter	CICS
Target	CICS
Create Error Document	If on, an error document will be returned when an error occurs <input type="checkbox"/> On
Persist Connection	If on, adapter connection will be reused between executes <input type="checkbox"/> On

- a. Select whether to return an error document when an error occurs.
  - b. Select whether an adapter connection will be reused between executes.
  - c. Review the connection information you specified in iWay Explorer. You can change or update any information.
7. Click *Next*.
  8. Provide a name and, optionally, a description, for the adapter, and click *Finish*.

The adapter appears in the adapters list, as shown in the following image.



**Procedure: How to Modify or Update an Adapter Connection**

The following image shows the Adapter Defines pane which displays the name of the adapter and the description (optional).



To modify or update an adapter connection:

1. From the Adapters list, click the adapter reference you defined, in this example, CICS\_iSM. The pane that displays the target connection information opens. You cannot change the name of the adapter or the target, but you can edit the connection information.
2. After you modify the connection information, click *Update Connection Properties*.
3. After you make changes or additions to the adapter target in iWay Explorer, click *Update Adapter Data*.
4. Click *Finish*.



## Running the Adapter Using LU6.2 Communication

This section contains information that you can use as a guide to ensure LU6.2 communication to the CICS region.

### In this appendix:

- [MVS OS/390 APPC Communication](#)
- [Microsoft SNA Server Communication](#)
- [Application Run-Time Requirements](#)

## MVS OS/390 APPC Communication

The following topics describe the set up for LU6.2 definitions for communication to the CICS region. You can use the following information when consulting with VTAM and CICS administrators.

### LU6.2 Set up on MVS

To correctly set the APPC/MVS LU definition, use the following VTAM APPC settings as a guide.

#### VTAM LU Definitions

```

T39HPAPU PU      PUTYPE=2,                +
                  ADDR=01,                +
                  DISCNT=NO,              +
                  IDBLK=05D,              +
                  IDNUM=54435,            +
                  MAXPATH=2,              +
                  MAXOUT=7,               +
                  MODETAB=IBILM,          +
                  DLOGMOD=SNA3270,        +
                  USSTAB=USSIBSNA,        +
                  ISTATUS=ACTIVE          +

T39HPAI0 LU      LOCADDR=0, MODETAB=MTOS2EE, DLOGMOD=PARALLEL
T39HPAI1 LU      LOCADDR=0, MODETAB=MTOS2EE, DLOGMOD=PARALLEL
T39HPAI2 LU      LOCADDR=0, MODETAB=MTOS2EE, DLOGMOD=PARALLEL
T39HPAI3 LU      LOCADDR=0, MODETAB=MTOS2EE, DLOGMOD=PARALLEL
T39HPAT1 LU      LOCADDR=2
T39HPAT2 LU      LOCADDR=3
T39HPAT3 LU      LOCADDR=4, DLOGMOD=LU62
T39HPAT4 LU      LOCADDR=5, DLOGMOD=LU62

```

#### Sample LogMode Definition

```
PARALLEL MODEENT LOGMODE=PARALLEL,FMPROF=X'13',TSPROF=X'07',
PRIPROT=X'B0',SECPROT=X'B0',COMPROT=X'50B1',TYPE=X'00',
RUSIZES=X'8787',PSERVIC=X'06020000000000000002F00'
```

In iWay Explorer, based on the previous definitions, the LogMode entered is Parallel and the Local LU entered is T39HPAI1.

**Note:** An SNA Stack must be configured with Local LU T39HPAI1 on the adapter platform.

## LU6.2 Set up on CICS

The following image shows a sample Object Characteristics screen for a CICS connection.

```
OBJECT CHARACTERISTICS                                CICS RELEASE = 0530
CEDA View Connection( HP64 )
  Connection      : HP64
  Group          : KUPYN
  Description     : HP64 CONNECTION
CONNECTION IDENTIFIERS
  Netname        : T39HPAI1
  INDSys         :
REMOTE ATTRIBUTES
  REMOTESYSTEM   :
  REMOTENAME     :
  REMOTESYSNET   :
CONNECTION PROPERTIES
  ACcessmethod   : Vtam      Vtam | IRc | INdirect | Xm
  PROtocol       : Appc      Appc | Lu61 | Exci
  Conntype       :           Generic | Specific
  SInglesess     : No        No | Yes
  DATastream     : User      User | 3270 | SCs | STRfield | Lms
  RECordformat   : U         U | Vb
                                                    SYSID=CICS APPLID=IWAYDEMO
```

The following image shows a sample Object Characteristics screen for a CICS session.

```

OBJECT CHARACTERISTICS                                CICS RELEASE = 0530
CEDA View Sessions( T39HPAI1 )
  Sessions      : T39HPAI1
  Group         : KUPYN
  Description   : HP64 LU
SESSION IDENTIFIERS
  Connection    : HP64
  SESSName     :
  NETnameq     :
  MOdename     : PARALLEL
SESSION PROPERTIES
  Protocol     : Appc                Appc | Lu61 | Exci
  MAximum      : 004 , 002          0-999
  RECEIVEPfx  :
  RECEIVECount :                    1-999
  SENDPfx     :
  SENDCount   :                    1-999
  SENDSize    : 04096              1-30720
  RECEIVESize : 04096              1-30720
  *
SYSID=CICS APPLID=IWAYDEMO

```

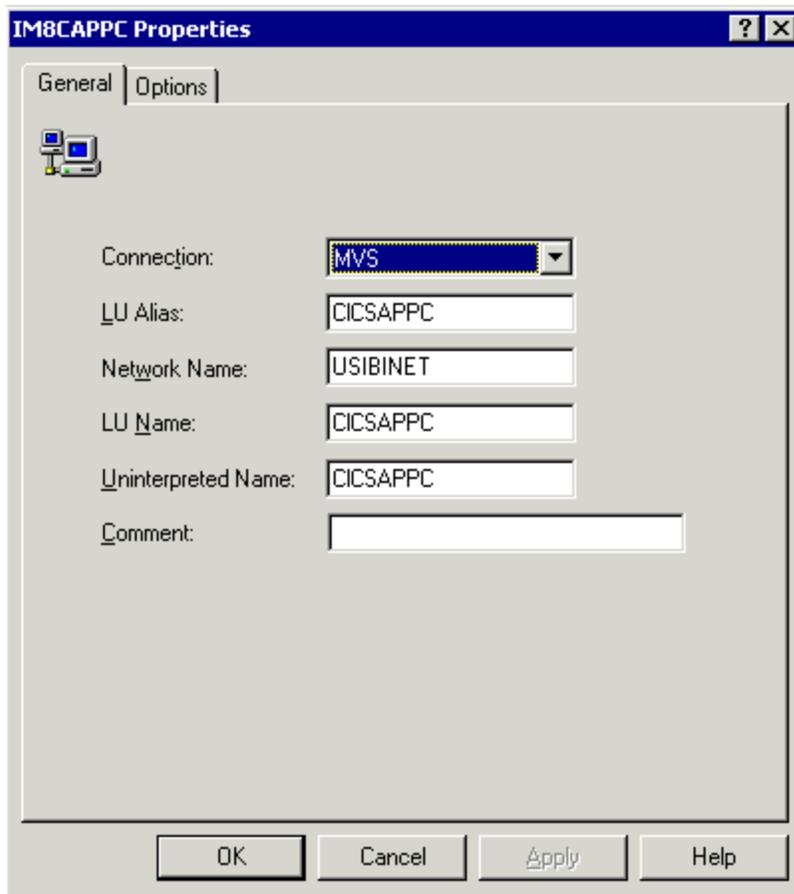
## Microsoft SNA Server Communication

The following topics describe the set up for Microsoft SNA Server communication to the CICS region.

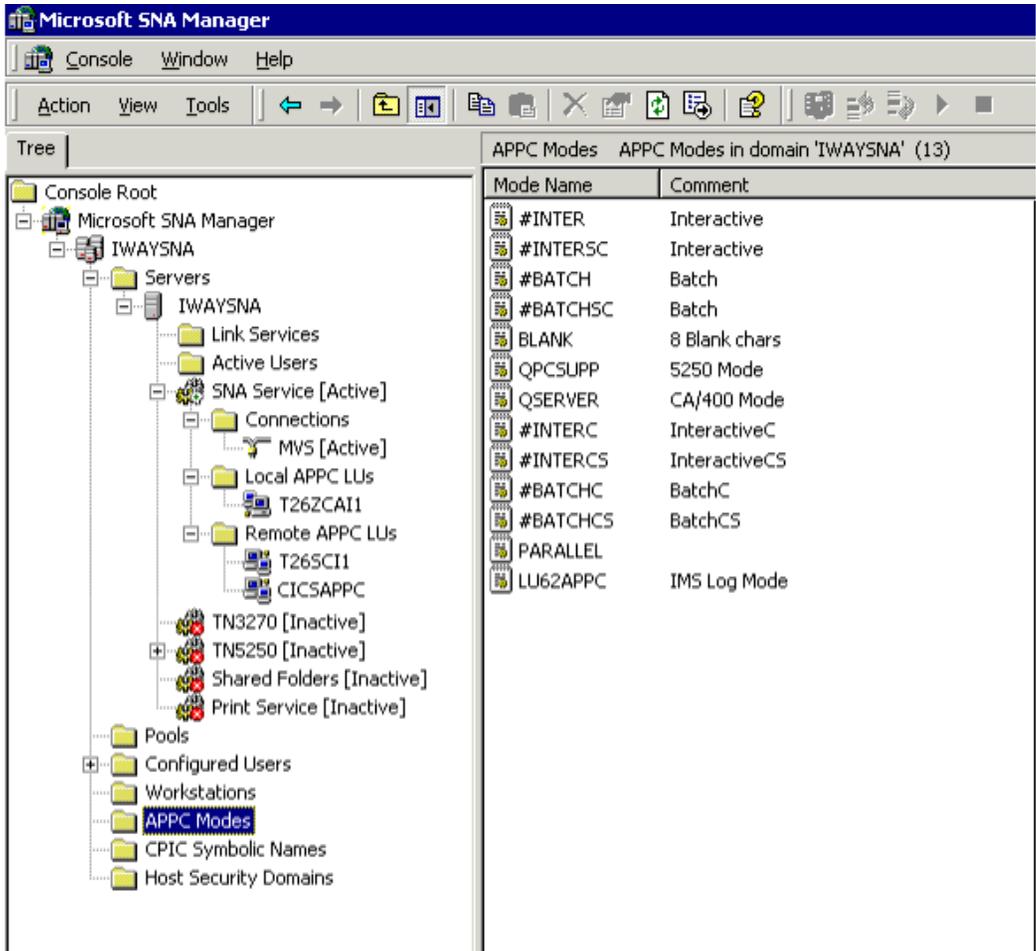
### LU6.2 Setup on a Windows SNA Server

You must verify that the SNA Service is active. You must create one or more remote LUs on the SNA Server with equivalent names for the remote LUs for the CICS region(s).

The following image displays the CICS LU, CICSAPPC, and the SNA Manager properties for the Remote LU on the General tab.



The following image shows SNA Manager on a Windows NT platform with remote LUs and an active SNA Service. A remote LU for the CICS region was created (CICSAPPC). The SNA Server Local LU is T26ZCAI1.



### Application Run-Time Requirements

The following are run-time requirements for applications that invoke the iWay Transaction Adapter for CICS using SNA Services. The adapter engine, iBSP, or any application server must include the required SNA DLLs.

The following image shows the Set connection info pane in iWay Explorer.

**Set connection info**

CICS LU: IWAYDEMO

Local LU: T26ZCAI1

LogMode: PARALLEL

Help < Back Finish Cancel

The remote LU is the CICS LU (IWAYDEMO) that matches the Remote LU on the SNA server.

The Local LU (T26ZCAI1) matches the SNA server local LU.

PARALLEL log mode is used for the connection. You must specify a valid log mode for the CICS region (VTAM mode table entry for APPC sessions).

## Running the Adapter Using TCP/IP Communication

---

This section contains information that you can use as a guide to ensure TCP/IP communication to the CICS region.

### In this appendix:

- [MVS OS/390 TCP/IP Communication](#)
- 

## MVS OS/390 TCP/IP Communication

The following topic describes the requirements to ensure TCP/IP communication to the CICS region. You can use the following information when consulting with VTAM and CICS administrators.

### TCP/IP Requirements

The requirements for TCP/IP communication to the CICS region are:

- TCP must be installed on z/OS.
- The CICS system initialization must specify TCPIP=YES.

The default is NO.

- You must define a TCPIPSERVICE resource definition for the CICS region.

You must specify PROTOCOL(ECI) and the ATTACHSEC(<value>). The ATTACHSEC(<value>) should match what you configure for the adapter connection parameters.

**Note:** The default transaction is CIEP. This can be changed. You may want to set STATUS(OPEN) so that the transaction is started when installed. The default port is 1435. You can change this using the PORT parameter.

For the transaction to run, the TCPIPSERVICE must be installed and started. This can be accomplished automatically (by ensuring that the TCPIPSERVICE is defined in a group that is in a CICS start up list) or manually from CEDA.



## Using Adabas/Natural Programs

---

This section describes how to use Adabas/Natural programs with the iWay Transaction Adapter for CICS.

### In this appendix:

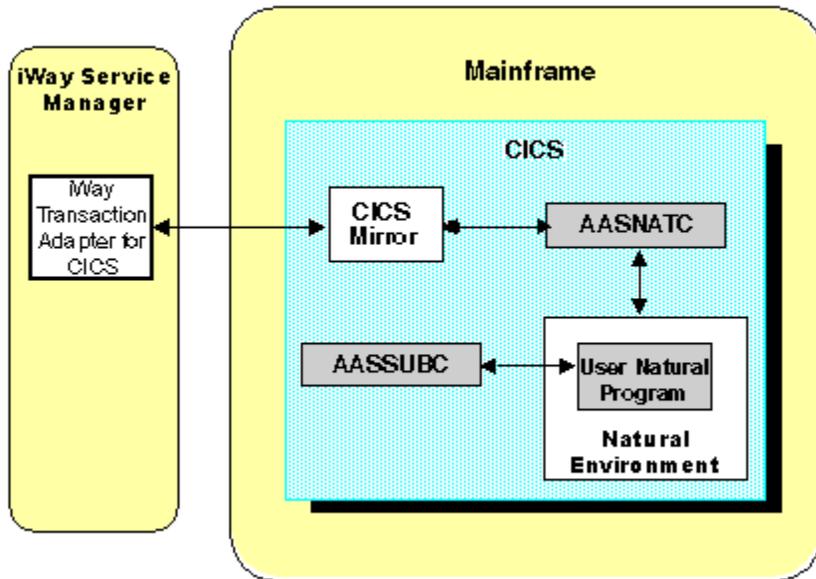
- [Adabas/Natural Programs Overview](#)
  - [Installing the Adabas/Natural Interface](#)
  - [Writing and Configuring a Natural Program](#)
- 

### Adabas/Natural Programs Overview

Adabas/Natural programs can be executed using the iWay Transaction Adapter for CICS. Two programs, AASNATC and AASSUBC, which comprise the Adabas/Natural interface, are packaged with the iWay Service Manager (SM) installation and must be installed in the CICS region. These programs function as an interface between the Natural environment and the CICS/DPL environment, and have the following roles:

- AASNATC** - This proxy program is invoked by the adapter and starts the Natural Nucleus to run your program.
- AASSUBC** - The Natural program uses AASSUBC to read and write the COMMAREA on a field-by-field basis.

The following diagram illustrates the end-to-end data flow for a service request to a Natural program.



The following steps outline the call flow:

1. The iWay Transaction Adapter for CICS receives an input XML document.
2. The adapter wraps the user COMMAREA with Natural Control information, and sends it to AASNATC.
3. The AASNATC program invokes the Natural environment with the Natural program name and logon parameters.
4. The Natural program receives control and reads the input COMMAREA using AASSUBC. After processing the data, it uses AASSUBC again to write the output COMMAREA and returns control to AASNATC.
5. AASNATC constructs the return COMMAREA and returns it to the adapter (via the mirror).
6. The adapter returns the output XML document to the caller.

## Installing the Adabas/Natural Interface

The mainframe programs necessary to create Adabas/Natural programs are packaged as an archive with the iWay Service Manager (SM) installation.

By default, on Windows, the archive location is:

`C:/Program Files/iWay7/etc/setup/natural.zip`

This archive provides the `iway.natural.bin` binary file, which contains the following programs:

- ❑ AASNATC
- ❑ AASSUBC
- ❑ AASNATD

AASNATC and AASSUBC are components of the Adabas/Natural interface used by the iWay Transaction Adapter for CICS. AASNATD is a diagnostic program to be used under the guidance of Customer Support Services.

### ***Procedure:* How to Upload the Adabas/Natural Interface Programs**

To upload the Adabas/Natural programs:

1. Unzip the `natural.zip` file and extract the `iway.natural.bin` binary file.
2. Allocate a dataset on MVS to which the binary file will be uploaded.  
Use `lrecl 80, recfm fb, blksize 3120, organization PS`. Five TRKs are adequate.
3. FTP the binary file to the MVS dataset allocated in step 2.

If you are running the FTP on MVS, use the following FTP subcommands:

```
binary
```

```
LOCSITE LRECL=80 BLKSIZE=3120 RECFM=FB
```

```
get iway.natural.bin mvs.allocated.bin (replace
```

If you are running the FTP command on another host, use:

```
binary
```

```
quote site LRECL=80 BLKSIZE=3120 RECFM=FB
```

```
put iway.natural.bin mvs.allocated.bin (replace
```

4. Unpack the uploaded file into a preexisting MVS PDS load library (not PDSE).
  - a. Enter the following command from the TSO READY prompt:
 

```
RECEIVE indataset(mvs.allocated.bin)
```
  - b. When prompted for an input, respond with:
 

```
DATASET(mvs.loadlib)
```
5. Confirm that the expected members have been added to the target load library.

6. Add the target load library (PDS) to DFHRPL or copy the programs to an existing DFHRPL library so the programs are made available to the CICS region.

**Procedure: How to Add the CICS Definitions**

To add the CICS definitions:

1. Define the programs AASNATC and AASSUBC to CICS.
2. Define an alternate mirror transaction with program DFHMIRS and TWA 128.

A mirror transaction with TWA of 128 is required for the AASNATC program to invoke the Natural Nucleus. You may copy the other parameters from system transaction CPML.

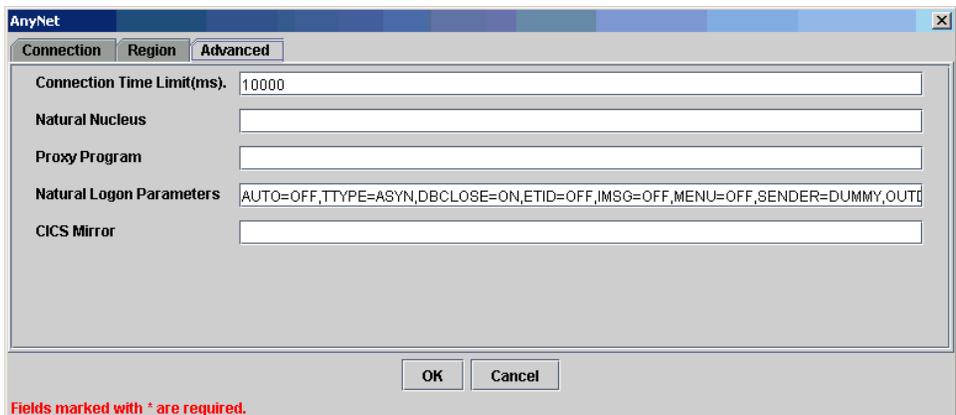
A transaction name of IWAY is suggested.

**Writing and Configuring a Natural Program**

To write and configure a Natural Program for use with the iWay Transaction Adapter for CICS:

1. Write a Natural program to use the AASSUBC calls to send and receive data.  
For more information, see [Using the AASSUBC Calling API](#) on page 99 and [Natural Program](#) on page 132.
2. Using iWay Explorer, define a new CICS target with any supported communications type, for example, AnyNet, SNA, or TCP/IP.
3. Click the *Advanced* tab after you have entered your connection parameters.

The Advanced tab opens.



Perform the following steps:

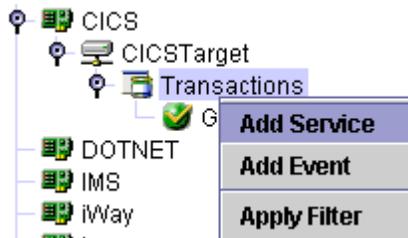
- a. Specify the name of the Natural Nucleus installed on the CICS region.

- b. Specify the Natural proxy program AASNATC.
- c. Specify the alternate mirror transaction in the CICS Mirror field.
- d. Adjust the default Natural Logon Parameters, as needed. The CICS adapter will automatically append a suitable STACK parameter to the end of Logon Parameters when each request is run, using the Natural library and the Program Name defined in the Service Node. A user may also include his own STACK= on the Logon Parameters and reference the Natural library and service program name via the strings #NATLIB and #NATPROG. For example:  
 AUTO=OFF,TTYTYPE=ASYN,DBCLOSE=ON,ETID=OFF,IMSG=OFF,MENU=OFF,SENDER DUMMY,OUTDEST=DUMMY,ID=/,STACK=(LOGON #NATLIB;#NATPROG;FIN). In this case, the automatic STACK parameter is not appended.

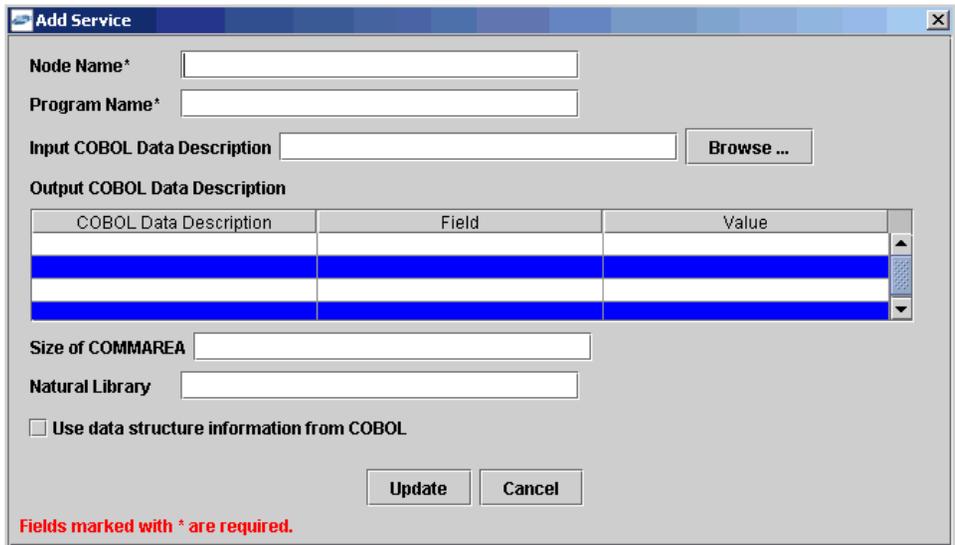
For more information on Natural Logon Parameters themselves, see the Natural documentation.

**Note:** Since this target invokes the Natural proxy program, it is not suitable for use with non-Natural, ECI CICS programs.

4. Connect to your CICS target, right-click the *Transactions* node, and select *Add Service* to create a service node that represents the Natural program.



The Add Service window opens.



**Add Service**

Node Name\*

Program Name\*

Input COBOL Data Description

Output COBOL Data Description

COBOL Data Description	Field	Value

Size of COMMAREA

Natural Library

Use data structure information from COBOL

Fields marked with \* are required.

Perform the following steps:

- a. Specify a name for the service node in the Node Name field.
- b. Specify the name of the Natural program to run in the Program Name field.
- c. Specify the library within the Natural subsystem where the program is stored in the Natural Library field.

**Note:** This value is usually SYSTEM by default.

- d. Specify a COMMAREA size for the service to be a maximum of:

The size of data sent to the Natural program + 300 and the maximum number of bytes to be received from the Natural program.

The iWay Transaction Adapter for CICS requires this extra space on output to send the data required by the proxy program to invoke the Natural Nucleus.

**Note:** This value is usually SYSTEM by default.

The size of data sent to the Natural program + 300 and the maximum number of bytes to be received from the Natural program.

The iWay Transaction Adapter for CICS requires this extra space on output to send the data required by the proxy program to invoke the Natural Nucleus.

For more information on Natural logon parameters, see the Natural documentation.

**Note:** This value is usually SYSTEM by default.

The size of data sent to the Natural program + 300 and the maximum number of bytes to be received from the Natural program.

The iWay Transaction Adapter for CICS requires this extra space on output to send the data required by the proxy program to invoke the Natural Nucleus.

**Reference: Using the AASSUBC Calling API**

A Natural program must use program AASSUBC for all access to input and output buffers.

Calls to AASSUBC all use the following control block as their first parameter:

```
1 #REQUEST-PARMS
2 #FUNCTION      (A2)  GT,PT,LC,LI
2 #OFFSET        (I2)  DATA OFFSET OF INPUT/OUTPUT
2 #LENGTH        (I2)  LENGTH OF DATA TO GET OR PUT
2 #RESPONSE-CODE (I4)
2 #ERR-MESSAGE   (A72)
```

The following table lists and describes the implemented functions:

Function	Description
GT	<p>Get input by offset.</p> <p>The offset must be set to 0 for the first call only. It is incremented following each call by the number of bytes read.</p> <p>The length of the requested data must be provided for each call and must match the length of the area provided to receive the input data. For example:</p> <pre>MOVE #FUNC-GT TO #FUNCTION MOVE 8 TO #LENGTH MOVE 0 TO OFFSET - FIRST CALL ONLY CALL 'AASSUBC' #FUNCTION #EMP-NUM1</pre> <p>WHERE #EMP-NUM1 IS AN EIGHT BYTE FIELD</p>

Function	Description
PT	<p>Put output by offset.</p> <p>The offset must be set to 0 for the first call only. It is incremented following each call by the number of bytes written.</p> <p>The length of the requested data must be provided for each call and must match the length of the area provided containing the output data. For example:</p> <pre>MOVE #FUNC-PT TO #FUNCTION MOVE 8 TO #LENGTH MOVE 0 TO OFFSET - FIRST CALL ONLY CALL 'AASSUBC' #FUNCTION #EMP-NUM1</pre> <p>WHERE #EMP-NUM1 IS AN EIGHT BYTE FIELD</p>
LI	<p>Get length of input data.</p> <p>This function updates the #LENGTH field with the total length of all input parameters. No additional parameters are required. For example:</p> <pre>MOVE #FUNC-LI TO #FUNCTION CALL 'AASSUBC' #FUNCTION</pre>
LC	<p>Get length of COMMAREA used to send data.</p> <p>This function updates the #LENGTH field with the COMMAREA length. No additional parameters are required. For example:</p> <pre>MOVE #FUNC-LC TO #FUNCTION CALL 'AASSUBC' #FUNCTION</pre>

The following table lists and describes the implemented response codes:

Response Code	Description
0	Operation completed successfully.
4	ENDDATA - OFFSET exceeds end of input data and indicates that no further input is available.
8	OFFSET + LENGTH is greater than COMMAREA.

**Note:** Always use the first field of a group when calling AASSUBC. Notice all calls are made with #FUNCTION and not #REQUEST-PARMS.

All input fields must be read before output fields are written. The sample program aasnaln.natural shows how AASSUBC is used. On Windows, the Natural sample program is located in:

```
C:\Program Files\iWay7\etc\samples\cics\natural
```

For more information on using this program, see [Natural Program](#) on page 132.



## Installing the Sample IWAYIVP and IWAYSRV0 Programs in CICS

---

This section describes how to verify that you have correctly installed the iWay Transaction Adapter for CICS.

**In this appendix:**

- [Installing and Configuring IWAYIVP](#)
  - [Installing and Configuring IWAYSRV0](#)
- 

### Installing and Configuring IWAYIVP

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 [Sample CICS Programs](#) on page 119. For specific information for your site, see your CICS Systems Administrator.

**Procedure: How to Install and Configure IWAYIVP**

To install and configure the sample CICS program, IWAYIVP:

1. Use the source code provided in [Sample CICS Programs](#) on page 119, compile the program, and make it available to CICS.
2. Define the COBOL program to the CICS region, as follows:
  - a. Log onto the CICS region.
  - b. Issue the following command:

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

The Define Program (IWAYIVP) mainframe screen appears, as shown in the following image.

```

DEF PROG(IWAYIVP) GROUP(IWAY) CICS RELEASE = 0530
OVERTYPE TO MODIFY
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
a 06/022
    
```

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

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

The following image shows a sample installation mainframe screen.

```

Session A - [24 x 80]
File Edit Transfer Appearance Communication Assist Window Help
PrtScr Copy Paste Send Recv Display Color Map Record Stop Play Quit Clipbrd Index

INST PROG(IWAYIVP)  GROUP(IWAY)
OVERTYPE TO MODIFY
CEDA Install
Connection ==>
DB2Conn ==>
DB2Entry ==>
DB2Tran ==>
D0ctemplate ==>
Enqmodel ==>
File ==>
Journalmodel ==>
LSrpool ==>
Mapset ==>
PARTIitionset ==>
PARTNer ==>
PROcesstype ==>
PROFile ==>
PROGram ==> IWAY
Requestmodel ==>
+ Sessions ==>

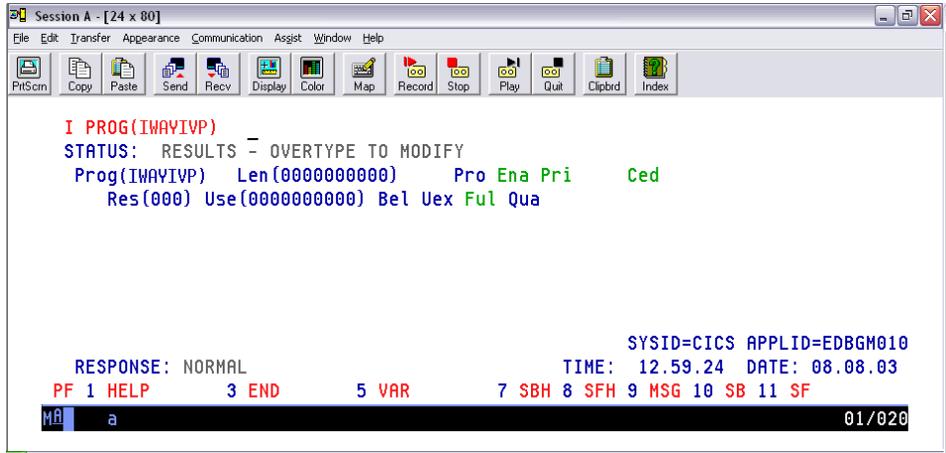
INSTALL SUCCESSFUL
PF 1 HELP 3 END 6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
SYSID=CICS APPLID=EDBGM010
TIME: 12.58.56 DATE: 03.220
a ^ 04/022

```

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

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

The following image shows a sample results mainframe screen.



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

```
I PROG(IWAYIVP)
STATUS: RESULTS - OVERTYPE TO MODIFY
Prog(IWAYIVP) Len(000000000) Pro Ena Pri Ced
Res(000) Use(000000000) Bel Uex Ful Qua

RESPONSE: NORMAL
SYSID=CICS APPLID=EDBGM010
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
```

The terminal prompt is "MA" followed by a cursor and the character "a". The bottom right corner of the terminal shows "01/020".

## Installing and Configuring IWAYSRVO

The following procedure demonstrates how to install and configure the sample CICS program, IWAYSRVO. Sample source code for the COBOL program IWAYSRVO is provided in [Sample CICS Programs](#) on page 119. For specific information for your site, see your CICS Systems Administrator.

### **Procedure:** How to Install and Configure IWAYSRVO

To install and configure the sample CICS program, IWAYSRVO:

1. Use the source code provided in [Sample CICS Programs](#) on page 119, compile the program, and make it available to CICS.
2. Define the COBOL program to the CICS region, as follows:
  - a. Log onto the CICS region.
  - b. Issue the following command:

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

The Define Program (IWAYSrv0) screen appears as shown in the following image.

```

DEF PROG(IWAYSrv0) GROUP(IWAY)
OVERTYPE TO MODIFY                                CICS RELEASE = 0530
CEDA DEFINE PROGRAM( IWAYSrv0 )
PROGRAM      : IWAYSrv0
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.

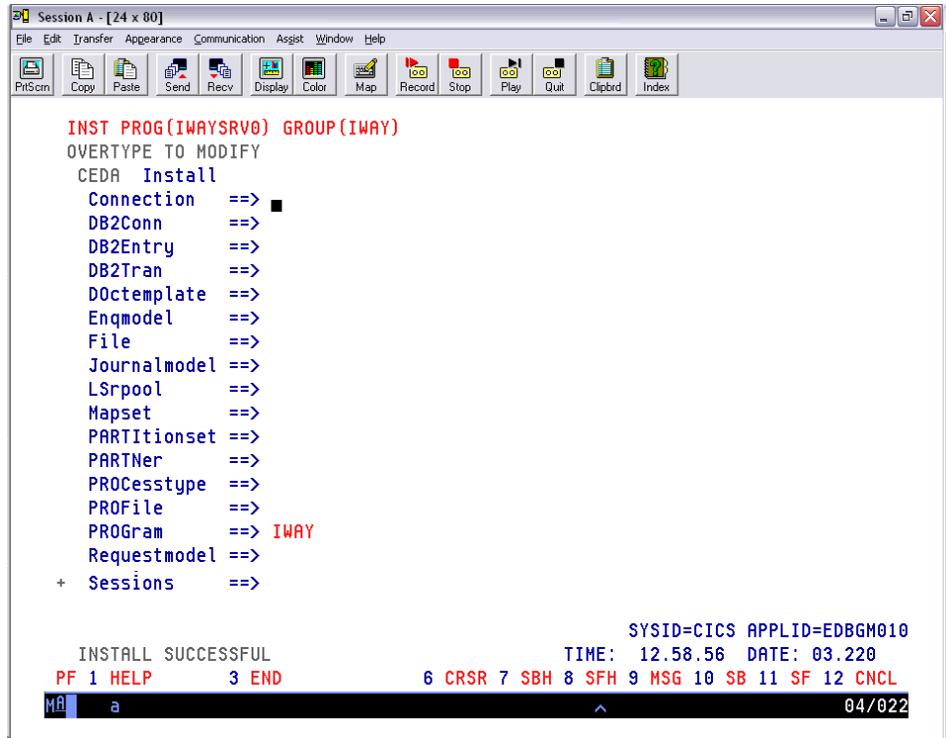
                                SYSID=CICS APPLID=EDBGM010
DEFINE SUCCESSFUL                TIME: 12.58.17 DATE: 03.220
PF 1 HELP 2 COM 3 END           6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
MA a                             06/022

```

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

```
CEDA INST PROG(IWAYSrv0) GROUP(IWAY)
```

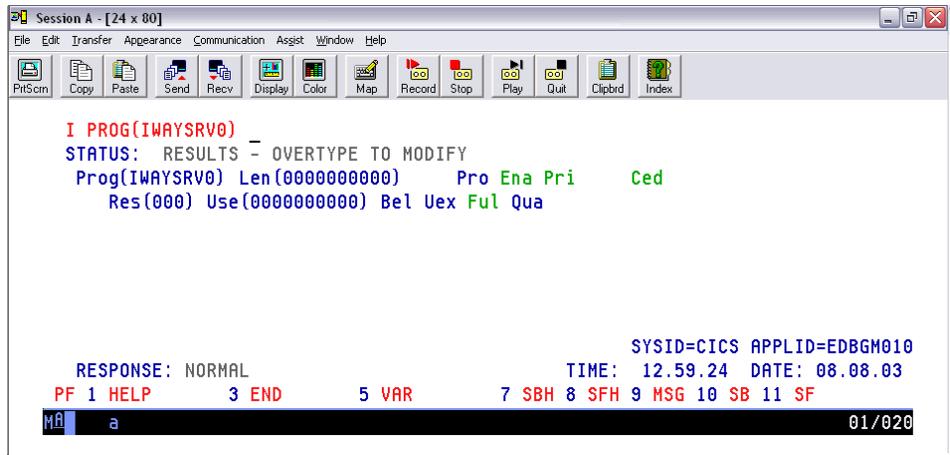
The following image shows a sample installation screen for IWAYSRV0.



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

```
CEMT I PROG(IWAYSrv0)
```

The following image shows a sample of the IWAYSIVO results screen.



```
Session A - [24 x 80]
File Edit Transfer Appearance Communication Assst Window Help
PrintScreen Copy Paste Send Recv Display Color Map Record Stop Play Quit Clipbrd Index

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

RESPONSE: NORMAL
SYSID=CICS APPLID=EDBGM010
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
a 01/020
```



## Sample Requests, Schemas, and COBOL File Descriptions

---

After you create a connection to CICS, you can add CICS transactions using iWay Explorer. The generic transaction is always 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. In addition, the COBOL descriptions that were used as input for the sample CICS transactions are shown.

### In this appendix:

- [Request Document for the Generic Transaction, IWAYIVP](#)
  - [Request Schema for IWAYIVP](#)
  - [Response Schema for IWAYIVP](#)
  - [Request Documents for IWAYSRV0](#)
  - [Request Schema for IWAYSRV0](#)
  - [Response Schema for IWAYSRV0](#)
  - [Request Document for AASNATN](#)
  - [Request Schema for AASNATN](#)
  - [Response Schema for the Program AASNATN](#)
  - [Sample COBOL File Descriptions](#)
- 

### Request Document for the Generic Transaction, IWAYIVP

The following is a sample XML request document to run the generic transaction, IWAYIVP.

```
<?xml version="1.0" encoding="UTF-8" ?>
<CICS>
  <Transaction location="CICS/Transactions/IWAYIVP">
    <CommArea>
      <INPUT>anything you want</INPUT>
    </CommArea>
  </Transaction>
</CICS>
```

## Request Schema for IWAYIVP

The following is a sample XML request schema for the generic transaction, IWAYIVP.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Generated by the iBSE 2004-11-04T16:19:05Z -->
<xsd:schema xml:lang="en" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="urn:iwaysoftware:CICS/Transactions/IWAYIVP_Request"
  attributeFormDefault="unqualified" elementFormDefault="qualified">
  <xsd:element name="CICS">
    <xsd:complexType>
      <xsd:all>
        <xsd:element name="Transaction">
          <xsd:complexType>
            <xsd:all>
              <xsd:element name="CommArea">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element minOccurs="1"
                      name="DFHCOMMAREA" maxOccurs="1">
                      <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                          <xsd:length value="1"/>
                        </xsd:restriction>
                      </xsd:simpleType>
                    </xsd:element>
                  </xsd:sequence>
                </xsd:complexType>
              </xsd:element>
            </xsd:all>
            <xsd:attribute type="xsd:string" use="required"
              fixed="CICS/Transactions/IWAYIVP" name="location"/>
          </xsd:complexType>
        </xsd:element>
      </xsd:all>
    </xsd:complexType>
  </xsd:element>
</xsd: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 IWAYIVP

The following is a sample XML response schema for the generic transaction, IWAYIVP.

```

<?xml version="1.0" encoding="UTF-8" ?>
<!-- Generated by the iBSE 2004-11-04T16:19:05Z -->
<xsd:schema xml:lang="en" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="urn:iwaysoftware:CICS/Transactions/IWAYIVP_Response"
  attributeFormDefault="unqualified" elementFormDefault="qualified">
  <xsd:element name="CICS">
    <xsd:complexType>
      <xsd:all>
        <xsd:element name="Transaction">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="CommArea" maxOccurs="unbounded">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element minOccurs="1"
                      name="DFHCOMMAREA" maxOccurs="1">
                      <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                          <xsd:length value="1"/>
                        </xsd:restriction>
                      </xsd:simpleType>
                    </xsd:element>
                  </xsd:sequence>
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:all>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>

```

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

## Request Documents for IWAYSRV0

The following are sample XML request documents to run the IWAYSRV0 program.

```

<?xml version="1.0" encoding="UTF-8" ?>
<CICS>
  <Transaction location="/CICS/Transaction/IWAYSRV0">
    <CommArea>
      <COMMAND>SHORT</COMMAND>
    </CommArea>
  </Transaction>
</CICS>

```

```

<?xml version="1.0" encoding="UTF-8" ?>
<CICS>
  <Transaction location="/CICS/Transaction/IWAYSRV0">
    <CommArea>
      <COMMAND>LONG</COMMAND>
    </CommArea>
  </Transaction>
</CICS>

```

## Request Schema for IWAYSRV0

The following is a sample XML request schema for the program, IWAYSRV0.

```

<?xml version="1.0" encoding="UTF-8" ?>
<!-- Generated by the iBSE 2004-11-04T16:20:49Z -->
<xsd:schema xml:lang="en" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="urn:iwaysoftware:CICS/Transactions/IWAYSRV0_Request"
  attributeFormDefault="unqualified" elementFormDefault="qualified">
  <xsd:element name="CICS">
    <xsd:complexType>
      <xsd:all>
        <xsd:element name="Transaction">
          <xsd:complexType>
            <xsd:all>
              <xsd:element name="CommArea">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element minOccurs="1"
                      name="DFHCOMMAREA" maxOccurs="1">
                      <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                          <xsd:length value="1"/>
                        </xsd:restriction>
                      </xsd:simpleType>
                    </xsd:element>
                  </xsd:sequence>
                </xsd:complexType>
              </xsd:element>
            </xsd:all>
            <xsd:attribute type="xsd:string" use="required"
              fixed="CICS/Transactions/IWAYSRV0" name="location"/>
          </xsd:complexType>
        </xsd:element>
      </xsd:all>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>

```

## Response Schema for IWAYSRV0

The following is a sample XML response schema for the program, IWAYSRV0.

```

<?xml version="1.0" encoding="UTF-8" ?>
<!-- Generated by the iBSE 2004-11-04T16:20:49Z -->
<xsd:schema xml:lang="en" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="urn:iwaysoftware:CICS/Transactions/IWAYSRV0_Response"
  attributeFormDefault="unqualified" elementFormDefault="qualified">
  <xsd:element name="CICS">
    <xsd:complexType>
      <xsd:all>
        <xsd:element name="Transaction">
          <xsd:complexType>
            <xsd:choice>
              <xsd:element name="message1">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element minOccurs="1" name="COMMAND"
maxOccurs="1">
                      <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                          <xsd:length value="1"/>
                        </xsd:restriction>
                      </xsd:simpleType>
                    </xsd:element>
                  </xsd:sequence>
                </xsd:complexType>
              </xsd:element>
              <xsd:element name="message2">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element minOccurs="1" name="COMMAND"
maxOccurs="1">
                      <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                          <xsd:length value="1"/>
                        </xsd:restriction>
                      </xsd:simpleType>
                    </xsd:element>
                  </xsd:sequence>
                </xsd:complexType>
              </xsd:element>
            </xsd:choice>
          </xsd:complexType>
        </xsd:element>
      </xsd:all>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>

```

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

## Request Document for AASNATN

The following is a sample XML request document for the AASNATN program.

```
<?xml version="1.0" encoding="UTF-8"?>
<CICS>
  <Transaction location="CICS/Transactions/AASNATN">
    <commarea>
      <STARTEMP>11111111</STARTEMP>
      <ENDEMP>55555555</ENDEMP>
    </commarea>
  </Transaction>
</CICS>
```

## Request Schema for AASNATN

The following is a sample XML request schema for the AASNATN program.

```
<?xml version="1.0" encoding="UTF-8" ?>
<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="STARTEMP" maxOccurs="1"/>
                    <xs:element minOccurs="1" name="ENDEMP" maxOccurs="1"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:all>
            <xs:attribute type="xs:string" use="required"
              fixed="CICS/Transactions/AASNATN" name="location"/>
          </xs:complexType>
        </xs:element>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

## Response Schema for the Program AASNATN

The following is a sample XML response schema for the AASNATN program.

```

<?xml version="1.0" encoding="UTF-8" ?>
<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:sequence>
              <xs:element name="message" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element minOccurs="1" name="EMPNUM" maxOccurs="1"/>
                    <xs:element minOccurs="1" name="FIRSTNAME" maxOccurs="1"/>
                    <xs:element minOccurs="1" name="LASTNAME" maxOccurs="1"/>
                    <xs:element minOccurs="1" name="MARITALSTATUS" maxOccurs="1"/>
                    <xs:element minOccurs="1" name="SEX" maxOccurs="1"/>
                    <xs:element minOccurs="1" name="BIRTH" maxOccurs="1"/>
                    <xs:element minOccurs="1" name="DEPARTMENT" maxOccurs="1"/>
                    <xs:element minOccurs="1" name="JOBTITLE" maxOccurs="1"/>
                    <xs:element minOccurs="1" name="CCODE1" maxOccurs="1"/>
                    <xs:element minOccurs="1" name="CCODE2" maxOccurs="1"/>
                    <xs:element minOccurs="1" name="CCODE3" maxOccurs="1"/>
                    <xs:element minOccurs="1" name="CCODE4" maxOccurs="1"/>
                    <xs:element minOccurs="1" name="CCODE5" maxOccurs="1"/>
                    <xs:element minOccurs="1" name="SALARY1" maxOccurs="1"/>
                    <xs:element minOccurs="1" name="FILLER" maxOccurs="1"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>

```

## Sample COBOL File Descriptions

The following sample COBOL File Descriptions are used as input for the CICS transactions in [Creating XML Schemas and iWay Business Services](#) on page 31.

### IWAYSIVO\_in.cbl

```
05  COMMAND                PIC X(5).
```

### IWAYSIVO\_out\_S.cbl

```

05 RECORDTYPE          PIC X(1).
05 DATA40             PIC X(39)

```

**IWAYSRV0\_out\_L.cbl**

```

05 RECORDTYPE          PIC X(1).
05 DATA60             PIC X(59).

```

**AASNATN\_in.cbl**

```

***** 08800000
*      INPUT to NATURAL PROGRAM AASNATN
***** 23100000
02 NATREC.             24000000
03 START-EMP          PIC X(8).    32000000
03 END-EMP            PIC X(8).    32000000

```

**AASNATN\_out.cbl**

```

***** 08800000
*      OUTPUT FROM NATURAL PROGRAM AASNATN
***** 23100000
02 NATREC.             24000000
03 EMP-NUM            PIC X(8).    32000000
03 FIRSTNAME          PIC X(20).   40000000
03 LASTNAME           PIC X(20).   40000000
03 MARITAL-STATUS    PIC X(1).    40000000
03 SEX                PIC X(1).    40000000
03 BIRTH              PIC X(4).    40000000
03 DEPARTMENT        PIC X(6).    40000000
03 JOB-TITLE         PIC X(25).   40000000
03 CCODE1             PIC X(3).    40000000
03 CCODE2             PIC X(3).    40000000
03 CCODE3             PIC X(3).    40000000
03 CCODE4             PIC X(3).    40000000
03 CCODE5             PIC X(3).    40000000
03 SALARY1            PIC X(9).    40000000
03 FILLER             PIC X(2).    40000000

```

## Sample CICS Programs

---

This section contains the source code and other supporting files for the sample programs that are provided with the iWay Transaction Adapter for CICS.

By default, on Windows, the sample programs are available in:

```
C:\Program Files\iWay7\etc\samples\cics
```

The following subdirectories are available and contain sample programs:

- ❑ **iwayivp** - Demonstrates basic request/response.
- ❑ **iwaysrv0** - Demonstrates multiple record types.
- ❑ **iwayevt0** - Demonstrates sending a binary record to the adapter.
- ❑ **iwayevt1** - Demonstrates sending an XML document to the adapter.
- ❑ **natural** - Demonstrates using the Natural interface.

### In this appendix:

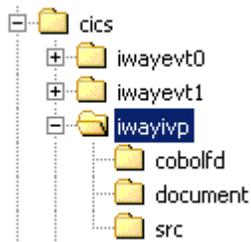
- ❑ [IWAYIVP Program](#)
  - ❑ [IWAYSrv0 Program](#)
  - ❑ [IWAYEVT0 Program](#)
  - ❑ [IWAYEVT1 Program](#)
  - ❑ [Natural Program](#)
- 

## IWAYIVP Program

The IWAYIVP sample program tests the adapter installation and populates the COMMAREA with a "CONGRATULATIONS!!!" message. This program takes arbitrary input and returns a fixed string as output.

On Windows, the IWAYIVP sample program is located in:

C:\Program Files\iWay7\etc\samples\cics\iwayivp



The following supporting files are provided:

**cobolfd**

- iwayivp\_in.cbl - The input record for the service.
- iwayivp\_out.cbl - The output record for the service.

**document**

- iwayivp.xml - The input XML document.

**src**

- iwayivp.c- The source code for the sample program written in C.
- iwayivp.cobol - The source code for the sample program written in COBOL.

The text versions of these source code files are included below for your review:

**iwayivp.c**

```
/*
 * iwayivp - This installation verification program populates
 * the commarea with the 'Congratulations!!!' message.
 *
 * uses: iwayivp_in.cbl   (input record)
 *       iwayivp_out.cbl (output record)
 *       iwayivp.xml     (input document)
 */
#include <stdio.h>
int main ()
{
    char *commarea;
    EXEC CICS ADDRESS EIB(dfheiptr);
    EXEC CICS ADDRESS COMMAREA(commarea);
    strcpy(commarea, "Congratulations!!!");
    EXEC CICS RETURN;
}
```

**iwayivp.cobol**

```

*****
* IWAYIVP - THIS INSTALLATION VERIFICATION PROGRAM POPULATES *
* THE COMMAREA WITH THE 'CONGRATULATIONS!!!' MESSAGE. *
* *
* USES: IWAYIVP_IN.CBL (INPUT RECORD) *
* IWAYIVP_OUT.CBL (OUTPUT RECORD) *
* IWAYIVP.XML (INPUT DOCUMENT) *
* *
*****
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.

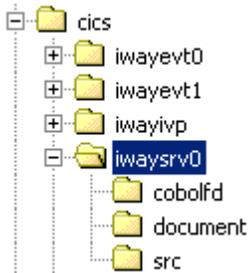
```

## IWAYSRV0 Program

The IWAYSRV0 sample program demonstrates how different output record types can be distinguished using positional fields. The input COMMAREA indicates whether a 40 byte short record or 60 byte long record is to be returned using the 'SHORT' or 'LONG' command. The service is configured to return an output document that is specific to the type.

On Windows, the IWAYSRV0 sample program is located in:

C:\Program Files\iWay7\etc\samples\cics\iwaysrv0



The following supporting files are provided:

**cobolfd**

- iwaysrv0\_in.cbl - Input record for the service.
- iwaysrv0\_out\_L.cbl - Long output record for the service.
- iwaysrv0\_out\_S.cbl - Short output record for the service.

**document**

- iwaysrv0\_L.xml - The long input XML document.
- iwaysrv0\_S.xml - The short input XML document.

**src**

- iwaysrv0.cobol - The source code for the sample program written in COBOL.

The text version of the source code file is included below for your review:

```

*****
* IWAYSrv0 - THIS SAMPLE PROGRAM DEMONSTRATES THAT DIFFERENT *
* OUTPUT RECORD TYPES MAY BE DISTINGUISHED BY POSITIONAL *
* FIELDS. THE INPUT COMMAREA INDICATES WHETHER A 40 BYTE S *
* OR 60 BYTE L RECORD IS TO BE RETURNED BY USING THE COMMAND *
* 'SHORT' OR 'LONG'. THE CICS ADAPTER SERVICE IS CONFIGURED *
* TO RETURN AN OUTPUT DOCUMENT SPECIFIC TO THE TYPE. *
*
* USES: IWAYSrv0_IN.CBL (INPUT RECORD) *
* IWAYSrv0_OUT_S.CBL (SHORT OUTPUT RECORD) *
* IWAYSrv0_OUT_L.CBL (LONG OUTPUT RECORD) *
* IWAYSrv0_S.XML (SHORT INPUT DOCUMENT) *
* IWAYSrv0_L.XML (LONG INPUT DOCUMENT) *
*
*****

```

```
IDENTIFICATION DIVISION.
PROGRAM-ID. IWAYSRV0.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 WS-DATA-OUT.
   05 WS-DATA-OUT-MSG    PIC X(30).
   05 WS-DATA-OUT-ARROW PIC X(30).
LINKAGE SECTION.
01 DFHCOMMAREA.
   05 CA-DATA.
      10 CA-BYTES                PIC X(60).
   05 CA-INPUT REDEFINES CA-DATA.
      10 CA-INPUT-COMMAND       PIC X(5).
      10 CA-INPUT-FILLER       PIC X(55).
   05 CA-OUTPUT REDEFINES CA-DATA.
      10 CA-OUTPUT-RECORD-TYPE  PIC X(1).
      10 CA-OUTPUT-MSG         PIC X(59).
PROCEDURE DIVISION.
EXEC CICS HANDLE ABEND
      LABEL (030-ABEND)
END-EXEC
IF EIBCALEN = 0
      MOVE ' EIBCALEN IS = ZERO ' TO CA-OUTPUT-MSG
      PERFORM 020-RETURN
END-IF.
```

```

*-----*
* THE CICS SERVICE IS CONFIGURED DIFFERENTIATE TO 'S' AND 'L' *
* RECORDS. *
*-----*
      IF CA-INPUT-COMMAND = 'SHORT'
          MOVE 'S' TO CA-OUTPUT-RECORD-TYPE
          MOVE 'RECORD TYPE S RETURNS 40 BYTES'
              TO WS-DATA-OUT-MSG

          MOVE '=====>'
              TO WS-DATA-OUT-ARROW
          MOVE WS-DATA-OUT TO CA-OUTPUT-MSG
          GO TO 020-RETURN
      END-IF.
      IF CA-INPUT-COMMAND = 'LONG '
          MOVE 'L' TO CA-OUTPUT-RECORD-TYPE
          MOVE 'RECORD TYPE L RETURNS 60 BYTES'
              TO WS-DATA-OUT-MSG
          MOVE '=====>'
              TO WS-DATA-OUT-ARROW
          MOVE WS-DATA-OUT TO CA-OUTPUT-MSG
          GO TO 020-RETURN
      END-IF.
      MOVE ' SUPPLY L OR S IN FIRST BYTE OF INPUT...PLEASE '
          TO CA-OUTPUT-MSG.
020-RETURN.
      EXEC CICS RETURN
      END-EXEC.
      GOBACK.
030-ABEND.
      GO TO 020-RETURN.

```

## IWAYEVTO Program

The IWAYEVTO sample program demonstrates event handling by sending a record to the iWay Transaction Adapter for CICS using CICS sockets. No response is returned and each data record that is mapped by the COPYBOOKS must be preceded by a 4 byte binary length.

You must configure the adapter with an event to receive this data. Using iWay Explorer, select the *Is Length Prefix* checkbox and *Request* from the Synchronization Type drop-down list. Specify IWAYEVTO.CBL in the Preparser FD field. The host and port must match the value set used in the sample program.

The EZASOKET interface is documented in the Z/OS communications server IP sockets guide.

**Note:** For other platforms, use the socket interface appropriate for that platform.

On Windows, the IWAYEVTO sample program is located in:

C:\Program Files\iWay7\etc\samples\cics\iwayevt0



The following supporting files are provided:

**cobolfd**

- iwayevt0\_in.cbl - Input record for the service.

**src**

- iwayevt0.cobol - The source code for the sample program written in COBOL.

The text version of the source code file is included below for your review.

```

CBL TRUNC(BIN)
ID DIVISION.
PROGRAM-ID. IWAYEVT0.
*****
* IWAYEVT0 - THIS SAMPLE PROGRAM DEMONSTRATES SENDING A      *
* RECORD TO THE IWAY CICS ADAPTER USING CICS SOCKETS. NO     *
* RESPONSE IS RETURNED. DATA RECORDS MAPPED BY COPYBOOKS  *
* MUST EACH BE PRECEDED BY A 4 BYTE BINARY LENGTH.          *
*                                                             *
* THE CICS ADAPTER MUST BE CONFIGURED WITH AN EVENT TO      *
* RECEIVE THIS DATA. SELECT "IS LENGTH PREFIX", SYNCHRON-  *
* IZATION TYPE "REQUEST", AND USE IWAYEVT0.CBL AS THE       *
* PREPARSER FD. HOST AND PORT MUST MATCH THE VALUES SET   *
* BELOW.                                                     *
*                                                             *
* THE EZASOKET INTERFACE IS DOCUMENTED IN THE Z/OS         *
* COMMUNICATIONS SERVER IP CICS SOCKETS GUIDE.             *
*                                                             *
* USES: IWAYEVT0_IN.CBL      (INPUT RECORD)                 *
*                                                             *
*****

```

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 SOCKET-GROUP.
    05 SOC-FUNCTION                PIC X(16) VALUE SPACES.
    05 ERRNO                       PIC 9(8) BINARY VALUE ZEROES.
    05 RETCODE                     PIC S9(8) BINARY VALUE ZEROES.
    05 AF                          PIC 9(8) BINARY VALUE 2.
    05 SOCTYPE                     PIC 9(8) BINARY VALUE 1.
    05 PROTO                       PIC 9(8) BINARY VALUE 0.
    05 NAMELEN                     PIC 9(8) BINARY.
    05 HOSTNAME                    PIC X(255).
    05 HOSTENT                     POINTER.
    05 NAME.
        10 FAMILY                  PIC 9(4) BINARY VALUE 2.
        10 PORT                    PIC 9(4) BINARY.
        10 IP-ADDRESS              PIC 9(8) BINARY.
        10 IP-ADDRESS-ALPHA REDEFINES IP-ADDRESS PIC X(4).
        10 RESERVED                PIC X(8) VALUE LOW-VALUES.
    05 FLAGS                       PIC 9(8) BINARY VALUE 0.
    05 SOCKET                      PIC 9(4) BINARY.
    05 NBYTE                       PIC 9(8) BINARY.
    05 CMD                         PIC 9(8) BINARY.
    05 REQARG                      PIC 9(8) BINARY.
01 WORKAREA.
    05 LLEN                        PIC 9(8) BINARY VALUE 4.
    05 ERRMSG                      PIC X(41)
        VALUE 'ERROR ENCOUNTERED DURING '.
    05 TMSG                        PIC X(44)
        VALUE 'EVENTCBL: RECORD TRANSMISSION WAS SUCCESSFUL'.
```

```

*****
* SAMPLE INBOUND DATA RECORD WITH VARIOUS COBOL TYPES.      *
*****
01 INBOUND-RECORD.
   05 ALPHA01                PIC X(8)
      VALUE 'ABCDEFGH' .
   05 INT01                  PIC S9(4) BINARY VALUE 25.
   05 PACK01                 PIC S9(15) PACKED-DECIMAL VALUE 50.
   05 ZONE01                 PIC 9(4) VALUE 75.
LINKAGE SECTION.
01 HOSTENT-STRUCT.
   05 HOSTNAME-PTR          POINTER.
   05 HOSTALIASL-PTR        POINTER.
   05 HOSTFAMILY            PIC S9(8) BINARY.
   05 HOSTADR-LEN           PIC S9(8) BINARY.
   05 HOSTADRL-PTR          POINTER.
01 HOST-ENTRY-PTR          POINTER.
01 HOST-ENTRY              PIC 9(8) BINARY.
PROCEDURE DIVISION.
MAINLINE.
*****
* CHANGE HOSTNAME AND PORT TO SITE SPECIFIC LOCATION OF THE  *
* CICS ADAPTER.                                             *
*****
      MOVE 'YOUR.DNS.NAME' TO HOSTNAME
      MOVE 4772 TO PORT
      PERFORM GETSOCK
      PERFORM GETHOSTBYNAME
      PERFORM SETBLOCK
      PERFORM CONNECTTOHOST
      PERFORM SENDDATA
      PERFORM CLOSESOCK
      EXEC CICS SEND TEXT FROM(TMSG)
          LENGTH(LENGTH OF TMSG)
      END-EXEC
      EXEC CICS RETURN END-EXEC
      GOBACK.

```

```
GETSOCK.  
  MOVE 'SOCKET          ' TO SOC-FUNCTION  
  CALL 'EZASOKET' USING SOC-FUNCTION,  
    AF,  
    SOCTYPE,  
    PROTO,  
    ERRNO,  
    RETCODE  
  MOVE RETCODE TO SOCKET  
  IF RETCODE < 0  
    PERFORM WRITERR-EXIT  
  END-IF.  
GETHOSTBYNAME.  
  MOVE 'GETHOSTBYNAME  ' TO SOC-FUNCTION  
  MOVE LENGTH OF HOSTNAME TO NAMELEN  
  CALL 'EZASOKET' USING SOC-FUNCTION NAMELEN HOSTNAME  
  HOSTENT RETCODE  
  IF RETCODE EQUAL ZERO  
    SET ADDRESS OF HOSTENT-STRUCT TO HOSTENT  
    SET ADDRESS OF HOST-ENTRY-PTR TO HOSTADRL-PTR  
    SET ADDRESS OF HOST-ENTRY TO HOST-ENTRY-PTR  
  ELSE  
    PERFORM WRITERR-EXIT  
  END-IF.  
SETBLOCK.  
  MOVE 'FCNTL          ' TO SOC-FUNCTION  
  MOVE 4 TO CMD  
  MOVE 0 TO REQARG  
  CALL 'EZASOKET' USING SOC-FUNCTION, SOCKET, CMD, REQARG,  
    ERRNO, RETCODE.  
CONNECTTOHOST.  
  MOVE HOST-ENTRY TO IP-ADDRESS  
  MOVE 'CONNECT        ' TO SOC-FUNCTION  
  CALL 'EZASOKET' USING SOC-FUNCTION,  
    SOCKET,  
    NAME,  
    ERRNO,  
    RETCODE  
  IF RETCODE = 0  
    CONTINUE  
  ELSE  
    PERFORM WRITERR-EXIT  
  END-IF.  
SENDDATA.
```

```

*****
* PRECEDE THE RECORD WITH 4 BYTE BINARY RECORD LENGTH *
*****
      MOVE 'SEND                ' TO SOC-FUNCTION
      MOVE LENGTH OF INBOUND-RECORD TO NBYTE
      MOVE 4 TO LLEN
      MOVE 0 TO RETCODE
      CALL 'EZASOCKET' USING SOC-FUNCTION,
          SOCKET,
          FLAGS,
          LLEN,
          NBYTE,
          ERRNO,
          RETCODE
      IF RETCODE = -1
          PERFORM WRITERR-EXIT
      END-IF
*****
* SEND THE ACTUAL RECORD *
*****
      CALL 'EZASOCKET' USING SOC-FUNCTION,
          SOCKET,
          FLAGS,
          NBYTE,
          INBOUND-RECORD,
          BY REFERENCE ERRNO,
          RETCODE
      IF RETCODE = -1
          PERFORM WRITERR-EXIT
      END-IF
      .
CLOSESOCK.
      MOVE ZEROES TO RETCODE ERRNO
      MOVE 'CLOSE                ' TO SOC-FUNCTION
      CALL 'EZASOCKET' USING SOC-FUNCTION,
          SOCKET,
          ERRNO,
          RETCODE
      IF RETCODE < 0
          PERFORM WRITERR-EXIT
      END-IF.
WRITERR-EXIT.
      MOVE SOC-FUNCTION TO ERRMSG(26:15)
      EXEC CICS SEND TEXT FROM(ERRMSG)
          LENGTH(LENGTH OF ERRMSG)
      END-EXEC
      EXEC CICS RETURN END-EXEC.

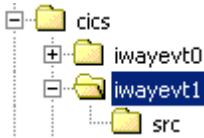
```

## IWAYEVT1 Program

The IWAYEVT1 sample program demonstrates sending an XML document to the iWay Transaction Adapter for CICS from a CICS/TX sockets interface. The program will echo the return document to the CICS terminal.

On Windows, the IWAYEVT1 sample program is located in:

C:\Program Files\iWay7\etc\samples\cics\iwayevt1



The following supporting file is provided:

**src**

- iwayevt1.css - The source code for the sample program written in C.

The text version of the source code file is included below for your review:

```

#include <cics_api.h>
#include <cics_eib.h>
/*****
/* IWYEVT1 - This sample program demonstrates sending an xml          */
/* document to the iWay CICS adapter from CICS/TX. The program will  */
/* echo the return document to the CICS terminal.                    */
/*                                                                    */
/* Author John Schlesinger                                          */
/* Version 1.0                                                       */
/*****
/* Updated 22-Feb-2006 - Lori Pieper                                 */
/* Changes include:                                                 */
/*                                                                    */
/* 1) In main, removed the declaration of                           */
/* int cics_api_temp_var = cics_api_edf_init_c_extended(0, &CicsArgs); */
/* since it seemed to be generated for us which caused a syntax    */
/* error when compiling the code.                                    */
/*                                                                    */

/* 2) Also in main, changed:                                        */
/* // SendData("168.135.63.14", 4773);                               */
/* to use the proper C language comment syntax as the former was   */
/* causing syntax errors.                                           */
/*                                                                    */
/* 3) To the includes section, added                                */
/* #include<netinet/in.h>                                           */
/* since this is needed in order to define sockaddr_in, which is   */
/* being used in the SendData subroutine.                            */
/*                                                                    */
/*****
/* #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<netinet/in.h>
#include<arpa/inet.h>
#include<sys/socket.h>
#endif
/*****
/* #defines */
/*****
#define InsertCursor '\x13' /* 3270 Insert cursor */
#define ebdic_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
{
    ics_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 */
/*****

```

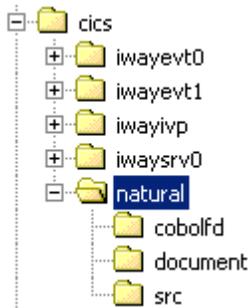
## Natural Program

The NATURAL sample program returns one or many records within a range from the Employee file.

For more information on how to use Adabas/Natural programs with the iWay Transaction Adapter for CICS, see [Using Adabas/Natural Programs](#) on page 93.

On Windows, the Natural sample program is located in:

`C:\Program Files\iWay7\etc\samples\cics\natural`



The following supporting files are provided:

### **cobolfd**

- aasnatin.cbl - Input record for the service.
- aasnatin\_out.cbl - Output record for the service.

### **document**

- aasnatin.xml - The input XML document.

### **src**

- aasnatin.natural - The sample natural program.

The text version of the source code file is included below for your review:

```

/*****/
/* AASNATN - SAMPLE NATURAL PROGRAM FOR IWAY CICS ADAPTER. */
/* RETURN ONE OR MANY RECORDS WITHIN A RANGE FROM THE */
/* EMPLOYEE FILE. */
/* */
/* NATURAL PROGRAMS RUNNING WITH THE IWAY CICS ADAPTER ARE */
/* INVOKED INDIRECTLY VIA THE PROXY PROGRAM, AASNATC. */
/* */
/* ALL ACCESS TO INPUT/OUTPUT BUFFERS ARE HANDLED BY CALLS TO */
/* THE DATA MOVER PROGRAM, AASSUBC, DESCRIBED BELOW. */
/* */
/* COMMUNICATION WITH THE DATA MOVER PROGRAMS REQUIRE THE */
/* CONTROL BLOCK: */
/*     1 #REQUEST-PARMS */
/*     2 #FUNCTION      (A2) GT,PT,LC,LI */
/*     2 #OFFSET       (I2) DATA OFFSET OF INPUT/OUTPUT */
/*     2 #LENGTH       (I2) LENGTH OF DATA TO GET OR PUT */
/*     2 #RESPONSE-CODE (I4) */
/*     2 #ERR-MESSAGE  (A72) */
/* */
/* THE IMPLEMENTED FUNCTIONS ARE: */
/* GT - GET INPUT BY OFFSET. */
/*     THE OFFSET MUST BE SET TO 0 FOR THE FIRST CALL ONLY!! */
/*     THE OFFSET IS INCREMENTED FOLLOWING EACH CALL BY THE */
/*     DATA MOVER PROGRAM */
/*     THE LENGTH OF THE REQUESTED DATA MUST BE PROVIDED FOR */
/*     EACH CALL. IT SHOULD MATCH THE LENGTH OF THE AREA */
/*     PROVIDED TO RECEIVE THE INPUT DATA */
/*     IE:  MOVE #FUNC-GT TO #FUNCTION */
/*           MOVE 8 TO #LENGTH */
/*           MOVE 0 TO OFFSET - FIRST CALL ONLY */
/*           CALL 'AASSUBC' #FUNCTION #EMP-NUM1 */
/* */
/*           WHERE #EMP-NUM1 IS AN EIGHT BYTE FIELD */
/* */
/* PT - PUT OUTPUT BY OFFSET */
/*     THE OFFSET MUST BE SET TO 0 FOR THE FIRST CALL ONLY!! */
/*     THE OFFSET IS INCREMENTED FOLLOWING EACH CALL BY THE */
/*     DATA MOVER PROGRAM */
/*     THE LENGTH OF THE REQUESTED DATA MUST BE PROVIDED FOR */
/*     EACH CALL. IT SHOULD MATCH THE LENGTH OF THE AREA */
/*     PROVIDED CONTAINING THE OUTPUT DATA */
/*     IE:  MOVE #FUNC-PT TO #FUNCTION */
/*           MOVE 8 TO #LENGTH */
/*           MOVE 0 TO OFFSET - FIRST CALL ONLY */
/*           CALL 'AASSUBC' #FUNCTION #EMP-NUM1 */
/* */
/*           WHERE #EMP-NUM1 IS AN EIGHT BYTE FIELD */
/* */

```

```

/* LI - GET LENGTH OF INPUT DATA                                */
/*   UPDATES THE #LENGTH FIELD WITH THE TOTAL LENGTH OF        */
/*   ALL INPUT PARMS.  NO OTHER PARMS ARE NEEDED              */
/*   IE: MOVE #FUNC-LI TO #FUNCTION                             */
/*       CALL 'AASSUBC' #FUNCTION                               */
/*                                                             */
/* LC - GET LENGTH OF COMMAREA USED TO SEND DATA.            */
/*   UPDATES THE #LENGTH FIELD WITH THE COMMAREA LENGTH.      */
/*   NO OTHER PARMS ARE NEEDED                                 */
/*   IE: MOVE #FUNC-LC TO #FUNCTION                             */
/*       CALL 'AASSUBC' #FUNCTION                               */
/*                                                             */
/* IMPLEMENTED RESPONSE-CODES:                                  */
/* 0 - OPERATION COMPLETED SUCCESSFULLY                       */
/* 4 - ENDDATA - OFFSET EXCEEDS END OF INPUT DATA.           */
/*   INDICATES NO FURTHER INPUT IS AVAILABLE                  */
/* 8 - ENDBUFF - OFFSET + LENGTH IS GREATER THEN COMMAREA     */
/*                                                             */
/* NOTE:                                                        */
/*   ALWAYS USE THE FIRST FIELD OF A GROUP WHEN CALLING       */
/*   DATA MOVER PROGRAMS.  NOTICE ALL CALLS ARE MADE WITH  */
/*   #FUNCTION NOT #REQUEST-PARMS.                             */
/*                                                             */
/*   ALL INPUT PARMS MUST BE PROCESSED BEFORE OUTPUT.        */
/*-----*/

```

```

DEFINE DATA LOCAL
1 #FUNC-TYPE
  2 #FUNC-GT (A2) INIT<'GT'>
  2 #FUNC-PT (A2) INIT<'PT'>
  2 #FUNC-LC (A2) INIT<'LC'>
  2 #FUNC-LI (A2) INIT<'LI'>
1 #REQUEST-PARMS
  2 #FUNCTION      (A2) /*GT,PT,LC,LI
  2 #OFFSET        (I2) /*DATA OFFSET OF INPUT/OUTPUT
  2 #LENGTH        (I2) /*LENGTH OF DATA TO GET OR PUT
  2 #RESPONSE-CODE (I4)
  2 #ERR-MESSAGE   (A72)
1 #PARMS
  2 #EMP-NUM1      (A8)
  2 #EMP-NUM2      (A8)
1 EMPLOY-VIEW VIEW OF EMPLOYEES1
  2 PERSONNEL-ID
  2 FIRST-NAME
  2 NAME
  2 MAR-STAT
  2 SEX
  2 BIRTH
  2 DEPT
  2 JOB-TITLE
  2 CURR-CODE(1:5)
  2 SALARY(N9/1:5)
1 #ERROR-PARMS
  2 #NATPROG (A8)
  2 #NATMSG  (A65)
  2 #NATERR  (A7)
END-DEFINE
/* USE LI FUNCTION TO THE GET LENGTH OF INPUT PARAMETERS */
MOVE #FUNC-LI TO #FUNCTION
CALL 'AASSUBC' #REQUEST-PARMS
IF #LENGTH LT 16 /*REQUIRED FOR THIS PROGRAM*/
THEN TERMINATE
END-IF
/* USE GET FUNCTION TO RETRIEVE DATA PARMS */
MOVE 8 TO #LENGTH
MOVE 0 TO #OFFSET
MOVE #FUNC-GT TO #FUNCTION
CALL 'AASSUBC' #FUNCTION #EMP-NUM1
CALL 'AASSUBC' #FUNCTION #EMP-NUM2
MOVE 0 TO #OFFSET
MOVE 147 TO #LENGTH
MOVE #FUNC-PT TO #FUNCTION

```

```

FIND ALL EMPLOY-VIEW WITH PERSONNEL-ID = #EMP-NUM1 THRU #EMP-NUM2
  IF NO RECORDS FOUND
    MOVE *PROGRAM TO #NATPROG
    MOVE ' REQUESTED EMPLOYEE NUMBERS NOT IN THE DATABASE' TO #NATMSG
    MOVE 80 TO #LENGTH
    MOVE #FUNC-PT TO #FUNCTION
    CALL 'AASSUBC' #FUNCTION #NATPROG
    TERMINATE
  END-NOREC
  CALL 'AASSUBC' #FUNCTION PERSONNEL-ID
END-FIND
ON ERROR
  MOVE *PROGRAM TO #NATPROG
  DECIDE FOR FIRST CONDITION
    WHEN *ERROR-NR = 1106
      MOVE ' EMPLOYEE NUMBER IS TOO LARGE. 8 BYTES IS '
        TO #NATMSG
      MOVE 'THE MAX' TO #NATERR
    WHEN *ERROR-NR = 3061
      MOVE ' INVALID EMPLOYEE NUMBER RANGE SPECIFIED '
        TO #NATMSG
      MOVE ' ' TO #NATERR
    WHEN NONE
      MOVE ' HAS DETECTED THE FOLLOWING ERROR NUMBER: '
        TO #NATMSG
      MOVE *ERROR-NR TO #NATERR
  END-DECIDE
  MOVE 80 TO #LENGTH
  CALL 'AASSUBC' #FUNCTION #NATPROG
  TERMINATE
END-ERROR
END

```

## Transaction Adapter for CICS Debugging and Troubleshooting

---

This section includes tips and techniques for debugging the adapter.

**In this appendix:**

- [Transaction Adapter for CICS Troubleshooting](#)
  - [CICS Data Type Conversions](#)
- 

### Transaction Adapter for CICS Troubleshooting

Certain situations may cause the adapter to return error messages. This topic describes error messages, possible causes, and solutions.

**Error message**

An Error has occurred running {program name}: CICS has returned no CommArea

**Accompanied by:**

```
at com.ibi.tcpappc.DirectTCPConnection.execute(DirectTCPConnection.java:nnn)
at com.ibi.cics.CICSIPServer.execute(CICSIPServer.java:nmm)
at com.ibi.cics.CICSConnection.execute(CICSConnection.java:nmm)
at com.ibi.cics.CICSAdapter.process(CICSAdapter.java:nnn)
at
com.iwaysoftware.ibse.iwse.Adapter2Runner.process(Adapter2Runner.java:nnn)
at com.iwaysoftware.ibse.iwse.Adapter2Runner.<init>(Adapter2Runner.java:nmm)
at
com.iwaysoftware.ibse.iwse.XDSOAPRouter.handleAdapter(XDSOAPRouter.java:nnn)
at com.iwaysoftware.ibse.iwse.XDSOAPRouter.process(XDSOAPRouter.java:nnn)
at com.iwaysoftware.ibse.iwse.IBSEServlet.doPost(IBSEServlet.java:nnn)
```

where:

*nnn*

Is the line number in the Java program.

**Possible Cause:** Occurs when a program abends.

**Solution:** Verify that the input request document accurately describes the input required for the program and that the COBOL description matches what the program is using.

**Error message**

Unable to execute Transaction {program name}

**Accompanied by:**

{applid of region} While performing an attach for node {nodename} a security violation was detected.

**Possible Cause:** Occurs when the password sent to CICS is lowercase.

**Solution:** Change the password from lowercase to uppercase.

**Error message**

Unable to execute Transaction {program name}

**Accompanied by:**

VTAM and CICS error message:

VTAM RETURN CODE 1001 SENSE CODE 8004 0000

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

**Solution:** Flush DNS.

**Procedure: How to Flush DNS**

To flush DNS:

1. At the command prompt, ping the host address for the machine where the adapter 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 type the following command:

```
nbtstat -A 172.30.166.90
```

The following image shows the DOS screen that lists the host addresses when the nbtstat command is executed.

```
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

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
    ----                -
    PGMMGY               <00> UNIQUE            Registered
    EDA                  <00> GROUP            Registered
    PGMMGY               <20> UNIQUE            Registered
    EDA                  <1E> GROUP            Registered
    PGMMGY               <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.

## CICS Data Type Conversions

When the adapter parses the COBOL structure, it creates the following field values in the byte array it either creates or receives. The values are visible in the log, for example, the CANTV log.

```
<outputFields>
  <fldData>
    <recfd="C:\Operaciones_ADMSAP\IWAY_ADAPTERS_FILES\OLRCOMU_OUT.FD">
      <field fdtype="6" type="string" length="3" name="CODIGOTRANSACCION" />
      <field fdtype="6" type="string" length="8" name="LUGARTRANSACCION" />
      <field fdtype="6" type="string" length="4" name="CODIGOAREA" />
      <field fdtype="6" type="string" length="7" name="TELEFONO" />
      <field fdtype="6" type="string" length="6" name="FECHAULTIMAFACTURACION" />
      <field fdtype="5" type="string" length="13" name="DEUDAMES" />
      <field fdtype="6" type="string" length="6" name="FECHAINDETERMINADA" />
      <field fdtype="5" type="string" length="13" name="SALDOVENCIDO" />
      <field fdtype="6" type="string" length="76" name="FILLER1" />
      <field fdtype="5" type="string" length="13" name="DEUDATOTAL" />
      <field fdtype="6" type="string" length="30" name="NOMBRECLIENTE" />
      <field fdtype="6" type="string" length="1" name="TIPOCLIENTE" />
      <field fdtype="6" type="string" length="2" name="CODIGORETORNO" />
      <field fdtype="6" type="string" length="1" name="TIPOSERVICIO" />
      <field fdtype="6" type="string" length="8" name="FECHAVENCIMIENTOFACTURA" />
      <field fdtype="6" type="string" length="4" name="FILLER2" />
      <field fdtype="6" type="string" length="1" name="TIPOTELEFONO" />
      <field fdtype="6" type="string" length="44" name="FILLER3" />
    </recfd>
  </fldData>
</outputFields>
```



## Feedback

*Customer success is our top priority. Connect with us today!*

---

Information Builders Technical Content Management team is comprised of many talented individuals who work together to design and deliver quality technical documentation products. Your feedback supports our ongoing efforts!

You can also preview new innovations to get an early look at new content products and services. Your participation helps us create great experiences for every customer.

To send us feedback or make a connection, contact Sarah Buccellato, Technical Editor, Technical Content Management at [Sarah\\_Buccellato@ibi.com](mailto:Sarah_Buccellato@ibi.com).

To request permission to repurpose copyrighted material, please contact Frances Gambino, Vice President, Technical Content Management at [Frances\\_Gambino@ibi.com](mailto:Frances_Gambino@ibi.com).

# iWay

## / iWay Transaction Adapter for CICS User's Guide

Version 7.0.x and Higher

DN3502277.0418

Information Builders, Inc.  
Two Penn Plaza  
New York, NY 10121-2898