Information Builders

iWay

iWay Transaction Adapter for IMS User's Guide

Version 7.0.x and Higher

August 16, 2018

Contents

Preface

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

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*.

How This Manual Is Organized

This manual includes the following chapters:

	Chapter/Appendix	Contents
1	Introducing the Transaction Adapter for IMS	Introduces the adapter environment.
2	Configuring the iWay Transaction Adapter for IMS	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 create iWay Business Services, which expose functionality as web services.
4	Event Processing With the IMS Adapter	Describes how to configure and use IMS Events.
A	Configuring the Transaction Adapter for IMS in an iWay Environment	Describes how to configure the adapter in the iWay Service Manager.
B	Using the IMS Sample Programs	Describes how to use the sample IMS programs (IWTEST01, IWTEST05, IWTEST10, IWAYEVT0, and IWAYEVT2) that are provided with the iWay Transaction Adapter for IMS.
C	Sample Requests, Schemas, and COBOL File Descriptions	Provides documents and schemas for the sample transaction, PART and the COBOL descriptions used as input for the sample IMS/TM transactions.
D	Transaction Adapter for IMS 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.
this typeface	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.
underscore	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 (...).
. . .	Indicates that there are (or could be) intervening or additional commands.

Related Publications

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

Customer Support

Do you have any 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 and documentation. 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.

Platform	
Operating System	
OS Version	
JVM Vendor	
JVM Version	

The following table lists the deployment information our consultants require.

Adapter Deployment	For example, JCA, Business Services Provider, iWay Service Manager
Container	For example, WebSphere

Version	
Enterprise Information System (EIS) - if any	
EIS Release Level	
EIS Service Pack	
EIS Platform	

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

iWay Adapter	
iWay Release Level	
iWay Patch	

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).	

Request/Question	Error/Problem Details or Information
Any change in the application environment: software configuration, EIS/database configuration, application, and so forth?	
Under what circumstance does the problem *not* 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. Please use the Reader Comments form at the end of this document to communicate your feedback to us or to suggest changes that will support improvements to our documentation. You can also 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.

Chapter 1

Introducing the Transaction Adapter for IMS

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

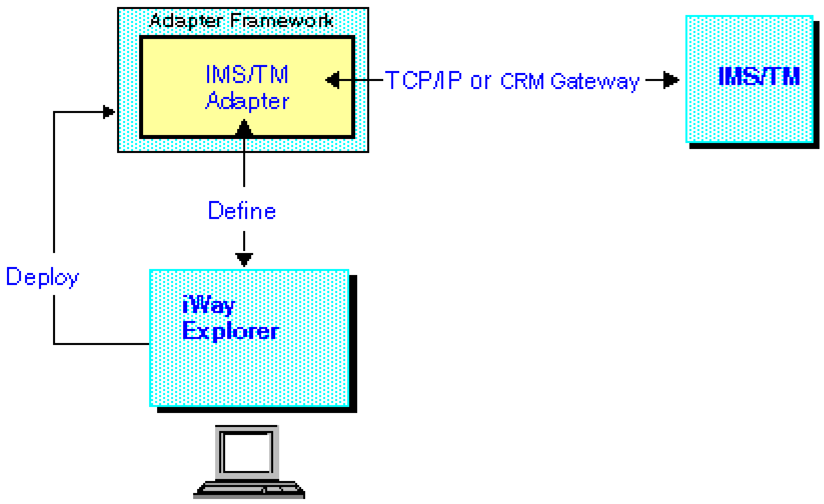
In this chapter:

iWay Transaction Adapter for IMS Overview

The iWay Transaction Adapter for IMS enables you to execute IMS/TM transactions. The advantages of the adapter include the following:

- ❑ No modification required to existing IMS/TM transactions.
- ❑ No installation of new code required on IMS/TM.
- ❑ Adapter processing performed off of the mainframe.
- ❑ Configuration by metadata—no coding required.
- ❑ Support for older versions of IMS/TM.
- ❑ Support for IMS/TM transactions.

The following diagram illustrates the framework for executing IMS/TM programs with the iWay Explorer and the iWay Transaction Adapter for IMS.

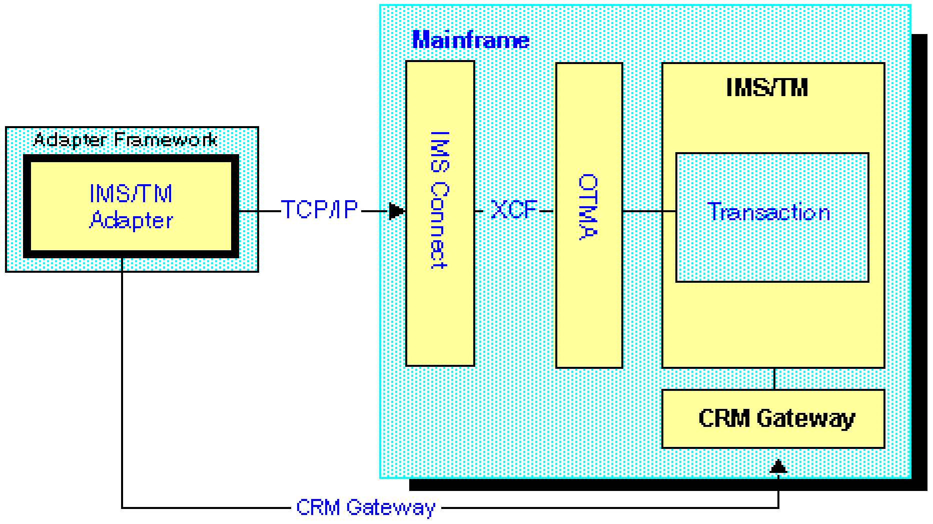


The following bidirectional scenarios are supported by the adapter:

- IMS/TM services
- IMS/TM events

Understanding the iWay Transaction for IMS

The following diagram illustrates the adapter connection to the mainframe using TCP/IP.



The adapter is the component that connects to IMS/TM. It is hosted in a container that supports events. The adapter enables the following functions:

- ❏ Connecting to IMS/TM.
- ❏ Executing IMS/TM transactions.
- ❏ Mapping XML messages to and from IMS/TM data structures.

The adapter enables you to invoke an IMS/TM transaction by sending a request and retrieving the response.

IMS Connect was introduced with IMS Version 7.1 to execute IMS/TM transactions from a TCP/IP client.

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

IMS/TM Transactions

There are two types of IMS/TM transactions:

- Non-conversational.
- Conversational (for example, when a user interacts with a terminal screen (3270).

The iWay Transaction Adapter for IMS can process conversational transactions when using TCP/IP. To execute 3270 conversational programs, you require a screen scraper, such as the iWay Emulation Adapter for 3270. For many years, IMS/TM applications were structured so that the business processing, as opposed to the screen dialogue, was in non-conversational transactions. In many cases, executing a non-conversational transaction is recommended for application integration. When the application program must be integrated in a conversation, and multiple requests are presented across a single process, the iWay Transaction Adapter for IMS can maintain the processing required for this IMS conversational transaction. Screen-scraping for 3270 cannot be involved in this integration because the adapter provides application integration only.

Local transactions have limited support. IMS resources are not involved in the local transactions.

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

Software Requirements for the Adapter

This section lists the software requirements for the iWay Transaction Adapter for IMS:

- z/OS Version 1.2, 1.3, 1.4, 1.5, 1.6, and 1.7.

For TCP/IP:

- IMS Version 4, 5, 6, 7, 8, and 9.
- IMS Connect and Open Transaction Manager Access (OTMA) installed and configured on the remote IMS/TM system.

Deployment Information for the iWay Transaction Adapter for IMS

The iWay Transaction Adapter for IMS 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 adapter connections, create web services, and configure event capabilities. Service Manager can access this configuration information through the iWay 7.0 Service Manager repository to create a robust integration solution.

When the adapter is hosted in a third party application server environment, iWay Explorer, used to configure IMS/TM connections, create web services, and configure event capabilities, can be configured to work in a web services environment in conjunction with the iBSP.

Deployment Information Roadmap

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

Deployed Component	For more information, see
iWay Service Manager	Chapter 6 of this guide *iWay Service Manager User's Guide*
iWay Explorer	Chapters 2 and 3 of this guide *iWay Installation and Configuration*
iWay Business Services Provider (iBSP)	*iWay Installation and Configuration*

iWay Service Manager

iWay Service Manager is the heart of the Universal Adapter Framework. It is an open transport service bus that uses graphical tools to create metadata from target applications, transform and map interfaces, and manage stateless processes to create sophisticated integration services without writing custom integration code. 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 an explorer metaphor to browse the IMS/TM 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 IMS/TM. External applications that access IMS/TM through the iWay Transaction Adapter for IMS use either XML schemas or web services to pass data between the external application and the adapter.

iWay Business Services Provider

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

Web services is a distributed programming architecture that solves Enterprise Application Integration (EAI) hurdles that other programming models cannot. It enables programs to communicate with one another using a text-based, platform and language independent message format called XML.

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.

Chapter 2 Configuring the iWay Transaction Adapter for IMS

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 IMS/TM.

In this chapter:

Starting iWay Explorer (Java Servlet)

The following section describes how to start iWay Explorer (Java Servlet).

Procedure: How to Start iWay Explorer (Java Servlet)

To start iWay Explorer:

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

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

   where:

   `hostname`

   Is the server where Application Explorer the server is installed.

   `port`

   Is the port number where the server listens. The default port is 807001.

   The 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 screen. From left to right they are:

   ❑ iWay Adapters, where you create and manage connections to IMS/TM.

   ❑ iWay Events, where you configure 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 will define and modify adapter functions and services.

  The Available Hosts drop-down list specifies the Servlet iBSP instance you can connect to.

You are now ready to define a new target to IMS/TM.

Configuring a Connection to IMS/TM

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

The following target types are available when you configure a connection to IMS/TM:

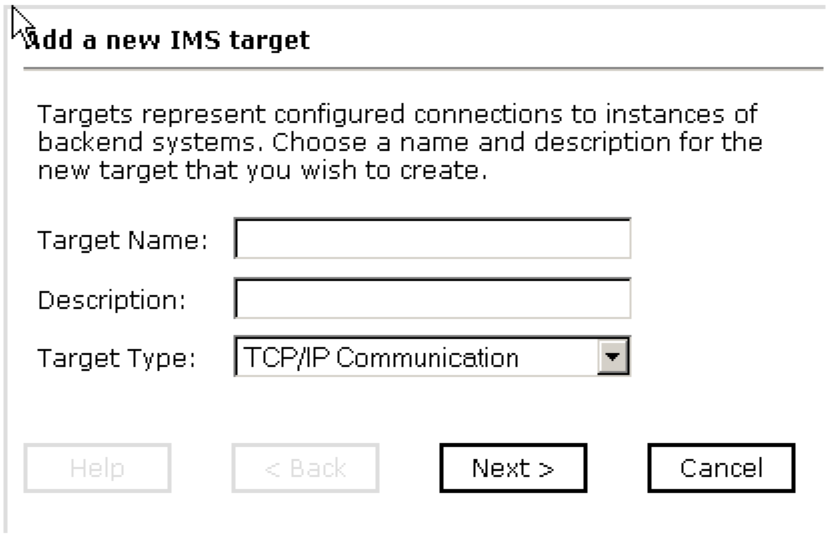
- TCP/IP Communication
- CRM Gateway

Procedure: How to Configure a TCP/IP Connection to IMS/TM

To configure a TCP/IP connection to IMS/TM:

1. In the left pane of iWay Explorer, expand the *iWay Adapters* node.
2. Select the *IMS* node.
3. In the right pane, move your pointer over Operations and select *Define a new target*.

The Add a new IMS 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 IMS 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: TCP/IP Communication

a. In the Target Name field, type a name for the connection.

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

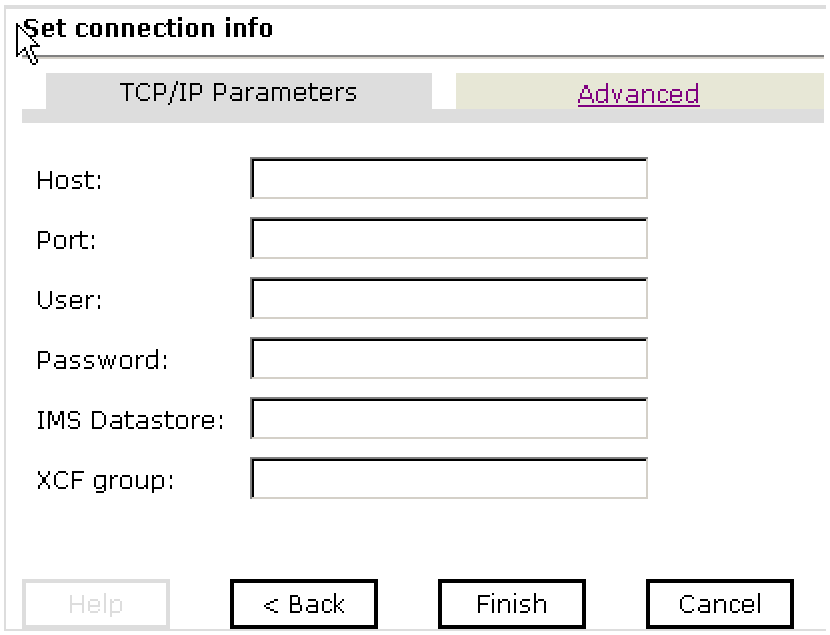
b. In the Description field, type a description for the target name you just created.

c. From the Target Type drop-down list, select *TCP/IP Communication* as the type of target.

4. Click *Next*.

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

The following image shows the Set connection info dialog box, which opens to the TCP/IP Parameters tab window and contains six entry fields. The following step describes the entries needed.



Note: The IMS connection parameters are consistent with those found in your IMS system. For more information on parameter values that are specific to your IMS configuration, consult your IMS system administrator.

5. Enter the TCP/IP parameters to configure a new connection to IMS/TM.

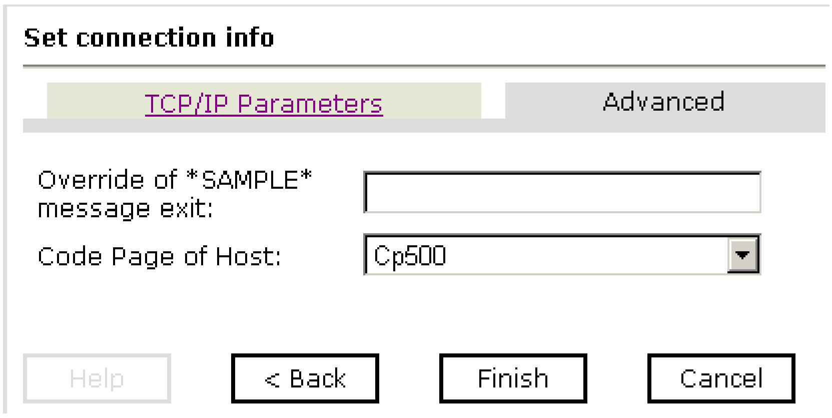
   You can obtain this information from the IMS/TM Systems Administrator. This information should be the same for all programs in a single IMS/TM system.

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

Parameter	Description
Host	Host name, or IP address, for the computer where IMS/TM is running.
Port	Port number on which IMS Connect is listening.
User	Valid user ID for IMS/TM.
Password	Valid password associated with the IMS/TM user ID.

Parameter	Description
IMS Datastore	Name of the IMS/TM datastore.
XCF Group	Name of the XCF group.

6. If you click the Advanced tab, the following parameters appear, as shown in the following image.



Override of *SAMPLE* message exit:

Code Page of Host: Cp500

Enter the Advanced parameters. The following table lists and describes the Advanced parameters.

Parameter	Description
Override of *SAMPLE* message exit	Specify a message exit if you want to override the sample message exit.
Code Page of Host	Select the codepage from the drop-down menu. Cp500 is the default value.

7. Click *Finish*.

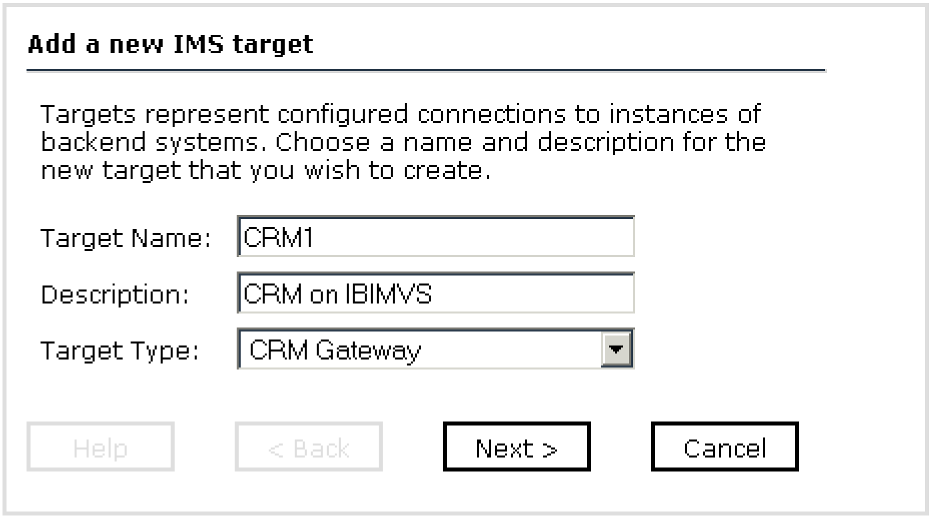
The newly created connection, TCPIP_Connection, appears as a node under the IMS service adapter. The configuration information is stored in the repository for the configuration you defined during installation.

Procedure: How to Configure a CRM Gateway Connection to IMS/TM

To configure a CRM gateway connection to IMS/TM:

1. In the left pane of iWay Explorer, expand the *iWay Adapters* node.
2. Select the *IMS* node.
3. In the right pane, move your pointer over Operations and select *Define a new target*.

   The Add a new IMS 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 IMS 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: CRM1

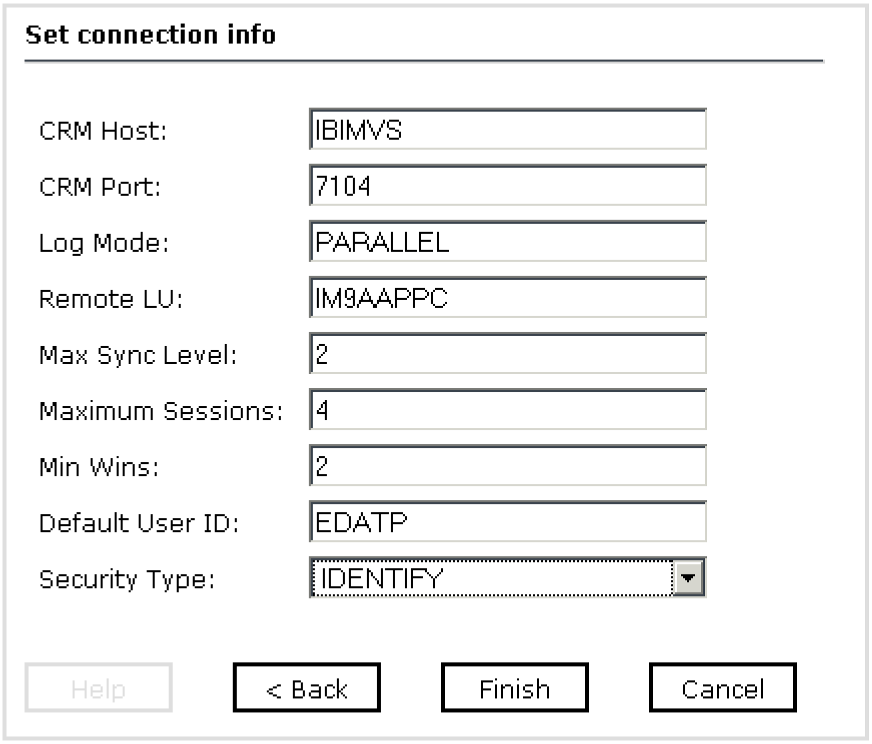
Description: CRM on IBIMVS

Target Type: CRM Gateway

   a. In the Target Name field, type a name for the connection (for example, CRM1).
   b. In the Description field, type a description for the target name you just created (for example, CRM on IBIMVS).
   c. From the Target Type drop-down list, select *CRM Gateway*.

4. Click *Next*.

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


Set connection info

CRM Host:	IBIMVS
CRM Port:	7104
Log Mode:	PARALLEL
Remote LU:	IM9AAPPC
Max Sync Level:	2
Maximum Sessions:	4
Min Wins:	2
Default User ID:	EDATP
Security Type:	IDENTIFY

5. Enter the CRM gateway parameters to configure a new connection to IMS/TM.

   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.

Parameter	Description
Log Mode	Defines the characteristics of the APPC sessions that the CRM gateway establishes across a CRM Link. 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. Use the same logon mode for the CRM gateway logical unit and the back-end application logical unit.
Remote LU	LU of IMS.
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	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 *minimum winner sessions*. The owner of these pre-allocated sessions has priority for its winner sessions. However, they may be reassigned depending on system load. 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. 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.
Default User ID	If IDENTIFY is selected from the Security Type drop-down list, this value represents the user ID that is transmitted to the LU.

Parameter	Description
Security Type	The security level of a CRM Link defines what security credentials are required for all requests that utilize this link. The following options are available from the drop-down list: ❑ **LOCAL** (No security credentials provided) In the simplest case, no security credentials are provided with requests sent on the link. This implies that the user has been authenticated in the originating domain, and that no further authentication is required. The value for the CRM would be *LOCAL*. The corresponding value for the back-end system must be defined in the connection definitions. ❑ **IDENTIFY** (User ID is provided) It is possible that the application processing the request needs to identify the user making the request for access control to application resources. For this case, only a user ID is provided with the request. The value for the CRM would be *IDENTIFY*. The corresponding value for the back-end system must be defined in the connection definitions.

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

❑ 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.

6. Click *Finish*.

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

Connecting to an IMS/TM Target

You can connect to a defined IMS/TM target (for example, TCPIP_Connection) from iWay Explorer.

Procedure: How to Connect to a Defined IMS/TM Target

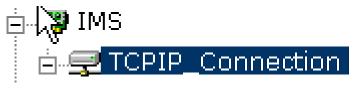
To connect to a defined IMS/TM target:

1. Expand the *iWay Adapters* node.
2. Expand the *IMS* node.
3. Click the target name (for example, TCPIP_Connection) under the IMS node.
4. Move your pointer over Operations and select *Connect*.

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

5. Verify your connection parameters. If required, provide the password and then click *OK*.

   The x icon disappears, indicating that the node is connected.



Managing a Connection to IMS/TM

To manage IMS/TM 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.

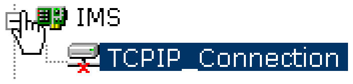
Procedure: How to Disconnect From a Connection to IMS/TM

To disconnect from a connection to IMS/TM:

1. Expand the *iWay Adapters* node.
2. Expand the *IMS* node.
3. Click the connection, for example, *TCPIP_Connection*, move your pointer over *Operations*, and select *Disconnect*.

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

The x icon appears, indicating that the node is disconnected.

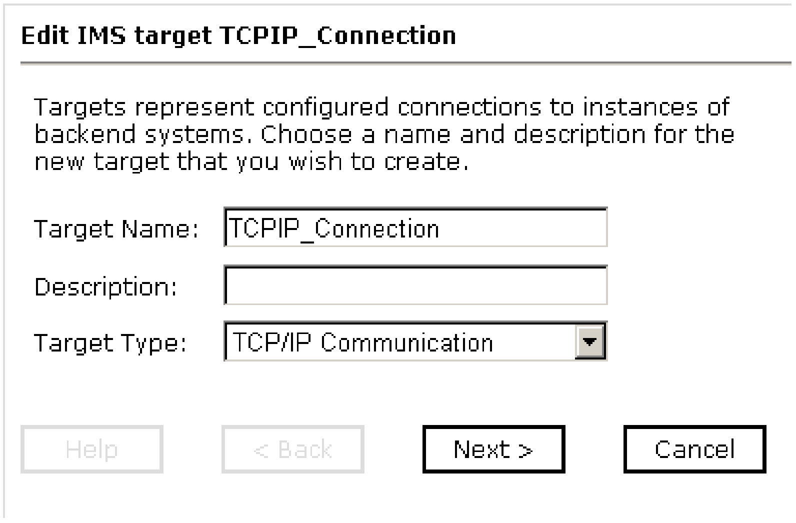


Procedure: How to Edit a Connection to IMS/TM

To edit a connection to IMS/TM:

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

   The Edit dialog box opens in the right pane containing three fields (Target Name, Description, and Target Type) and two action buttons (Next and Cancel).

   Edit IMS target TCPIP_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: TCPIP_Connection

   Description:

   Target Type: TCP/IP Communication

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

   The Set connection info dialog box opens in the right pane containing the TCP/IP Parameters and Advanced tabs.

5. Modify the information as needed and then click *Finish*.

Procedure: How to Delete a Connection to IMS/TM

To delete a connection to IMS/TM:

1. Expand the *iWay Service Adapter*s node.
2. Expand the *IMS* node.
3. Click the connection, for example, *TCPIP_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.

Chapter 3 Creating XML Schemas and iWay Business Services

This section describes how to use iWay Explorer (Java Servlet) to create IMS/TM transactions and generate request and response XML schemas for new or existing transactions. These schemas are used to represent a transaction for integration with external systems.

In addition, this section provides information on how to use the generated schemas to create iWay Business Services, which expose functionality as web services.

In this chapter:

Creating an Adapter Transaction

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

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

For example, the IMS transaction PART connects to IMS/TM and returns PART information on successful adapter installation and configuration. One request and response schema is applicable for this generic transaction. The request schema for the generic transaction is in *Sample Requests, Schemas, and COBOL File Descriptions* on page 115.

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 IMS segment must be in the <message> tag, which can be a maximum of 32,500 bytes.

The generic response schema is constructed from the data received from IMS/TM. If the <message> element has more than 80 bytes, the received IMS segment is split into 80-byte messages. Illegal XML characters ('<', '/', and '&') are converted to XML entities. For example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<IMS>
  <Transaction tpname="PART" noreply="NO">
    <messages>
      <message>*</message>
    </messages>
  </Transaction>
</IMS>
```

For transactions that require input and output and a formatted response, which is usually the case, you must add your own adapter transactions, as described in *How to Create an Adapter Transaction* on page 40. XML request messages must specify the transaction to use in the location attribute of the <Transaction> tag. For example, if you create an IMS/TM transaction called PART, the location is "IMS/Transactions/PART".

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 115.

Specifying IMS/TM Transaction Input and Output

The iWay Transaction Adapter for IMS can execute regular IMS/TM transactions and transactions that use Message Format Services (MFS). Therefore, the input to, and output from, IMS/TM transactions can be specified by using COBOL copybooks or MFS XML descriptors. Any combination of the two can be used, provided that only one method is used to specify the input and only one method is used to specify the output.

Input	**Output**
COBOL copybook	COBOL copybook
COBOL copybook	MFS XML descriptor
MFS XML descriptor	COBOL copybook
MFS XML descriptor	MFS XML descriptor

Note: Only field format 1 (OPT=1 on the MSG macro definition) is supported. Short records, allowed under OPT 1, are not currently supported. In addition, only one segment can be sent when using MFS.

Specifying the Input and Output For MFS IMS/TM Transactions

The following section describes how to specify the input and output for MFS IMS/TM transactions. Examples are also provided.

Using an MFS XML Descriptor to Define IMS/TM Transaction Input

The iWay Transaction Adapter for IMS uses an MFS XML input descriptor to create the proper IMS/TM transaction input record from the input XML document. All segments that are sent to IMS are full length; there are no short fields or short segments. The input MFS XML specifies how to format each element in the input XML document before adding it to the IMS/TM input record by specifying the attributes of the MFLD element.

When using an input XML document, the adapter does not use the value that is specified in the Program Name field of the Add Service dialog box in iWay Explorer. The IMS/TM transaction code must be specified in some other manner.

The first MFLD element in the MFS XML input descriptor must describe the IMS/TM transaction code used to invoke the IMS/TM program. The transaction code itself must either appear in the input XML document, as the value of the element whose name is equal to the value of the name attribute of the first MFLD element in the MFS XML input descriptor, or it must appear as the value of the default attribute of the first MFLD element in the MFS XML descriptor.

Example: Input MFS XML Descriptor

The following is a sample input MFS XML descriptor.

```
<?xml version="1.0" encoding="UTF-8" ?>
<MSG name="IW01MI" opt="1" type="INPUT" next="IW01MO">
  <SEG>
     <MFLD name="TCODE"
         Length="9"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="no"
         fill=" "
         InputMFSDocumentUsage="Required" />

     <MFLD name="CSRPOS"
         Length="4"
         iwayFieldType="cursor"
         default="7,2"
         InputMFSDocumentUsage="Forbidden" />
```

```
        <MFLD name="BCODE"
            Length="12"
            iwaySuppress="no"
            iwayFieldType="alpha"
            justification="right"
            attribute="yes"
            fill="0"
            InputMFSDocumentUsage="Optional" />
      </SEG>
   </MSG>
```

The following table lists and describes the attribute types in the input MFS XML descriptor.

Attribute	Description
name	Specifies the name of the field (Required).
Length	Specifies the length of the field (Required). It includes 2 for attributes if attribute="yes". For the *cursor* type, it must always be set to 4.
InputMFSDocumentUsage	Specifies the input disposition (Optional). This attribute specifies the disposition of the field from the input XML document. If absent, the field is always included in the data that is returned to IMS. The following values are available: ❏ *Required* - The field must be included in the inbound document. ❏ *Forbidden* - The field must not be included in the inbound document. The data that is sent to IMS is supplied from the default attribute. ❏ *Optional* - The field can be included in the inbound document. If it is not, the data sent to IMS is supplied from the *default* attribute.
iwaySuppress	This attribute is reserved for future use (Optional).
iwayFieldType	The following values are available: ❏ *alpha* (default) ❏ *cursor*

Attribute	Description
attribute	Specifies whether 3270 attribute bytes are included (Required). The following values are available: ❏ *yes* ❏ *no* (default) If *yes* is selected, the length specified by the Length attribute includes these 2 bytes.
justification	Specifies left or right justification. The following values are available: ❏ *left* (default) - Indicates left justify and right truncate. ❏ *right* - Indicates right justify and left truncate.
fill	Specifies the pad byte. The default is " ". For example, if the attribute is missing or specified as "" (empty string).
default	Specifies the default value of the field when the field is omitted from the document. The default must be in a format that is acceptable for the type specified by the *iwayFieldType* attribute. For *alpha* type fields, the default, if specified, is populated with the fill byte, and justified according to the justification attribute. **Note:** For *alpha* type fields, conceptually, the fill byte is set first. Then the value of the field is set. The field is then padded, justified, and truncated. If the value specified for the field in the input document is null (<field/>), the default is not used; the field is populated with the pad byte. The *default* attribute is required if the *InputMFSDocumentUsage* attribute is set to *Forbidden*.

If the MFLD element describes the IMS/TM transaction code, and the CRM gateway is used as the communication method, only the following attributes are used:

❏ name

- length
- InputMFSDocumentUsage
- default

If TCP/IP communication method is used, the other attributes are used in the expected manner.

Notes Regarding iwayFieldType="cursor"

The expected format of type cursor elements in the input MFS XML document is "row,column". The row and the column must be integers. Only the following attributes, if present, are used from the MFLD element:

- name
- Length
- iwayFieldType
- default

The other attributes may be present, but are not used.

If the cursor element is not in the input document, or if its value is null (for example, <csr/>), the default is used instead, unless there is no default, in which case the IMS "1,2" default cursor position is used. Whichever values are used, they are converted to 2 byte integers before being inserted in the record that is sent to IMS.

Example: **Input MFS XML Descriptor I**

```
<MFLD name="SomeField"
      Length="10"
      iwaySuppress="no"
      iwayFieldType="alpha"
      attribute="no"
      fill=" "
      default="WHATEVER" />
```

Input XML: <SomeField>nothing</SomeField>
Sent to IMS: 'nothing '

Input XML: <SomeField></SomeField> (or < SomeField/>)
Sent to IMS: ' ' (10 blanks)

Input XML: <SomeField>nothingofimportance</SomeField>
Sent to IMS: 'nothingofi' (10 chars)

Input XML: Field not in the input XML
Sent to IMS: 'WHATEVER ' (10 chars)

Example: **Input MFS XML Descriptor II**

```
<MFLD name="csr"
      Length="4"
      iwayFieldType="cursor"
      default="10,23" />
```

Note: Only the type, and the default attributes are used. The cursor values sent to IMS are 2 byte integers, not strings.

Input XML: <csr>3,17</csr> Sent to IMS: the 2 half integers 3, and 17.

Input MFS: <csr/> Sent to IMS: the 2 half integers 10, 23.

Using an MFS XML Descriptor to Define IMS/TM Transaction Output

The iWay Transaction Adapter for IMS uses an MFS XML input descriptor to convert the records received from IMS/TM to an XML message properly. Currently, it is assumed that IMS sends full length segments every time. If a short segment is sent from IMS, it will result in a length mismatch error.

Attribute	Description
name	Specifies the name of the XML element that will be generated.
Length	Specifies the length of the field in the IMS output.
attribute	Specifies whether 3270 attribute bytes are included (Optional). The following values are available: ❑ *yes* ❑ *no* (default) If *yes* is selected, the length specified by the Length attribute includes these 2 bytes.

Attribute	Description
OutputMFSDocumentUsage	Specifies the output disposition. This attribute specifies the disposition of the field. If absent, the field is always included in the output. The following values are available: ❑ *Always* - The field is always included in the output. ❑ *Never* - The field is omitted from the output. ❑ *IfNotDefault* - The field is included in the output if it is different from the value of the *default* attribute, or if there is no *default* attribute.
iwayFieldType	The following values are available: ❑ *alpha* ❑ *cursor* If *cursor* is used, two 2-byte integers are read and converted in a string of the form "row,column".
default	Used in conjunction with the *OutputMFSDocumentUsage* attribute.

COBOL Descriptions for Defining IMS/TM Transaction Output

The adapter uses COBOL descriptions to properly create an XML structure of the message that is returned from the IMS/TM transaction. For transactions that return one or more messages (for example, a message that has its own message layout), the adapter transforms the message into the proper structure based on COBOL descriptions.

If the application that executes the adapter requires proper formatting of each returned message from the transaction:

- ❑ You must examine all possible outputs of the transaction so you can create COBOL descriptions correctly.

  As a reference, it helps to use the COBOL description of the output message or the MFS message output descriptor area for the transaction to create the COBOL descriptions.

- You must supply COBOL descriptions of each type of output using iWay Explorer when configuring the transaction.

  Note: For generic transactions where the format of the output is of no consequence, you are not required to supply COBOL descriptions.

  For the PART transaction, you require three COBOL descriptions. A RECTYPE field in each description is used by the adapter to determine which of the three messages is being returned by the adapter.

Sample Transaction PART

IBM supplies the PART transaction with an IMS system. This document uses the PART transaction for illustration purposes and as a reference for the adapter. The PART transaction accepts an input part number with a length of 17 characters or less. Based on what is passed to the PART transaction, an answer set is returned from the DP21PART database.

- If a part number is passed and found in IMS, the transaction returns detail information.
- If an asterisk ('*') is passed in the request, the transaction returns all part numbers in the database with their descriptions.
- If the part number is not found in IMS, the message, "PART NOT FOUND," is returned.

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

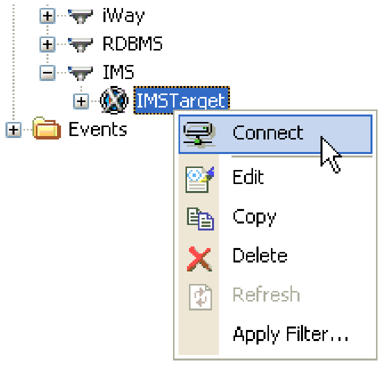
Sample request documents, including sample response schemas for the PART transaction, are in *Sample Requests, Schemas, and COBOL File Descriptions* on page 115. You specify the output as explained in *Creating an Adapter Transaction* on page 31. You must know the field in the COBOL description that 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.

The previous is also true for events to determine the layout returned from IMS when you configure an IMS event. For more information on configuring IMS events, see *Event Processing With the IMS Adapter* on page 59.

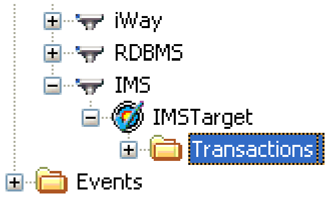
Procedure: How to Create an Adapter Transaction

Perform the following steps to create an adapter transaction.

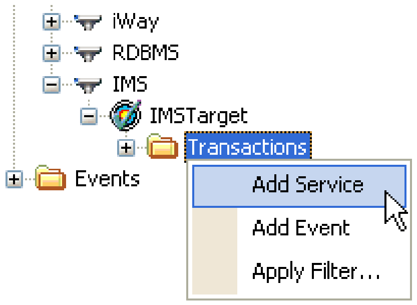
1. Expand the *IMS* adapter node and connect to an available IMS target (for example, IMSTarget).



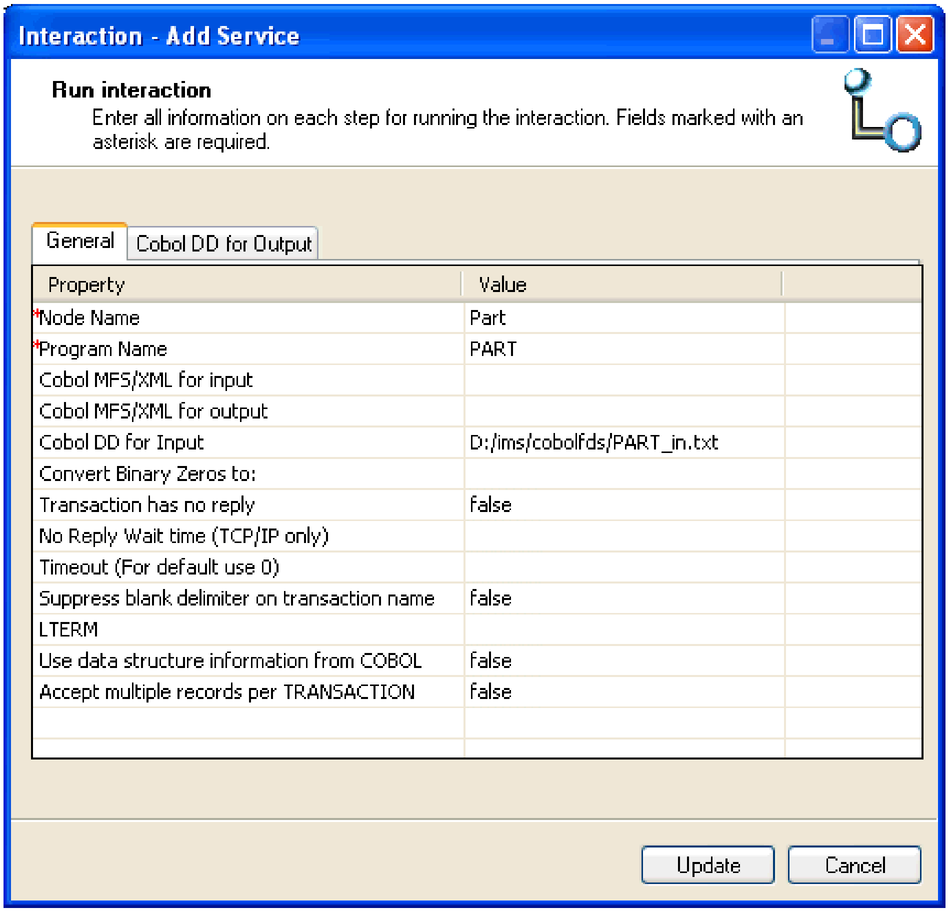
The Transaction node appears under the connected target.



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 COBOL DD for Output). The following image shows the parameters in the General tab that will enable you to map the COBOL descriptions for the IMS/TM transaction.



Run interaction

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

Property	Value
*Node Name	Part
*Program Name	PART
Cobol MFS/XML for input	
Cobol MFS/XML for output	
Cobol DD for Input	D:/ims/cobolfds/PART_in.txt
Convert Binary Zeros to:	
Transaction has no reply	false
No Reply Wait time (TCP/IP only)	
Timeout (For default use 0)	
Suppress blank delimiter on transaction name	false
LTERM	
Use data structure information from COBOL	false
Accept multiple records per TRANSACTION	false

3. To map the COBOL descriptions for the IMS/TM transaction, type values for the parameters in the General tab, as defined in the following table.

Parameter	Description
Node Name	Name to describe the adapter transaction you create. This name, for example, Part, appears under the Transactions node for the current connection. The name to use in the <Transaction location="..."> attribute.

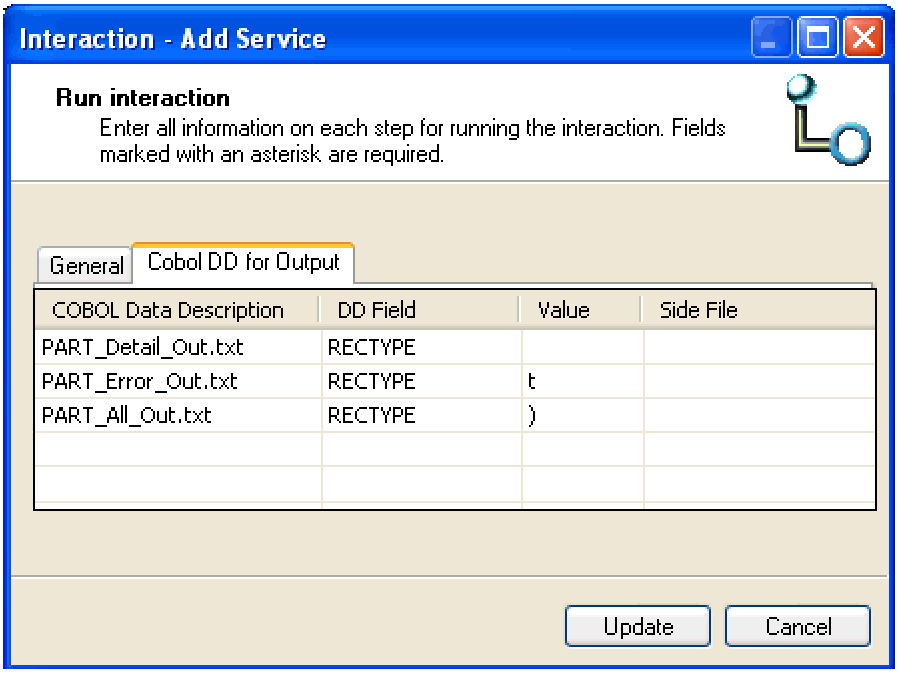
Parameter	Description
Program Name	Name of the program to be called in IMS/TM, for example, PART. The PART input DD is shown in *Sample Requests, Schemas, and COBOL File Descriptions* on page 115.
COBOL MFS/XML for input	Specifies the MFS XML input descriptor. For more information, see *Specifying IMS/TM Transaction Input and Output* on page 32.
COBOL MFS/XML for output	Specifies the MFS XML output descriptor. For more information, see *Specifying IMS/TM Transaction Input and Output* on page 32.
COBOL DD for Input	Location of the COBOL description that describes the input parameters of the IMS transaction to execute. Converted by the adapter to an XML schema that the adapter uses to map from XML to the format required by IMS/TM at run time.
Convert Binary Zeros to	Character to convert binary zeros to in output.
Transaction has no reply	Set this parameter to *true* when you do not want to wait for a response from the program.
No Reply Wait time (TCP/IP only)	Specify a wait time, in milliseconds, for IMS to send an error message during service interactions defined for IMS/TCPIP targets that use TCPIP connections when the *Transaction has no reply* check box is selected.

Parameter	Description
Timeout (For default use 0)	Specifies, in hundredths of a second, how long the adapter waits for a response before assuming IMS will not respond. The minimum is two seconds. To get the default value, enter 0. For TCP/IP connections, the default is in the IMS Connect configuration file. For CRM Gateway connections, the default is one minute. **Note:** The implementation of the Timeout parameter depends on whether the adapter connects to IMS using a TCP/IP connection, or using the CRM Gateway. If using a TCP/IP connection, the adapter uses the IMS Connect timeout service. It indicates the length of the timeout by setting the IRM_TIMER field in the IRM header to an appropriate value. The IRM_TIMER field is only one byte long, and the actual mapping from the IRM_TIMER value to the actual wait time is best explained in the *IMS Connect Guide and Reference* manual. The adapter maps the user-specified timeout to the IRM_TIMER value that would result in the wait time closest to the value entered. If using a CRM Gateway, the adapter sets the socket timeout, and the user-specified value is the value used for the timeout, without any modifications.
Suppress blank delimiter on transaction name	When this parameter is set to *true*, and the target uses COBOL copybooks, the adapter sends all eight character transaction codes to IMS Connect without appending a trailing blank. This parameter only applies to TCP/IP targets, and it only affects transactions with eight character names. Shorter transactions keep their trailing blank regardless of this setting. It has no effect on CRM Gateway targets. By default this parameter is set to *false*.
LTERM	Logical terminal. Blank is the default unless specified by the user.

Parameter	Description
Use data structure information from COBOL	When this parameter is set to *true*, the adapter creates request and response schemas that reflect COBOL group levels (for example, 05, 10, 20, and so on). The COBOL grouping will be reflected in the XML request and response schemas. You must enable this parameter when COBOL input or output descriptions contain the COBOL OCCURS statement. When this parameter is enabled and the program COMMAREA contains an OCCURS statement, the output COBOL definition must also contain an OCCURS statement. Do not "flatten out" the COBOL output description, since the adapter relies on the number of OCCURS when formulating the program output.
Accept multiple records per TRANSACTION	When this parameter is set to *true*, multiple COMMAREA records are read by the PreParser. Otherwise, any data in excess of the COBOL definition is truncated.

Note: You must transfer the COBOL descriptions to a location accessible to iWay Explorer. For the correct COBOL descriptions to use for the program, contact your IMS/TM Administrator or application developer.

4. Click the *COBOL DD for Output* tab, as shown in the following image.


Interaction - Add Service
Run interaction
Enter all information on each step for running the interaction. Fields marked with an asterisk are required.
General
Cobol DD for Output
Update
Cancel

COBOL Data Description	DD Field	Value	Side File
PART_Detail_Out.txt	RECTYPE		
PART_Error_Out.txt	RECTYPE	t	
PART_All_Out.txt	RECTYPE	)	

The COBOL DD for Output tab allows you to specify the path that corresponds to the message you want returned from the IMS/TM transaction. 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 46.

If the transaction can return multiple types of messages for each output COBOL description, enter the COBOL description field and value to determine the schema that is created and used 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 PART populates the field called RECTYPE. Depending on program logic, iWay Explorer creates the correct response schema.

Value in RECTYPE Field	COBOL Description
space ' '	PART_Detail_Out
't'	PART_Error_Out
parenthesis ')'	PART_All_Out

The PART_Detail_Out, PART_Error_Out, and PART_All_Out COBOL descriptions appear in *Sample Requests, Schemas, and COBOL File Descriptions* on page 115.

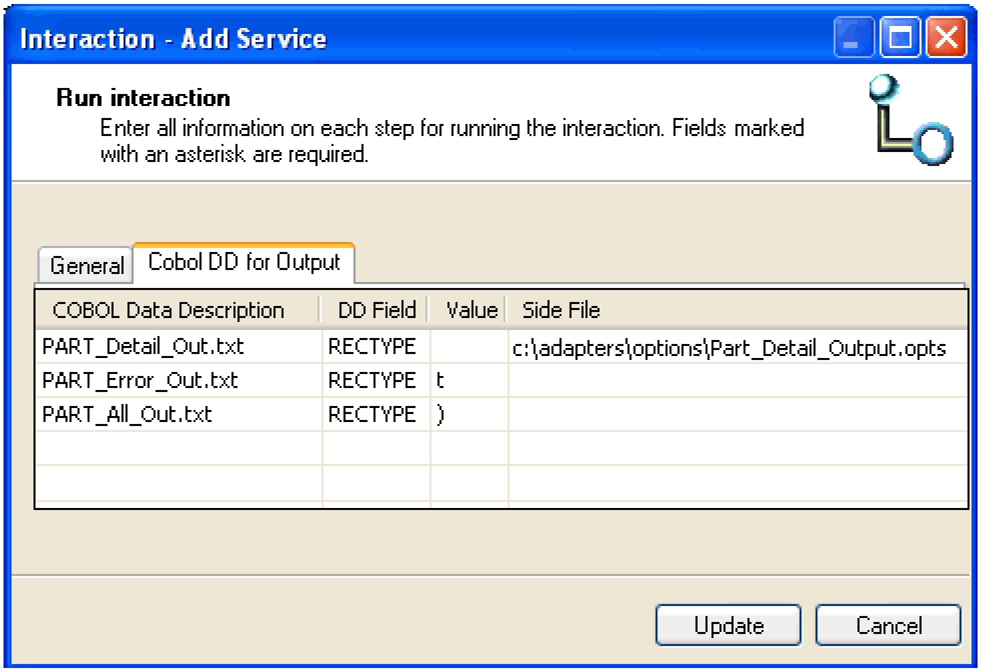
5. Click *Update* when you have finished configuring all the parameters.

   The new IMS/TM transaction 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 COBOL DD for Output tab in the Add Service dialog for the iWay Transaction Adapter for IMS. A side file is specified in the Side File column.


Interaction - Add Service
Run interaction
Enter all information on each step for running the interaction. Fields marked with an asterisk are required.
General
Cobol DD for Output
Update
Cancel

COBOL Data Description	DD Field	Value	Side File
PART_Detail_Out.txt	RECTYPE		c:\adapters\options\Part_Detail_Output.opts
PART_Error_Out.txt	RECTYPE	t	
PART_All_Out.txt	RECTYPE	)	

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

IMS Connect Considerations: When the IMS Connect returns data (that is, an IMS Segment) it prefixes the data with LL, which represents the length of the returned data, and ZZ, which is a reserved field of binary zeros (00). The LL is called the IRM_LEN and is the length of the IRM structure, while ZZ is called IRM_RSV and is a reserved field.

The format of the IRM structure is

```
LLZZDATA
```

where:

`LL`

Is the total length of this segment.

`ZZ`

Are binary zeros (00).

`DATA`

Is the IMS transcode followed by transaction data.

Important: The adapter does not require these prefixes in the COBOL input and output definitions. Only describe the data portion in the COBOL input and output definitions.

Example: COBOL Descriptions

You must use the following syntax for binary and packed 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 IMS 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:

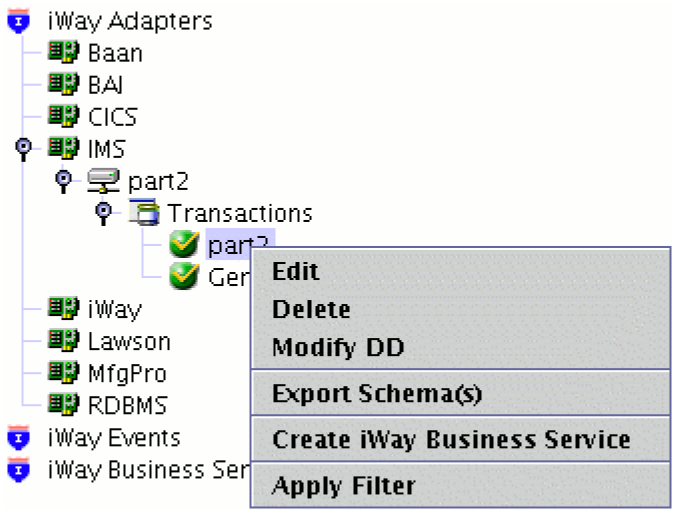
```
CHARS PIC X 4

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

Procedure: How to Modify COBOL DD Field Definitions

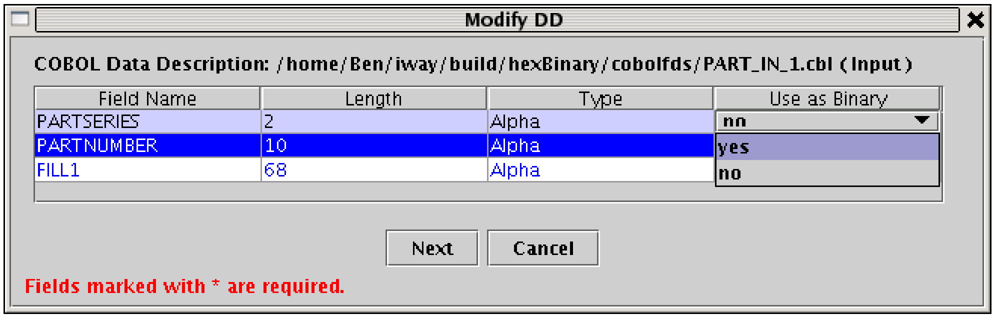
To modify COBOL DD field definitions:

1. Open iWay Explorer and connect to your IMS target.
2. Expand the *Transactions* node and select a defined transaction.



3. 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.

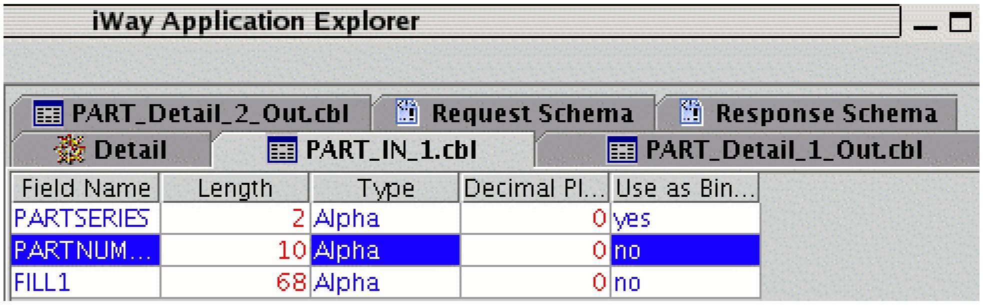


COBOL Data Description: /home/Ben/iway/build/hexBinary/cobolfds/PART_IN_1.cbl (Input)

Field Name	Length	Type	Use as Binary
PARTSERIES	2	Alpha	no
PARTNUMBER	10	Alpha	yes
FILL1	68	Alpha	no

Fields marked with * are required.

4. Select the Alpha field in the table that you want to use as Binary.
5. Click the *Use as Binary* column and select *yes* or *no* from the drop-down list.
6. Click *Next* to browse through the remaining input and output record layouts for the transaction.
7. 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.

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 in the Sun Java System Application Server installation directory, for example,

```
C:\Program Files\iway7\config\base\wsdl\schemas\service\IMS
\TCPIP_Connection
```

where:

TCPIP_Connection

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

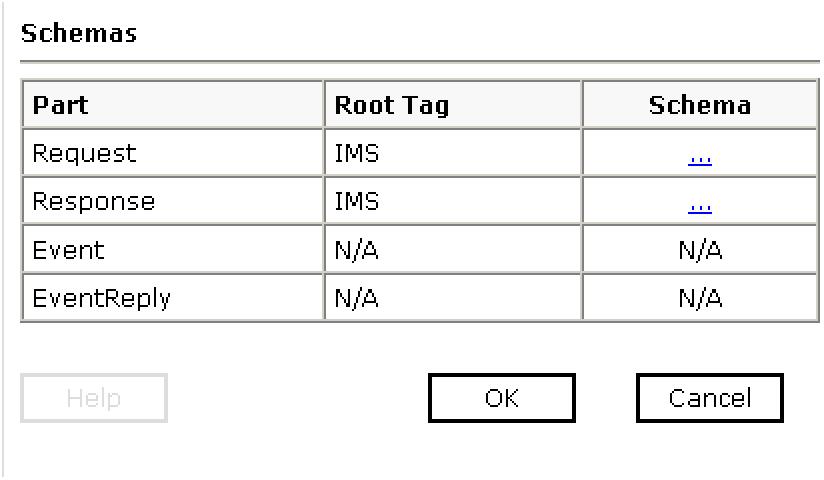
Procedure: How to Create Schemas for an Adapter Transaction

The following procedure describes how to create request and response 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*.

   Note: 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 115.

The Schemas table appears in the right pane. The following image shows the Schemas window which is a table of four rows and three columns, Part, Root Tag and Schema. Only the first two rows are applicable, Request and Response under the Part column that have IMS as the value located under the Root Tag column and clickable ellipsis under the Schema column.


Schemas

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

a. To view the Request schema, click the ellipsis symbol that is located in the third column of the Request row.

b. To view the Response schema, click the ellipsis symbol that is located in the third column of the Response row.

3. Click *OK* to exit the Schemas window.

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 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.

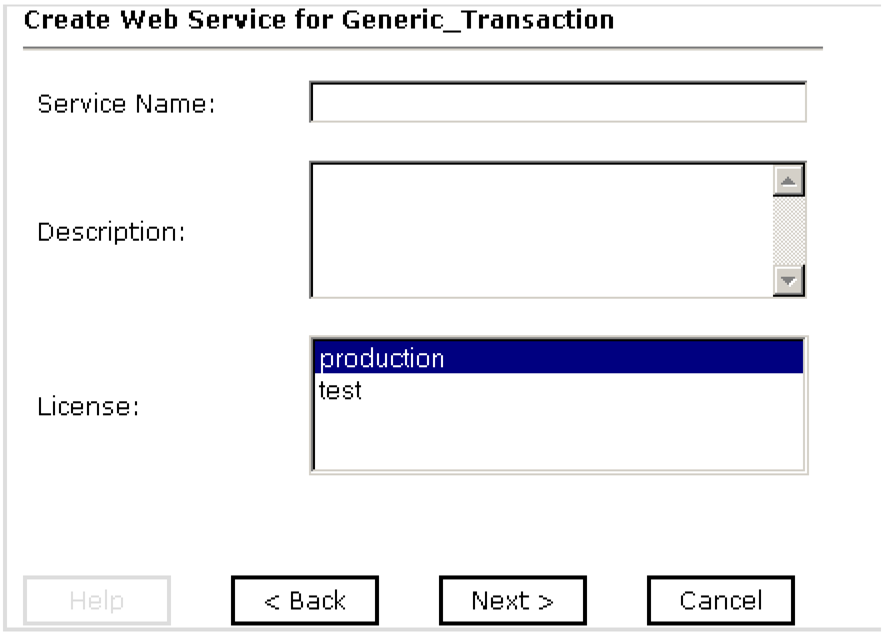
   The iWay Adapters window opens.

2. In the left pane, expand the *IMS* node.
3. Connect to an *IMS* target (for example, TCPIP_TCPIP_Connection).
4. Expand the node to which you connected.

   The Transaction node appears under the connected node.

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

The Create Web Service dialog box opens in the right pane. The following image shows the Create Web Service for Generic_Transaction dialog box which contains three fields for entry.


Create Web Service for Generic_Transaction

Service Name:

Description:

License: production, test

7. Enter information that is specific to the web service you are defining.

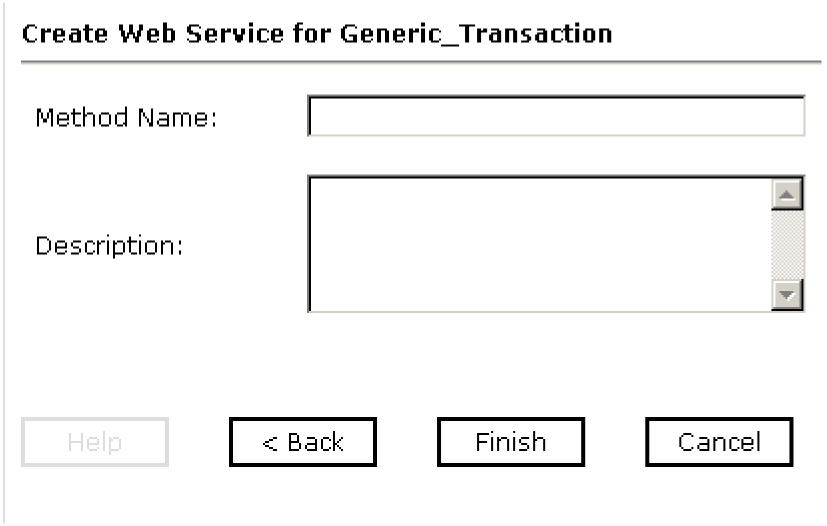
   a. In the Service Name field, type a descriptive name for the web service.

   b. In the Description field, type a brief description for the web service (optional).

   c. In the License field, select one or more license codes to assign to the web service. To select more than one, hold down the *Ctrl* key and click the licenses.

8. Click *Next*.

Another dialog box with the Method Name and Description fields opens. The following image shows another Create Web Service for Generic_Transaction dialog box that have two fields for entry.



9. Enter information that is specific to the method you are defining.
   a. In the Method Name field, type a descriptive name for the method.
   b. In the Description field, type a brief description for the method.
10. 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 a business service is created, test it to ensure that it functions properly. iWay Explorer provides a test tool for testing the business service.

Procedure: How 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.
2. If it is not expanded, expand the list of business services under iWay Business Services.
3. Expand the *Services* node.

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

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

5. In the right pane, click the named business services.

   The test option appears in the right pane.

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

   ```
   <?xml version="1.0" encoding="UTF-8" ?>
   <IMS>
    <Transaction location="/IMS/Transaction/PART">
      <message>AN960C10</message>
    </Transaction>
   </IMS>
   ```

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

7. 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.

Procedure: How to Generate WSDL From a Web Service

1. If you are not already in 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 Service for which you want to generate WSDL.

3. Select the business service.

   The hyperlink for the service appears in the right pane.

4. Right-click *Service Description* and choose *Save Target As*.

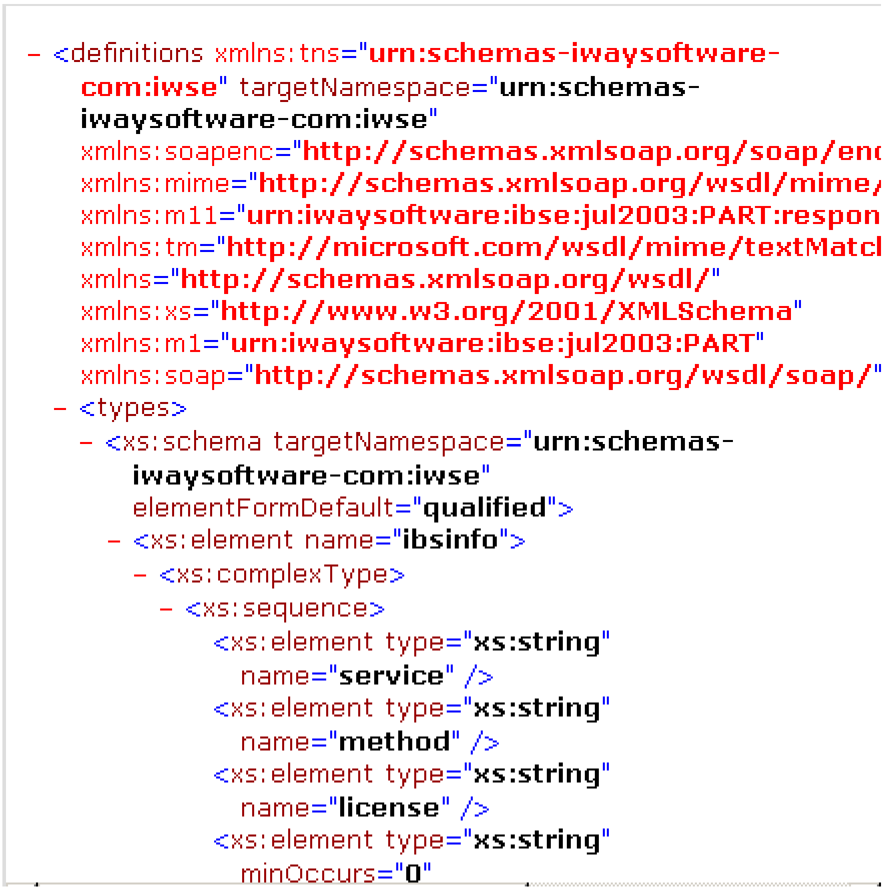
5. Choose a location for the file and specify .wsdl for the extension.

   Note: The file extension must be .wsdl.

6. Click *Save*.

Example: **Viewing WSDL Generated From a Web Service**

After generating a WSDL file from the PART.ibs serialized object, the PART.wsdl file looks similar to the following image which is a sample XML file.



```
- <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:PART:respon
    xmlns:tm="http://microsoft.com/wsdl/mime/textMatcl
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:m1="urn:iwaysoftware:ibse:jul2003:PART"
    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"
```

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 IMS. The user name and password values that you provided for IMS 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:ibsinfo 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:ibsinfo>
</SOAP-ENV:Header>
```

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

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

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

Chapter 4

Event Processing With the IMS Adapter

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

In this chapter:

Understanding IMS Events

An IMS Event is the processing defined on a particular message received from IMS. An IMS 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.

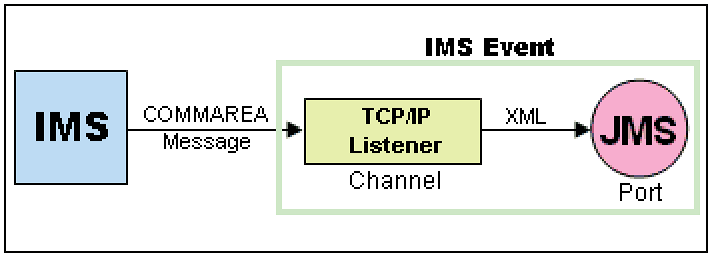
All processing details for a message are configured under a channel, which include:

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

An IMS 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:

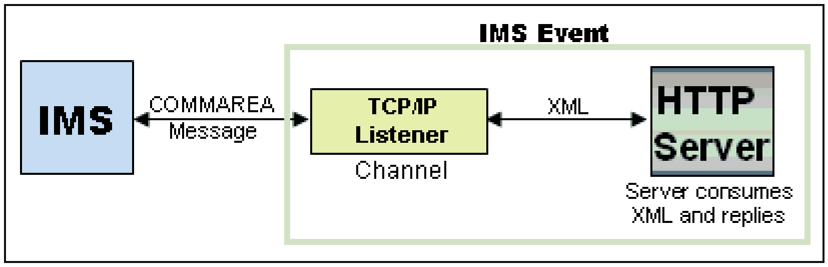
REQUEST



The following sequence outlines how request messages are processed.

1. A message is created by an IMS 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 an IMS 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 IMS 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 property:

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

Supported Environments

IMS Event handling is supported under Servlet iBSP.

Configuring IMS Events

IMS Events are configured using a design time tool such as iWay Explorer.

To configure an IMS Event, you must:

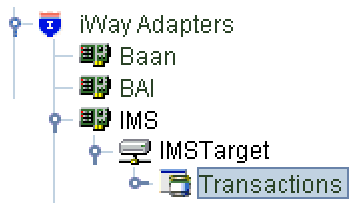
1. Create metadata in the form of an XML schema if you are using COBOL Copybooks.
2. Create one or more ports for use as message endpoints.
3. Create a channel to receive messages and associate it with ports and COBOL Copybooks.

Procedure: How to Create XML Schemas From COBOL Copybook Metadata

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

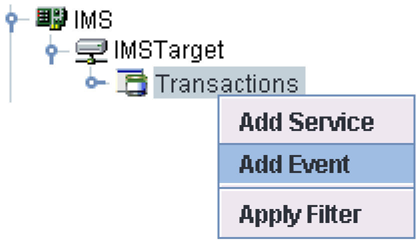
Note: You can skip this section if your message is in XML format.

1. Connect to your iBSP.
2. Create and connect to an IMS target.

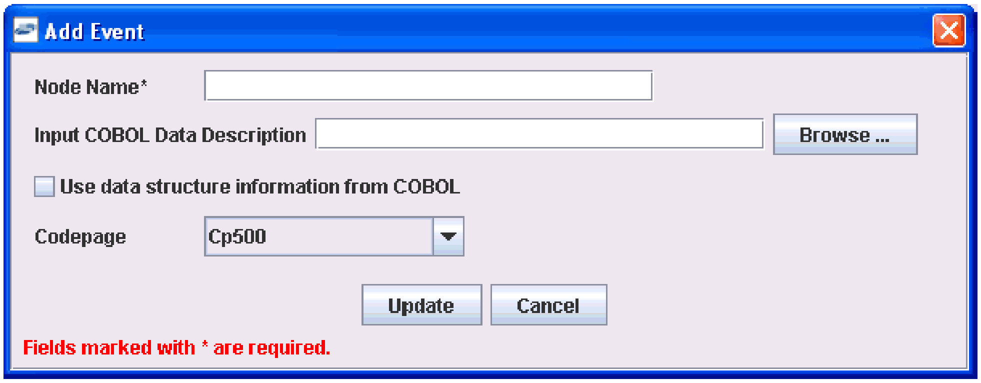


Note: The connectivity being used here is outbound connectivity (IMS adapter), and not strictly necessary for Events.

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



The Add Event dialog box opens.

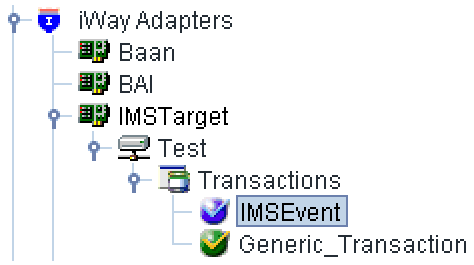

Add Event
Node Name*
Input COBOL Data Description
Browse ...
Use data structure information from COBOL
Codepage
Cp500
Update
Cancel

Fields marked with * are required.

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 IMS 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, IMSEvent, 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.
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

You can create, edit, or delete an Event port using iWay Explorer.

You create a port for an IMS Message from the iWay Adapters tab or from the iWay Events tab. You can use an iBSE using the Available Hosts drop-down list in iWay Explorer.

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

- File
- iBSP

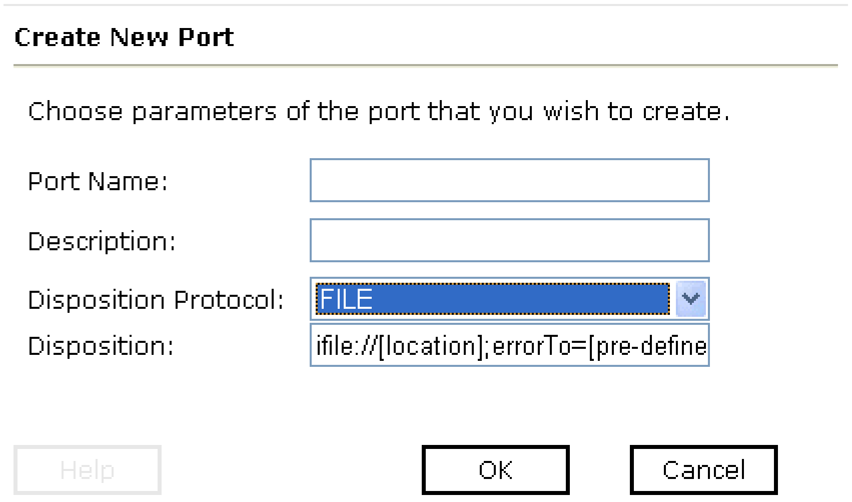
- MSMQ
- JMSQ
- SOAP
- HTTP
- MQ Series

Procedure: How to Create an Event Port for File

To create an event port for File:

1. Click the *iWay Events* tab.
2. In the left pane, expand the *IMS* 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 where you choose parameters for the new port.

   Create New Port

   Choose parameters of the port that you wish to create.

   Port Name:

   Description:

   Disposition Protocol: FILE

   Disposition: ifile://[location];errorTo=[pre-define

   a. In the Port Name field, type a name for the event.

      Note: Ensure that you specify a name that conforms to standards set by IMS. For example, when using IMS, periods are not allowed. You must remove all instances of this character.

b. In the Description field, type a brief description.

c. From the Disposition Protocol drop-down list, select *FILE*.

d. In the Disposition field, type a File destination to which 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 describes the parameters for the disposition.

Parameter	Description
location	Destination and file name of the document where event data is written, for example, `ifile://D:\in\x.txt;errorTo=ifile://D:\error.`
errorTo	Predefined port name or another disposition URL where error logs are sent.

5. Click *OK*.

   The event 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. Click the *iWay Events* tab.
2. In the left pane, expand the *IMS* 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.

   a. In the Port Name field, type a name for the event.

   b. In the Description field, type a brief description.

   c. From the Disposition Protocol drop-down list, select *iBSE*.

   d. In the Disposition field, enter an iBSE destination in the form of:

```
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 defines the parameters for the disposition.

Parameter	Description
svcName	Name of the service created with iBSE.
mthName	Name of the method created for the business service.
responseTo	Location where responses to the business service are posted. Predefined port name or another full URL. Optional.
errorTo	Location where error documents are sent. Predefined port name or another full URL. Optional.

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.

Procedure: How to Create a Port for the MSMQ Disposition

To create a port for an MSMQ disposition using iWay Explorer:

1. Click the *iWay Events* tab.
2. In the left pane, expand the *IMS* 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.

   a. In the Port Name field, type a name for the event.

   b. In the Description field, type a brief description.

   c. From the Disposition Protocol drop-down list, select *MSMQ*.

   d. In the Disposition field, enter a MSMQ destination in the form of:

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

   The following table defines the parameters for the disposition.

Parameter	Description
host	Machine name where the Microsoft Queuing system is running.
Queue Type	For private queues, enter *Private$*. 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.
qName	Name of the private queue where messages are placed.
errorTo	Location where error documents are sent. Predefined port name or another full URL. Optional.

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.

Procedure: How to Create an Event Port for the JMS Queue Disposition

To create a port for a JMS queue disposition using iWay Explorer:

1. Click the *iWay Events* tab.
2. In the left pane, expand the *IMS* 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.

   a. In the Port Name field, type a name for the event.

   b. In the Description field, type a brief description.

   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 defines the parameters for the disposition.

Parameter	Description
queue	Name of a queue to which events are emitted.
Connection Factory	A resource that contains information about the JMS Server.
jndi_url	The URL to use to contact the JNDI provider. The syntax of this URL depends on the JNDI provider that is used. This value corresponds to the standard JNDI property: `java.naming.provider.url`
jndi_factory	Is JNDI context.INITIAL_CONTEXT_FACTORY and is provided by the JNDI service provider.
user	User ID associated with this queue.
password	Password for the user ID.
errorTo	Location where 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.

Procedure: How to Create a Port for the SOAP Disposition

To create a port for a SOAP disposition:

1. Click the *iWay Events* tab.
2. In the left pane, expand the *IMS* 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.

   a. In the Port Name field, type a name for the event.
   b. In the Description field, type a brief description.
   c. From the Disposition Protocol drop-down list, select *SOAP*.
   d. In the Disposition field, enter a SOAP destination, using 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 defines the parameters for the disposition.

Parameter	Description
wsdl-url	The URL to the WSDL file that is required to create the SOAP message, for example: `http://localhost:7001/ibse/IBSEServlet/test/` *`webservice`*`.ibs?wsdl` where: *`webservice`* Is the name of the web service you created using iWay Explorer. To find this value, navigate to the iWay Business Services tab and open the *Service Description* hyperlink in a new window. The WSDL URL appears in the Address field. 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.
soapactio n	The method called by the SOAP disposition, for example: *`webservice.method@test@@`* where: *`webservice`* Is the name of the web service you created using iWay Explorer. *`method`* Is the method being used.

The event 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 HTTP

To create an Event port for HTTP:

1. In the left pane of iWay Explorer, select the *iWay Events* tab.
2. Expand the *IMS* 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.
5. In the Port Name field, type a name for the Event port.
6. In the Description field, type a brief description for the port (optional).
7. From the Disposition Protocol drop-down list, select *HTTP*.
8. 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, `http://myhost:1234/docroot`
responseTo	Location to which responses are posted. Optional. Predefined port name or another disposition URL. The URL must be complete, including the protocol.
9. 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 a Port for the MQSeries Disposition

To create a port for an MQSeries disposition using iWay Explorer:

1. Click the *iWay Events* tab.
2. In the left pane, expand the *IMS* 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.

   a. In the Port Name field, type a name for the event.

   b. In the Description field, type a brief description.

   c. From the Disposition Protocol drop-down list, select *MQSeries*.

   d. In the Disposition field, enter an MQSeries destination.

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

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

   The following table lists and defines the parameters for the 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). The default channel name for MQSeries is SYSTEM.DEF.SVRCONN.
errorTo	Location where error documents are sent. Predefined port name or another full URL. Optional.

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. Select the event port you want to edit.
2. In the right pane, move the pointer over Operations, and select *Edit*.

   The Edit Port pane opens on the right.
3. Make the required changes to the event port configuration fields.
4. Click *OK*.

Procedure: How to Delete an Event Port

To delete an event port:

1. Select the event port you want to delete.
2. In the right pane, move the pointer over Operations, and select *Delete*.

   A confirmation dialog box opens.
3. 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 IMS 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 IMS 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.

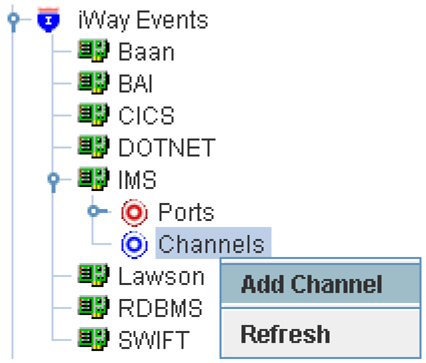
When using a REQUEST/RESPONSE configuration, input and output data must be either both Copybook or XML formatted.

Procedure: How to Create a Channel

To create a channel:

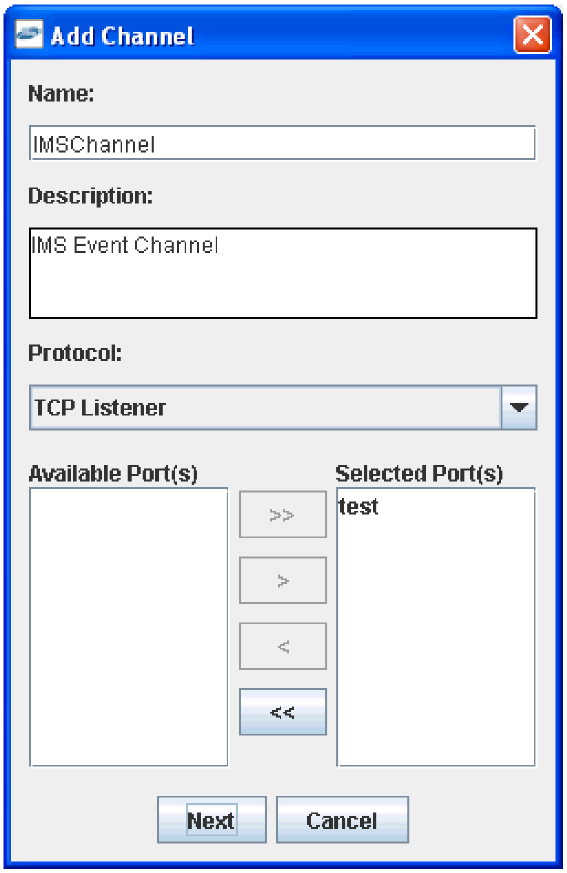
1. In the left pane of iWay Explorer, expand the *iWay Events* node.
2. Expand the *IMS* node.
3. Select the *Channels* node.

   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, IMSChannel.

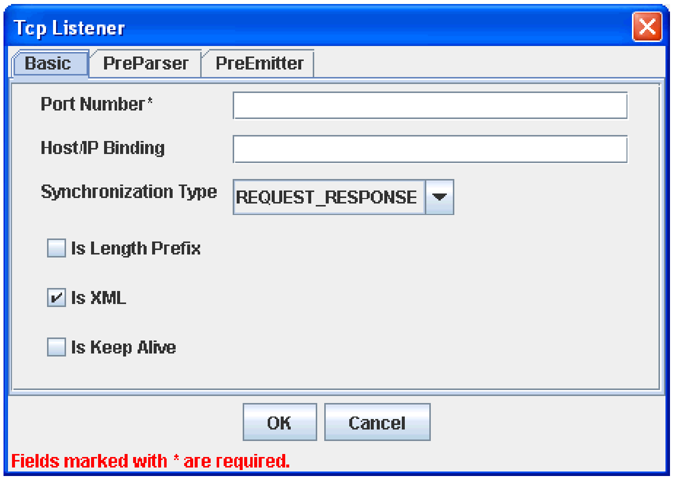
b. In the Description field (optional), type a brief description for the channel, for example, IMS Event Channel.

c. From the Protocol drop-down list, select a channel type. TCP Listener is the only type supported by the IMS 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.



Port Number*

Host/IP Binding

Synchronization Type: REQUEST_RESPONSE

[ ] Is Length Prefix

[✓] Is XML

[ ] Is Keep Alive

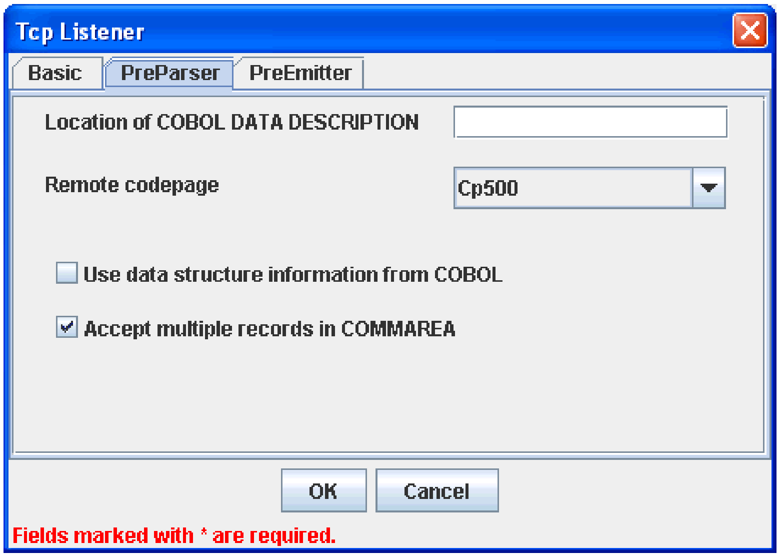
Fields marked with * are required.

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
Port Number	Port number where your IMS connection node listens for Events generated by IMS.
Host/IP Binding	Name or IP address of the host computer where the IMS application region is running.
Synchronization Type	Choose one of the following: ❑ If the Event application expects a reply sent back to it, select REQUEST_RESPONSE. ❑ If the Event application expects a TCP/IP acknowledgment (ACK), select REQUEST_ACK. ❑ If the Event application does not expect a reply, select REQUEST.

Property	Description
Is Length Prefix	For IMS events that return data, which is not in XML format. The TCP/IP event application must prefix the data with a 4-byte binary length field when writing data to the TCP/IP port.
Is XML	For IMS events that return data in XML format. No preparser is required.
Is Keep Alive	Maintains continuous communication between the Event transaction and the channel.



Location of COBOL DATA DESCRIPTION

Remote codepage: Cp500

☐ Use data structure information from COBOL

☑ Accept multiple records in COMMAREA

Fields marked with * are required.

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 IMS. For more information, see *Sample Requests, Schemas, and COBOL File Descriptions* on page 115.

Property	Description
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 others). 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.

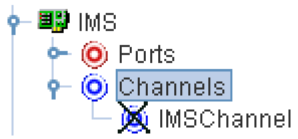
The following table lists and describes the properties in the PreEmitter tab.

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 IMS.
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 others). 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 IMSChannel 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 *IMS* 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 *IMS* 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 IMS 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 *IMS* 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.
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 *IMS* 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 IMS 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.

Appendix A

Configuring the Transaction Adapter for IMS 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 IMS in iWay Service Manager

Before configuring the adapter in iWay Service Manager, you must first create a target, which represents a connection to a back-end system, using iWay Explorer. For more information on configuring targets and connections using iWay Explorer, see *Configuring the iWay Transaction Adapter for IMS* on page 19 or the *iWay Explorer User's Guide*.

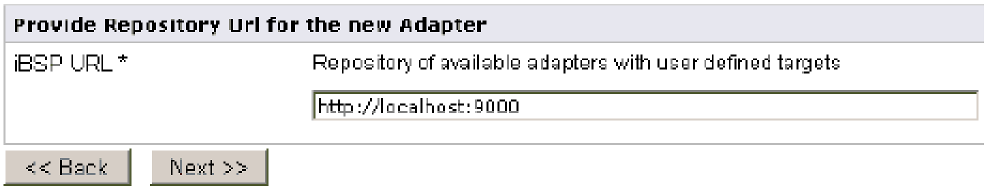
You configure the adapter in the iWay Service Manager console. The configuration process creates run-time connection and persistent data files within Service Manager. The configuration process interrogates the Service Manager repository entries that were built when the target and connection were created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

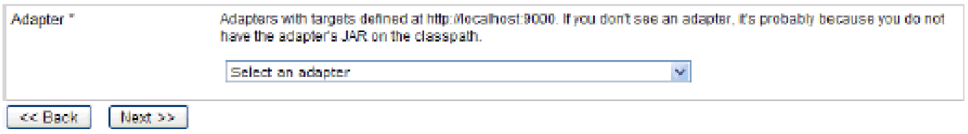


Provide Repository Url for the new Adapter

iBSP URL *	Repository of available adapters with user defined targets
	http://localhost:9000

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.

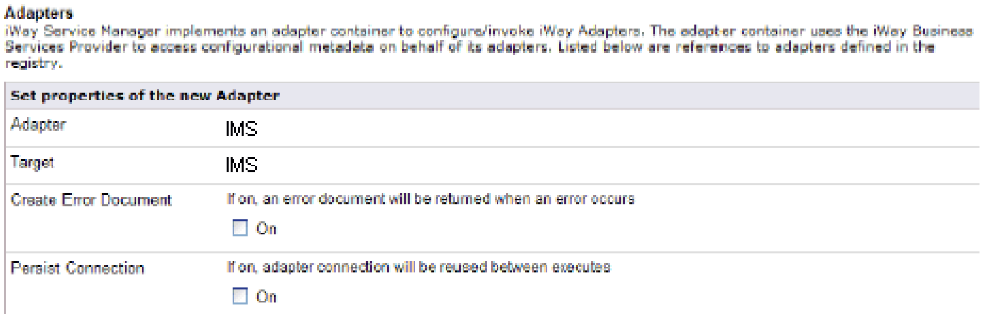


Adapter *	Adapters with targets defined at http://localhost:9000. If you don't see an adapter, it's probably because you do not have the adapter's JAR on the classpath. Select an adapter

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	IMS
Target	IMS
Create Error Document	If on, an error document will be returned when an error occurs ☐ On
Persist Connection	If on, adapter connection will be reused between executes ☐ On

   a. Select whether to return an error document when an error occurs.

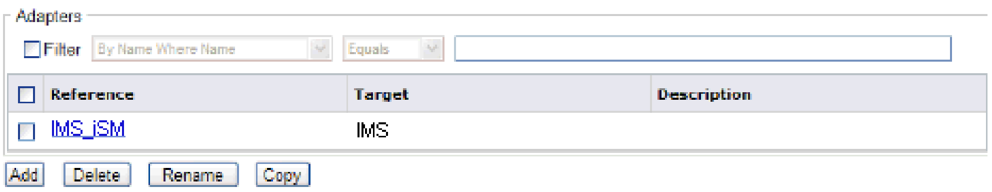
   b. Select whether an adapter connection will be reused between executes.

   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.

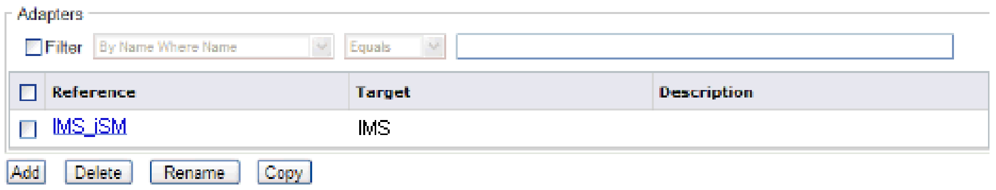


Adapters

☐	Reference	Target	Description
☐	IMS_iSM	IMS	

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).



Reference	Target	Description
IMS_iSM	IMS	

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*.

Appendix B

Using the IMS Sample Programs

This appendix describes how to use the sample IMS programs (IWTEST01, IWTEST05, IWTEST10, IWAYEVT0, and IWAYEVT2) that are provided with the iWay Transaction Adapter for IMS.

Note: Before they can be used, these sample programs must be installed on an IMS system. It is recommended that this installation is performed by IMS system administrators where the specific IMS system resides. This is different from the PART transactions, which are almost always installed on every IMS system, since they are provided by IBM.

In this appendix:

IWTEST01 Sample Program

The IWTEST01 sample program generates a new random balance every time it is called and a new comment. Otherwise, it is an echo program, which is used to test the fill, justification, truncation, defaults, and cursor.

- Transaction name: IWTEST01
- MFS Non-conversational
- One segment input
- One segment output
- Terminal defs: iwtmfs01

Sample Program Location and Directory Structure

The IWTEST01 sample program and related program files are located in the following directory:

```
<iway_home>\etc\samples\ims\iwtest01
```

where:

<iway_home>

Is the location on your system where iWay Service Manager is installed.

The following subdirectories are included for the IWTEST01 sample program:

- **document**
- **mfs**
- **src**

Document Subdirectory

The **document** subdirectory contains an XML input document for the adapter.

Reference: IWTEST01_IN.XML Input Document

The following is the structure of the IWTEST01_IN.XML input document:

```
<?xml version="1.0" encoding="UTF-8" ?>
<IMS>
  <Transaction location="IMS/Transactions/IWTEST01">
    <IW01MI>
      <Segment>
        <TCODE>IWTEST01 </TCODE>
        <WHATEVER>WHATEVER  </WHATEVER>
        <BCODE>11</BCODE>
        <NAME>MUSS</NAME>
      </Segment>
    </IW01MI>
  </Transaction>
</IMS>
```

MFS Subdirectory

The **mfs** subdirectory contains the MFS XML files, which describe the mapping of the input document to the data sent to IMS and the mapping of the incoming data from IMS to the output document. The adapter takes the input document, formats it according to the input MFS XML file, and sends it to IMS. It then takes the data IMS sends back, formats it according to the output MFS XML file, and passes the result to the caller. These descriptions are derived from the real IMS MFS definitions.

Reference: IW01MI.XML Input Document

The following is the structure of the IW01MI.XML input document:

```
<?xml version="1.0" encoding="UTF-8" ?>
<MSG name="IW01MI" opt="1" type="INPUT" next="IW01MO">
    <SEG>
        <MFLD name="TCODE"
         Length="9"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="no"
         fill=" "
         InputMFSDocumentUsage="Required" />
  <MFLD name="WHATEVER"
         Length="10"
         iwaySuppress="no"
         iwayFieldType="alpha"
         attribute="no"
         fill=" " />
  <MFLD name="PFKEYIND"
         Length="1"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="no"
         fill=" " />
  <MFLD name="CSRPOS"
         Length="4"
         iwayFieldType="cursor"
         default="7,2"
         InputMFSDocumentUsage="Forbidden" />
  <MFLD name="BCODE"
         Length="12"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="right"
         attribute="yes"
         fill="0"
         InputMFSDocumentUsage="Optional" />
  <MFLD name="USERID"
         Length="8"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="no"
         fill="U" />
  <MFLD name="LTERM"
         Length="8"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="no"
         fill="P" />
  <MFLD name="NAME"
         Length="38"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="yes"
         fill="S" />
  <MFLD name="ACCT"
         Length="12"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="yes"
         fill="#" />
  <MFLD name="BALANCE"
         Length="12"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="right"
        attribute="no"
         fill="0" />
  <MFLD name="COMM"
         Length="42"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="yes"
         fill=" " />
  </SEG>
</MSG>
```

Reference: IW01MO.XML Output Document

The following is the structure of the IW01MO.XML output document:

```
<?xml version="1.0" encoding="UTF-8" ?>
<MSG name="IW01MO" opt="1" type="OUTPUT" next="IW01MI">
    <SEG>
        <MFLD name="TCODE"
         Length="9"
         iwaySuppress="no"
         iwayFieldType="alpha"
         attribute="no"
         fill=" " />
  <MFLD name="WHATEVER"
         Length="10"
         iwaySuppress="no"
         iwayFieldType="alpha"
         attribute="no"
         fill=" " />
  <MFLD name="PFKEYIND"
         Length="1"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="no"
         fill=" " />
  <MFLD name="CSRPOS"
         Length="4"
         iwayFieldType="cursor"
        OutputMFSDocumentUsage="IfNotDefault"
         default="1,2" />
  <MFLD name="BCODE"
         Length="12"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="right"
         attribute="yes"
         fill="0"/>
  <MFLD name="USERID"
         Length="8"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="no"
         fill=" " />
  <MFLD name="LTERM"
         Length="8"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="no"
         fill=" "
         OutputMFSDocumentUsage="Never" />
  <MFLD name="NAME"
         Length="38"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="yes"
         fill="Z" />
  <MFLD name="ACCT"
         Length="12"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="yes"
         fill=" " />
  <MFLD name="BALANCE"
         Length="12"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="right"
        attribute="no"
         fill="0" />
  <MFLD name="COMM"
         Length="42"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="yes"
         fill=" "/>
  <!-- To test mismatch error
  <MFLD name="MISMATCH"
         Length="42"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="no"
         fill=" " />  -->
  </SEG>
</MSG>
```

SRC Subdirectory

The **src** subdirectory contains the following IMS components that are required to run the IWTEST01 sample program:

- ❏ **iwdef01** contains the APPLCTN and the TRANSACT macro definitions of the IWTEST01 sample program and the IWTEST01 transaction.
- ❏ **iwmfs01** contains the ‘real’ MFS definitions used by the IWTEST01 sample program.
- ❏ **iwtest01.c** is the program source code.

IWTEST05 Sample Program

The IWTEST05 sample program generates a new random balance the first time it is called. ADD and SUB may be used in the comment field to modify the balance.

- ❏ Transaction name: IWTEST05
- ❏ MFS Conversational, SPA size 256 bytes
- ❏ One segment input
- ❏ One segment output
- ❏ Terminal defs: iwtmfs01

Sample Program Location and Directory Structure

The IWTEST05 sample program and related program files are located in the following directory:

`<iway_home>\etc\samples\ims\iwtest05`

where:

`<iway_home>`

Is the location on your system where iWay Service Manager is installed.

The following subdirectories are included for the IWTEST05 sample program:

- ❏ **document**
- ❏ **mfs**
- ❏ **src**

Document Subdirectory

The **document** subdirectory contains the following XML input documents for the adapter.

- IWTEST05ADD0.XML
- IWTEST05ADD100.XML
- IWTEST05END.XML
- IWTEST05SUB100.XML

Reference: IWTEST05ADD0.XML Input Document

The following is the structure of the IWTEST05ADD0.XML input document:

```
<?xml version="1.0" encoding="UTF-8" ?>
<IMS>
  <Transaction location="IMS/Transactions/IWTEST05">
    <IW01MI>
      <Segment>
        <TCODE>IWTEST05 </TCODE>
        <WHATEVER>WHATEVER  </WHATEVER>
        <BCODE>11</BCODE>
        <NAME>MUSS</NAME>
        <COMM>add</COMM>
        <BALANCE>0</BALANCE>
      </Segment>
     </IW01MI>
  </Transaction>
</IMS>
```

Reference: IWTEST05ADD100.XML Input Document

The following is the structure of the IWTEST05ADD100.XML input document:

```
<?xml version="1.0" encoding="UTF-8" ?>
<IMS>
  <Transaction location="IMS/Transactions/IWTEST05">
    <IW01MI>
      <Segment>
        <TCODE>IWTEST05 </TCODE>
        <WHATEVER>WHATEVER  </WHATEVER>
        <BCODE>11</BCODE>
        <NAME>MUSS</NAME>
        <COMM>add</COMM>
        <BALANCE>100</BALANCE>
      </Segment>
     </IW01MI>
  </Transaction>
</IMS>
```

Reference: IWTEST05END.XML Input Document

The following is the structure of the IWTEST05END.XML input document:

```
<?xml version="1.0" encoding="UTF-8" ?>
<IMS>
  <Transaction location="IMS/Transactions/IWTEST05">
    <IW01MI>
      <Segment>
        <TCODE>IWTEST05 </TCODE>
        <WHATEVER>WHATEVER  </WHATEVER>
        <BCODE>11</BCODE>
        <NAME>MUSS</NAME>
        <COMM>end</COMM>
      </Segment>
    </IW01MI>
  </Transaction>
</IMS>
```

Reference: IWTEST05SUB100.XML Input Document

The following is the structure of the IWTEST05SUB100.XML input document:

```
<?xml version="1.0" encoding="UTF-8" ?>
<IMS>
  <Transaction location="IMS/Transactions/IWTEST05">
    <IW01MI>
      <Segment>
        <TCODE>IWTEST05 </TCODE>
        <WHATEVER>WHATEVER  </WHATEVER>
        <BCODE>11</BCODE>
        <NAME>MUSS</NAME>
        <COMM>sub</COMM>
        <BALANCE>100</BALANCE>
      </Segment>
    </IW01MI>
  </Transaction>
</IMS>
```

MFS Subdirectory

The **mfs** subdirectory contains the MFS XML files, which describe the mapping of the input document to the data sent to IMS and the mapping of the incoming data from IMS to the output document. The adapter takes the input document, formats it according to the input MFS XML file, and sends it to IMS. It then takes the data IMS sends back, formats it according to the output MFS XML file, and passes the result to the caller. These descriptions are derived from the real IMS MFS definitions.

Reference: IW01MI.XML Input Document

The following is the structure of the IW01MI.XML input document:

```
<?xml version="1.0" encoding="UTF-8" ?>
<MSG name="IW01MI" opt="1" type="INPUT" next="IW01MO">
  <SEG>
    <MFLD name="TCODE"
         Length="9"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="no"
         fill=" "
         InputMFSDocumentUsage="Required"/>
  <MFLD name="WHATEVER"
         Length="10"
         iwaySuppress="no"
         iwayFieldType="alpha"
         attribute="no"
         fill=" " />
  <MFLD name="PFKEYIND"
         Length="1"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="no"
         fill=" " />
  <MFLD name="CSRPOS"
         Length="4"
         iwayFieldType="cursor"
         default="7,2"
         InputMFSDocumentUsage="Forbidden" />
  <MFLD name="BCODE"
         Length="12"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="right"
         attribute="yes"
         fill="0"
         InputMFSDocumentUsage="Optional" />
  <MFLD name="USERID"
         Length="8"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="no"
         fill="U" />
  <MFLD name="LTERM"
         Length="8"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="no"
         fill="P" />
  <MFLD name="NAME"
         Length="38"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="yes"
         fill="S" />
  <MFLD name="ACCT"
         Length="12"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="yes"
         fill="#" />
  <MFLD name="BALANCE"
         Length="12"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="right"
         attribute="no"
         fill="0" />
  <MFLD name="COMM"
         Length="42"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="yes"
         fill=" " />
  </SEG>
</MSG>
```

Reference: IW01MO.XML Output Document

The following is the structure of the IW01MO.XML output document:

```
<?xml version="1.0" encoding="UTF-8" ?>
<MSG name="IW01MO" opt="1" type="OUTPUT" next="IW01MI">
  <SEG>
    <MFLD name="TCODE"
         Length="9"
         iwaySuppress="no"
         iwayFieldType="alpha"
         attribute="no"
         fill=" " />
  <MFLD name="WHATEVER"
         Length="10"
         iwaySuppress="no"
         iwayFieldType="alpha"
         attribute="no"
         fill=" " />
  <MFLD name="PFKEYIND"
         Length="1"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="no"
         fill=" " />
  <MFLD name="CSRPOS"
         Length="4"
         iwayFieldType="cursor"
         OutputMFSDocumentUsage="IfNotDefault"
         default="1,2" />
  <MFLD name="BCODE"
         Length="12"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="right"
         attribute="yes"
         fill="0" />
  <MFLD name="USERID"
         Length="8"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="no"
         fill=" " />
  <MFLD name="LTERM"
         Length="8"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="no"
         fill=" "
         OutputMFSDocumentUsage="Never" />
  <MFLD name="NAME"
         Length="38"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="yes"
         fill="Z" />
  <MFLD name="ACCT"
         Length="12"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="yes"
         fill=" " />
  <MFLD name="BALANCE"
         Length="12"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="right"
         attribute="no"
         fill="0" />
  <MFLD name="COMM"
         Length="42"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="yes"
         fill=" " />
  <!-- To test mismatch error
  <MFLD name="MISMATCH"
         Length="42"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="no"
         fill=" " /> -->
  </SEG>
</MSG>
```

SRC Subdirectory

The **src** subdirectory contains the following IMS components that are required to run the IWTEST05 sample program:

- **iwdef05** contains the APPLCTN and the TRANSACT macro definitions of the IWTEST05 sample program and the IWTEST05 transaction.
- **iwmfs01** contains the ‘real’ MFS definitions used by the IWTEST01 sample program.
- **iwtest05.c** is the program source code.

IWTEST10 Sample Program

The IWTEST10 sample program accesses a database (IWSTDB). Three database operations are used (add, delete, and get), (key, description), key length = 16, and data length = 34.

- Transaction names: IWT10, IWT10A, IWT10D, IWT10G. Currently, using only transaction IWT10 is recommended.
- MFS Non-conversational
- One segment input. Can be mapped by an mfs.xml, or 'freeform', with the fields separated by blanks (for example, op key description).

  Examples:

  - `GET dodo`
  - `add dodo deadbird`

  Timeout example: If a digit is stuck at the end of the operation, the program will execute the operation and then wait for that many seconds before returning.

  - `add9 dodo extinct`

    Adds 'dodo extinct' to the database, waits nine seconds, and then returns.
- One segment output
- Terminal defs: iwtmfs01

Note: When accessing this transaction via the CRM Gateway, the security type **must** be IDENTIFY. This security type is required since it is related to RACF and UNIX System Services.

Sample Program Location and Directory Structure

The IWTEST10 sample program and related program files are located in the following directory:

```
<iway_home>\etc\samples\ims\iwtest10
```

where:

<iway_home>

Is the location on your system where iWay Service Manager is installed.

The following subdirectories are included for the IWTEST01 sample program:

- **document**
- **mfs**
- **src**

Document Subdirectory

The **document** subdirectory contains an XML input document for the adapter.

Reference: **IWTEST10_IN.XML Input Document**

The following is the structure of the IWTEST10_IN.XML input document:

```
<?xml version="1.0" encoding="UTF-8" ?>
<IMS>
  <Transaction location="IMS/Transactions/IWT10">
    <IW10MI>
      <Segment>
        <TCODE>IWT10</TCODE>
        <OP>add</OP>
        <NAME>burst</NAME>
        <COMM>inTears</COMM>
      </Segment>
    </IW10MI>
  </Transaction>
</IMS>
```

MFS Subdirectory

The **mfs** subdirectory contains the MFS XML files, which describe the mapping of the input document to the data sent to IMS and the mapping of the incoming data from IMS to the output document. The adapter takes the input document, formats it according to the input MFS XML file, and sends it to IMS. It then takes the data IMS sends back, formats it according to the output MFS XML file, and passes the result to the caller. These descriptions are derived from the real IMS MFS definitions.

Reference: IW10MI.XML Input Document

The following is the structure of the IW10MI.XML input document:

```
<?xml version="1.0" encoding="UTF-8" ?>
<MSG name="IW10MI" opt="1" type="INPUT" next="IW10MO">
  <SEG>
    <MFLD name="TCODE"
         Length="6"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="no"
         fill=" " />
  <MFLD name="OP"
         Length="4"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="no"
         fill=" " />
  <MFLD name="NAME"
         Length="16"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="no"
         fill=" " />
  <MFLD name="COMM"
         Length="34"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="no"
         fill=" " />
  <MFLD name="MSG"
         Length="50"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="no"
         fill=" " />
  </SEG>
</MSG>
```

Reference: IW10MO.XML Output Document

The following is the structure of the IW10MO.XML output document:

```
<?xml version="1.0" encoding="UTF-8" ?>
<MSG name="IW10MO" opt="1" type="OUTPUT" next="IW10MI">
  <SEG>
    <MFLD name="TCODE"
         Length="8"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="no"
         fill=" " />
  <MFLD name="OP"
         Length="4"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="no"
         fill=" " />
  <MFLD name="NAME"
         Length="16"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="no"
         fill=" " />
  <MFLD name="COMM"
         Length="34"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="no"
         fill=" " />
  <MFLD name="MSG"
         Length="50"
         iwaySuppress="no"
         iwayFieldType="alpha"
         justification="left"
         attribute="no"
         fill=" " />
  </SEG>
</MSG>
```

SRC Subdirectory

The **src** subdirectory contains the following IMS components that are required to run the IWTEST10 sample program:

- **iwdef10** contains the APPLCTN and the TRANSACT macro definitions for the IWTEST10 sample program and the following transactions that invoke it:
  - iwt10
  - iwt10a

  - iwt10d
  - iwt10g

  Note: Currently, using only transaction IWT10 is recommended.

- **iwmfs10** contains the 'real' MFS definitions used by the IWTEST10 sample program.
- **iwpsb10** contains the program definitions (psbgen) iwtest10. This program uses the *iwstdb* database, which must be defined separately.
- **iwstdb** contains the database definitions (dbdgen) for the *iwstdb* database.
- **iwtest10.c** is the program source code.

IWAYEVT0 Sample Program

The IWAYEVT0 sample program (iwayevt0.cobol) is a sample program that tests event handling in the iWay Transaction Adapter for IMS. It is identical to the version that is packaged with the iWay Transaction Adapter for CICS, only modified to run on IMS.

Sample Program Location and Directory Structure

The IWAYEVT0 sample program and related program files are located in the following directory:

```
<iway_home>\etc\samples\ims\iwayevt0
```

where:

<iway_home>

Is the location on your system where iWay Service Manager is installed.

The following subdirectories are included for the IWAYEVT0 sample program:

- **cobolfd**
- **src**

Cobolfd Subdirectory

The **cobolfd** subdirectory contains the COBOL copybook (IWAYEVT0_IN.CBL) to map the data that is sent from IMS.

Reference: **IWAYEVT0_IN.CBL COBOL Copybook**

The following is the structure of the IWAYEVT0_IN.CBL COBOL copybook:

```
05 ALPHA01                 PIC X(8).
05 INT01                   PIC S9(4) BINARY.
05 PACK01                  PIC S9(15) PACKED-DECIMAL.
05 ZONE01                  PIC  9(4).
```

SRC Subdirectory

The **src** subdirectory contains the COBOL program (IWAYEVT0.COBOL).

Reference: IWAYEVT0.COBOL Program

The following is the structure of the COBOL program (IWAYEVT0.COBOL):

```
CBL TRUNC(BIN)
       ID DIVISION.
       PROGRAM-ID. IWAYEVT0.
      *************************************************************
      * IWAYEVT0 - THIS SAMPLE PROGRAM DEMONSTRATES SENDING A       *
      * RECORD TO THE IWAY IMS ADAPTER USING CICS SOCKETS.  NO      *
      * RESPONSE IS RETURNED.  DATA RECORDS MAPPED BY COPYBOOKS     *
      * MUST EACH BE PRECEDED BY A 4 BYTE BINARY LENGTH.            *
      *                                                             *
      * THE IMS 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.
       77  GU                            PIC   X(04) VALUE 'GU  '.
       77  CHNG                          PIC   X(04) VALUE 'CHNG'.
       77  ISRT                          PIC   X(04) VALUE 'ISRT'.
       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.
 01 MSG-OUT.
    05  IMS-LL              PIC 9(4) BINARY VALUE 70.
    05  IMS-ZZ              PIC 9(4) BINARY.
    05  MSG                 PIC X(70).
 01  PARM-IN.
    05  SLEN                PIC S9(4) BINARY.
    05  SCRN-IOA            PIC X(255).
    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.
*********************************************************
*    I/O PCB                                            *
*********************************************************
    01  IOPCB.
        05  LTERM                    PIC   X(08).
        05  FILLER                   PIC   X(02).
        05  IOPCB-STATUS             PIC   X(02).
        05  FILLER                   PIC   X(28).
 PROCEDURE DIVISION.
 MAINLINE.
     ENTRY 'DLITCBL' USING IOPCB
     PERFORM GETPARM
***************************************************************
* 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
     MOVE SPACE TO MSG
     MOVE TMSG TO MSG
     CALL 'CBLTDLI' USING ISRT, IOPCB, MSG-OUT
     GOBACK.
 GETPARM.
     CALL 'CBLTDLI' USING GU, IOPCB, SCRN-IOA
```

IWAYEVT2 Sample Program

The IWAYEVT2 sample program (iwayevt2.cobol) is a sample program that tests request/ response event handling in the iWay Transaction Adapter for IMS. It sends out a message, prefixed by a 4-byte length, and expects the exact same message as the response.

Sample Program Location and Directory Structure

The IWAYEVT2 sample program and related program files are located in the following directory:

```
<iway_home>\etc\samples\ims\iwayevt2
```

where:

<iway_home>

Is the location on your system where iWay Service Manager is installed.

The following subdirectories are included for the IWAYEVT2 sample program:

- ❏ **cobolfd**
- ❏ **java**
- ❏ **src**

Cobolfd Subdirectory

The **cobolfd** subdirectory contains the COBOL copybook (IWAYEVT0_IN.CBL) to map the data that is sent from, and returned to, IMS.

Reference: IWAYEVT0_IN.CBL COBOL Copybook

The following is the structure of the IWAYEVT0_IN.CBL COBOL copybook:

```
05 ALPHA01                  PIC X(8).
05 INT01                    PIC S9(4) BINARY.
05 PACK01                   PIC S9(15) PACKED-DECIMAL.
05 ZONE01                   PIC  9(4).
```

Java Subdirectory

The **java** subdirectory contains the message-driven Java Bean (InboundEchoBean.java) that actually performs the echo.

Reference: InboundEchoBean.Java Bean

The following is the structure of the Java Bean (InboundEchoBean.java):

```
/*
 * Created on Feb 11, 2005
 *
 * Copyright (C) 2003-2004, iWay Software/Information Builders, Inc.
 * All Rights Reserved.
 */
package com.iwaysoftware.afjca15.samples;
import javax.ejb.EJBException;
import javax.ejb.MessageDrivenBean;
import javax.ejb.MessageDrivenContext;
import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.Enumeration;
import java.util.Iterator;
import java.util.Properties;
import java.util.logging.*;
import javax.naming.*;
import javax.resource.cci.MessageListener;
import javax.resource.cci.Record;
import com.ibi.common.IContext;
import com.iwaysoftware.afjca15.IIWAFRecord;
import com.iwaysoftware.afjca15.CCIRecordFactory;
/**
 * A simple message-driven bean. The code illustrates the requirements of a
 * message-driven bean class:
 * <ul>
 * <li>It implements the <code>MessageDrivenBean</code> and
 * <code>CCI MessageListener</code> interfaces.
 * <li>The class is defined as <code>public</code>.
 * <li>The class cannot be defined as <code>abstract</code> or
 * <code>final</code>.
 * <li>It implements one <code>onMessage</code> method.
 * <li>It implements one <code>ejbCreate</code> method and one
 * <code>ejbRemove</code> method.
 * <li>It contains a public constructor with no arguments.
 * <li>It must not define the <code>finalize</code> method.
 * </ul>
 *
 * Unlike session and entity beans, message-driven beans do not have the
remote
```

```
 * or local interfaces that define client access. Client components do not
 * locate message-driven beans and invoke methods on them. Although
 * message-driven beans do not have business methods, they may contain
helper
 * methods that are invoked internally by the onMessage method.
 *
 */
public class InboundEchoBean implements MessageDrivenBean, MessageListener {
  static final Logger logger = Logger
   .getLogger("com.iwaysoftware.afjca15.samples");
  private transient MessageDrivenContext mdc = null;
  private Context context;
  /**
   * Default constructor. Creates a bean.
  */
  public InboundEchoBean() {
    logger.info("<MDB> In InboundEchoBean.InboundEchoBean()");
  }
  /**
   * Sets the context for the bean.
   *
   * @param mdc the message-driven-bean context
  */
  public void setMessageDrivenContext(MessageDrivenContext mdc) {
    logger.info("<MDB> In InboundMessageBean.setMessageDrivenContext()");
    this.mdc = mdc;
  }
  /**
   * Creates a bean. Required by EJB spec.
  */
  public void ejbCreate() {
    logger.info("<MDB> In InboundEchoBean.ejbCreate()");
  }
  /**
   * When the IWAF Channel receives a record, the EJB container invokes the
   * <code>onMessage</code> method of the message-driven bean. The
   * <code>onMessage</code> method displays data from the record
   *
   * @param record incoming record
  */
  public Record onMessage(Record record) {
    if (!(record instanceof IIWAFRecord)){
      throw new EJBException("Received 'Record' must be of type
'IWAFRecord'");
    }
  IIWAFRecord iwafRecord = (IIWAFRecord)record;
  try {
    logger.info("<MDB> ---- Got a message ");
    //Event context
    IContext iCtx = iwafRecord.getIContext();
    if (iCtx == null) {
      logger.info("IContext is null");
    } else {
       StringBuffer buf = new StringBuffer();
       buf.append("IContext is [");
       Iterator i = iCtx.attributeNames();
       while (i.hasNext()) {
         String key = (String)i.next();
         Object value = iCtx.getAttribute(key);
         buf.append(" ");
         buf.append(key);
         buf.append("=");
         buf.append(value);
         buf.append(" ");
```

SRC Subdirectory

The **src** subdirectory contains the COBOL program (IWAYEVT2.COBOL).

Reference: IWAYEVT2.COBOL Program

The following is the structure of the COBOL program (IWAYEVT2.COBOL):

```
CBL TRUNC(BIN)
       ID DIVISION.
       PROGRAM-ID. IWAYEVT2.
      ***************************************************************
      * IWAYEVT2 - THIS SAMPLE PROGRAM DEMONSTRATES SENDING A       *
      * RECORD TO THE IWAY IMS ADAPTER USING CICS SOCKETS, AND      *
      * THEN RECEIVING THE SAME DATA BACK (ECHO).  THE DATA RECORD, *
      * MAPPED BY A COPYBOOK, MUST BE PRECEEDED BY A 4 BYTE BINARY  *
      * LENGTH.                                                     *
      *                                                             *
      * THE IMS 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 AND PREEMITTER FD.                                *
      *                                                             *
      * HOST AND PORT MUST BE SET BELOW (PROCEDURE DIVISION)        *
      *                                                             *
      * THE EZASOKET INTERFACE IS DOCUMENTED IN THE Z/OS            *
      * COMMUNICATIONS SERVER IP IMS SOCKETS GUIDE.                 *
      *                                                             *
      * USES: IWAYEVT0_IN.CBL     (INPUT RECORD)                    *
      *                                                             *
      ***************************************************************
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       77  GU                          PIC   X(04) VALUE 'GU  '.
       77  CHNG                        PIC   X(04) VALUE 'CHNG'.
       77  ISRT                        PIC   X(04) VALUE 'ISRT'.
       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 'IWAYEVT2: DATA SENT SUCCESSFULLLY.          '.
          05 RMSG                         PIC X(44)
             VALUE 'IWAYEVT2: DATA RECEIVED SUCCESSFULLY        '.
      ***************************************************************
      * 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.
      ***************************************************************
      * ECHO RESPONSE.                                              *
      ***************************************************************
       01 RESPONSE-BUF.
          02 RESPONSE-LEN          PIC 9(8) BINARY.
          02 RESPONSE              REDEFINES RESPONSE-LEN
                                   PIC X OCCURS 4 TIMES.
          02 OUTBOUND-RECORD.
             05 ALPHA01            PIC X(8).
             05 INT01              PIC S9(4) BINARY.
             05 PACK01             PIC S9(15) PACKED-DECIMAL.
             05 ZONE01             PIC 9(4).
       01 MSG-OUT.
          05  IMS-LL               PIC 9(4) BINARY VALUE 70.
          05  IMS-ZZ               PIC 9(4) BINARY.
          05  MSG                  PIC X(70).
       01  PARM-IN.
          05  SLEN                 PIC S9(4) BINARY.
          05  SCRN-IOA             PIC X(255).
       01 USED                     PIC 9(8) BINARY.
          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.
```

```
*********************************************************
*    I/O PCB                                            *
*********************************************************
    01  IOPCB.
        05  LTERM                  PIC   X(08).
        05  FILLER                 PIC   X(02).
        05  IOPCB-STATUS           PIC   X(02).
        05  FILLER                 PIC   X(28).
 PROCEDURE DIVISION.
 MAINLINE.
     ENTRY 'DLITCBL' USING IOPCB
     PERFORM GETPARM
***************************************************************
* CHANGE HOSTNAME AND PORT TO SITE SPECIFIC LOCATION OF THE   *
* IMS ADAPTER.                                                *
***************************************************************
     MOVE 'INFORMAT-16C9AB.ibi.com' TO HOSTNAME
     MOVE 4773 TO PORT
     PERFORM GETSOCK
     PERFORM GETHOSTBYNAME
     PERFORM SETBLOCK
     PERFORM CONNECTTOHOST
     PERFORM SENDDATA
     MOVE SPACE TO MSG
     MOVE TMSG  TO MSG
     CALL 'CBLTDLI' USING ISRT, IOPCB, MSG-OUT
     PERFORM RECVDATA
     PERFORM CLOSESOCK
     MOVE RMSG TO MSG
     CALL 'CBLTDLI' USING ISRT, IOPCB, MSG-OUT
     MOVE SPACE TO MSG
     MOVE ALPHA01 OF OUTBOUND-RECORD  TO MSG
     MOVE INT01 OF OUTBOUND-RECORD    TO MSG(10:4)
     MOVE PACK01 OF OUTBOUND-RECORD   TO MSG(16:15)
     MOVE ZONE01 OF OUTBOUND-RECORD   TO MSG(34:4)
     CALL 'CBLTDLI' USING ISRT, IOPCB, MSG-OUT
     GOBACK.
 GETPARM.
     CALL 'CBLTDLI' USING GU, IOPCB, SCRN-IOA
     IF IOPCB-STATUS NOT = SPACES
        PERFORM WRITERR-EXIT
     END-IF
*    DISPLAY 'INPUT PARM: ' SCRN-IOA
     .
 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 'EZASOKET' USING SOC-FUNCTION,
           SOCKET,
           FLAGS,
           LLEN,
           NBYTE,
           ERRNO,
           RETCODE
     IF RETCODE = -1
        PERFORM WRITERR-EXIT
     END-IF
```

```
****************************************************************
* SEND THE ACTUAL RECORD                                       *
****************************************************************
     CALL 'EZASOKET' USING SOC-FUNCTION,
          SOCKET,
          FLAGS,
          NBYTE,
          INBOUND-RECORD,
          BY REFERENCE ERRNO,
          RETCODE
     IF RETCODE = -1
        PERFORM WRITERR-EXIT
     END-IF
     .
 RECVDATA.
****************************************************************
* RECEIVE A RESPONSE                                           *
****************************************************************
     MOVE LOW-VALUES TO RESPONSE-BUF.
     MOVE LENGTH OF RESPONSE-BUF TO NBYTE.
     MOVE 1 TO USED.
*
     PERFORM READ-BYTES UNTIL NBYTE < 1.
*
 READ-BYTES.
     MOVE 'READ            ' TO SOC-FUNCTION
     CALL 'EZASOKET' USING SOC-FUNCTION,
           SOCKET,
           NBYTE,
           RESPONSE(USED),
           ERRNO,
           RETCODE.
     IF RETCODE < 0
         PERFORM WRITERR-EXIT
     END-IF
*
     IF RETCODE = 0
       MOVE 0 TO NBYTE
     ELSE
       ADD RETCODE TO USED
       SUBTRACT RETCODE FROM NBYTE
     END-IF
     .
 CLOSESOCK.
     MOVE ZEROES TO RETCODE ERRNO
     MOVE 'CLOSE           ' TO SOC-FUNCTION
     CALL 'EZASOKET' USING SOC-FUNCTION,
        SOCKET,
        ERRNO,
        RETCODE
     IF RETCODE < 0
        PERFORM WRITERR-EXIT
     END-IF.
 WRITERR-EXIT.
     MOVE SOC-FUNCTION TO ERRMSG(26:15)
     DISPLAY 'ERROR IN PROGRAM FUNCTION: ' ERRMSG.
     GOBACK.
```

Appendix C

Sample Requests, Schemas, and COBOL File Descriptions

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

The following topics illustrate the request and response documents for the transaction, PART. The COBOL descriptions that were used as input for the sample IMS/TM transactions also are shown.

In this appendix:

Request Document to Run PART as a Generic Transaction

```
<?xml version="1.0" encoding="UTF-8" ?>
<IMS>
  <Transaction tpname="PART" noreply= "NO">
    <message>
      <message>*</message>
    </message>
  </Transaction>
</IMS>
```

Request Schema for Generic Transaction PART

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Generated by the iBSE 2004-11-04T16:29:41Z -->
<xs:schema xml:lang="en" targetNamespace="urn:iwaysoftware:IMS/
Transactions/Generic_Transaction_Request"
attributeFormDefault="unqualified" xmlns:xs="http://www.w3.org/2001/
XMLSchema" elementFormDefault="qualified">
   <xs:element name="IMS">
      <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="message"/>
                           </xs:sequence>
                        </xs:complexType>
                     </xs:element>
                  </xs:all>
                  <xs:attribute type="xs:string"
                  use="optional" name="hexConv"/>
                  <xs:attribute type="xs:integer"
                  use="optional" name="bufferLimit"/>
                  <xs:attribute type="xs:boolean"
                  use="optional" name="noRepl"/>
                  <xs:attribute type="xs:string"
                  use="required" name="tpname"/>
                  <xs:attribute type="xs:string"
                  use="optional" name="lterm"/>
               </xs:complexType>
            </xs:element>
         </xs:all>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

Response Schema for Generic Transaction PART

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Generated by the iBSE 2004-11-04T16:29:41Z -->
<xs:schema xml:lang="en"
targetNamespace="urn:iwaysoftware:IMS/Transactions/
Generic_Transaction_Response" attributeFormDefault="unqualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
   <xs:element name="IMS">
      <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="message"/>
                           </xs:sequence>
                        </xs:complexType>
                     </xs:element>
                  </xs:all>
               </xs:complexType>
            </xs:element>
         </xs:all>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

Request Documents for Adapter Transaction PART

The following are the sample XML request documents to run the transaction, PART:

PART_All_Request.xml

```
<IMS>
 <Transaction location="IMS/Transactions/Part">
  <message>
   <MESSAGE>*</MESSAGE>
  </message>
 </Transaction>
</IMS>
```

PART_Detail_Request.xml

```
<IMS>
 <Transaction location="IMS/Transactions/Part">
  <message>
   <MESSAGE>PartNumberHere</MESSAGE>
  </message>
 </Transaction>
</IMS>
```

PART_Error_Request.xml

```
<IMS>
 <Transaction location="IMS/Transactions/Part">
  <message>
   <MESSAGE>WillNotFind</MESSAGE>
  </message>
 </Transaction>
</IMS>
```

Request Schema for Adapter Transaction PART

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Generated by the iBSE 2004-11-04T16:44:28Z -->
<xs:schema xml:lang="en" targetNamespace="urn:iwaysoftware:IMS/
Transactions/Part_Request" attributeFormDefault="unqualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
   <xs:element name="IMS">
      <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="MESSAGE"/>
                           </xs:sequence>
                        </xs:complexType>
                     </xs:element>
                  </xs:all>
                  <xs:attribute type="xs:string" use="required"
                  fixed="IMS/Transactions/Part" name="location"/>
               </xs:complexType>
            </xs:element>
         </xs:all>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

Response Schema for Adapter Transaction PART

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Generated by the iBSE 2004-11-04T16:44:28Z -->
<xs:schema xml:lang="en" targetNamespace="urn:iwaysoftware:IMS/
Transactions/Part_Response" attributeFormDefault="unqualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
   <xs:element name="IMS">
      <xs:complexType>
         <xs:all>
            <xs:element name="Transaction">
               <xs:complexType>
                  <xs:choice>
                     <xs:element name="message1">
                        <xs:complexType>
                           <xs:sequence>
                              <xs:element minOccurs="1"
                              name="FILL" maxOccurs="1"/>
                              <xs:element minOccurs="1"
                              name="RECTYPE" maxOccurs="1"/>
                              <xs:element minOccurs="1"
                              name="FILL1" maxOccurs="1"/>
                              <xs:element minOccurs="1"
                              name="PARTNUMBER" maxOccurs="1"/>

                              <xs:element minOccurs="1"
                              name="FILL2" maxOccurs="1"/>
                              <xs:element minOccurs="1"
                              name="DESCRIPTION" maxOccurs="1"/>
                              <xs:element minOccurs="1"
                              name="FILL3" maxOccurs="1"/>
                              <xs:element minOccurs="1"
                              name="PROCCODE" maxOccurs="1"/>
                              <xs:element minOccurs="1"
                              name="FILL4" maxOccurs="1"/>
                              <xs:element minOccurs="1"
                              name="INVCODE" maxOccurs="1"/>
                              <xs:element minOccurs="1"
                              name="FILL5" maxOccurs="1"/>
                              <xs:element minOccurs="1"
                              name="MAKEDEPT" maxOccurs="1"/>
                              <xs:element minOccurs="1"
                              name="FILL6" maxOccurs="1"/>
                              <xs:element minOccurs="1"
                              name="PREVNO" maxOccurs="1"/>
                              <xs:element minOccurs="1"
                              name="FILL7" maxOccurs="1"/>
```

```
                                          <xs:element minOccurs="1"
                                          name="MAKETIME" maxOccurs="1"/>
                                          <xs:element minOccurs="1"
                                          name="FILL8" maxOccurs="1"/>
                                          <xs:element minOccurs="1"
                                          name="CCODE" maxOccurs="1"/>
                                       </xs:sequence>
                                    </xs:complexType>
                                 </xs:element>
                                 <xs:element name="message2">
                                    <xs:complexType>
                                       <xs:sequence>
                                          <xs:element minOccurs="1"
                                          name="FILL" maxOccurs="1"/>
                                          <xs:element minOccurs="1"
                                          name="RECTYPE" maxOccurs="1"/>
                                       </xs:sequence>
                                    </xs:complexType>
                                 </xs:element>
                                 <xs:element name="message3">
                                    <xs:complexType>
                                       <xs:sequence>
                                          <xs:element minOccurs="1"
                                          name="REC-NUM" maxOccurs="1"/>
                                          <xs:element minOccurs="1"
                                          name="RECTYPE" maxOccurs="1"/>
                                          <xs:element minOccurs="1"
                                          name="PART-NUM" maxOccurs="1"/>
                                          <xs:element minOccurs="1"
                                          name="PART-DESC" maxOccurs="1"/>
                                       </xs:sequence>
                                    </xs:complexType>
                                 </xs:element>
                              </xs:choice>
                           </xs:complexType>
                        </xs:element>
                     </xs:all>
                  </xs:complexType>
               </xs:element>
            </xs:schema>
```

Sample COBOL File Descriptions for PART

The following sample COBOL File Description is used as input for the IMS/TM transactions in *Creating XML Schemas and iWay Business Services* on page 31.

PART_IN.cbl

```
01  PARTIN.
    05  MESSAGE                  PIC X(80)   VALUE SPACE.
```

PART_All_Out.cbl

```
01  PARTALL.
    05  FILL                        PIC X(3).
    05  RECTYPE                     PIC X(1).
    05  MESSAGE                     PIC X(76).
```

PART_Detail_Out.cbl

```
01  PARTDETAIL.
    05  FILL                        PIC X(3)   VALUE SPACE.
    05  RECTYPE                     PIC X(1)   VALUE SPACE.
    05  FILL1                       PIC X(22)  VALUE SPACE.
    05  PARTNUMBER                  PIC X(12)  VALUE SPACE.
    05  FILL2                       PIC X(18)  VALUE SPACE.
    05  DESCRIPTION                 PIC X(20)  VALUE SPACE.
    05  FILL3                       PIC X(26)  VALUE SPACE.
    05  PROCCODE                    PIC X(12)  VALUE SPACE.
    05  FILL4                       PIC X(18)  VALUE SPACE.
    05  INVCODE                     PIC X(8)   VALUE SPACE.
    05  FILL5                       PIC X(26)  VALUE SPACE.
    05  MAKEDEPT                    PIC X(12)  VALUE SPACE.
    05  FILL6                       PIC X(18)  VALUE SPACE.
    05  PREVNO                      PIC X(8)   VALUE SPACE.
    05  FILL7                       PIC X(26)  VALUE SPACE.
    05  MAKETIME                    PIC X(12)  VALUE SPACE.
    05  FILL8                       PIC X(18)  VALUE SPACE.
    05  CCODE                       PIC X(8)   VALUE SPACE.
```

PART_Error_Out.cbl

```
01  PARTERROR.
    05  FILL                        PIC X(3)   VALUE SPACE.
    05  RECTYPE                     PIC X(1)   VALUE SPACE.
    05  MESSAGE                     PIC X(46)  VALUE SPACE.
```

Conversational Transaction Sample

The sample shows a conversational transaction for the IMS verification sample for customer information. The IMS transaction, known as IVTCV, is supplied with the IMS product. The sample adds a customer segment in the first message and then ends the request with the second message.

The conversational transaction must be configured in using both iWay Explorer and Service Manager. With iWay Explorer you can design transaction information, creating one transaction for each potential segment of a conversation. You can then use Service Manager to create one listener for each transaction. For more information on using Service Manager, see the *iWay Service Manager's User's Guide*.

Each message in the conversation is processed by the same IVTCV application. In this sample, two transactions are designed in iWay Explorer and unique request and response schemas are generated for each transaction. In Service Manager, a listener is configured for each transaction.

One listener processes each message segment for a total of two listeners. The listeners must be connected in a local transaction. The IMS resources are not part of the local transaction. However, the local transaction allows the listeners to maintain a state required for a conversational transaction with the IMS application.

Configuring Conversational Transactions

Create a transaction in iWay Explorer for the first part of the conversation. This transaction is used to add a customer to the IMS customer database.

The following files represent the request documents, associated schemas, and .cpy files configured for the first part of the conversation.

- Request document: Segment1_request_doc.xml
- Request schema: Segment1_request.xsd
- Response schema: Segment1_response.xsd
- Sample copybook for request: ivtcv_add.cpy
- Sample copybook for response: ivtcv_end.cpy

Create a transaction in iWay Explorer for the second part of the conversation. This part of the conversation ends the customer information conversation with IMS. The message sent to the IMS customer application is simply an action of "END".

The following files represent the request documents, associated schemas, and .cpy files configured for the second part of the conversation.

- Request document: Segment2_request_doc.xml
- Request schema: Segment2_request.xsd
- Response schema: Segment2_response.xsd
- Sample copybook for request: ivtcv_end.cpy
- Sample copybook for response: ivtcv_response.cpy

The following schemas and XML instance documents illustrate the segment requests and segment responses referenced above.

For examples of the .cpy files, see *lvtcv_add.cpy* on page 135, *lvtcv_end.cpy* on page 135, and *lvtcv_response.cpy* on page 135.

Reference: Segment One Request Schema

The following is an example of the segment one request schema:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <!--  Generated by the iBSE 2005-05-02T19:16:28Z
  -->
- <xsd:schema xml:lang="en" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:iwaysoftware:IMS/Transactions/Segment1_Request"
attributeFormDefault="unqualified" elementFormDefault="qualified">
- <xsd:element name="IMS">
- <xsd:complexType>
- <xsd:all>
- <xsd:element name="Transaction">
- <xsd:complexType>
- <xsd:sequence>
- <xsd:element name="Segment" maxOccurs="unbounded">
- <xsd:complexType>
- <xsd:sequence>
- <xsd:element minOccurs="1" name="INPUT-MSG" maxOccurs="1">
- <xsd:complexType>
- <xsd:sequence>
- <xsd:element minOccurs="1" name="IN-FILL" maxOccurs="1">
- <xsd:simpleType>
- <xsd:restriction base="xsd:string">
  <xsd:length value="4" />
  </xsd:restriction>
  </xsd:simpleType>
  </xsd:element>
- <xsd:element minOccurs="0" name="IN-COMMAND" maxOccurs="1">
- <xsd:simpleType>
- <xsd:restriction base="xsd:string">
  <xsd:length value="8" />
  </xsd:restriction>
  </xsd:simpleType>
  </xsd:element>
- <xsd:element minOccurs="0" name="TEMP-COMMAND" maxOccurs="1">
- <xsd:complexType>
- <xsd:sequence>
- <xsd:element minOccurs="1" name="TEMP-IOCMD" maxOccurs="1">
- <xsd:simpleType>
- <xsd:restriction base="xsd:string">
  <xsd:length value="3" />
  </xsd:restriction>
  </xsd:simpleType>
  </xsd:element>
```

```
- <xsd:element minOccurs="1" name="TEMP-FILLER" maxOccurs="1">
- <xsd:simpleType>
- <xsd:restriction base="xsd:string">
    <xsd:length value="5" />
    </xsd:restriction>
```

```
  </xsd:simpleType>
  </xsd:element>
  </xsd:sequence>
  </xsd:complexType>
  </xsd:element>
- <xsd:element minOccurs="1" name="IN-LAST-NAME" maxOccurs="1">
- <xsd:simpleType>
- <xsd:restriction base="xsd:string">
  <xsd:length value="10" />
  </xsd:restriction>
  </xsd:simpleType>
  </xsd:element>
- <xsd:element minOccurs="1" name="IN-FIRST-NAME" maxOccurs="1">
- <xsd:simpleType>
- <xsd:restriction base="xsd:string">
  <xsd:length value="10" />
  </xsd:restriction>
  </xsd:simpleType>
  </xsd:element>
- <xsd:element minOccurs="1" name="IN-EXTENSION" maxOccurs="1">
- <xsd:simpleType>
- <xsd:restriction base="xsd:string">
  <xsd:length value="10" />
  </xsd:restriction>
  </xsd:simpleType>
  </xsd:element>
- <xsd:element minOccurs="1" name="IN-ZIP-CODE" maxOccurs="1">
- <xsd:simpleType>
- <xsd:restriction base="xsd:string">
  <xsd:length value="7" />
  </xsd:restriction>
  </xsd:simpleType>
  </xsd:element>
  </xsd:sequence>
  </xsd:complexType>
  </xsd:element>
  </xsd:sequence>
  </xsd:complexType>
  </xsd:element>
  </xsd:sequence>
  <xsd:attribute type="xsd:string" use="required" fixed="IMS/
Transactions/Segment1" name="location" />
  </xsd:complexType>
  </xsd:element>
  </xsd:all>
  </xsd:complexType>
  </xsd:element>
  </xsd:schema>
```

Reference: Segment One XML Request Document

The following is an example of a segment one XML request document:

```
 <?xml version="1.0" encoding="UTF-8" ?>
- <IMS>
- <Transaction location="IMS/Transactions/Segment1">
- <Segment>
- <INPUT-MSG>
  <IN-FILL />
- <TEMP-COMMAND>
  <TEMP-IOCMD>ADD</TEMP-IOCMD>
  <TEMP-FILLER />
  </TEMP-COMMAND>
  <IN-LAST-NAME>Doe</IN-LAST-NAME>
  <IN-FIRST-NAME>John</IN-FIRST-NAME>
  <IN-EXTENSION>1111111111</IN-EXTENSION>
  <IN-ZIP-CODE>2222222</IN-ZIP-CODE>
  </INPUT-MSG>
  </Segment>
  </Transaction>
  </IMS>
```

Reference: Segment One Response Schema

The following is an example of the segment one response schema:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <!--  Generated by the iBSE 2005-05-02T19:16:28Z
  -->
- <xsd:schema xml:lang="en" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:iwaysoftware:IMS/Transactions/Segment1_Response"
attributeFormDefault="unqualified" elementFormDefault="qualified">
- <xsd:element name="IMS">
- <xsd:complexType>
- <xsd:all>
- <xsd:element name="Transaction">
- <xsd:complexType>
- <xsd:sequence>
- <xsd:element name="Segment" maxOccurs="unbounded">
- <xsd:complexType>
- <xsd:sequence>
- <xsd:element minOccurs="1" name="INPUT-MSG" maxOccurs="1">
- <xsd:complexType>
- <xsd:sequence>
- <xsd:element minOccurs="1" name="IN-FILL" maxOccurs="1">
- <xsd:simpleType>
- <xsd:restriction base="xsd:string">
  <xsd:length value="4" />
  </xsd:restriction>
  </xsd:simpleType>
  </xsd:element>
- <xsd:element minOccurs="0" name="IN-COMMAND" maxOccurs="1">
- <xsd:simpleType>
- <xsd:restriction base="xsd:string">
  <xsd:length value="8" />
  </xsd:restriction>
  </xsd:simpleType>
  </xsd:element>
- <xsd:element minOccurs="0" name="TEMP-COMMAND" maxOccurs="1">
- <xsd:complexType>
- <xsd:sequence>
- <xsd:element minOccurs="1" name="TEMP-IOCMD" maxOccurs="1">
- <xsd:simpleType>
- <xsd:restriction base="xsd:string">
  <xsd:length value="3" />
  </xsd:restriction>
  </xsd:simpleType>
  </xsd:element>
- <xsd:element minOccurs="1" name="TEMP-FILLER" maxOccurs="1">
- <xsd:simpleType>
```

```
 - <xsd:restriction base="xsd:string">
  <xsd:length value="5" />
  </xsd:restriction>
  </xsd:simpleType>
  </xsd:element>
  </xsd:sequence>
  </xsd:complexType>
  </xsd:element>
- <xsd:element minOccurs="1" name="IN-LAST-NAME" maxOccurs="1">
- <xsd:simpleType>
- <xsd:restriction base="xsd:string">
  <xsd:length value="10" />
  </xsd:restriction>
  </xsd:simpleType>
  </xsd:element>
- <xsd:element minOccurs="1" name="IN-FIRST-NAME" maxOccurs="1">
- <xsd:simpleType>
- <xsd:restriction base="xsd:string">
  <xsd:length value="10" />
  </xsd:restriction>
  </xsd:simpleType>
  </xsd:element>
- <xsd:element minOccurs="1" name="IN-EXTENSION" maxOccurs="1">
- <xsd:simpleType>
- <xsd:restriction base="xsd:string">
  <xsd:length value="10" />
  </xsd:restriction>
  </xsd:simpleType>
  </xsd:element>
- <xsd:element minOccurs="1" name="IN-ZIP-CODE" maxOccurs="1">
- <xsd:simpleType>
- <xsd:restriction base="xsd:string">
  <xsd:length value="7" />
  </xsd:restriction>
  </xsd:simpleType>
  </xsd:element>
  </xsd:sequence>
  </xsd:complexType>
  </xsd:element>
  </xsd:sequence>
  </xsd:complexType>
  </xsd:element>
  </xsd:sequence>
  </xsd:complexType>
  </xsd:element>
  </xsd:all>
  </xsd:complexType>
  </xsd:element>
  </xsd:schema>
```

Reference: Segment One XML Response Document

The following is an example of the segment one XML response document:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <IMS xmlns="uri:iwaysoftware:IMS/Transactions/Segment1_Response">
- <Transaction>
- <Segment>
- <RESPONSE-MSG>
  <FILLER1>ENTRY WAS ADDED ADD Doe John 111111111122</FILLER1>
  </RESPONSE-MSG>
  </Segment>
  </Transaction>
  </IMS>
```

Reference: Segment Two Request Schema

The following is an example of the segment two request schema:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <!--  Generated by the iBSE 2005-05-02T19:24:45Z
  -->
- <xsd:schema xml:lang="en" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:iwaysoftware:IMS/Transactions/Segment2_Request"
attributeFormDefault="unqualified" elementFormDefault="qualified">
- <xsd:element name="IMS">
- <xsd:complexType>
- <xsd:all>
- <xsd:element name="Transaction">
- <xsd:complexType>
- <xsd:sequence>
- <xsd:element name="Segment" maxOccurs="unbounded">
- <xsd:complexType>
- <xsd:sequence>
- <xsd:element minOccurs="1" name="INPUT-MSG" maxOccurs="1">
- <xsd:complexType>
- <xsd:sequence>
- <xsd:element minOccurs="1" name="IN-FILL" maxOccurs="1">
- <xsd:simpleType>
- <xsd:restriction base="xsd:string">
  <xsd:length value="4" />
  </xsd:restriction>
  </xsd:simpleType>
  </xsd:element>
- <xsd:element minOccurs="0" name="IN-COMMAND" maxOccurs="1">
- <xsd:simpleType>
- <xsd:restriction base="xsd:string">
  <xsd:length value="8" />
  </xsd:restriction>
  </xsd:simpleType>
  </xsd:element>
- <xsd:element minOccurs="0" name="TEMP-COMMAND" maxOccurs="1">
- <xsd:complexType>
- <xsd:sequence>
- <xsd:element minOccurs="1" name="TEMP-IOCMD" maxOccurs="1">
- <xsd:simpleType>
- <xsd:restriction base="xsd:string">
  <xsd:length value="3" />
  </xsd:restriction>
  </xsd:simpleType>
  </xsd:element>
- <xsd:element minOccurs="1" name="TEMP-FILLER" maxOccurs="1">
- <xsd:simpleType>
- <xsd:restriction base="xsd:string">
  <xsd:length value="5" />
  </xsd:restriction>
  </xsd:simpleType>
  </xsd:element>
  </xsd:sequence>
  </xsd:complexType>
  </xsd:element>
- <xsd:element minOccurs="1" name="IN-LAST-NAME" maxOccurs="1">
- <xsd:simpleType>
- <xsd:restriction base="xsd:string">
  <xsd:length value="10" />
  </xsd:restriction>
  </xsd:simpleType>
  </xsd:element>
- <xsd:element minOccurs="1" name="IN-FIRST-NAME" maxOccurs="1">
- <xsd:simpleType>
- <xsd:restriction base="xsd:string">
  <xsd:length value="10" />
  </xsd:restriction>
  </xsd:simpleType>
  </xsd:element>
- <xsd:element minOccurs="1" name="IN-EXTENSION" maxOccurs="1">
- <xsd:simpleType>
- <xsd:restriction base="xsd:string">
  <xsd:length value="10" />
  </xsd:restriction>
  </xsd:simpleType>
  </xsd:element>
- <xsd:element minOccurs="1" name="IN-ZIP-CODE" maxOccurs="1">
- <xsd:simpleType>
- <xsd:restriction base="xsd:string">
  <xsd:length value="7" />
  </xsd:restriction>
  </xsd:simpleType>
  </xsd:element>
  </xsd:sequence>
  </xsd:complexType>
  </xsd:element>
  </xsd:sequence>
  </xsd:complexType>
  </xsd:element>
  </xsd:sequence>
  <xsd:attribute type="xsd:string" use="required" fixed="IMS/
Transactions/Segment2" name="location" />
  </xsd:complexType>
  </xsd:element>
  </xsd:all>
  </xsd:complexType>
  </xsd:element>
  </xsd:schema>
```

Reference: Segment Two XML Request Document

The following is an example of the segment two XML request document:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <IMS>
- <Transaction location="IMS/Transactions/Segment2">
- <Segment>
- <INPUT-MSG>
  <IN-FILL />
- <IN-COMMAND>
  <TEMP-IOCMD>END</TEMP-IOCMD>
  <TEMP-FILLER />
  </IN-COMMAND>
  </INPUT-MSG>
  </Segment>
  </Transaction>
  </IMS>
```

Reference: Segment Two Response Schema

The following is an example of the segment two response schema:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <!--  Generated by the iBSE 2005-05-02T19:24:45Z
  -->
- <xsd:schema xml:lang="en" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:iwaysoftware:IMS/Transactions/Segment2_Response"
attributeFormDefault="unqualified" elementFormDefault="qualified">
- <xsd:element name="IMS">
- <xsd:complexType>
- <xsd:all>
- <xsd:element name="Transaction">
- <xsd:complexType>
- <xsd:sequence>
- <xsd:element name="Segment" maxOccurs="unbounded">
- <xsd:complexType>
- <xsd:sequence>
- <xsd:element minOccurs="1" name="INPUT-MSG" maxOccurs="1">
- <xsd:complexType>
- <xsd:sequence>
- <xsd:element minOccurs="1" name="IN-FILL" maxOccurs="1">
- <xsd:simpleType>
- <xsd:restriction base="xsd:string">
  <xsd:length value="4" />
  </xsd:restriction>
  </xsd:simpleType>
  </xsd:element>
- <xsd:element minOccurs="0" name="IN-COMMAND" maxOccurs="1">
- <xsd:simpleType>
- <xsd:restriction base="xsd:string">
  <xsd:length value="8" />
  </xsd:restriction>
  </xsd:simpleType>
  </xsd:element>
- <xsd:element minOccurs="0" name="TEMP-COMMAND" maxOccurs="1">
- <xsd:complexType>
- <xsd:sequence>
- <xsd:element minOccurs="1" name="TEMP-IOCMD" maxOccurs="1">
- <xsd:simpleType>
- <xsd:restriction base="xsd:string">
  <xsd:length value="3" />
  </xsd:restriction>
  </xsd:simpleType>
  </xsd:element>
- <xsd:element minOccurs="1" name="TEMP-FILLER" maxOccurs="1">
- <xsd:simpleType>
- <xsd:restriction base="xsd:string">
  <xsd:length value="5" />
  </xsd:restriction>
  </xsd:simpleType>
  </xsd:element>
  </xsd:sequence>
  </xsd:complexType>
  </xsd:element>
- <xsd:element minOccurs="1" name="IN-LAST-NAME" maxOccurs="1">
- <xsd:simpleType>
- <xsd:restriction base="xsd:string">
  <xsd:length value="10" />
  </xsd:restriction>
  </xsd:simpleType>
  </xsd:element>
- <xsd:element minOccurs="1" name="IN-FIRST-NAME" maxOccurs="1">
- <xsd:simpleType>
- <xsd:restriction base="xsd:string">
  <xsd:length value="10" />
  </xsd:restriction>
  </xsd:simpleType>
  </xsd:element>
- <xsd:element minOccurs="1" name="IN-EXTENSION" maxOccurs="1">
- <xsd:simpleType>
- <xsd:restriction base="xsd:string">
  <xsd:length value="10" />
  </xsd:restriction>
  </xsd:simpleType>
  </xsd:element>
- <xsd:element minOccurs="1" name="IN-ZIP-CODE" maxOccurs="1">
- <xsd:simpleType>
- <xsd:restriction base="xsd:string">
  <xsd:length value="7" />
  </xsd:restriction>
  </xsd:simpleType>
  </xsd:element>
  </xsd:sequence>
  </xsd:complexType>
  </xsd:element>
  </xsd:sequence>
  </xsd:complexType>
  </xsd:element>
  </xsd:sequence>
  </xsd:complexType>
  </xsd:element>
  </xsd:all>
  </xsd:complexType>
  </xsd:element>
  </xsd:schema>
```

Reference: Segment Two XML Response Document

The following is an example of the segment two XML response document:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <IMS xmlns="uri:iwaysoftware:IMS/Transactions/Segment2_Response">
- <Transaction>
- <Segment>
- <RESPONSE-MSG>
  <FILLER1>CONVERSATION HAS ENDED</FILLER1>
  </RESPONSE-MSG>
  </Segment>
  </Transaction>
  </IMS>
```

Reference: Ivtcv_add.cpy

The following is an example of an ivtcv_add.cpy file:

```
01  INPUT-MSG.
       02  IN-FILL        PICTURE X(4).
       02  IN-COMMAND     PICTURE X(8).
       02  TEMP-COMMAND REDEFINES IN-COMMAND.
           04  TEMP-IOCMD    PIC X(3).
           04  TEMP-FILLER   PIC X(5).
       02  IN-LAST-NAME   PICTURE X(10).
       02  IN-FIRST-NAME  PICTURE X(10).
       02  IN-EXTENSION   PICTURE X(10).
       02  IN-ZIP-CODE    PICTURE X(7).
```

Reference: Ivtcv_end.cpy

The following is an example of an ivtcv_end.cpy file:

```
01  INPUT-MSG.
       02  IN-FILL        PICTURE X(4).
       02  IN-COMMAND.
           04  TEMP-IOCMD    PIC X(3).
           04  TEMP-FILLER   PIC X(5).
```

Reference: Ivtcv_response.cpy

The following is an example of an ivtcv_response.cpy file:

```
  01  RESPONSE-MSG.
              05  FILLER         PICTURE X(80).
```

Appendix D

Transaction Adapter for IMS Debugging and Troubleshooting

This appendix includes tips and techniques for debugging the adapter.

In this appendix:

Transaction Adapter for IMS Troubleshooting

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

```
Unable to process request:java.lang.IllegalStateException: Error reading
segment ret: 4 res: Datastore not found
```

Accompanied by:

```
IMS Connect Message:
Message failed, ORIGIN=6686 _35775912 to DESTID=IMS8A ; R=4, S=NF
```

Possible Cause: Datastore is not active.

Solution: Start or cycle imsconnect job.

Index

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.*

To request permission to repurpose copyrighted material, please contact Frances Gambino, Vice President, Technical Content Management at *Frances_Gambino@ibi.com.*

Information Builders

iWay

iWay Transaction Adapter for IMS User's Guide

Version 7.0.x and Higher

DN3502278.0418

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