IBM WebSphere InterChange Server

Access Development Guide

Version 4.2

Note!
Before using this information and the product it supports, read the information in "Notices" on page 103.

20March2004

This edition of this document applies to IBM WebSphere InterChange Server, version 4.2.2, and to all subsequent releases and modifications until otherwise indicated in new editions.

To send us your comments about this document, email doc-comments@us.ibm.com. We look forward to hearing from you.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

About this document

IBM(R) WebSphere(R) InterChange Server and its associated toolset are used with IBM WebSphere Business Integration Adapters to provide business process integration and connectivity among leading e-business technologies and enterprise applications.

This document describes how to use the IBM Server Access Interface APIs to enable a call-triggered flow capability. A call-triggered flow is one that is initiated by an access client process, which can then create business objects and execute collaborations.

Audience

This document is for IBM WebSphere customers, consultants, or resellers who create or modify collaborations. Before you start, you should understand all the concepts explained in the manual *Technical Introduction to IBM WebSphere InterChange Server.*

To implement the Server Access Interface APIs, you should know standard programming concepts and practice as well as the Java (TM) programming language. The Server Access APIs are based on the Java programming language.

Prerequisites for this document

This manual assumes that you are starting with a specification, flow chart, or pencil design. It does not cover analysis of business processes, development of collaborations or connectors, or design of business objects.

Note: In this document backslashes (\) are used as the convention for directory paths. For UNIX installations, substitute backslashes with slashes (/). All file path names are relative to the directory where the IBM CrossWorlds product is installed on your system.

How to use this manual

The *Server Access Interface Development Guide* is organized as follows:

Part I: Getting Started

Chapter 1, "Introduction to the Server Access Interface," on page 3	Is an overview of the Server Access Interface.
Chapter 2, "Setting up the access-client environments," on page 11	Tells you how to install and set up your development and run-time environment.
Chapter 3, "Configuring collaborations for call-triggered flows," on page 15	Shows you how to configure collaborations for use with access clients.
Chapter 4, "Implementing an access client," on page 21	Provides an overview of how to implement an access client to execute a collaboration.

Part II: Tutorial

	Chapter 5, "A sample servlet with HTML data-handling capabilities," on page 29	Shows a servlet written in Java that uses the APIs.
Part III: Server Access Interface API Reference		
	Chapter 6, "IAccessEngine interface," on page 51	Contains syntax and code snippets that show how to use methods in the `IAccessEngine` interface.
	Chapter 7, "IInterchangeAccessSession interface," on page 53	Contains syntax and code snippets that show how to use methods in the `IInterchangeAccessSession` interface.
	Chapter 8, "IBusinessObject interface," on page 61	Contains syntax and code snippets that show how to use methods in the `IBusinessObject` interface.
	Chapter 9, "IBusinessObjectArray interface," on page 87	Contains syntax and code snippets that show how to use methods in the `IBusinessObjectArray` interface.
	Chapter 10, "Server Access Interface exceptions," on page 91	Describes the exceptions of the Server Access Interface API.

Related documents

The complete set of documentation available with this product describes the features and components common to all WebSphere InterChange Server installations, and includes reference material on specific components.

This document contains many references to two other documents: the *System Installation Guide for Windows* or for *UNIX* and the *Implementation Guide for WebSphere InterChange Server*. If you choose to print this document, you may want to print those guides as well.

Before using this document, you should read the *Technical Introduction to IBM WebSphere InterChange Server* to understand how collaborations and connectors use business objects and maps.

You can install the documentation from the following sites:

- For InterChange Server documentation:
  http://www.ibm.com/websphere/integration/wicserver/infocenter
- For collaboration documentation:
  http://www.ibm.com/websphere/integration/wbicollaborations/infocenter
- For WebSphere Business Integration Adapters documentation:
  http://www.ibm.com/websphere/integration/wbiadapters/infocenter

These sites contain simple directions for downloading, installing, and viewing the documentation.

Typographic conventions

This document uses the following conventions:

`courier font`	Indicates a literal value, such as a command name, file name, information that you type, or information that the system prints on the screen.
bold	Indicates a new term the first time that it appears.

italic	Indicates a variable name or a cross-reference. When you view a IBM CrossWorlds document as a PDF file, cross references are both italic and blue. You can click on a cross-reference to jump to the target information.
`italic courier`	Indicates a variable name within literal text.
`boxed courier`	Separates a code fragment from the rest of the text.
blue text	Blue text, which is visible only when you view the manual online, indicates a cross-reference hyperlink. Click any blue text to jump to the object of the reference.
{ }	In a syntax line, curly braces surround a set of options from which you must choose one and only one.
`[ ]`	In a syntax line, brackets surround an optional parameter.
`...`	In a syntax line, ellipses indicate a repetition of the previous parameter. For example, `option[,...]` means that you can enter multiple, comma-separated options.

New in this release

This chapter describes the following new features of the Server Access Interface Development Guide for the IBM WebSphere InterChange Server development environment.

New in WebSphere InterChange Server version 4.2.2

February 2004

This release provides additional information on the `OAport` configuration property, including how to set it within the InterChange Server configuration file. For more information, see "Generating a persistent .ior file" on page 12.

December 2003

For this release of InterChange Server, the following changes have been made to this guide:

- Terminology, product names, file names, path names, and copyright information were updated in this manual for the WebSphere InterChange Server version 4.2.2 release.
- WebSphere Server Access now uses the IBM Java Object Request Broker (ORB) to handle communication between the access client and InterChange Server. For more information, see "Generating a persistent .ior file" on page 12.

New in release 4.1.1

Updated in March, 2003. The "CrossWorlds" name is no longer used to describe an entire system or to modify the names of components or tools, which are otherwise mostly the same as before. For example "CrossWorlds System Manager" is now "System Manager," and "CrossWorlds InterChange Server" is now "IBM WebSphere InterChange Server."

This product has been internationalized. For details in this guide, see the following:

- "Locales and encoding" on page 23
- "setLocale(String)" on page 60
- "Internationalization considerations," on page 97

New in release 4.1

The changes made in IBM WebSphere InterChange Server version 4.1 do not affect the content of this document. However, the following changes have been made to the document itself:

- The introductory material in Chapter 1, "Introduction to the Server Access Interface," on page 3 has been updated to use more consistent terminology.
- Chapters on the interfaces of the Server Access Interface API (Chapter 6 through Chapter 10) have been updated to provide more information and more consistent terminology.
- A index has been prepared for the guide.

New in release 4.0.1

The changes made in IBM WebSphere InterChange Server version 4.0.1 do not affect the content of this document.

New in release 4.0.0

The IBM WebSphere InterChange Server version 4.0.0 of this guide includes the following changes:

- The sample servlet in Chapter 5, "A sample servlet with HTML data-handling capabilities," on page 29 has been updated to include new initialization parameters.
- Several corrections and clarifications have been made throughout the guide.

Part 1. Getting started

Chapter 1. Introduction to the Server Access Interface

The IBM WebSphere business integration system **Server Access Interface** is an API that allows an external process to request execution of a collaboration inside IBM WebSphere InterChange Server (ICS). This external process, called an **access client**, sends an access request to initiate a call-triggered flow.

This chapter provides an overview of the Server Access Interface, how it enables business-to-business connectivity, and how to begin developing site-specific solutions using the Server Access Interface API.

The chapter contains the following sections:

Call-triggered flow

The Server Access Interface is an API that allows an external process to request execution of a collaboration inside IBM WebSphere InterChange Server. A **collaboration** represents a business process that can involve several applications. By using Server Access Interface, this external process, called an **access client**, can obtain data from applications that ICS handles through executing a collaboration.

The Server Access Interface makes it possible for WebSphere InterChange Server to receive requests for execution of a collaboration directly, without receiving a triggering event from a connector. The requests that the access client sends are called **access requests**. To send an access request, an access client issues a call to a method in the Server Access Interface instead of actually sending an event. Therefore, the flow trigger that an access client initiates is called a **call-triggered flow**, instead of the event-triggered flow that a connector initiates (see Figure 1).

The call-triggered flow is handled with the economy and transparency of an event-triggered flow. The main operational distinction is that call-triggered flows are processed synchronously and are therefore *not* persistent within the WebSphere InterChange Server system. By contrast, the event-triggered flows are processed asynchronously and are persistent. For more on how these flows are processed in the system, see the *Technical Introduction to IBM WebSphere InterChange Server*.

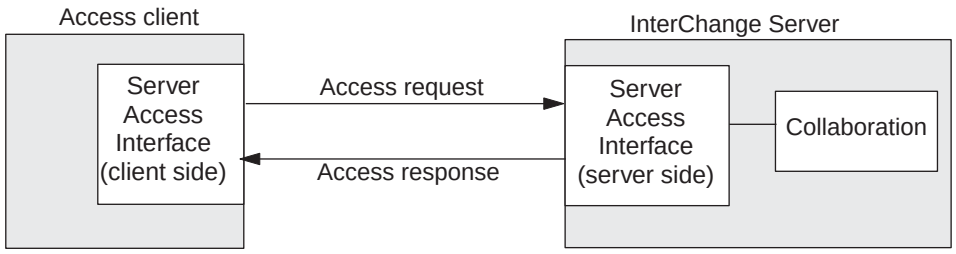

Figure 1. Call-triggered flow

As Figure 1 shows, an access request that an access client initiates involves the following steps:

1. The access client creates the **triggering access data**, which it sends to WebSphere InterChange server during the access request. This data is what triggers the specified collaboration; that is, the collaboration requires this data to begin execution.
2. The access client calls a method of the Server Access Interface API to send a **triggering access call** to the Server Access Interface within InterChange Server. The triggering access call includes the triggering access data and the name of the collaboration to execute. Through this method call, the access client performs an access request, which initiates the call-triggered flow.
3. The Server Access Interface within IBM WebSphere InterChange Server receives the triggering access call, performing any needed conversion of its triggering access data to a system business object. For more information on this data conversion, see "The role of IBM WebSphere business integration data handlers" on page 4.
4. The Server Access Interface within InterChange Server sends the triggering access data to the specified collaboration to trigger its execution.
5. Once the collaboration completes, it sends the resulting business object to the Server Access Interface.
6. The Server Access Interface performs any needed conversion from the resulting business object to the triggering access data's original format then performs the **access response** to send the access-response data back to the access client. For more information on this data conversion, see "The role of IBM WebSphere business integration data handlers" on page 4.

This section provides the following additional information about call-triggered flow:

- The role of IBM WebSphere business integration data handlers
- Call-triggered flow example

The role of IBM WebSphere business integration data handlers

An IBM WebSphere business integration data handler converts between serialized data and an IBM WebSphere business object. These data handlers support a variety of data formats for the serialized data. The Server Access Interface API allows the access client to send a triggering event formatted in one of several different formats. If the triggering access data is in XML, the Server Access Interface within WebSphere InterChange server makes calls to the XML data handler, which parses the triggering access data and converts it to the IBM WebSphere data format: a business object. Optionally, the access client can pass the resulting business object

from a collaboration response to the Server Access Interface, which calls the appropriate the data handler for conversion back to the incoming (in this case, XML) format.

To invoke a data handler, the Server Access Interface must first locate a top-level data-handler meta-object that it uses to create an instance of a data handler. The top-level meta-object for InterChange Server is `MO_Server_DataHandler` and it resides on the same machine as WebSphere InterChange server. The Server Access Interface Development software includes the XML data handler, EDI data handler, NameValue data handler, FixedWidth data handler, and Delimiter data handler. It also supports development of custom data handlers. By default, the `MO_Server_DataHandler` meta-object is configured so that the Server Access Interface automatically calls the XML data handler when it receives serialized data from an access client. If your access client uses serialized data in some format other than XML, you need to make sure that this `MO_Server_DataHandler` meta-object is modified to support the appropriate data handler. For more information, see the *Data Handler Guide*.

Call-triggered flow example

The Server Access Interface supports business-to-business transactions that require secure, reliable, external access by suppliers, vendors, or networked corporate units to backend applications. What follows is a business-to-business example involving two fictional firms, Firm A and Firm B.

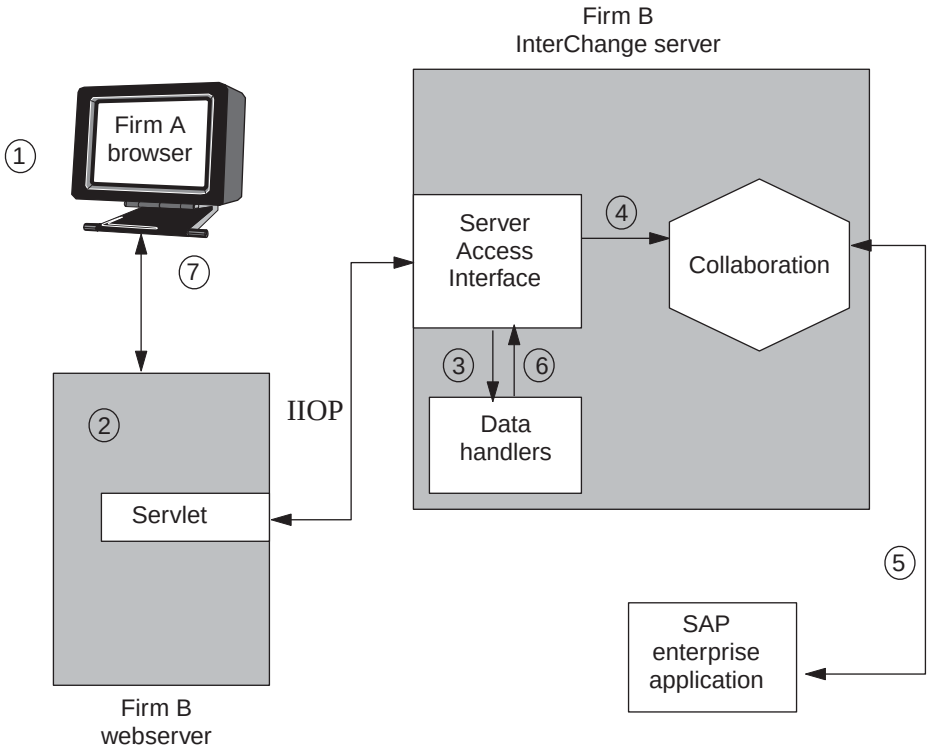

Figure 2. Business-to-business example

In this example, Firm A wishes to order 1,000 ICs from Firm B. For authorized suppliers such as Firm A, Firm B supports call-triggered flows to its IBM WebSphere InterChange Server-integrated backend. The process unfolds as follows:

1. A Firm A employee logs in to the Firm B Web site, entering an account ID and password. The employee then places an order for 1,000 ICs. The Firm B Web server authenticates the user as an authorized vendor.
2. The access client initiates a call-triggered flow at Firm B's e-business server (IBM WebSphere InterChange Server). Firm B's Server Access Interface receives and processes the API calls from the access client. The triggering access call indicates that the data is in XML format.
3. Firm A's call-triggered flow passes data to the XML data handler. This data handler converts the serialized data into Firm B's generic business-object format. Business object definitions are extracted from the DTDs in the XML data stream and from the data-handler meta-object.
4. Firm A's access client executes the collaboration inside the Firm B InterChange Server, launching an `Order_Generation` process. The business object uses a IBM WebSphere collaboration that is appropriately configured—one that is bound to a port with an access-client capability and that has a map to convey data to and from that port.
5. The business object is routed to a IBM WebSphere connector for SAP, which accesses Firm B's SAP/R3 application and places the order. (Firm B routes the order to its supplier sites for fulfillment). The result—order confirmation—is generated and passed via a connector back to the access client.
6. Firm A's access client sends the resulting business object to the XML data handler. The XML data handler parses and converts the result into an XML data stream.
7. The result is streamed to the Web server site, which launches a separate process to e-mail the Firm A employee with confirmation of the transaction, including the order number.

Overview of access-client development procedure

To develop an access client, you code the access-client source file and complete other tasks. The task of creating an access client includes the following general steps:

1. Set up the development environment. Install the IBM WebSphere business integration software including the `AccessInterfaces.idl` file and then use a utility to generate either Java or C++ stubs from the `AccessInterfaces.idl` file.
2. Configure a port of a collaboration for access and execution by a call-triggered flow. This step involves configuring external collaboration ports, which can handle access clients.
3. Implement and debug the access client (such as a web servlet) that executes the Server Access Interface API calls. Import the `IdlAccessInterfaces.*` classes, and implement Java code to do the following:
   - Get an access session to IBM WebSphere InterChange Server.
   - Send a triggering access call to a specified collaboration, including data handler calls
   - Execute a collaboration.
4. Configure the top-level data handler meta-object `MO_Server_DataHandler` to point to the data handler instances needed to convert data from the external format (sent from the access client) to the IBM WebSphere business-object format. For more information, see the *IBM WebSphere Business Integration Data Handler Guide*.

Figure 3 provides an overview of the access-client development process and provides a quick reference to chapters where you can find information on specific

topics. Note that if a team of people is available for access-client development, the major tasks of developing an access client can be done in parallel by different members of the development team.

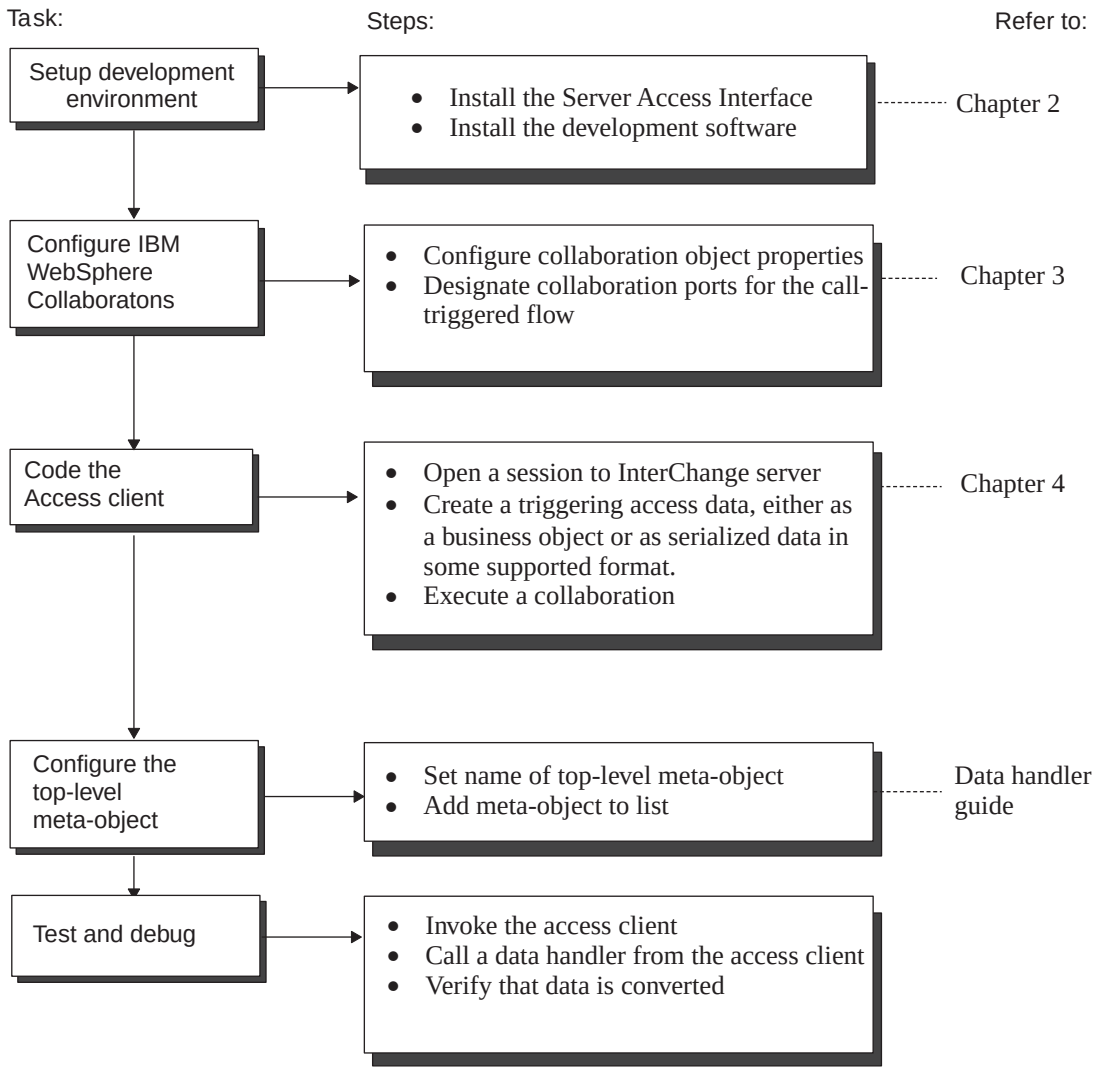

Figure 3. Overview of the access-client development task

Tools for access-client development

Because access clients are written Java, you can develop them on either a Windows or UNIX system. The following table lists the tools that IBM WebSphere provides for access-client development.

IBM WebSphere tool	Description
E-Business Development Kit (EDK)	Includes the following: • Sample data handlers • Stub file for extending `DataHandler` class
Server Access Interface API	Contains Java classes to access InterChange Server from within an access client.
Data Handler API	Contains a single class, `DataHandler`, which you extend to create a custom data handler.

IBM WebSphere tool	Description
IBM WebSphere Java Connector Development Kit (JCDK)	Includes Java classes to work with business objects.

E-Business development kit

The WebSphere Business Integration E-Business Development Kit (EDK) provides developers with tools to develop the custom software in the following table.

Custom software	Subdirectory of DevelopmentKits\edk
Connector	`ConnectorAgent`
Data handler	`DataHandler`
Protocol handler	`ProtocolHandler`
Access client	`ServerAccessInterfaces`
Utilities, including the XMLBORGEN utility (used by the XML data handler)	`Utilities`

As the previous table shows, the tools to develop access clients are in the `ServerAccessInterfaces` directory, under the `DevelopmentKits\edk` subdirectory of the *ProductDir* directory.

Sample access client

To assist with the development of an access client, EDK includes a sample access client in the IBM WebSphere directory: DevelopmentKits\edk\ServerAccessInterfaces\AccessSample.

This directory contains:

- The sample access client, `ATPServlet.java`, is a servlet that provides the ability to convert HTML data into a business object, which can then be sent to a collaboration in InterChange Server.
- A custom data handler, `HtmlDataHandler.java`, handles conversion between HTML data and a WebSphere Interchange Server business object.
- The `SampleRepos.jar` file, which contains repository definitions of the components used by the Access sample.
- The subdirectories listed in the following table contain additional sample files.

Name	Description
`collaborations`	Contains collaborations configured for call-triggered flows.
`DLMs`	Contains the required native maps.

Note: While this sample is useful to examine, it does *not* provide examples of all the functionality supported in the Server Access Interface API.

For more information, see Chapter 5, “A sample servlet with HTML data-handling capabilities,” on page 29

IBM WebSphere Server Access Interface API

The IBM WebSphere Server Access Interface API provides the following interfaces:

Server Access Interface	Description	For more information
`IAccessEngine`	Provides a method to bind the access client to InterChange Server	Chapter 6, "IAccessEngine interface," on page 51
`IInterchangeAccessSession`	Provides methods to control access to an access session in InterChange Server	Chapter 7, "IInterchangeAccessSession interface," on page 53
`IBusinessObject`	Provides methods to perform business object operations such as getting, setting, and comparing attribute values	Chapter 8, "IBusinessObject interface," on page 61
`IBusinessObjectArray`	Provides methods that allow an access client to interact with and manipulate business object arrays	Chapter 9, "IBusinessObjectArray interface," on page 87

Note: The methods in the interfaces that the previous table lists throw the exceptions described in Chapter 10, "Server Access Interface exceptions," on page 91.

IBM WebSphere data handler API

The IBM WebSphere data handler API provides a single class, called `DataHandler`. The abstract `DataHandler` base class facilitates the development of a custom data handler. This class contains the methods that populate a business object with values extracted from serialized input data, and methods that serialize a business object into a string or a stream. The class also includes utility methods that a custom data handler can use. You derive a custom data handler from this `DataHandler` class. For information on the methods in the `DataHandler` class, see the *Data Handler Guide*.

Note: You only need to consider developing a custom data handler if your access client formats its serialized data in some format other that one supported by existing IBM WebSphere Business Integration Server data handlers. For a list of these data handlers, see "The role of IBM WebSphere business integration data handlers" on page 4.

IBM WebSphere Java connector development kit

If you develop a custom data handler, you must use methods of some of the classes in the IBM WebSphere Java Connector Development Kit (JCDK) to work with business objects. As you develop your data handler, you might need to import additional JCDK classes, such as `CxCommon.CxObjectContainerInterface` or `CxCommon.CxObjectAttr`. For reference information on the JCDK methods, see the *Connector Reference: Java Class Library*.

Chapter 2. Setting up the access-client environments

This chapter shows you how to set up your environment to develop and run access clients. The chapter contains the following sections:

Setting up the development environment

The development environment for you access client requires that you have access to the Server Access Interface API stubs, which are part of the software that IBM WebSphere installer installs. Therefore, to include calls to the Server Access Interface API in your access client, you must have access to the following software:

- A IBM Java ORB development environment (version 4.5 or later; consult your *IBM WebSphere Business Integration System Installation Guide* for the current release)
- A Java development environment and JDK 1.3.1
- The current release of IBM WebSphere software
- InterChange Server that is booted and running
- An IBM WebSphere repository with collaborations that have been configured for call-triggered flow (For more information on how to perform this configuration, see Chapter 3, "Configuring collaborations for call-triggered flows," on page 15)

Once you have access to the software listed above, setting up the development environment for an access client involves the following steps:

- "Installing IBM WebSphere Server Access interface"—Install the Server Access Interface on the development machine.
- "Compiling the access client" on page 12—Create an executable for the access client.

Installing IBM WebSphere Server Access interface

To be able to develop an access client, you must install the Server Access Interface on the development machine. IBM WebSphere Installer installs the files associated with the IBM WebSphere Server Access Interface. It installs the directories and files shown in Table 1.

Table 1. Installed file structure for the IBM WebSphere Server Access Interface

Directory	Description
`DevelopmentKits\edk\ ServerAccessInterfaces`	Contains the `AccessInterfaces.idl` file for access clients.
`DevelopmentKits\edk\ ServerAccessInterfaces\ AccessSample`	Contains source code for the sample access client.
`repository\edk`	Contains file for `MO_Server_DataHandler` meta-object that defines which data handlers the Server Access Interface supports.

IBM WebSphere Installer installs the files in Table 1 automatically when it installs the IBM WebSphere software. To ensure that the Server Access Interface API is installed, make sure that the Server and Tools component is selected on the Select

Components screen of IBM WebSphere installer. When the installer installs this component, it automatically installs the directories and files listed in Table 1 on page 11. For information on IBM WebSphere Installer, see the *IBM WebSphere System Installation Guide for UNIX* or *for Windows*.

Note: IBM WebSphere installer also installs files needed by the IBM WebSphere-delivered data handlers. For more information, see the installation chapter in the *IBM WebSphere Data Handler Guide*.

Compiling the access client

When you are ready to compile your access client, you must make sure that the paths to the following files are on your classpath:

- The IBM WebSphere `crossworlds.jar` file
- The IBM Java Object Request Broker (ORB) jar files

You can use the `javac` compiler or any Integrated Development Environment (IDE).

Setting up the run-time environment

At run time, the access client does not need to reside on a machine that contains IBM WebSphere InterChange Server, nor does it need to reside on the same machine as the development environment. However, for the access client to be able to locate the InterChange Server instance it needs at run time, it must be able to locate the Object Request Broker (ORB) server, which keeps track of the locations of different CORBA objects (including InterChange Server instances) and communicates this information to ORB clients (such as an access client). To obtain the location of the ORB server, the access client can use the Interoperable Object Reference File that its ICS instance generates. When ICS starts or reboots, it generates an Interoperable Object Reference file, which has the `.ior` extension. The access client can use this file to locate the ORB server, and, in turn, to communicate with its ICS instance.

Therefore, for the access client to locate its ICS instance, you must take the following steps:

1. Request that InterChange Server generate a persistent `.ior` file.
2. Ensure that the machine on which the access client resides is able to locate the `.ior` file for its InterChange Server instance.

Each of these steps is described in more detail in the following sections.

Generating a persistent .ior file

When InterChange Server version 3.1.0 or later is booted up, it generates a new `.ior` file. However, InterChange Server dynamically assigns a port number for the ORB server. If the port number changes each time the server boots, the access client cannot depend on the `.ior` file to locate the ORB Server. Therefore, an access client needs InterChange Server to generate a **persistent** `.ior` file.

To have InterChange Server generate a persistent `.ior` file, you must edit the ICS configuration file (`InterchangeSystem.cfg`) in an XML editor and add a subsection for CORBA, if one does not already exist. Figure 4 on page 13 shows the XML code

that defines an *empty* CORBA subsection (one with *no* configuration parameter defined).

```
<tns:property>
   <tns:name>CORBA</tns:name>
   <tns:isEncrypted>false</tns:isEncrypted>
   <tns:updateMethod>system restart</tns:updateMethod>
   <tns:location>
      <tns:reposController>false</tns:reposController>
      <tns:reposAgent>false</tns:reposAgent>
      <tns:localConfig>true</tns:localConfig>
   </tns:location>
   XML definitions of CORBA properties go here
</tns:property>
```

Figure 4. XML definition of CORBA subsection

The CORBA subsection specifies the static port number with the `OAport` configuration parameter, which has the following syntax:

`OAport=`*`portNumber`*

For example, if the static port number is to be `15000`, assign a value of 15000 to its `OAport` parameter in the CORBA subsection. The following XML fragment would appear within the `<tns:property>` tag for the CORBA subsection, in the place indicated in Figure 4 with the string *"XML definitions of CORBA properties go here"*:

```
<tns:property>
  <tns:name>OAport</tns:name>
  <tns:value xml:space="preserve">15000</tns:value>
  <tns:isEncrypted>false</tns:isEncrypted>
  <tns:updateMethod>system restart</tns:updateMethod>
  <tns:location>
     <tns:reposController>false</tns:reposController>
     <tns:reposAgent>false</tns:reposAgent>
     <tns:localConfig>true</tns:localConfig>
  </tns:location>
</tns:property>
```

Important: The ICS configuration file is an XML file. To add the CORBA subsection and its configuration parameter, you must use an XML editor or must correctly format the appropriate XML tags.

For more information on the CORBA subsection in the configuration file, see the *IBM WebSphere System Installation Guide for UNIX* or *for Windows*.

Locating the .ior file

For the access client to locate the ORB server at run time, it must be able to locate the `.ior` file for its InterChange Server instance. Locating this file is not a problem if the access client and InterChange Server are on the same machine. However, if these two components are *not* on the same machine, you must take *one* of the following actions to ensure that the access-client machine can access the `.ior` file:

- Copy the `.ior` file that InterChange Server has generated to the machine on which the access client resides.
- Create a shared directory on the machine with InterChange Server and point the access-client machine to the directory.

Toggling event sequencing for access requests

When synchronous requests are sent to the collaboration using the access framework, the sequence of requests may not be important, especially when tuning for performance. By default, event sequencing is turned on at the collaboration level for synchronous access requests. To turn event sequencing off for synchronous access requests, edit the `InterchangeSystem.cfg` file and add the following lines:

```
[ACCESS]
EVENT_SEQUENCING=FALSE
```

Chapter 3. Configuring collaborations for call-triggered flows

This chapter shows you how to configure collaborations for call-triggered flows. You must configure the collaborations *before* executing them from an access client. Topics in this chapter include:

Important: To configure a collaboration for call-triggered flow, you must have installed all IBM WebSphere software and have InterChange Server up and running.

Using System Manager to implement a call-triggered flow option

You use System Manager to configure a collaboration for a call-triggered flow. To implement a call-triggered flow option for a collaboration, you must first create a new collaboration object from one of the existing collaboration templates in the repository. To create the new collaboration object, follow these steps:

1. In System Manager, right-click on the Collaboration Objects folder and choose New Collaboration Object.

   The Create New Collaboration Object dialog box opens, listing the installed templates in the Template Name column.
2. Click the name of a collaboration template from which you want to configure a collaboration object to support a call-triggered flow.
3. Enter a name for the collaboration object in the Collaboration object name field. Click Next.

   The Bind Collaboration Ports dialog box opens.

Once the collaboration object exists, you can configure it for call-triggered flow with the steps listed in Table 2.

Table 2. Configuring a collaboration port for a call-triggered flow

Configuration step	For more information
Designate collaboration ports for call-triggered flows and bind the port to the collaboration.	"Designating collaboration ports for call-triggered flows" on page 16
Select maps that associate the business object flow with the collaboration.	"Associating business objects and maps" on page 18
Set properties for the new collaboration object.	"Configuring collaboration object properties" on page 20

Note: A collaboration can have multiple call-triggered flow ports configured.

Table 2 on page 15 provides a summary of how to configure a collaboration for a call-triggered flow. For more on collaboration configuration and System Manager, see the *System Implementation Guide* and the *Collaboration Development Guide*.

Designating collaboration ports for call-triggered flows

For each collaboration you wish to configure for a call-triggered flow, you must perform the port configuration on the collaboration object.

To configure a collaboration port for a call-triggered flow:

1. Make sure that the Collaboration Object View window for your collaboration object displays in the Server Monitor area of the System Manager.

   If this window is not currently displaying, navigate to the Collaboration Objects folder in the System Manager object browser, and double-click the collaboration object that you want to configure.
2. Right-click the port you want to configure for call-triggered flow and choose Bind Port.

   The Configure Port dialog box opens (see Figure 5). Its default setting for type of port is `Internal`.

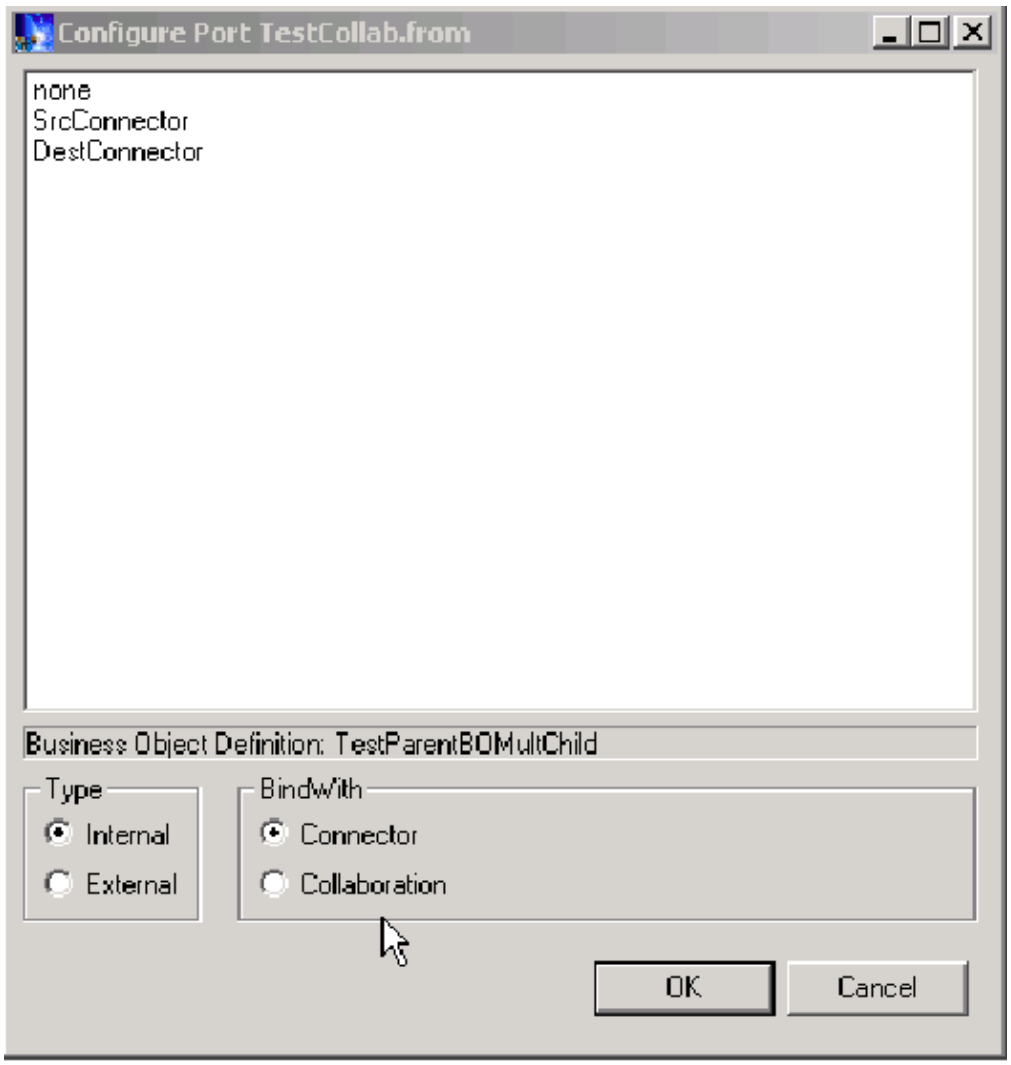

Figure 5. Configure port (internal) dialog box

3. Click `External` in the Type area.

   This displays the Configure Port (External) dialog box as shown Figure 6. In the Configure as area, the dialog box displays the type of port you have chosen to configure: Incoming or Outgoing.

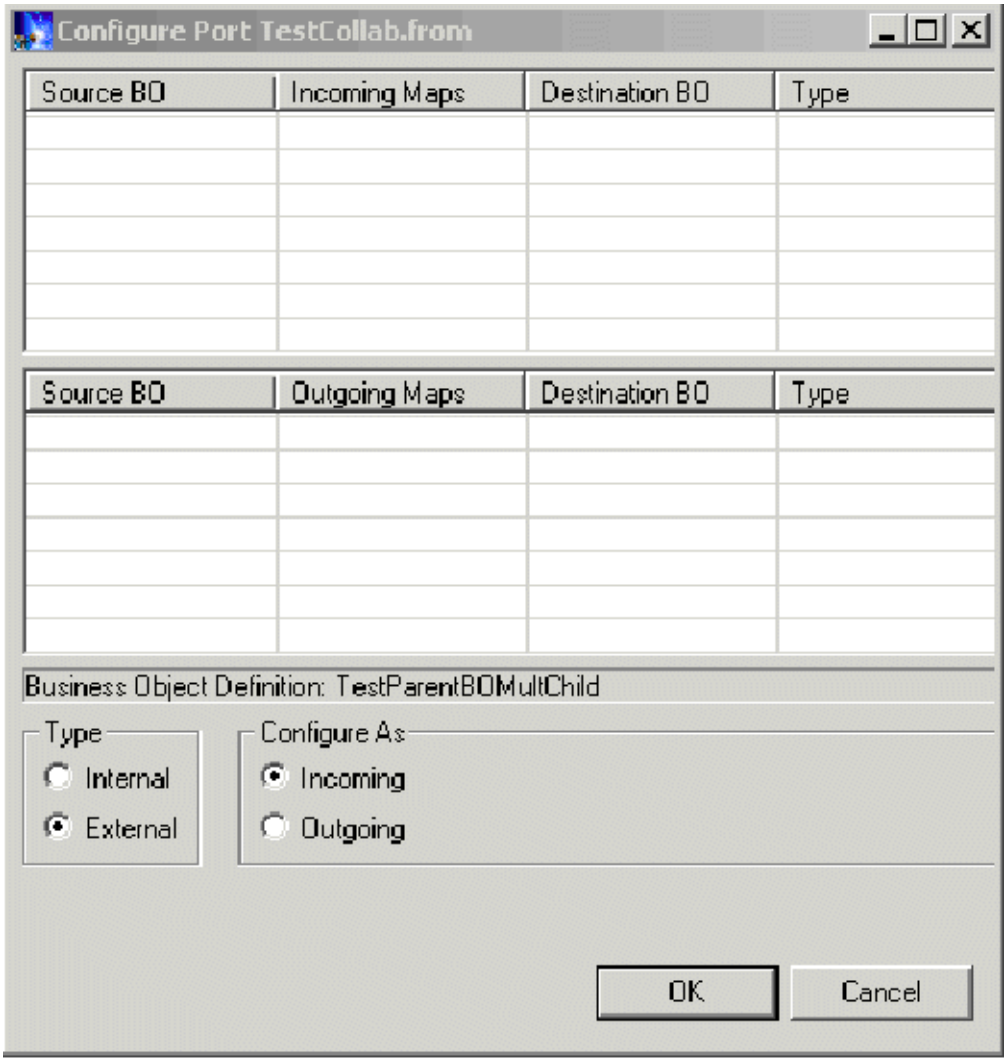

Figure 6. Configure port (external) dialog box

Associating business objects and maps

You must associate a map and a business object with the collaboration that you have configured for call-triggered flow. You do this in one of three ways:

- By clicking and dragging a business object from the Business Object folder in the Designer Directory at the left of the Object View area (Figure 7) to the Destination pocket in the Configure Port dialog box. When the business object is dropped into this pocket, all the maps that go from the source to destination are displayed in a pop-up window. You select the appropriate map.

or

- If you know the map you wish to associate with the collaboration, you can click and drag a map to the Incoming Maps or Outgoing Maps pocket in the Configure Port dialog box.

or

- If you don't want to associate any maps to this collaboration when executed by an external entity, leave the incoming and outgoing business objects columns empty.

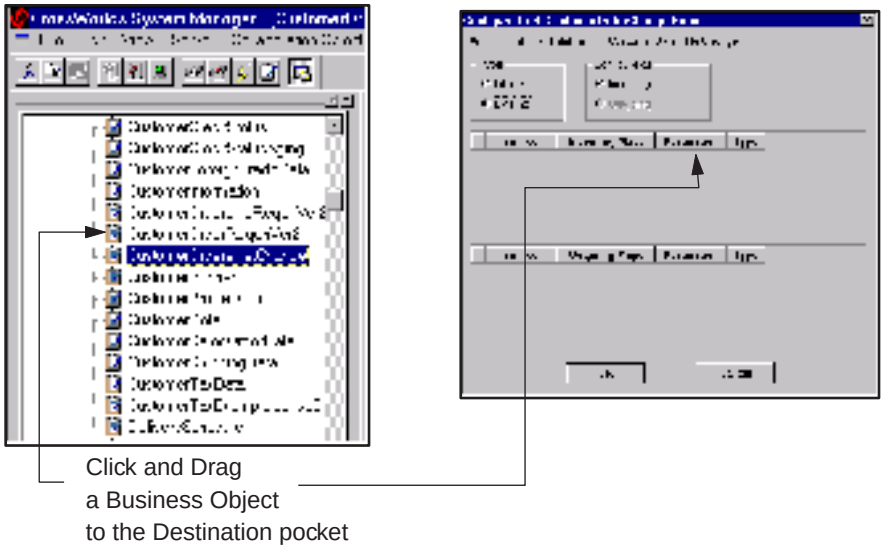

Figure 7. Mapping a business object to a call-triggered flow collaboration

Note: See the *System Implementation Guide* and the *Collaboration Development Guide* for more on collaborations, business objects, and maps.

Flow direction: Into the collaboration

Dragging a business object—The business object is used as the destination type for the collaboration. Select an appropriate map from the pop-up window. The destination will always be the business object definition that is shown in the Configure Port window.

Dragging a Map—The map is used when the call is made to the collaboration. Select a map that supports the destination business object.

Flow direction: Out of the collaboration

Dragging a business object—The business object options are used when the collaboration is returning the result.

Dragging a map—The map is used when the collaboration returns data or attributes to the requesting process.

Dragging a business object

To bind a business object type and map to a collaboration by dragging a business object:

1. Double-click the Business Object folder in the Designer Directory.

   This displays a list of business objects.
2. Click, drag, and drop the business object into the Destination pocket in the Configure Port dialog box. Choose Incoming or Outgoing Maps, depending on the port you are configuring.

This displays the map and business object in the Configure Port dialog box.

3. Choose a map from those displayed with the business object (there may be only one map displayed). Click OK.

Configuring collaboration object properties

For each collaboration you wish to configure for a call-triggered flow, set its number of concurrent events to zero (`0`). To configure the properties of a collaboration for a call-triggered flow:

1. Make sure that the Collaboration Object View window for your collaboration object displays in the Server Monitor area of the System Manager.

   If this window is not currently displaying, navigate to the Collaboration Objects folder in the System Manager object browser, and double-click the collaboration object that you want to configure.

2. Right-click the collaboration's icon (the center icon) and choose Properties.

   The Collaboration Properties dialog box opens.

3. Configure the properties of the collaboration object as desired.

   Important: Make sure that the property `Maximum number of concurrent events` is set to a value of `0`. Call-triggered flows are by default multi-threaded, so setting this property to `0` ensures that no additional threads are spawned by InterChange Server to provide the multi-threading capability. Consult the *WebSphere InterChange Server System Administration Guide* for further details about this property.

4. Click OK to close the Collaboration Properties dialog box.

Chapter 4. Implementing an access client

This chapter provides an overview of how to implement an access client, which can request execution of a collaboration with InterChange Server through the Server Access Interface API. Topics in this chapter include:

Creating an access session

Before an access client can issue an access request, it must first establish an **access session** with InterChange Server. To allow the access client to connect to InterChange Server, the `IAccessEngine` interface provides the IgetInterchangeAccessSession() method. This method creates the access session, which provides the access client with access to the Server Access Interface within InterChange Server (ICS). You must provide a valid ICS user name and password to the `IgetInterchangeAccessSession()` method as arguments.

Important: The ICS user name must be `admin`.

For a more detailed explanation of the `IAccessEngine` interface, see Chapter 6, "IAccessEngine interface," on page 51.

Issuing the access request

Once the access client has created an access session, it can send an access request to ICS. The access request is what initiates the call-triggered flow within ICS. Before it can send its triggering access call, the access client must generate the triggering access data that is sent to the collaboration. The Server Access Interface provides the following ways for an access client to issue an access request, based on the format of the triggering access data:

- "Sending a business object"
- "Sending serialized data" on page 23

Sending a business object

The access client can send its triggering access data encapsulated in an IBM WebSphere InterChange Server business object. The `IInterchangeAccessSession` interface provides methods for creating business objects and executing collaborations. For a more detailed explanation of this interface, see Chapter 7, "IInterchangeAccessSession interface," on page 53.

Sending a business object as triggering access data involves the following steps:

- "Creating the business object"
- "Operating on the business object"
- "Requesting execution of the collaboration"

Creating the business object

Table 3 shows the methods that the Server Access Interface API provides in the `IInterchangeAccessSession` interface for the access client to create a business object.

Table 3. IInterchangeAccessSession methods for creating a business object

Creating the business object	IInterchangeAccessSession method
Create a business object	IcreateBusinessObject()
Create a business object with a verb that specifies an operation on the object attributes.	IcreateBusinessObjectWithVerb()
Create a business object array that contains one or more attributes, each attribute having a business object as its type.	IcreateBusinessObjectArray()
Create a business object from data that is formatted in a specified MIME type.	IcreateBusinessObjectFrom()

Operating on the business object

Once the access client has created the business object, it can use the interfaces in Table 4 to perform any operations required to put the triggering access data into this object.

Table 4. Interfaces to access a business object

Type of business object	Server Access Interface API	For more information
Business object (single cardinality)	`IBusinessObject` Allows the access client to perform business object operations such as getting, setting, and comparing attribute values.	Chapter 8, "IBusinessObject interface," on page 61
Business object array	`IBusinessObjectArray` Allows the access client to interact with and manipulate business object arrays. The methods include setting or getting business object array elements, copying an array, adding a business object to an array, or fetching the number of elements in an business object array.	Chapter 9, "IBusinessObjectArray interface," on page 87

Requesting execution of the collaboration

The `IInterchangeAccessSession` interface provides the IexecuteCollaboration() method for sending a business object as the triggering access data in the triggering access call. This method tells the Server Access Interface within ICS to send the business object as the triggering access data to the specified collaboration.

Note: The collaboration, port, and business object must be configured and mapped for direct call access and manipulation.

Sending serialized data

The access client can send its triggering access data as serialized data in a specified MIME type. The Server Access Interface within the WebSphere InterChange Server (ICS) performs the data conversion necessary from the serialized data to an IBM WebSphere business object. Sending a serialized data involves a call to a single method of Server Access Interface API, IexecuteCollaborationExtFmt(). This method provides the following tasks for the access client:

- Specify a data handler (based on the MIME type of the serialized data) to convert the serialized data to a business object.
- Create the business object that triggers the collaboration.
- Set the verb to a specified value.
- Execute the collaboration.

Locales and encoding

By default, the access session uses the Locale value of the ICS. However, you may wish to change the Locale value to match the Locale value of a business object or collaboration that you are creating or executing through the access session.

Input data sent to the Server Access Interface must use Unicode encoding.

For an overview of Locales, see Appendix A, Internationalization Considerations.

For a description of the method for setting Locale values, see setLocale(String) in Chapter 7, "IInterchangeAccessSession interface," on page 53.

Obtaining the access response

A collaboration returns an access response to the access client through the return value of one of the methods in Table 5. The format of this access request depends on the method that the access client used to send the access request.

Table 5. Methods for obtaining the access response

Access request	Server Access Interface method	Format of access response
Sends triggering access data as a business object	IexecuteCollaboration()	Business object
Sends triggering access data as serialized data in a specified MIME type	IexecuteCollaborationExtFmt()	Serialized data (in the same MIME format as the access request)

Note: If your access response is in the form of an IBM WebSphere InterChange Server business object, you can use the methods of the interfaces listed in Table 4 on page 22 to operate on this business object.

Closing the access session

When the access client have completed its access request, it should take the steps in Table 6.

Table 6. Closing the access session

Task	Server Access Interface method
Release resources that the Server Access Interface within ICS is using for business objects and business object arrays	`IInterchangeAccessSession` methods: IreleaseBusinessObject(), IreleaseBusinessObjectArray()
Close the access session	`IAccessEngine` method: IcloseSession()

Note: A call to `IcloseSession()` releases the resources that the access session is using.

An example of implementing a call-triggering flow

Figure 8 shows a more detailed of a call-triggered flow, initiated, in this case, by an access client that is a client browser.

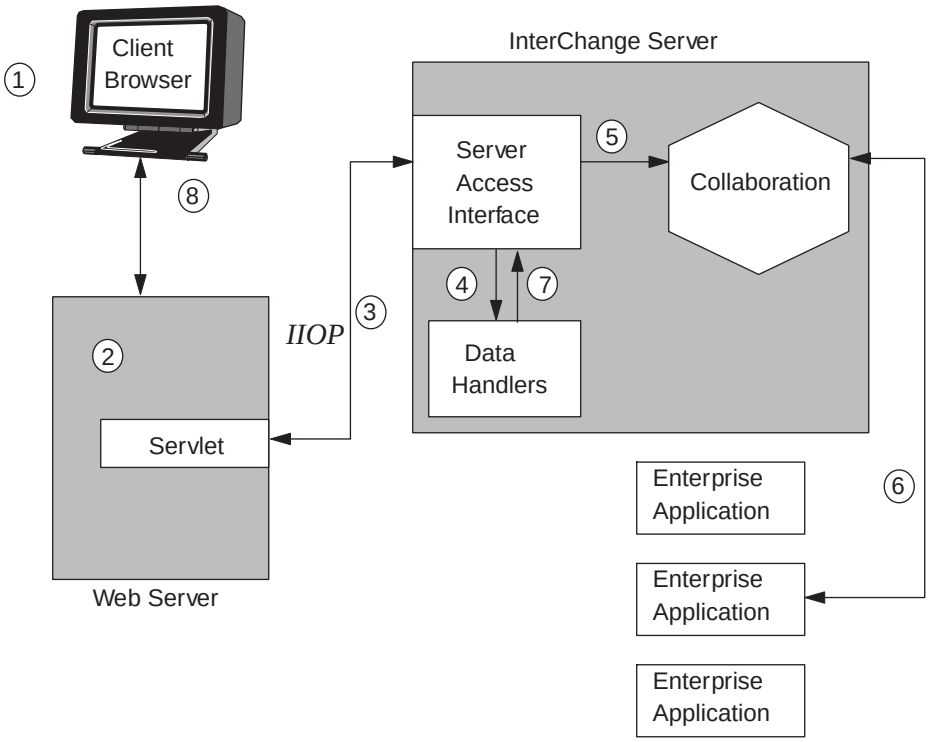

Figure 8. Sample call-triggered flow initiated by a client browser

As shown in Figure 8:

1. The client browser issues a request in a specific protocol and format (for example, an HTTP protocol and an XML data format).
2. The enterprise Web server loads a servlet to handle the request. This servlet is the access client. It is programmed to look for the name of the CORBA-compliant Interchange Server (from the CORBA registry).

3. The access client, via an IIOP connection, logs in to InterChange Server by creating an access session with the `IgetInterchangeAccessSession()` method of the `IAccessEngine` interface of the Server Access Interface API.

   Note: To execute the collaboration, InterChange Server does *not* create any threads of its own, but uses the CORBA thread. See the *Collaboration Development Guide* for details on how collaborations use threads.

4. The access client uses the `IcreateBusinessObjectFrom()` method in the `IInterchangeAccessSession` interface to transform the XML data into a generic IBM WebSphere InterChange Server business object. In response to this method call, the Server Access Interface within ICS invokes the XML data handler to perform the data transformation and then returns the business object to the access client.
5. The access client uses the `IexecuteCollaboration()` method in the `IInterchangeAccessSession` interface to send the triggering access call, which contains the business object as the triggering access data. This process requests the execution of a collaboration that manipulates the business object.

   Note: The Server Access Interface API also provides the `IexecuteCollaborationExtFmt()` method, which combines step 4 and step 5 into a single method call.

6. Traversing connectors, the collaboration places requests, sorts, and fetches data, manipulating enterprise applications as required. The collaboration returns requested data, or results of requested actions, to the access client in business-object format.
7. If the access client has used the `IexecuteCollaborationExtFmt()` method to issue the access request, it does not need to explicitly perform the actions in step 6. The `IexecuteCollaborationExtFmt()` method automatically transforms the business object back to its original format (in this case, the XML format) and returns this serialized data to the access client.
8. The results are delivered to the client browser.

As shown in Figure 8, the Web server handling the call loads a servlet to handle the call, which connects to WebSphere InterChange Server.

Part 2. Example

Chapter 5. A sample servlet with HTML data-handling capabilities

This chapter presents a typical e-commerce scenario and sample code that uses the Server Access Interface APIs. Topics covered in this chapter include:

The scenario

A common problem encountered in e-commerce environments is that of item availability and the prospect of assured delivery by a requested date. This class of problems is commonly known as available to promise, or ATP.

An enterprise that uses a supply chain optimization system or enterprise resource planning (ERP) system will generally query their system to determine whether a product can be delivered by the requested delivery date. Some firms, particularly those with online trading relationships with several vendors, may wish to determine product availability before committing to order the products.

An ATP capability means effectively accessing a firm's ERP or supply chain optimization system. In the following example, the Server Access Interface APIs are utilized to perform the following tasks:

- **Data conversion** - Convert an incoming quote object from its HTML format to an IBM WebSphere business object.
- **Collaboration execution** - Trigger a collaboration that retrieves the ATP data for each item encountered in the incoming quote object.
- **Results retrieval** - Return results as an HTML table.

Figure 9 depicts a single available to promise collaboration.

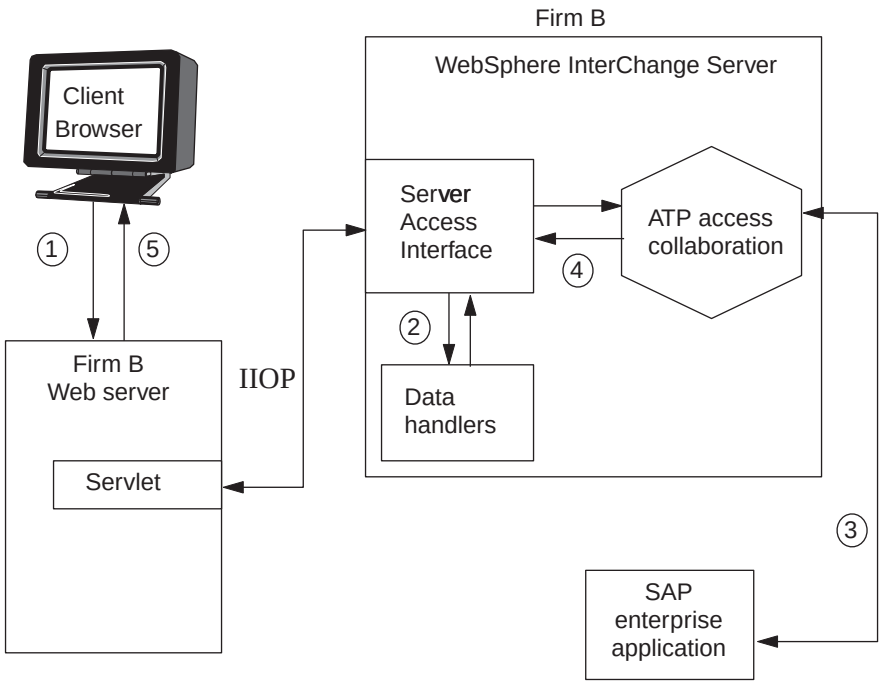

Figure 9. An available-to-promise e-commerce scenario

1. The browser client sends an HTML form that contains the data corresponding to an IncomingQuote object. The IncomingQuote object is HTML formatted data supplied by a third-party application.
2. The servlet (see example code below) uses the Server Access Interface APIs to convert the HTML to a generic SalesQuote object and then send it to the collaboration.
3. The ATP Access Collaboration then retrieves the available-to-promise date from the SAP connector.
4. The collaboration returns this information to the servlet.
5. The servlet constructs an HTML table containing an ATP date for each requested item and displays this table on the client browser.

Running the sample on a web server

You can load and run the sample Server Access Interface code samples. This section shows you how.

1. Install Server Access Development software and go to `DevelopmentKits\edk\ServerAccessInterfaces\AccessSample` to locate the following:
   - The two java code samples:

     `HtmlDataHandler.java`

     `ATPServlet.java`
   - The HTML Sales Quote Inquiry form: `Example2.html`
   - The sample repository: `SampleRepos.jar`
   - The `collaborations` subdirectory contains the collaborations and classes.
   - The `DLMs` directory contains the native map classes.
2. Load the `SampleRepos.in` with the `repos_copy` utility. For help with loading files into the repository, see the *System Administration Guide*.

3. Compile the servlet file `ATPServlet.java`.
4. Deploy the compiled servlet into your web server. Set the initialization parameter values appropriately for your configuration. Refer to your web server documentation for details about deployment and initialization of servlets.
5. If you have a Solaris or HP-UX operating system, add the `ibmorb.jar`, located in `<ProductDirectory>jre\lib\ext` (IBM Java ORB class files) to the classpath of the client and the web server. Restart the web server if needed. For details, consult your web server documentation.
6. Make `Example2.html` available to your web server.
7. Copy the `AccessSample\collaborations` directory to *`ProductDir`*`\collaborations`.
8. Copy the `AccessSample\DLMs` directory to *`ProductDir`*`\DLMs`.
9. Compile `HtmlDataHandler.java`.
10. Create a `.jar` file and save it as `HtmlDataHandler.jar`, maintaining the output directory structure.
11. Copy the `HtmlDataHandler.jar` file to *`ProductDir`*`\lib`.
12. Modify the `start_server` batch file, adding *`ProductDir`*`\lib\HtmlDataHandler.jar` to the class path.
13. Restart InterChange Server.
14. Make the Interoperable Object Reference (`.ior`) file available to your web server.

    For more information, see "Setting up the run-time environment" on page 12.
15. Launch a browser, and open the `example2.html` page (see Figure 10).
16. Start the test connector, and open and add the "`SampleSapConnector`" profile. Press the Connect button to bring the connector up.
17. Enter data in at least one row of fields (see "Sample HTML data handler" for more on the sample HTML page) and perform a sample Retrieve operation.

The following sections describe the data handler and servlet used in this example:

- "Sample HTML data handler"
- "Sample Java code—ATP servlet" on page 40

Sample HTML data handler

In the sample, the HTML data handler converts the incoming HTML query string into an IBM WebSphere InterChange Server business object. For more on the IBM WebSphere data handler capability, see the *IBM WebSphere Business Integration Data Handler Guide*. These are among the noteworthy features of the data handler component:

- **The datahandler base class** - The sample HTML data handler extends the IBM WebSphere InterChange Server supplied `DataHandler` base class and is automatically loaded at run time when an access request is encountered for a MIME type of "`text/html`".
- **Metadata-based configuration** - Metadata tells the system where to find the data handler and how to call it. Accordingly, multiple data handlers can execute concurrently in a single InterChange Server.
- **Generic transformation** - The HTML data handler is generic in nature and can be re-used without modification to transform any type of HTML query string.

Figure 10 shows the HTML page as it might appear on a client browser. The HTML data handler relies on the properties associated with text boxes on the page.

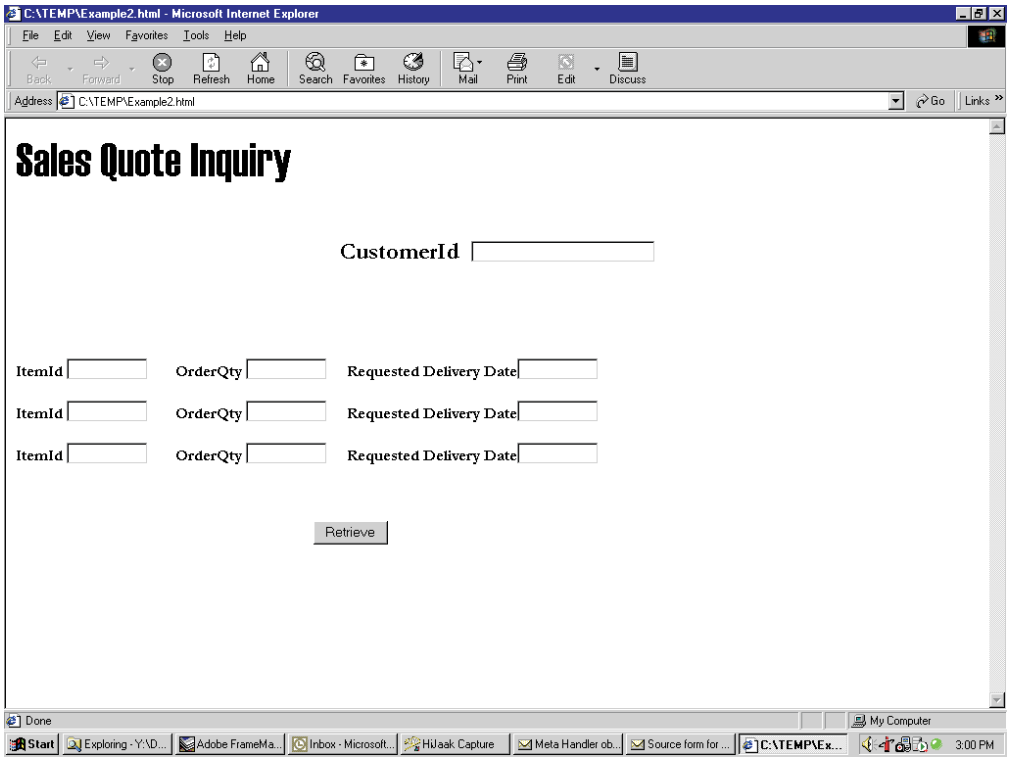

Figure 10. The HTML sales quote inquiry page

In Figure 10, each text box has an HTML property associated with it. The HTML text box property contains IBM WebSphere InterChange Server business object grammar. This grammar enables the HTML data handler to convert the data associated with the property into a business object.

For example, the properties associated with the first item are the following:

- **ItemId** - OrderItems[0].ItemID
- **OrderQty** - OrderItems[0].orderQty
- **Requested delivery date** - OrderItems[0].deliveryDate

As shown in Figure 11, the data handler converts the data on the HTML page to a hierarchical `SalesQuote` business object with child (`orderQty`, `deliveryDate`, and so on) business objects.

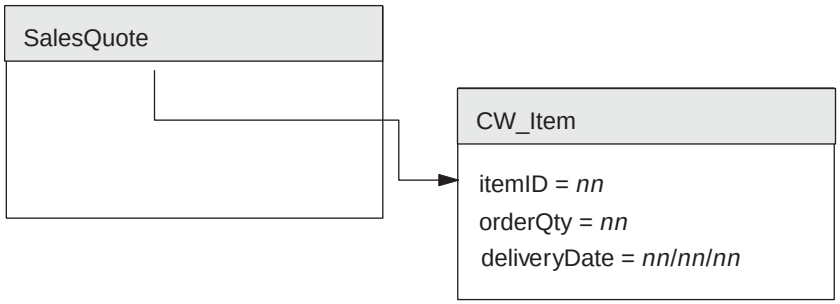

Figure 11. Hierarchical parent-child business objects

Data-handler meta-object

IBM WebSphere business integration software delivers two top-level data-handler meta-objects, one for the server and one for connectors. In addition, there is a child meta-object for each data handler, several of which are delivered with IBM WebSphere business integration software. When you configure your environment, you can:

- Modify the top-level server meta-object attribute name.

  The top-level data-handler meta-object used with data handlers called in the context of the Server Access Interface is `MO_Server_DataHandler.`
- Modify the default values of the child meta-object to reflect the data handler instance you need to create.

You define an attribute in the top-level meta-object for the MIME type and any subtype (`BOPrefix`) you want to support. This attribute represents a child meta-object, which has attributes to provide the class name and configuration properties required by the data handler to do its work.

Figure 12 on page 34 shows the text format of two meta-objects:

- The top-level data-handler meta-object, `MO_Server_DataHandler`.

  Note that this meta-object contains an attribute named for the MIME supported by the HTML data handler (`text.html`). This attribute represents the child data-handler meta-object for the HTML data handler, `MO_DataHandler_DefaultHtmlConfig`.
- The child data-handler meta-object for the HTML data handler, `MO_DataHandler_DefaultHtmlConfig`.

  The child meta-object declares a `ClassName` attribute, whose `DefaultValue` attribute property lists the name of the data handler class (`com.crossworlds.DataHandlers.Html.HtmlDataHandler`) to use to invoke for the HTML data handler.

```
[BusinessObjectDefinition]
Name = MO_Server_DataHandler
Version = 1.0.0

     [Attribute]
     Name = text.html
     Type = MO_DataHandler_DefaultHtmlConfig
     ContainedObjectVersion = 1.0.0
     Relationship = Containment
     Cardinality = 1
     MaxLength = 1
     IsKey = true
     IsForeignKey = false
     IsRequired = false
     IsRequiredServerBound = false
     [End]
     [Attribute]
     Name = ObjectEventId
     Type = String
     MaxLength = 255
     IsKey = false
     IsForeignKey = false
     IsRequired = false
     IsRequiredServerBound = false
     [End]

     [Verb]
     Name = Create
     [End]

     [Verb]
     Name = Delete
     [End]

     [Verb]
     Name = Retrieve
     [End]

     [Verb]
     Name = Update
     [End]
[End]
```

Figure 12. Text Format of HTML meta-objects (Part 1 of 2)

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<xsd:schema elementFormDefault="qualified"
   targetNamespace="http://www.ibm.com/websphere"
   xmlns:bx="http://www.ibm.com/websphere"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:annotation><xsd:documentation>

Tue Mar 11 14:25:46 PST 2003

</xsd:documentation>

</xsd:annotation>

<xsd:element name="TestChildBO">d
<xsd:annotation>
<xsd:appinfo>
<bx:boDefinition version="3.0.0" />
</xsd:appinfo>
</xsd:annotation>
<xsd:complexType><xsd:sequence>
<xsd:element name="FirstName" minOccurs="0">
<xsd:annotation>
<xsd:appinfo>
<bx:boAttribute>
<bx:attributeInfo isForeignKey="false" isKey="false" />
</bx:boAttribute>
</xsd:appinfo>
</xsd:annotation>
<xsd:simpleType>
<xsd:restriction base="xsd:string">
<xsd:maxLength value="255" />
</xsd:restriction>
</xsd:simpleType>
</xsd:element>
<xsd:element name="LastName" minOccurs="1">
<xsd:annotation>
<xsd:appinfo><bx:boAttribute>
<bx:attributeInfo isForeignKey="false" isKey="true" />
</bx:boAttribute>
|</xsd:appinfo>
</xsd:annotation>
<xsd:simpleType>
<xsd:restriction base="xsd:string">
<xsd:maxLength value="255" />
</xsd:restriction>
</xsd:simpleType>
</xsd:element>
<xsd:element name="ObjectEventId" type="xsd:string" minOccurs="0" />
</xsd:sequence>
<xsd:attribute name="version" type="xsd:token" default="0.0.0" />
<xsd:attribute name="delta" type="xsd:boolean" default="false" />
<xsd:attribute name="verb" use="required"><xsd:simpleType>
<xsd:restriction base="xsd:NMTOKEN">
<xsd:enumeration value="Create" />
<xsd:enumeration value="Delete" />
<xsd:enumeration value="Retrieve" />
<xsd:enumeration value="Update" />
</xsd:restriction>
</xsd:simpleType>
</xsd:attribute>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

Figure 12. Text Format of HTML meta-objects (Part 2 of 2)

Sample code for HTML data handler

Here is the HTML data handler Java code sample.

```
/**
 * @(#) HtmlDataHandler.java
 *
 * Copyright (c) 1997-2000 CrossWorlds Software, Inc.
 * All rights reserved.
 *
 * This software is the confidential and proprietary information of IBM, Inc.
 * You shall not disclose such Confidential information and shall
 * use it only in accordance with the terms of the license agreement you entered into
 * with CrossWorlds Software.
*/
import com.crossworlds.DataHandlers.*;
import com.crossworlds.DataHandlers.Exceptions.*;
import AppSide_Connector.JavaConnectorUtil;

import CxCommon.BusinessObjectInterface;

// java classes
import java.util.*;
import java.io.*;
/**
 ** This is a html data handler which converts a html query
 ** string to a Crossworlds Business object. This example is
 ** assumes the incoming html query is structured in a specific
 ** format as explained in the program below. See the comments
 ** associated with the method parse() in this class.
 */
public class HtmlDataHandler extends DataHandler
{

     /*
      **  A utility method to convert a HTML query string into a crossworlds BO.
      **  See comments associated with the parse() method for a detailed explanation
      **  @param String serializeddata
      **  @param Object the incoming mime type
      */
     public BusinessObjectInterface getBO(String serializedData,
       Object config)
          throws Exception
     {

          HashMap nameValuePairs = parse((String) serializedData);

          /*
           ** Get the BO to be created from the hidden tag BusObjName
           */
          String boName = (String) nameValuePairs.get("BusObjName");
          if (boName == null)
               throw new Exception("Unable to find business object name in "
                    + "serialized business object");

          BusinessObjectInterface bo = JavaConnectorUtil.createBusinessObject(boName);
          String verb = (String) nameValuePairs.get("Verb");
          if (verb == null)
               throw new Exception("Unable to find verb in serialized business object");

          bo.setVerb(verb);

          /*
           ** Get the elements from the HashMap and set it into the BO
           */
          setValues(bo, nameValuePairs);
          return bo;
     }
```

```
/*
**  Parse an HTML query string looking for tokens of the form &name=value.
**  The format of the incoming query string must conform to the &name=value
**  format as well as the following semantics:
**        if name does not contain syntax of the form name[X].attribute it is
**        assumed to be the name of an attribute in the parent object otherwise
**        the expression will be used AS IS to set the value of a child object
**        and attribute.
**
**  For example, the following query string can be successfully parsed by
**  this method:
**
** CustomerID=&items[0].itemID=44&items[0].orderQty=25&items[0].
**          deliveryDate1=12/12/00
**  &items[1].itemID=67&items[1].orderQty=2&items[1].
** deliveryDate=12/12/00&Verb=Retrieve&
**  BusObjName=SalesQuote&SubObjName=CwItem
**
**
**  @param String query sent from the webserver to be parsed
**  @return HashMap a hash map containing the name value pairs
*/

private HashMap parse(String queryString)
{
     HashMap nameValuePairs = new HashMap();
     String content = queryString.replace('+', ' ');
     StringTokenizer st = new StringTokenizer(content,"&");

     while (st.hasMoreTokens())
     {
          String token = st.nextToken();
          int i = token.indexOf("=");
          String name = token.substring(0, i);
          String value = token.substring(i+1);

          /*
           **  HTTP will encode certain ASCII values as their hex equivalents.
           ** Convert any of these encodings back to ASCII for both the name
           **  and the value strings (i.e. right hand side of = and left hand
           **  side of =)
           */
          name = replaceHexEncodedWithAscii(name);
          value = replaceHexEncodedWithAscii(value);

          /*
           **  Store these value in the hashmap so that our caller can look
           **  them up.
           */
          nameValuePairs.put(name, value);
     }

     return(nameValuePairs);
}

/*
 *  Given a Hashmap of name/value pairs, enumerate through the business
 *  object and set each attribute in the BO with the corresponding
 *  value from the Hashtable
 *  @param IBusinessObject target of the set
 *  @param Hashmap contains the name/value pairs
 */
private void setValues(BusinessObjectInterface bo, HashMap nameValuePairs)
     throws Exception
{
     String SubObjName = null;
```

```
        Iterator aIterator = nameValuePairs.keySet().iterator();
        // Save the SubObject name so we need to save it
        while (aIterator.hasNext())
        {
             String name = (String) aIterator.next();
             /*
              **  Ignore any hidden keywords that we parsed out of the HTML and
              **  stored in the hash map
              */

             if (name.equalsIgnoreCase("BusObjName")  ||
                       name.equalsIgnoreCase("Verb") ||
                       name.equalsIgnoreCase("SubObjName") ||
                       name.equalsIgnoreCase("ContainerAttrName"))
             {
                  System.out.println("Skipping Item : " + name);
                  continue;
             }

             /*
              **  All subobjects have a grammar in the form of object[X].attribute
              **  where X is the index of the contained subobject.  Therefore, if
              **  the name does not have this embedded string, it's an attribute
              **  of the parent object
              */
             if (name.indexOf("[") == -1)
              bo.setAttrValue(name, (String) nameValuePairs.get(name));
             else
              bo.setAttributeWithCreate(name, (String) nameValuePairs.get(name));
        }
   }

   /*
    *  Replace any hex encoded bytes with the ASCII char equivalent and return
    *  the new string to the caller.
    *  @param name The string to convert.
    */
   private String replaceHexEncodedWithAscii(String name)
   {
        int nameLength = name.length();

        /*
         **  Replace any hex values (HTTP may send over a hex value int
         **  the form of %XX for certain characters) with their
         **  corresponding char equivalents
         */
        StringBuffer nameBuffer = new StringBuffer();
        for (int i = 0; i < nameLength; ++i)
        {
             char c = name.charAt(i);
             switch (c)
             {
                  case '%':
                       byte[] b = { Byte.parseByte(name.substring(i+1, i+3),
                            16) };
                       nameBuffer.append(new String(b));
                       i += 2;
                       break;
                  default:
                       nameBuffer.append(c);
             }
        }
        return(nameBuffer.toString());
   }

   /**
    ** Implementation of abstract methods in the Data Handler class
```

```
     **  @param BusinessObjectInterface the actual business object
     **  @param Object config
     **  @return String string representation of the BO
     */

    public String getStringFromBO(BusinessObjectInterface theObj, Object config)
         throws Exception
    {
         throw new Exception("Not implemented");
    }

    /**
     ** Implementation of abstract methods in the Data Handler class
     *  @param Reader actual data
     *  @param BusinessObjectInterface the actual business object
     *  @param Object config
     */
    public void getBO(Reader serializedData, BusinessObjectInterface theObj,
         Object config)
         throws Exception
    {
         throw new Exception("Not Implemented");
    }

    /**
     ** Implementation of abstract methods in the Data Handler class
     *  @param String actual data
     *  @param BusinessObjectInterface the actual business object
     *  @param Object config
     */
    public void getBO(String serializedData, BusinessObjectInterface theObj,
         Object config)
         throws Exception
    {
         throw new Exception("Not Implemented");
  }

    /**
     ** Implementation of abstract methods in the Data Handler class
     *  @param BusinessObjectInterface the actual business object
     *  @return InputStream a handle to the stream
     */
    public InputStream getStreamFromBO(BusinessObjectInterface theObj,
         Object config)
         throws Exception
    {
         throw new Exception("Not Implemented");
    }

    /**
     ** Implementation of abstract methods in the Data Handler class
     *  @param Reader actual data
     *  @param BusinessObjectInterface the actual business object
     *  @return BusinessObjectInterface the translated BO
     */
    public BusinessObjectInterface getBO(Reader serializedData, Object config)
         throws Exception
    {
         throw new Exception("Not Implemented");
    }
}
```

Sample Java code—ATP servlet

Here is the sample ATP servlet described in "The scenario" on page 29.

```
/**
 * @(#) ATPServlet.java
 *
 * Copyright (c) 1997-2000 CrossWorlds Software, Inc.
 * All rights reserved.
 *
 * This software is the confidential and proprietary information of
 * IBM. You shall not disclose such Confidential information and shall
 * use it only in accordance with the terms of the license agreement you
 * entered into with IBM Software.
*/
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
import java.util.*;
import java.text.*;
import IdlAccessInterfaces.*;
import CxCommon.BusinessObject;

/**
* Available To Promise Servlet example
*/
public class ATPServlet extends HttpServlet
{

     // Defines for some statics
     public static String DEFAULT_SERVER = "CrossWorlds";
     public static String DEFAULT_IOR = "CrossWorlds.ior";
     public static String DEFAULT_USER = "admin";
     public static String DEFAULT_PASSWD = "null";
     // User name to login into the IC Server
     private String userName = DEFAULT_USER;
     // Password
     private String passWord = DEFAULT_PASSWD;
     // ServerName
     private String serverName = DEFAULT_SERVER;
     // IOR File
     private String iorFile = DEFAULT_IOR;
     // AccessSession
     private IInterchangeAccessSession accessSession = null;
     // AccessEngine
     private IAccessEngine accessEngine = null;

     // Servlet Context for getting config information
     private ServletContext ctx;
     // A formatter to print the price with precision.
     private static DecimalFormat formatter;
     // MIME type
     private String mimeType = "text/html";

     /**
     * The init method. This method is used by the web server
     * when the Servlet is loaded for the first time.
     * @param ServletConfig Configuration information
     * associated with the servlet.
     * @exception ServletException is thrown when the
     * servlet cannot be initialized
     */
     public void init(ServletConfig aConfig)throws ServletException
     {
        super.init(aConfig);
        // Formatter for printing prices in the correct format
        formatter = new DecimalFormat();
```

```
      formatter.setDecimalSeparatorAlwaysShown(true);

      // Read up the initial parameters so we can connect to
      // the right ICS server
      String configuredServer = null;
      String configurediorFile = null;
      String configuredUser = null;
      String configuredpassWord = null;
      configuredServer = aConfig.getInitParameter("ICSNAME");
      if ( configuredServer != null)
             {
             this.serverName = configuredServer;
             }
      else
             {
             this.log(
                "No Interchange Server configured, using
                  default of CROSSWORLDS");
             }
      configurediorFile = aConfig.getInitParameter("IORFILE");
      if (iorFile != null)
             {
             this.iorFile = configurediorFile;
             }
      else
             {
             this.log(
               "IOR file not defined, will use CrossWorlds.ior
                from home directory");
             }

      try
             {
             initAccessSession();
             }
      catch(Exception e)
             {
             this.log("Encountered Initialization error", e);
             throw new ServletException(e.toString());
             }
  }
  /**
  * Cleanup method called when the servlet is unloaded from the Web Server
  */
  public void destroy()
{
       // Release our session
       if ( ( accessEngine != null) && (accessSession != null))
             {
             accessEngine.IcloseSession(accessSession);
             accessEngine = null;
             accessSession = null;
             }
  }

  /*
  **  Utility method which creates an access session with InterChange Server.
  **  If one has already been established then return that.
  **  @exception Exception when an error occurs while establishing
  **  the connection to InterChange Server.
  */
  private synchronized void initAccessSession() throws Exception
  {
       try
             {
             /*
             **  If the access session has already been established then
```

```
**  see if the session is still valid (i.e. InterChange
**  Server could have been rebooted since the last time
**   we used the session).
**  If it's not still valid, then open up a new one.
*/
if (accessSession != null)
     {
     try {
          accessSession.IcreateBusinessObject("");
     } catch (ICxAccessError e) {
          /*
          ** Cached session is still valid. We expect
              to get this exception
          */
               return;
     }
     // Catch Corba SystemException
     catch (org.omg.CORBA.SystemException se) {
          /*
           **  The session is invalid.
           **  Open a new one below
           */
          this.log("Re-establishing sessions to ICS");
     }
     }
/**
* Add the relevant Visigenic ORB properties to initialize the
* visigenic ORB.
*/
Properties orbProperties = new java.util.Properties();
orbProperties.setProperty("org.omg.CORBA.ORBClass",
     "com.ibm.CORBA.iiop.ORB");
orbProperties.setProperty("org.omg.CORBA.ORBSingletonClass",
     "com.ibm.rmi.corba.ORBSingleton");
org.omg.CORBA.ORB  orb =
     org.omg.CORBA.ORB.init((String[])null, orbProperties);
/*
**  Use the file that contains the Internet Inter-Orb Reference.
**  This object reference will be a serialized CORBA object
**  reference to the running Interchange Server that
**  we wish to talk to.
*/
LineNumberReader input =
     new LineNumberReader(new FileReader(iorFile));
/*
**  Create a memory resident CORBA object reference from the IOR
**  in the file
*/
org.omg.CORBA.Object object = orb.string_to_object
   (input.readLine());

/*
**  Now get create a real session with the running object
*/
accessEngine = IAccessEngineHelper.narrow(object);
if (accessEngine == null)
     throw new Exception("Unable to communicate with server
       " + serverName + " using IOR from " + iorFile);

/*
**  Now that we have an object reference to a running
**  server, we must authenticate ourselves before we
**  can get a session that is useful.
*/
accessSession = accessEngine.IgetInterchangeAccessSession(
     userName,
     passWord);
```

```
            if (accessSession == null)
                 throw new Exception("Invalid user name and password");

            }
       catch (Exception e)
            {
            this.log("Encountered orb Initialization error" , e);
            if (e instanceof org.omg.CORBA.SystemException)
                 throw new Exception(e.toString());
            else
                 throw e;
            }
}

/**
* Get method called by the WebServer whenever a GET action
* is requested by an HTML page.
* @param HttpServletRequest handle to the http request
* object@param HttpServletResponse handle to the http response
* object @exception ServletException is thrown when the servlet
* encounters an error @exception is thrown when the
* Webserver cannot communicate to the calling
* html page
*/
public void doGet(HttpServletRequest req, HttpServletResponse res)
     throws ServletException, IOException
{
     // String serializedHTMLQuote = null;

     // A BusinessObject to hold our incoming BO from the
     // requesting HTML page
     IBusinessObject aBO = null;

     // A BusinessObject to hold our resultant BO from the
     // result of the Collaboration execution

     IBusinessObject returnedQuoteBusObj = null;

     /*
     **  Make sure we have a valid access session with the interchange
     **  server first
     */
     try
          {
           initAccessSession();
          }
     catch(Exception e)
          {
           throw new ServletException
              ("InitAccessSession Failed " + e.toString());
          }

     // Create a BO from the data provided by the HTML page
     try {
          aBO =
               accessSession.IcreateBusinessObjectFrom
                 (req.getQueryString(),
                   mimeType);
     } catch (ICxAccessError e) {
           throw new ServletException
             (" Creating Business Object Failed : " +
               e.IerrorMessage);
     }

     if (aBO == null)
          {
           throw new ServletException("Attempting to use Null Bo ");
```

```
     }
/*
**  Execute the collaboration.  We'll get back a
**  CrossWorlds business object that contains an ATP
**  date for each item.
*/
try
     {
     returnedQuoteBusObj = accessSession.IexecuteCollaboration(
          "ATPExample","From", aBO);
     }
catch(IExecuteCollaborationError ae)
     {
     String error = "Collaboration Error :
          " + ae.IerrorMessage
                         + ae.status;
     this.log("Collaboration Error", ae);
     throw new ServletException(error);
     }
/*
**  Now create a table to send back that has:
**    ItemNumber    Quantity    Price
*/
res.setContentType(mimeType);
PrintWriter out = res.getWriter();
out.println("<body>");
out.println("<TABLE BORDER=\"1\">");
out.println("<caption align=\"center\" > " +
     "<font face=e=\"Haettenschweiler\" size=\"7\">" +
     "Sales Quote Response</caption>");

out.println("<TR> <TH>Item ID" +
     "<TH> Item Description"    +
     "<TH> Quantity " +
     "<TH> Item Price" +
     "<TH> Available Date " +
     "<TH> Total Price " +
     "</TH> </TR>");
IBusinessObjectArray itemContainer = null;
try {
     itemContainer =
          returnedQuoteBusObj.
           IgetBusinessObjectArrayAttribute
            ("OrderItems");
} catch (IInvalidAttributeTypeException e) {
     throw new ServletException(e.IerrorMessage);
} catch (IInvalidAttributeNameException e) {
     throw new ServletException(e.IerrorMessage);
} catch (IAttributeBlankException e) {
     throw new ServletException(e.IerrorMessage);
} catch (IAttributeNotSetException e) {
     throw new ServletException(e.IerrorMessage);
}

// A subobject to hold each individual Item
IBusinessObject item = null;

int size = itemContainer.IgetSize();
// Loop thru the array and print each item
// separately
String attr = null;
int itemQuantity = 0;
double itemPrice = 0;
//Loop thru the array of returned items
for (int i = 0;i < size; i++)
     {
     try
```

```
{
// Get the item BusinessObject at the
    current indexitem =
    itemContainer.IgetBusinessObjectAtIndex(i);
if (item != null)
     {
     // Build a html table row beginning with ITemID
     // attribute
     try {
          attr = item.IgetStringAttribute("ItemID");
          out.print("<TR> <TD> " +
               attr +
               "</TD>" + "<TD>");
          // We have printed the value,
          // set it to null again
          attr = null;
     } catch (IAttributeNotSetException e) {
          attr = "N/A";
          out.print("<TR> <TD> ");
          out.print(attr + "</TD>" + "<TD>");
     } catch (IInvalidAttributeNameException e) {
               attr = "N/A";
               out.print("<TR> <TD> ");
               out.print(attr + "</TD>" + "<TD>");
     } catch (IInvalidAttributeTypeException e) {
          attr = "N/A";
          out.print("<TR> <TD> ");
          out.print(attr + "</TD>" + "<TD>");
     }
     // Get the ItemType attribute
     try {
          attr = item.IgetStringAttribute
            ("itemType");
          out.print(attr + "</TD>" + "<TD>");
          // We have printed the value,
          // set it to null again
          attr = null;
     } catch (IAttributeNotSetException e) {
          attr = "N/A";
          out.print(attr + "</TD>" + "<TD>");
     } catch (IInvalidAttributeNameException e) {
          attr = "N/A";
          out.print(attr + "</TD>" + "<TD>");
     } catch (IInvalidAttributeTypeException e) {
          attr = "N/A";
          out.print(attr + "</TD>" + "<TD>");
     }
     // Get the orderQty Attribute
     try {
          attr = item.IgetStringAttribute
             ("orderQty");
          try {
               itemQuantity = Integer.parseInt(attr);
          } catch (NumberFormatException e) {
               itemQuantity = -1;
          }
          out.print(attr + "</TD>" + "<TD>");
          // We have printed the value,
          // set it to null again
          attr = null;
     } catch (IAttributeNotSetException e) {
          attr = "N/A";
          itemQuantity = -1;
          out.print(attr + "</TD>" + "<TD>");
     } catch (IInvalidAttributeNameException e) {
          attr = "N/A";
          out.print(attr + "</TD>" + "<TD>");
```

```
} catch (IInvalidAttributeTypeException e) {
     attr = "N/A";
     out.print(attr + "</TD>" + "<TD>");
}
// Get the ItemPrice attribute
try {
     attr = item.IgetStringAttribute("itemPrice");
     int indexOfDollar = attr.indexOf("$");
     String priceToParse = null;
     // Locate if we have "$" in the value
     if (indexOfDollar == -1)
          priceToParse = attr;
     else
          priceToParse = attr.substring
              (indexOfDollar + 1);
     // Format the price so it looks like $NNNN.NN
     try {
          itemPrice = Double.parseDouble
          (priceToParse);
     } catch (NumberFormatException e) {
          itemPrice = -1;
     }
     out.print(attr + "</TD>" + "<TD>");
     // We have printed the value,
         set it to null again
     attr = null;
} catch (IAttributeNotSetException e) {
     attr = "N/A";
     itemPrice = -1;
     out.print(attr + "</TD>" + "<TD>");
} catch (IInvalidAttributeNameException e) {
     attr = "N/A";
     out.print(attr + "</TD>" + "<TD>");
} catch (IInvalidAttributeTypeException e) {
     attr = "N/A";
     out.print(attr + "</TD>" + "<TD>");
}
// Get the ATPDate and print it
try {
     attr = item.IgetStringAttribute("ATPDate");
     out.print(attr + "</TD>" + "<TD>");
} catch (IAttributeNotSetException e) {
     attr = "N/A";
     out.print(attr + "</TD>" + "<TD>");
} catch (IInvalidAttributeNameException e) {
     attr = "N/A";
     out.print(attr + "</TD>" + "<TD>");
} catch (IInvalidAttributeTypeException e) {
     attr = "N/A";
     out.print(attr + "</TD>" + "<TD>");
}
/*
**  Now print the total price for the item.
**  If we don't have sufficient information then
**  print N/A
*/
if ((itemPrice == -1) || (itemQuantity == -1))
     {
     out.println(attr + "</TD>" + "<TD>");
     // We have printed the value,
     // set it to null again
     attr = null;
     }
else
     {
     double totalPrice = itemQuantity
       * itemPrice;
```

```
                                out.println("$" + formatter.format
                                  (totalPrice).trim()
                                     + "</TD>"
                                     + "<TD>");
                                }
                           } // end if (Item != null)
                     } // End try
                     catch (IAttributeBlankException e2) {
                          continue;
                     } catch (IInvalidIndexException e) {
                          throw new ServletException(e.getMessage());
                     }

               }// End for loop
          // Close the HTML table
          out.println("</TABLE>");
          // Finish the page body
          out.println("</body></html>");
          } // end do get
     }
```

Part 3. Server Access Interface API reference

Chapter 6. IAccessEngine interface

The IAccessEngine interface provides methods to open and close an access session with InterChange Server. Table 7 summarizes the methods in the IAccessEngine interface.

Table 7. Member methods of the IAccessEngine interface

Method	Description	Page
IgetInterchangeAccessSession()	Creates an access session to InterChange Server for the access client.	51
IcloseSession()	Closes the access session with InterChange Server.	52

IgetInterchangeAccessSession()

Creates an access session to InterChange Server for the access client.

Syntax

```
IInterchangeAccessSession IgetInterchangeAccessSession(
     string userName, string password);
```

Parameters

userName The name of the IBM WebSphere InterChange Server user.

password The IBM WebSphere InterChange Server password for the user.

Return Values

An IInterchangeAccessSession object for the access session.

Exceptions

ICxAccessError Thrown when an invalid user name or password is encountered.

Notes

The IgetInterchangeAccessSession() method verifies that *userName* and *password* are valid for the ICS instance.

Important: The user name for this method must be admin.

Example

```
// Open the access session
String userName = "admin";
String password = "null";
IInterchangeAccessSession aSession =
     serverAccessEngine.IgetInterchangeAccessSession(
     userName,
     password);
```

IcloseSession()

Closes the access session with InterChange Server.

Syntax

```
void IcloseSession(IInterchangeAccessSession session);
```

Parameters

session The access-session object to close.

Return Values

None.

Example

```
// Close the access session
serverAccessEngine.IcloseSession(aSession);
```

Chapter 7. IInterchangeAccessSession interface

The IInterchangeAccessSession interface provides methods for creating business objects and executing collaborations. Table 8 summarizes the methods of the IInterchangeAccessSession interface.

Table 8. Member methods of the IInterchangeAccessSession interface

Method	Description	Page
IcreateBusinessObject()	Creates a business object from a specified business object definition.	53
IcreateBusinessObjectArray()	Creates the business object array that contains one or more elements, each element having a specified business object as its type.	54
IcreateBusinessObjectFrom()	Converts serialized data in the specified MIME format into an IBM WebSphere InterChange Server business object.	55
IcreateBusinessObjectWithVerb()	Creates a business object with a specified verb.	55
IexecuteCollaboration()	Executes a collaboration, sending in a business object as the triggering access data in the access request.	56
IexecuteCollaborationExtFmt()	Executes a collaboration, sending in serialized data as the triggering access data in the access request.	57
IreleaseBusinessObject()	Releases the resources of a business object.	59
IreleaseBusinessObjectArray()	Releases the resources of a business object array.	59
setLocale(String)	Sets the locale.	60

IcreateBusinessObject()

Creates a business object from a specified business object definition.

Syntax

```
IBusinessObject IcreateBusinessObject(string busObjName);
```

Parameters

busObjName The name of the business object definition to use when creating the business object.

Return Values

An IBusinessObject object to hold the new business object.

Exceptions

ICxAccessError	Thrown when the specified business object definition is *not* present in the IBM WebSphere InterChange Server repository.

Notes

The Server Access Interface creates a business object of type *busObjName* and sends it back to the access client.

Example

The following code fragment creates a business object:

```
// This method creates a business object
// Declare our object
IBusinessObject exampleObj = null;
exampleObj = aSession.IcreateBusinessObject("PayablesNetChange");
```

IcreateBusinessObjectArray()

Creates the business object array that contains one or more elements, each element having a specified business object as its type.

Syntax

```
IBusinessObjectArray IcreateBusinessObjectArray(string busObjName);
```

Parameters

busObjName	The name of the business object definition to use when creating the business objects in the business object array.

Return Values

An `IBusinessObjectArray` object to hold the new business object array.

Exceptions

ICxAccessError	Thrown when the specified business object definition is *not* present in the IBM WebSphere InterChange Server repository.

Notes

The Server Access Methods creates a business object array and sends it back to the access client. The `IcreateBusinessObjectArray()` method returns an `IBusinessObjectArray` object. Other methods in the `IBusinessObjectArray` interface allow you to manipulate the business object array.

Example

The following example creates a business object array:

```
// Declare the array
IBusinessObjectArray exampleObjArray = null;
// Create the business object array that holds "CustomerAcct"
// business objects
exampleObjArray =
     accessSession.IcreateBusinessObjectArray("CustomerAcct");
```

IcreateBusinessObjectFrom()

Converts serialized data in the specified MIME format into an IBM WebSphere InterChange Server business object.

Syntax

```
IBusinessObject IcreateBusinessObjectFrom(string serializedData,
     string mimeType);
```

Parameters

serializedData	The incoming serialized data.
mimeType	The MIME type of the *serializedData* data.

Return Values

An `IBusinessObject` object to hold the business object that the data handler creates from the *serializedData* data.

Exceptions

ICxAccessError	Thrown when the data cannot be converted into a business object or if the data handler cannot be accessed.

Notes

The `IcreateBusinessObjectFrom()` method sends the *serializedData* data in its specified *mimeType* MIME type to InterChange Server. The Server Access Interface within ICS invokes the necessary data handler to convert the specified MIME type into an IBM WebSphere InterChange Server business object, which is compatible with the IBM WebSphere InterChange Server environment. The *serializedData* data must specify the name of the business object definition to use when creating the business object. The data handler parses and converts the data into a business object, returning it to the Server Access Interface within ICS, which in turn returns it to the access client. The external format of the serialized data must be of a type that a data handler (IBM WebSphere InterChange Server-delivered or a custom data handler you have written) supports. For more information, see the *Data Handler Guide*.

Example

```
// Declare the object
String custData = "exampleXmlData";
String mimeType = "text/Xml";
IBusinessObject exampleObj = null;
// This method creates the business object from data in XML format
exampleObj =
     accessSession.IcreateBusinessObjectFrom(custData, mimeType);
```

IcreateBusinessObjectWithVerb()

Creates a business object with a specified verb.

Syntax

```
IBusinessObject IcreateBusinessObjectWithVerb(string busObjName,
     string verb);
```

Parameters

busObjName The name of the business object definition to use when creating the business object.

verb The verb for the new business object.

Return Values

An `IBusinessObject` object that holds the new business object with the specified *`verb`* value.

Exceptions

`ICxAccessError` Thrown when the specified business object definition is *not* present in the IBM WebSphere InterChange Server repository or if the *`verb`* passed is invalid for the business object definition.

Notes

The Server Access Interface creates a business object of type `busObjName` and initializes it with the *`verb`* verb. It then sends this business object back to the access client. Only verbs supported in the business object definition are valid.

Example

```
// Create the business object
IBusinessObject exampleobj = null
exampleObj =
     accessSession.IcreateBusinessObjectWithVerb("AcctsRecCurrent",
     "Retrieve");
```

IexecuteCollaboration()

Executes a collaboration, sending in a business object as the triggering access data in the access request.

Syntax

```
IBusinessObject IexecuteCollaboration
     (string collabName, string portName, IBusinessObject busObj);
```

Parameters

collabName The name of the collaboration to execute.

portName The name of the external collaboration port to which the access client is bound.

busObj The generic business object that contains the triggering access data for the collaboration.

Return Values

An `IBusinessObject` object that contains the business object that the collaboration returns.

Exceptions

IExecuteCollaborationError
: Thrown when the collaboration is not active or the maps have failed. This exception contains a status value set to one of the following constants to indicate the details of the call when the exception occurred. For more information on how to access this status, see "IExecuteCollaborationError" on page 92.

Constant Name	Description
UNKNOWNSTATUS	The status of the call to the IexecuteCollaboration() method is unknown.
FAILEDTOREACHCOLLABORATION	The access request did not reach the collaboration.
FAILEDINEXECUTIONOFCOLLABORATION	The access request failed while executing the collaboration.
FAILEDINRETURNTOCLIENT	The collaboration executed but an error occurred while delivering the response to the access client.

Notes

The IexecuteCollaboration() method requests execution of the *collabName* collaboration. To initiate the collaboration, Server Access Interface sends the triggering access data in the *busObj* business object to the *portName* port of the *collabName* collaboration. This port must be configured as external so that is supports call-triggered flow.

Note: The collaboration, port, and business object must be configured and mapped for call-triggered flow and manipulation.

Example

```
String portName = "From";
IBusinessObject srcBO =
     accessSession.IcreateBusinessObject ("payableNetChange");

// set srcBO attributes, verb, or both
...
// Execute the collaboration
IBusinessObject resultantBO = null;
resultantBO = accessSession.IexecuteCollaboration(
     "getCustAcctPayable",
     portName,
     srcBO);
```

IexecuteCollaborationExtFmt()

Executes a collaboration, sending in serialized data as the triggering access data in the access request.

Syntax

```
string IexecuteCollaborationExtFmt(string collabName, string portName,
          string serializedData, string mimeType, string verb);
```

Parameters

collabName	The name of the collaboration to execute.
portName	The name of the external collaboration port to which the access client is bound.
serializedData	The serialized data that represents the triggering access data.
mimeType	The external format (as a MIME type) of the serialized data.
verb	The value for the business object's verb.

Return Values

A `string` that contains the serialized version of the business object that the collaboration returns. This string is in the *`mimeType`* external format.

Exceptions

`IExecuteCollaborationError`

Thrown when the collaboration is not active or the maps have failed. This exception contains a status value set to one of the following constants to indicate the details of the call when the exception occurred. For more information on how to access this status, see "IExecuteCollaborationError" on page 92.

Constant Name	Description
`UNKNOWNSTATUS`	The status of the call to the `IexecuteCollaborationExtFmt()` method is unknown.
`FAILEDTOREACHCOLLABORATION`	The access request did not reach the collaboration.
`FAILEDINEXECUTIONOFCOLLABORATION`	The access request failed while executing the collaboration.
`FAILEDINRETURNTOCLIENT`	The collaboration executed but an error occurred while delivering the response to the access client.

Notes

The `IexecuteCollaborationExtFmt()` method performs the same basic task as `IexecuteCollaboration()`: it requests execution of the *`collabName`* collaboration. The main difference is that this method allows you to perform the following tasks with a single call:

- Convert the *`serializedData`* data to a business object, using the data handler appropriate for the data's *`mimeType`* MIME type. This business object represents the triggering access data for the collaboration.
- Set the business object's verb to the specified `verb` value.
- Send the business object to the *`portName`* port of the collaboration to initiate execution of the collaboration. This port must be configured as external so that is supports call-triggered flow.

Note: No CORBA objects are passing in or out of this method.

The collaboration and port must be configured and mapped for call-triggered flow and manipulation.

The *mimeType* parameter specifies the external format of the serialized data for the business object. The Server Access Interface uses this MIME type to determine which data handler it calls to parse and convert the data into an IBM WebSphere InterChange Server business object. The external format must be of a type that a data handler (IBM WebSphere InterChange Server-delivered or a custom data handler you have written) supports. For more on data handling, see the *Data Handler Guide*.

Example

```
String portName = "From";
// Execute the collaboration
IBusinessObject resultantBO = null;
resultantBO = accessSession.IexecuteCollaborationExtFmt(
     "getCustAcctPayable",
     portName,
     serializedXMLData,
     "text/xml",
     "Create");
```

IreleaseBusinessObject()

Releases the resources of a business object.

Syntax

```
void IreleaseBusinessObject(IBusinessObject releaseObject);
```

Parameters

releaseObject	The business object whose resources are released.

Return Values

None.

Notes

When the access client is finished using a business object, it should the `IreleaseBusinessObject()` method to free the `IBusinessObject` object in InterChange Server memory.

Example

```
// Create the business object
IBusinessObj anObject = null;
accessSession.IcreateBusinessObjectWithVerb("AcctsRecCurrent",
     "Retrieve");
// Release the object
accessSession.IreleaseBusinessObject(anObject);
```

IreleaseBusinessObjectArray()

Releases the resources of a business object array.

Syntax

```
void IreleaseBusinessObjectArray(IBusinessObjectArray releaseObject);
```

Parameters

releaseObject	The business object array whose resources are released.

Return Values

None.

Notes

When the access client is finished using a business object array, it should the `IreleaseBusinessObjectArray()` method to free the `IBusinessObjectArray` object in InterChange Server memory.

Example

```
// Create the array
IBusinessObjectArray exampleObjArray = null;
exampleObjArray =
     accessSession.IcreateBusinessObjectArray("CustomerAcct");
// Release the object array
accessSession.IreleaseBusinessObjectArray(exampleObjArray);
```

---

setLocale(String)

Sets the locale of the access interface session object.

Syntax

```
public String setLocale(String);
```

Parameters

A string designating the locale, in this format:

ll_TT

where *ll* is a two-character language code (usually in lower case) and *TT* is a an optional two-letter country and territory code (usually in upper case). For example, the following strings are valid locales:

```
en
de_DE
```

Notes

The `setLocale()` method sets the locale for the access interface session object. The locale defines cultural conventions for data according to language and country (or territory).

By default, the locale used in the beginning of a session object is the same as the locale used by the ICS. When you use a call on the setLocale() method to change to a new locale, calls on all subsequent methods in the session object will use the new locale.

Chapter 8. IBusinessObject interface

The `IBusinessObject` interface provides methods that operate on objects of the type `BusinesssObject`. These represent IBM WebSphere business integration system business objects that are defined in the IBM WebSphere repository. Table 9 summarizes the methods in the `IBusinessObject` interface.

Table 9. Member methods of the IBusinessObject interface

Method	Description	Page
Iduplicate()	Creates a clone of the business object.	62
Iequals()	Compares this business object's attribute values with those of the input business object.	63
IequalsKeys()	Compares this business object's key attribute values with those of the input business object.	64
IgetAppSpecificInfo()	Retrieves the application-specific information for the attribute.	64
IgetAttributeCount()	Retrieves the number of attributes in the business object.	65
IgetAttributeName()	Retrieves the attribute name at the specified position in the business object definition.	65
IgetAttributeType()	Retrieves the type of the attribute.	66
IgetAttributeTypeAtIndex()	Retrieves the type of the attribute at the specified position in the business object definition.	66
IgetBooleanAttribute()	Retrieves a `boolean` value of an attribute.	67
IgetBOAppSpecification()	Retrieves the value of an attribute that is a business object array (multiple cardinality).	68
IgetBusinessObjectArrayAttribute()	Retrieves the value of a business object attribute that is a business object array (multiple cardinality).	69
IgetBusinessObjectAttribute()	Retrieves the value of an attribute of single cardinality.	69
IgetDateAttribute()	Retrieves the value of the date attribute.	69
IgetDefaultValue()	Retrieves the default value of the attribute.	70
IgetDoubleAttribute()	Retrieves a `double` value of an attribute.	70
IgetFloatAttribute()	Retrieves a `float` value of an attribute.	71
IgetIntAttribute()	Retrieves an `int` value of an attribute.	72
IgetLongTextAttribute()	Retrieves a `longtext` value of an attribute.	73
IgetName()	Retrieves the name of the business object definition.	73
IgetStringAttribute()	Retrieves a `string` value of an attribute.	73

Table 9. Member methods of the IBusinessObject interface (continued)

Method	Description	Page
IgetVerb()	Retrieves the verb for the business object.	74
IisAttributeMultipleCardinality()	Determines whether the attribute has multiple cardinality.	75
IisBlankValue()	Determines whether the attribute value is a blank value.	75
IisIgnoreValue()	Determines whether the attribute value is "ignore".	76
IisKey()	Determines whether the attribute is a key.	76
IisRequired()	Determines whether the specified attribute is required.	77
Iserialize()	Returns the attribute data in a readable (serialized) format.	77
IsetAttributes()	Sets attributes in a business object from serialized data in a specified MIME type.	78
IsetAttributeToBlank()	Sets the attribute in a business object to a blank value.	78
IsetAttributeToIgnore()	Sets an attribute in a business object to "ignore".	79
IsetBooleanAttribute()	Sets an attribute to a `boolean` value.	79
IsetBusinessObjectArrayAttribute()	Sets the value of an attribute that is a business object array (multiple cardinality).	80
IsetBusinessObjectAttribute()	Sets the value of an attribute of single cardinality.	80
IsetDateAttribute()	Sets an attribute to a date value.	81
IsetDoubleAttribute()	Sets an attribute to a `double` value.	81
IsetFloatAttribute()	Sets an attribute to a `float` value.	82
IsetIntAttribute()	Sets an attribute to an `int` value.	82
IsetLongTextAttribute()	Sets an attribute to a `longtext` value.	83
IsetStringAttribute()	Sets an attribute to a `string` value.	83
IsetVerb()	Sets the verb for the business object.	84
ItoExternalForm()	Serializes the business object data into an external format of the specified MIME type.	84
ItoString()	Serializes the business object data using an IBM WebSphere InterChange Server format.	85

Iduplicate()

Creates a clone of the business object.

Syntax

```
IBusinessObject Iduplicate();
```

Parameters

None.

Return Values

An an `IBusinessObject` object that contains the duplicate business object.

Exceptions

`ICxAccessError`	Thrown when the object cannot be found.

Notes

The `Iduplicate()` method makes a clone of the business object and returns it. You must explicitly assign the return value of this method call to a declared variable of `IBusinessObject` type.

Example

The following example duplicates `sourceCustomer` to create `destCustomer`.

```
IBusinessObject destCustomer = sourceCustomer.Iduplicate();
```

Iequals()

Compares this business object's attribute values with those of the input business object.

Syntax

```
boolean Iequals(IBusinessObject obj2);
```

Parameters

obj2 The business object to compare.

Return Values

Returns `true` if the values of *all* attributes and the verbs are the same; otherwise, returns `false`.

Notes

The `Iequals()` method compares this business object's attribute values with those in the input business object. If the business objects are hierarchical, the comparison includes *all* attributes in the child business objects. The verbs and the attribute values must match.

In the comparison, a null value is considered equivalent to any value to which it is compared and does *not* prevent a return of `true`.

Example

The following example compares the verbs and attributes of `order2` to all attributes of `order1`:

```
boolean isEqual = false;
IBusinessObject order1 =
     accessSession.IcreateBusinessObjectwithVerb("salesorder",
     "create");
IBusinessObject order2 =
     accessSession.IcreateBusinessObjectwithVerb("salesorder",
     "create");
isEqual = order1.Iequals(order2);
if(isEqual)
```

```
     System.out.println("order1 is the same as order2")
else
     System.out.println("order1 is not the same as order2");
```

IequalsKeys()

Compares this business object's key attribute values with those of the input business object.

Syntax

```
boolean IequalsKeys(IBusinessObject obj2);
```

Parameters

obj2 A business object to evaluate for the comparison.

Return Values

Returns `true` if the values of *all* key attributes are the same; otherwise, returns `false`.

Notes

The `IequalsKeys()` method performs a shallow comparison; that is, it does *not* compare the keys in child business objects.

Example

The following example compares key attributes of `order2` with key attributes of `order1`, excluding the attributes of child business objects, if any.

```
boolean keyEqual = false;
IBusinessObject order1 =
     accessSession.IcreateBusinessObjectwithVerb("salesorder",
     "retrieve");
IBusinessObject order2 =
     accessSession.IcreateBusinessObjectwithVerb("salesorder",
     "retrieve");
keyEqual = order1.IequalsKeys(order2);
if(keyEqual)
     System.out.println("order1 is the same as order2")
else
     System.out.println("order1 is not the same as order2");
```

IgetAppSpecificInfo()

Retrieves the application-specific information for the attribute.

Syntax

```
string IgetAppSpecificInfo(string attributeName)
```

Parameters

attributeName The name of the attribute.

Return Values

A `string` that contains the application-specific information associated with the specified attribute.

Exceptions

`IInvalidAttributeNameException`	Thrown when the attribute name is invalid.
`IValueNotSetException`	Thrown when the attribute has no application-specific information.

Notes

The `IgetAppSpecificInfo()` method can return a null.

Example

```
// This method determines the app-specific info of an attribute
String appSpecificInfo;
appSpecificInfo = aBusObj.IgetAppSpecificInfor();
```

IgetAttributeCount()

Retrieves the number of attributes in the business object.

Syntax

```
long IgetAttributeCount();
```

Parameters

None.

Return Values

An integer value to indicate the number of attributes in the current business object.

Example

```
long attributeCount = 0;
attributeCount = aBusObj.IgetAttributeCount();
```

IgetAttributeName()

Retrieves the attribute name at the specified position in the business object definition.

Syntax

```
string IgetAttributeName(long position);
```

Parameters

position	The position of the attribute in a business object definition.

Return Values

A `string` that contains the name of the attribute at the specified position in the business object definition.

Exceptions

`IInvalidIndexException`	Thrown when the position index is invalid.

Example

```
int position = 1;
String attribute name;
attributeName = aBusObj.IgetAttributeName(position);
```

IgetAttributeType()

Retrieves the type of the attribute.

Syntax

```
long IgetAttributeType(string attributeName);
```

Parameters

attributeName The name of the attribute whose type is returned.

Return Values

An integer to indicate the data type of the specified attribute in the business object, as follows:

0 Object

1 boolean

2 int

3 float

4 double

5 string

6 date

7 longtext

Exceptions

IInvalidAttributeNameException
Thrown when the attribute name is invalid.

Example

```
String attributeName = "Name";
long attributeType = 0;
attributeType = aBusObj.IgetAttributeType(attributeName);
```

IgetAttributeTypeAtIndex()

Retrieves the type of the attribute at the specified position in the business object definition.

Syntax

```
long IgetAttributeTypeAtIndex(long position);
```

Parameters

position The position of the attribute in the business object definition.

Return Values

An integer to indicate the data type of the attribute at the specified position in the business object, as follows:

0 `Object`

1 `boolean`

2 `int`

3 `float`

4 `double`

5 `string`

6 `date`

7 `longtext`

Exceptions

`IInvalidIndexException` Thrown when the position index is invalid.

Example

```
int indexPosition = 1;
long attributeType = 0;
attributeType = aBusObj.IgetAttributeTypeAtIndex(indexPosition);
```

IgetBooleanAttribute()

Retrieves a `boolean` value of an attribute.

Syntax

```
boolean IgetBooleanAttribute(string attributeName);
```

Parameters

attributeName The name of the `boolean` attribute whose value is retrieved.

Return Values

The `boolean` value of the attribute.

Exceptions

`IAttributeNotSetException`
Thrown when the attribute value is not set.

`IInvalidAttributeNameException`
Thrown when the attribute name is invalid.

`IInvalidAttributeTypeException`
Thrown when the attribute is not of the `boolean` date type.

`IAttributeBlankException`
Thrown when the attribute has a blank value.

Example

```
// Call the boolean method
String booleanAttribute = "MyBooleanAttribute";
boolean value = exampleBusObj.IgetBooleanAttribute(booleanAttribute);
```

IgetBOAppSpecification()

Retrieves application-specific information.

Syntax

```
public String IgetBOAppSpecificInfo();
```

Parameters

This method has no input parameters.

Return Values

An`IgetBOAppSpecificInfo()`object that contains application specific information for the business application.

Exceptions

`IValueNotSetException`
Thrown when the attribute value is invalid.

IgetBusinessObjectArrayAttribute()

Retrieves the value of an attribute that is a business object array (multiple cardinality).

Syntax

```
IBusinessObjectArray IgetBusinessObjectArrayAttribute(
     string attributeName);
```

Parameters

attributeName
The name of the multiple-cardinality attribute whose value is retrieved.

Return Values

An `IBusinessObjectArray` object that contains the value of the multiple-cardinality attribute.

Exceptions

`IAttributeNotSetException`
Thrown when the attribute value is not set.

`IInvalidAttributeNameException`
Thrown when the attribute name is invalid.

`IInvalidAttributeTypeException`
Thrown when the attribute is not a single-cardinality attribute (it is of some other date type).

`IAttributeBlankException`
Thrown when the attribute has a blank value.

Example

```
// Call the BusinessObjectArray method and get the attribute
String arrayAttribute = "Account";
IBusinessObjectArray aBusObj =
     exampleBusObj.IgetBusinessObjectArrayAttribute(arrayAttribute);
```

IgetBusinessObjectAttribute()

Retrieves the value of an attribute of single cardinality.

Syntax

```
IBusinessObject IgetBusinessObjectAttribute(string attributeName);
```

Parameters

attributeName
The name of the single-cardinality attribute whose value is retrieved.

Return Values

An `IBusinessObject` object that contains the value of the single-cardinality attribute.

Exceptions

`IAttributeNotSetException`
Thrown when the attribute value is not set.

`IInvalidAttributeNameException`
Thrown when the attribute name is invalid.

`IInvalidAttributeTypeException`
Thrown when the attribute is not a single-cardinality attribute (it is of some other date type).

`IAttributeBlankException`
Thrown when the attribute has a blank value.

Example

```
// Call the get business object method and get the attribute
String busObjAttribute = "Customer";
IBusinessObject aBusObj =
     exampleBusObj.IgetBusinessObjectAttribute(busObjAttribute);
```

IgetDateAttribute()

Retrieves the value of the date attribute.

Syntax

```
string IgetDateAttribute(string attributeName);
```

Parameters

attributeName The name of the date attribute whose value is retrieved.

Return Values

A `string` that contains the value of the date attribute.

Exceptions

`IAttributeNotSetException`
Thrown when the attribute value is not set.

`IInvalidAttributeNameException`
Thrown when the attribute name is invalid.

`IInvalidAttributeTypeException`
Thrown when the attribute is not of the date type.

`IAttributeBlankException`
Thrown when the attribute has a blank value.

Example

```
//call the Date method and get the attribute
String dateAttributeName = "DateOfBirth";
String aDate;
aDate = exampleBusObj.IgetDateAttribute(dateAttributeName);
```

IgetDefaultValue()

Retrieves the default value of the attribute.

Syntax

```
string IgetDefaultValue(string attributeName);
```

Parameters

attributeName The name of the attribute whose default value is retrieved.

Return Values

A `string` that contains the default value of the attribute.

Exceptions

`IInvalidAttributeNameException`
Thrown when the attribute name is invalid.

`IValueNotSetException`
Thrown when the attribute has no default value.

Example

```
// Call the default value method
String attributeName = "Name";
String defaultAttributeValue;
defaultAttributeValue =
     exampleBusObj.IgetDefaultValue (attributeName);
```

IgetDoubleAttribute()

Retrieves a `double` value of an attribute.

Syntax

```
double IgetDoubleAttribute(string attributeName);
```

Parameters

attributeName The name of the attribute whose `double` value is retrieved.

Return Values

The `double` value of the attribute.

Exceptions

`IAttributeNotSetException`
Thrown when the attribute value is not set.

`IInvalidAttributeNameException`
Thrown when the attribute name is invalid.

`IInvalidAttributeTypeException`
Thrown when the attribute is not of the `double` type.

`IAttributeBlankException`
Thrown when the attribute has a blank value.

Example

```
// Call the double method and get the attribute
double doubleValue = 0;
String doubleAttributeName = "Average";
doubleValue = exampleBusObj.IgetDoubleAttribute(doubleAttributeName);
```

IgetFloatAttribute()

Retrieves a `float` value of an attribute.

Syntax

```
float IgetFloatAttribute(string attributeName);
```

Parameters

attributeName The name of the attribute whose `float` value is retrieved.

Return Values

The `float` value of the attribute.

Exceptions

`IAttributeNotSetException`
Thrown when the attribute value is not set.

`IInvalidAttributeNameException`
Thrown when the attribute name is invalid.

`IInvalidAttributeTypeException`
Thrown when the attribute is not of the `float` type.

`IAttributeBlankException`
Thrown when the attribute has a blank value.

Example

```
// Call the Float method and get the attribute
float floatValue = 0.0;
String floatAttributeName = "Height";
floatValue = exampleBusObj.IgetFloatAttribute(floatAttributeName);
```

IgetICSVersion()

Retrieves the InterChange framework version number.

Syntax

```
public String IgetICSVersion();
```

Parameters

No input parameters

Return Values

Returns the version number of the InterChange framework.

Exceptions

This method throws no exceptions.

IgetIntAttribute()

Retrieves an `int` value of an attribute.

Syntax

```
long IgetIntAttribute(string attributeName);
```

Parameters

attributeName The name of the attribute whose integer value is retrieved.

Return Values

A `long` value that holds the integer value of the attribute.

Exceptions

`IAttributeNotSetException`
: Thrown when the attribute value is not set.

`IInvalidAttributeNameException`
: Thrown when the attribute name is invalid.

`IInvalidAttributeTypeException`
: Thrown when the attribute is not of the integer type.

`IAttributeBlankException`
: Thrown when the attribute has a blank value.

Example

```
// Call the int method and get the attribute
int intValue = 1;
String intAttributeName = "priority";
intValue = exampleBusObj.IgetIntAttribute(intAttributeName);
```

IgetLongTextAttribute()

Retrieves a `longtext` value of an attribute.

Syntax

```
string IgetLongTextAttribute(string attributeName);
```

Parameters

attributeName The name of the attribute whose `longtext` value is retrieved.

Return Values

The `longtext` value of the attribute as a string.

Exceptions

`IAttributeNotSetException`
Thrown when the attribute value is not set.

`IInvalidAttributeNameException`
Thrown when the attribute name is invalid.

`IInvalidAttributeTypeException`
Thrown when the attribute is not of the `longtext` type.

`IAttributeBlankException`
Thrown when the attribute has a blank value.

Example

```
// Call the LongText method and get the attribute
long longValue = "net30";
String longAttributeName = "Customer";
longValue = exampleBusObj.IgetLongTextAttribute(longAttributeName);
```

IgetName()

Retrieves the name of the business object definition.

Syntax

```
string IgetName();
```

Parameters

None.

Return Values

A `string` that contains the name of the business object definition.

Example

```
// Get the name of the business object definition
String busObjName;
busObjName = exampleBusObj.IgetName();
```

IgetStringAttribute()

Retrieves a `string` value of an attribute.

Syntax

```
string IgetStringAttribute(string attributeName);
```

Parameters

attributeName The name of the attribute whose `string` value is retrieved.

Return Values

A `string` that contains the value of the attribute.

Exceptions

`IAttributeNotSetException`
: Thrown when the attribute value is not set.

`IInvalidAttributeNameException`
: Thrown when the attribute name is invalid.

`IInvalidAttributeTypeException`
: Thrown when the attribute is not of the `string` type.

`IAttributeBlankException`
: Thrown when the attribute has a blank value.

Example

```
// Call the String method and get the attribute
String stringValue = "declined";
String stringAttributeName = "SalesOrder";
stringValue = exampleBusObj.IgetStringAttribute(stringAttributeName);
```

IgetVerb()

Retrieves the verb for the business object.

Syntax

```
string IgetVerb();
```

Parameters

None.

Return Values

A `string` that contains the verb of the business object, which can be `null`.

Exceptions

`IVerbNotSetException`
: Thrown when the verb is not set.

Example

```
// Get the verb of the business object.
String busObjName;
busObjName = exampleBusObj.IgetVerb();
```

IisAttributeMultipleCardinality()

Determines whether the attribute has multiple cardinality.

Syntax

```
boolean IisAttributeMultipleCardinality(string attributeName);
```

Parameters

attributeName The name of the attribute whose cardinality is determined.

Return Values

Returns `true` if the attribute has multiple cardinality; otherwise, it returns `false`.

Exceptions

`IInvalidAttributeNameException`
Thrown when the attribute name is invalid.

Example

```
// Call the multiple cardinality method.
boolean multCard = false;
String busAttribute = "AttributeName";
multCard =
     exampleBusObj.IisAttributeMultipleCardinality(busAttribute);
if (multCard)
     System.out.println ("attribute is multiple cardinality");
else
     System.out.println ("attribute is not multiple cardinality");
```

IisBlankValue()

Determines whether the attribute value is a blank value.

Syntax

```
boolean IisBlankValue(string attributeName);
```

Parameters

attributeName The name of the attribute whose attribute value is tested for a blank value.

Return Values

Returns `true` if the attribute value is a blank value; otherwise, it returns `false`.

Exceptions

`IInvalidAttributeNameException`
Thrown when the attribute name is invalid.

Example

```
// See if attribute is blank
boolean isBlank = false;
String busAttribute = "AttributeName";
isBlank = exampleBusObj.IisBlankValue(busAttribute);
if (isBlank)
     ...
```

IisIgnoreValue()

Determines whether the attribute value is “ignore”.

Syntax

```
boolean IisIgnoreValue(string attributeName);
```

Parameters

attributeName The name of the attribute whose value is tested for “ignore”.

Return Values

Returns `true` if the attribute value is “ignore”; otherwise, it returns `false`.

Exceptions

`IInvalidAttributeNameException`
: Thrown when the attribute name is invalid.

`IValueNotSetException`
: Thrown when the attribute has no default value.

Example

```
// Call the attribute ignore method
boolean isIgnore = false;
String busAttribute = "AttributeName";
isIgnore = exampleBusObj.IisIgnoreValue(busAttribute);
if (isIgnore)
     ...
```

IisKey()

Determines whether the attribute is a key.

Syntax

```
boolean IisKey(string attributeName);
```

Parameters

attributeName The name of the attribute that is checked for a key.

Return Values

The method returns `true` if the attribute is a key, else it returns `false`.

Exceptions

`IInvalidAttributeNameException`
: Thrown when the attribute name is invalid.

Example

```
// See if attribute is key
boolean isKey = false;
String busAttribute = "AttributeName";
isKey = exampleBusObj.IisKey(busAttribute);
if (isKey)
     ...
```

IisRequired()

Determines whether the specified attribute is required.

Syntax

```
boolean IisRequired(string attributeName);
```

Parameters

attributeName The name of the attribute that is checked for whether it is required.

Return Values

Returns `true` if the attribute is required; otherwise, it returns `false`.

Exceptions

`IInvalidAttributeNameException`
Thrown when the attribute name is invalid.

Example

```
// Call the isRequired method
boolean isReq = false;
String busAttribute = "AttributeName";
isReq = exampleBusObj.IisRequired (busAttribute);
if (isReq)
...
```

Iserialize()

Serializes the business object data using the IBM WebSphere InterChange server serialization format.

Syntax

```
string Iserialize();
```

Parameters

None.

Return Values

A `string` that contains the serialized data for the business object.

Example

```
// Call the serialize data method
IBusinessObject srcBO =
     accessSession.IcreateBusinessObject("Customer");
...
String serializedCustomer = scrBO.Iserialize();
```

IsetAttributes()

Sets attributes in a business object from serialized data in a specified MIME type.

Syntax

```
void IsetAttributes(string serializedData, string mimeType);
```

Parameters

serializedData The serialized data in the specified MIME type format.

mimeType The MIME type that identifies the external format of the serialized data.

Return Values

None.

Exceptions

`IMalFormedDataException`
Thrown when the data is not formatted correctly.

Example

```
// Establish data format type
String externalData = "incomingData"
String mimeType = "text/xml";
exampleBusObj.IsetAttributes (externalData, mimeType);
```

IsetAttributeToBlank()

Sets the attribute in a business object to a blank value.

Syntax

```
void IsetAttributeToBlank(string attributeName);
```

Parameters

attributeName The name of the attribute whose value is set to blank.

Return Values

None.

Exceptions

`IInvalidAttributeNameException`
Thrown when the attribute name is invalid.

Example

```
// Call the set-attribute-to-blank method
String attributeName = "checkType";
exampleBusObj.IsetAttributeToBlank(attributeName);
```

IsetAttributeToIgnore()

Sets an attribute in a business object to “ignore”.

Syntax

```
void IsetAttributeToIgnore(string attributeName);
```

Parameters

attributeName The name of the attribute whose value is set to “ignore”.

Return Values

None.

Exceptions

`IInvalidAttributeNameException`
Thrown when the attribute name is invalid.

Example

```
// Set the Default Attribute to a CxIgnore value
String attributeName = "Ignore";
exampleBusObj.IsetAttributeToIgnore(attributeName);
```

IsetBooleanAttribute()

Sets an attribute to a `boolean` value.

Syntax

```
void IsetBooleanAttribute(string attributeName, boolean value);
```

Parameters

attributeName The name of the attribute whose value is set.

value The `boolean` value for the attribute.

Return Values

None.

Exceptions

`IInvalidAttributeTypeException`
Thrown when the attribute is not a `boolean` type.

`IInvalidAttributeNameException`
Thrown when the attribute name is invalid.

Example

```
// Call the Boolean method
String attributeName = "custID";
boolean value = false;
exampleBusObj.IsetBooleanAttribute(attributeName, false);
```

IsetBusinessObjectArrayAttribute()

Sets the value of an attribute that is a business object array (multiple cardinality).

Syntax

```
void IsetBusinessObjectArrayAttribute(string attributeName,
     IBusinessObjectArray value);
```

Parameters

attributeName The name of the multiple-cardinality attribute whose value is set.

value The business object array that is the value for the attribute.

Return Values

None.

Exceptions

`IInvalidAttributeTypeException`
Thrown when the attribute is not a business object array.

`IInvalidAttributeNameException`
Thrown when the attribute name is invalid.

Example

```
// Call the BusinessObjectArray attribute method
String arrayAttribute = "CustomerAddress";
IBusinessObject CustomerAddress =
     accessSession.IcreateBusinessObjectArray ("Address");
IBusinessObject exampleBO =
     accessSession.IcreateBusinessObject ("Customer");
exampleBO.IsetBusinessObjectArrayAttribute(arrayAttribute,
     CustomerAddress);
```

IsetBusinessObjectAttribute()

Sets the value of an attribute of single cardinality.

Syntax

```
void IsetBusinessObjectAttribute(string attributeName,
     IBusinessObject value);
```

Parameters

attributeName The name of the single-cardinality attribute whose value is set.

value The business object that is the value for the attribute.

Return Values

None.

Exceptions

`IInvalidAttributeTypeException`
Thrown when the attribute is not a business object.

`IInvalidAttributeNameException`
Thrown when the attribute name is invalid.

Example

```
// Call the BusinessObject attribute method
String attributeName = "AccountStatus";
String value = "delqnt";
exampleBusObj.IsetBusinessObjectAttribute(attributeName, value);
```

IsetDateAttribute()

Sets an attribute to a date value.

Syntax

```
void IsetDateAttribute(string attributeName, string value);
```

Parameters

attributeName The name of the attribute whose value is set.

value The date value for the attribute, in a string format.

Return Values

None.

Exceptions

`IInvalidAttributeTypeException`
Thrown when the attribute is not a date.

`IInvalidAttributeNameException`
Thrown when the attribute name is invalid.

Example

```
// Call the set Date attribute method
String dateAttribute = "DateofBirth";
String dateValue = "11/18/1966";
exampleBusObj.IsetDateAttribute(dateAttribute, dateValue);
```

IsetDoubleAttribute()

Sets an attribute to a `double` value.

Syntax

```
void IsetDoubleAttribute(string attributeName, double value);
```

Parameters

attributeName The name of the attribute whose value is set.

value The `double` value for the attribute.

Return Values

None.

Exceptions

`IInvalidAttributeTypeException`
Thrown when the attribute is not `double` type.

`IInvalidAttributeNameException`
Thrown when the attribute name is invalid.

Example

```
// Call the double method
String doubleAttributeName = "Average";
double value = 5.75;
exampleBusObj.IsetDoubleAttribute(doubleAttributeName, value);
```

IsetFloatAttribute()

Sets an attribute to a `float` value.

Syntax

```
void IsetFloatAttribute(string attributeName, float value);
```

Parameters

attributeName The name of the attribute whose value is set.

value The `float` value for the attribute.

Return Values

None.

Exceptions

`IInvalidAttributeTypeException`
Thrown when the attribute is not `float` type.

`IInvalidAttributeNameException`
Thrown when the attribute name is invalid.

Example

```
// Call the Float method
String floatAttributeName "FloatAttributeName";
float value = 0.999;
exampleBusObj.IsetFloatAttribute(floatAttributeName, value);
```

IsetIntAttribute()

Sets an attribute to an `int` value.

Syntax

```
void IsetIntAttribute(string attributeName, long value);
```

Parameters

attributeName The name of the attribute whose value is set.

value A `long` value for the integer attribute.

Return Values

None.

Exceptions

`IInvalidAttributeTypeException`
Thrown when the attribute is not an integer type.

`IInvalidAttributeNameException`
Thrown when the attribute name is invalid.

Example

```
// Call the int method
String intAttribute = "CustomerNumber";
int value = 5002;
exampleBusObj.IsetIntAttribute(intAttribute, value);
```

IsetLongTextAttribute()

Sets an attribute to a `longtext` value.

Syntax

```
void IsetLongTextAttribute(string attributeName, string value);
```

Parameters

attributeName The name of the attribute whose value is set.

value The value for the attribute, in string format.

Return Values

None.

Exceptions

`IInvalidAttributeTypeException`
Thrown when the attribute is not `longtext` type.

`IInvalidAttributeNameException`
Thrown when the attribute name is invalid.

Example

```
// Call the LongText method
String longTextAttributeName = "Description";
String value = "A very long text"
exampleBusObj.IsetLongTextAttribute(longTextAttributeName, value);
```

IsetStringAttribute()

Sets an attribute to a `string` value.

Syntax

```
void IsetStringAttribute(string attributeName, string value);
```

Parameters

attributeName The name of the attribute whose value is set.

value The `string` value for the attribute.

Return Values

None.

Exceptions

`IInvalidAttributeTypeException`
Thrown when the attribute is not `string` type.

`IInvalidAttributeNameException`
Thrown when the attribute name is invalid.

Example

```
// Call the String method
String stringAttribute = "CustomerName";
String value = "Greatest Customer";
exampleBusObj.IsetStringAttribute(stringAttribute, value);
```

IsetVerb()

Sets the verb for the business object.

Syntax

```
void IsetVerb(string verb);
```

Parameters

verb The verb for the business object

Return Values

None.

Exceptions

`IInvalidVerbException`
Thrown when the verb is not supported by the business object.

Example

```
// Set the verb
String verb = "Create";
exampleBusObj.IsetVerb(verb);
```

ItoExternalForm()

Serializes the business object data into an external format of the specified MIME type.

Syntax

```
string ItoExternalForm(string mimeType);
```

Parameters

mimeType The MIME type (of the access client) to convert the business object to.

Return Values

A `string` that contains the serialized version of the business object, in the specified MIME type.

Exceptions

`IMalFormedDataException`
Thrown when the conversion runs into an error.

Notes

The `ItoExternalForm()` method invokes a data handler, passing it the MIME type of the serialized data. The data handler parses and converts the InterChange Server business object into serialized data of the requested MIME type, returning the serialized data to the access client. The format of the serialized data must be of a type that IBM WebSphere InterChange Server Software supports or a custom data handler you have written. For more information, see the *Data Handler Guide*.

Example

```
// Serialize data into html
String mimeType = "text/html");
String htmldata = exampleBusObj.ItoExternalForm(mimeType);
```

ItoString()

Returns the dump o f the business object in the WebSphere InterChange Server broker serialization format.

Syntax

```
string ItoString();
```

Parameters

None.

Return Values

A `string` that contains the serialized data in an IBM WebSphere InterChange Server-compatible format.

Example

```
// Convert to IBM format
String stringBusObj;
stringBusObj = exampleBusObj.ItoString();
```

Chapter 9. IBusinessObjectArray interface

The IBusinessObjectArray interface provides methods to return a business object, an array, an array attribute or to set attributes or objects within an array. Table 10 summarizes the methods of the IBusinessObjectArray interface.

Table 10. Member methods of the IBusinessObjectArray interface

Method	Description	Page
Iduplicate()	Returns a clone of the business object array.	87
IdeleteBusinessObjectAtIndex()	Deletes the business object at the specified index of the business object array.	88
IgetBusinessObjectAtIndex()	Retrieves a business object at the given index of the business object array.	88
IgetSize()	Returns the size of the business object array.	88
IremoveAllElements()	Removes all the elements (business objects) in the business object array.	89
IsetBusinessObject()	Sets the business object at the end of the business object array.	89
IsetBusinessObjectAtIndex()	Sets the business object at the specified index of the business object array.	90

Iduplicate()

Returns a clone of the business object array.

Syntax

```
IBusinessObjectArray Iduplicate();
```

Parameters

None.

Return Values

An IBusinessObjectArray object that contains the duplicate business object array.

Exceptions

ICxAccessError	Thrown when the business object array cannot be accessed.

Example

The following example duplicates sourceCustomer in order to create destCustomer.

```
IBusinessObjectArray srcBOArray =
     accessSession.IcreateBusinessObjectArray ("Customer");
IBusinessObjectArray destBOArray = scrBOArray.Iduplicate();
```

IdeleteBusinessObjectAtIndex()

Deletes the business object at the specified index of the business object array.

Syntax

```
void IdeleteBusinessObjectAtIndex(long index);
```

Parameters

index The index in the business object array of the business object to delete.

Return Values

None.

Exceptions

`IInvalidIndexException`
Thrown when the index is invalid.

Example

```
//Delete the business object
long index = 5;
exampleBusObjArray.IdeleteBusinessObjectAtIndex(index);
```

IgetBusinessObjectAtIndex()

Retrieves a business object at the given index of the business object array.

Syntax

```
IBusinessObject IgetBusinessObjectAtIndex(long index);
```

Parameters

index The index in the business object array of the business object to retrieve.

Return Values

An `IBusinessObject` object that contains the business object at the specified index of the business object array.

Exceptions

`IInvalidIndexException`
Thrown when the index is invalid.

Example

```
// call the get business object at index method
IBusinessObject aBusinessObject = null;
long index = 1;
aBusinessObject = exampleBusObjArray.IgetBusinessObjectAtIndex(index);
```

IgetSize()

Returns the size of the business object array.

Syntax

```
long IgetSize();
```

Parameters

None.

Return Values

An integer to indicate the number of elements (business objects) in the business object array.

Example

```
// get the array size
long = arraySize = 0;
arraySize = exampleBusObjArray.IgetSize();
```

IremoveAllElements()

Removes all the elements (business objects) in the business object array.

Syntax

```
void IremoveAllElements()
```

Parameters

None.

Return Values

None.

Example

```
// remove array elements
exampleBusObjArray.IremoveAllElements();
```

IsetBusinessObject()

Sets the business object at the end of the business object array.

Syntax

```
void IsetBusinessObject(IBusinessObject value);
```

Parameters

value The business object to set at the end of the array.

Return Values

None.

Exceptions

`IInvalidBusinessObjectTypeException`
Thrown when the business object is not supported.

Example

```
// Set the business object at the end of the array
IBusinessObject srcBO = accessSession.IcreateBusinessObject(
     "PayableNetChange");
exampleBusObjArray.IsetBusinessObject(srcBO);
```

IsetBusinessObjectAtIndex()

Sets the business object at the specified index of the business object array.

Syntax

```
void IsetBusinessObjectAtIndex(long index, IBusinessObject inObj);
```

Parameters

index The index in the business object array.

inObj The business object to be placed in the array.

Exceptions

`IInvalidIndexException`
Thrown when the index is invalid.

`IInvalidBusinessObjectTypeException`
Thrown when the business object type is not supported by the business object array.

Example

```
// Set the business object at the index
long index = 1;
IBusinessObject aBusObj = accessSession.IcreateBusinessObject(
     "PayableNetChange");
exampleBusObjArray.IsetBusinessObjectAtIndex(index, aBusObj);
```

Chapter 10. Server Access Interface exceptions

This chapter describes the Server Access Interface exceptions. The exceptions thrown by methods of the Server Access Interface are subclasses of the following exception class:

```
org.omg.CORBA.UserException
```

Note: This `UserException` class is external class. It is *not* an IBM Crosswords exception class. Please consult the IBM Java ORB documentation for the members and methods of `UserException`.

All Server Access Interface exceptions contain a `string` error message member called `IerrorMessage`.

Table 11 summarizes the exceptions of the Server Access Interface.

Table 11. Exceptions summary

IAttributeBlankException

This exception is thrown when the attribute contains a blank value.

Members

```
string IerrorMessage;
```

IAttributeNotSetException

This exception is thrown when the attribute does not contain a value.

Members

```
string IerrorMessage;
```

ICxAccessError

This exception is thrown when an object cannot be accessed.

Members

```
string IerrorMessage;
```

IExecuteCollaborationError

This exception is thrown when execution of a collaboration fails.

Members

```
string IerrorMessage;
long status;
```

Notes

The two following methods, which request execution of a collaboration, can throw the `IExecuteCollaborationError` exception:

- IexecuteCollaboration()
- IexecuteCollaborationExtFmt()

This exception contains a public `int` variable called `status` to indicate the details of when the exception occurred. The Server Access Interface provides execution-status constants to represent the possible values of this `status` variable. The execution-status constants for this exception are listed in Table 12

Table 12. Values for the IExecuteCollaborationError Status

Constant Name	Description
`UNKNOWNSTATUS`	The status of the call to `IexecuteCollaboration()` or `IexecuteCollaborationExtFmt()` method.
`FAILEDTOREACHCOLLABORATION`	The access request did not reach the collaboration.
`FAILEDINEXECUTIONOFCOLLABORATION`	The access request failed while executing the collaboration.
`FAILEDINRETURNTOCLIENT`	The collaboration executed but an error occurred while delivering the response to the access client.

To obtain this value, dereference your exception variable as follows:

`this_exception_name_caught``.status`

IInvalidAttributeNameException

This exception is thrown when the attribute name is invalid.

Members

```
string IerrorMessage;
```

IInvalidAttributeTypeException

This exception is thrown when the attribute type is invalid.

Members

```
string IerrorMessage;
```

IInvalidBusinessObjectTypeException

This exception is thrown when the business object type does not match the container.

Members

```
string IerrorMessage;
```

IInvalidIndexException

This exception is thrown when the index is invalid.

Members

```
string IerrorMessage;
```

IInvalidVerbException

This exception is thrown when the verb is invalid.

Members

```
string IerrorMessage;
```

IMalFormedDataException

This exception is thrown when the data is malformed.

Members

```
string IerrorMessage;
```

IValueNotSetException

This exception is thrown when the attribute has no default value.

Members

```
string IerrorMessage;
```

IVerbNotSetException

This exception is thrown when the verb is not set.

Members

```
string IerrorMessage;
```

Part 4. Appendixes

Appendix. Internationalization considerations

An internationalized access client is one that has been written in such as way that it can be customized for a particular locale. A locale is the part of a user's environment that brings together information about how to handle data that is specific to the end user's particular country, language, or territory.

This section provides the following information on an internationalized access client:

- "What Is a locale?"
- "Designing an Access client for internationalization"

What Is a locale?

A **locale** is the part of a user's environment that brings together information about how to handle data that is specific to the end user's particular country, language, or territory. The locale is typically installed as part of the operating system.

A **locale** provides the following information for the user environment:

- Cultural conventions according to the language and country (or territory)
  - Data formats:

    Dates: define full and abbreviated names for weekdays and months, as well as the structure of the date (including date separator).

    Numbers: define symbols for the thousands separator and decimal point, as well as where these symbols are placed within the number.

    Times: define indicators for 12-hour time (such AM and PM indicators) as well as the structure of the time.

    Monetary values: define numeric and currency symbols, as well as where these symbols are placed within the monetary value.
  - Collation order indicates how to sort data for the particular character code set and language.
  - String handling includes tasks such as letter "case" (upper case and lower case) comparison, substrings, and concatenation.

Designing an Access client for internationalization

To use an access client in an internationalized context, take into account both Locale and character-encoding considerations.

Locale considerations

To be internationalized, a access client must be coded to be locale-sensitive; that is, its behavior must take the locale setting into consideration and perform the task appropriate to that locale.

Typically the access client should follow these locale-sensitive design principles:

- The text of any error, status, and trace messages should be isolated from the application-specific component in a message file and translated into the language of the locale.

- Sorting or collation of data uses a collation order appropriate for the language and country of the locale.
- String processing (such as comparison, substrings, and letter case) is appropriate for characters in the locale's language.
- Formats of dates, numbers, and times are appropriate for the locale.

Character-Encoding

The Server Access Interface uses UCS-2, a form of Unicode. Data that the access client transfers to the Server Access Interface must be use Unicode character-encoding.

Index

Special characters

A

B

C

D

E

F

I

J

K

L

M

S

T

U

V

Notices

IBM may not offer the products, services, or features discussed in this document in all countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Burlingame Laboratory Director
IBM Burlingame Laboratory
577 Airport Blvd., Suite 800

Burlingame, CA 94010
U.S.A

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not necessarily tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples may include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Programming interface information

Programming interface information, if provided, is intended to help you create application software using this program.

General-use programming interfaces allow you to write application software that obtain the services of this program's tools.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Warning: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks and service marks

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States or other countries, or both:

IBM
the IBM logo
AIX
CrossWorlds
DB2
DB2 Universal Database
Domino
Lotus
Lotus Notes
MQIntegrator
MQSeries
Tivoli
WebSphere

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

MMX, Pentium, and ProShare are trademarks or registered trademarks of Intel Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product or service names may be trademarks or service marks of others.

IBM WebSphere InterChange Server v4.2.2, IBM WebSphere Business Integration Toolset v4.2.2