

webMethods EntireX

Calling REST from COBOL

Version 10.9

April 2023

This document applies to webMethods EntireX Version 10.9 and all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 1997-2023 Software AG, Darmstadt, Germany and/or Software AG USA, Inc., Reston, VA, USA, and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA, Inc. and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software AG and/or its subsidiaries is located at <http://softwareag.com/licenses>.

Use of this software is subject to adherence to Software AG's licensing conditions and terms. These terms are part of the product documentation, located at <http://softwareag.com/licenses/> and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third-Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software AG Products / Copyright and Trademark Notices of Software AG Products". These documents are part of the product documentation, located at <http://softwareag.com/licenses> and/or in the root installation directory of the licensed product(s).

Use, reproduction, transfer, publication or disclosure is prohibited except as specifically provided for in your License Agreement with Software AG.

Document ID: EXX-EEXXSCENARIO-REST2COB-109-20230403

Table of Contents

Calling REST from COBOL	v
1 About this Documentation	1
Document Conventions	2
Online Information and Support	2
Data Protection	3
2 Calling REST from COBOL under z/OS	5
3 Calling REST from COBOL under z/OS CICS	7
Introduction	8
What do I need to Install for this Scenario?	9
Task 1: Create the Consumer REST API Descriptor, Connections and Listeners	9
Task 2: Generate Client Interface Objects and Build Client Application	16
Task 3: Execute the Call from COBOL to REST	21
4 Calling REST from COBOL under z/OS Batch	23
Introduction	24
What do I need to Install for this Scenario?	25
Task 1: Create the Consumer REST API Descriptor, Connections and Listeners	25
Task 2: Generate Client Interface Objects and Build Client Application	32
Task 3: Execute the Call from COBOL to REST	36
5 Calling REST from COBOL under z/OS IMS	39
Introduction	40
What do I need to Install for this Scenario?	41
Task 1: Create the Consumer REST API Descriptor, Connections and Listeners	41
Task 2: Generate Client Interface Objects and Build Client Application	48
Task 3: Execute the Call from COBOL to REST	52
6 Calling REST from COBOL under z/OS IDMS/DC	55
Introduction	56
What do I need to Install for this Scenario?	57
Task 1: Create the Consumer REST API Descriptor, Connections and Listeners	57
Task 2: Generate Client Interface Objects and Build Client Application	64
Task 3: Execute the Call from COBOL to REST	68
7 Calling REST from COBOL under BS2000	71
Introduction	72
What do I need to Install for this Scenario?	73
Task 1: Create the Consumer REST API Descriptor, Connections and Listeners	73
Task 2: Generate Client Interface Objects and Build Client Application	80
Task 3: Execute the Call from COBOL to REST	84
8 Calling REST from COBOL under z/VSE CICS	87
Introduction	88

What do I need to Install for this Scenario?	89
Task 1: Create the Consumer REST API Descriptor, Connections and Listeners	89
Task 2: Generate Client Interface Objects and Build Client Application	96
Task 3: Execute the Call from COBOL to REST	101
9 Calling REST from COBOL under z/VSE Batch	103
Introduction	104
What do I need to Install for this Scenario?	105
Task 1: Create the Consumer REST API Descriptor, Connections and Listeners	105
Task 2: Generate Client Interface Objects and Build Client Application	112
Task 3: Execute the Call from COBOL to REST	116
10 Calling REST from COBOL under IBM i	119
Introduction	120
What do I need to Install for this Scenario?	121
Task 1: Create the Consumer REST API Descriptor, Connections and Listeners	121
Task 2: Generate Client Interface Objects and Build Client Application	128
Task 3: Execute the Call from COBOL to REST	132

Calling REST from COBOL

1

About this Documentation

■ Document Conventions	2
■ Online Information and Support	2
■ Data Protection	3

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
Monospace font	Identifies service names and locations in the format <i>folder.subfolder.service</i> , APIs, Java classes, methods, properties.
<i>Italic</i>	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
Monospace font	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the symbol.
[]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Product Documentation

You can find the product documentation on our documentation website at <https://documentation.softwareag.com>.

In addition, you can also access the cloud product documentation via <https://www.software-ag.cloud>. Navigate to the desired product and then, depending on your solution, go to “Developer Center”, “User Center” or “Documentation”.

Product Training

You can find helpful product training material on our Learning Portal at <https://knowledge.softwareag.com>.

Tech Community

You can collaborate with Software AG experts on our Tech Community website at <https://tech-community.softwareag.com>. From here you can, for example:

- Browse through our vast knowledge base.
- Ask questions and find answers in our discussion forums.
- Get the latest Software AG news and announcements.
- Explore our communities.
- Go to our public GitHub and Docker repositories at <https://github.com/softwareag> and <https://hub.docker.com/publishers/softwareag> and discover additional Software AG resources.

Product Support

Support for Software AG products is provided to licensed customers via our Empower Portal at <https://empower.softwareag.com>. Many services on this portal require that you have an account. If you do not yet have one, you can request it at <https://empower.softwareag.com/register>. Once you have an account, you can, for example:

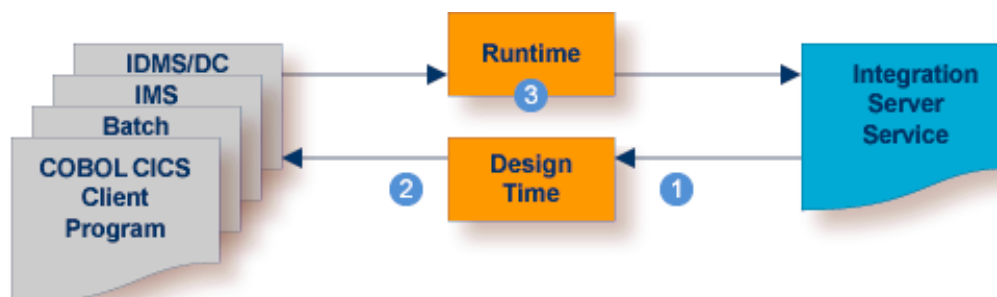
- Download products, updates and fixes.
- Search the Knowledge Center for technical information and tips.
- Subscribe to early warnings and critical alerts.
- Open and update support incidents.
- Add product feature requests.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

2 Calling REST from COBOL under z/OS

Under z/OS, the Integration Server can be called from a COBOL client in different environments.



- ① Use IDL Extractor for Integration Server to generate Integration Server connections and listeners from the Integration Server service to be called.
- ② Generate client interface objects and build COBOL client application.
- ③ Execute the call from the COBOL client to the Integration Server service.

Continue with the appropriate scenario:

- CICS
- Batch
- IMS
- IDMS/DC

3

Calling REST from COBOL under z/OS CICS

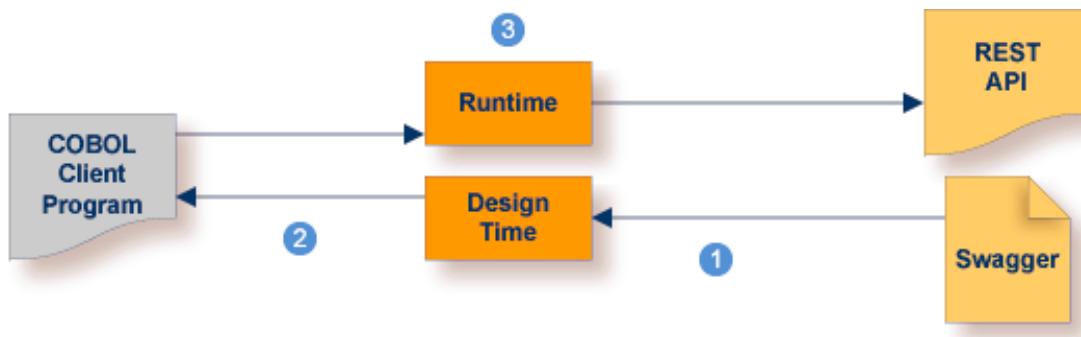
■ Introduction	8
■ What do I need to Install for this Scenario?	9
■ Task 1: Create the Consumer REST API Descriptor, Connections and Listeners	9
■ Task 2: Generate Client Interface Objects and Build Client Application	16
■ Task 3: Execute the Call from COBOL to REST	21

This scenario uses the following tools of the Designer:

- from the **Service Development** perspective: New REST API Descriptor
- from the **EntireX** perspective: *IDL Extractor for Integration Server* and *COBOL Wrapper*

Introduction

To call the REST API from COBOL, take an existing description of a REST API **1** and generate the integration logic **2** to call it from a COBOL application **3**, as shown below.



- 1** Create the consumer REST API descriptor in Integration Server. See *Service Development Help* in the webMethods Integration Server documentation. Then generate Integration Server connections and listeners. See *Using the IDL Extractor for Integration Server*.
- 2** Generate client interface objects and build COBOL client application.
- 3** Execute the call from the COBOL client to the REST API.

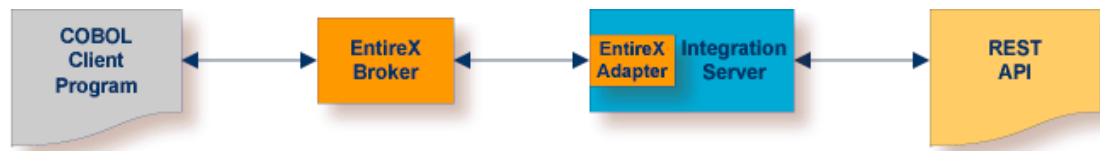
This scenario makes the following important assumptions:

For Task 1:

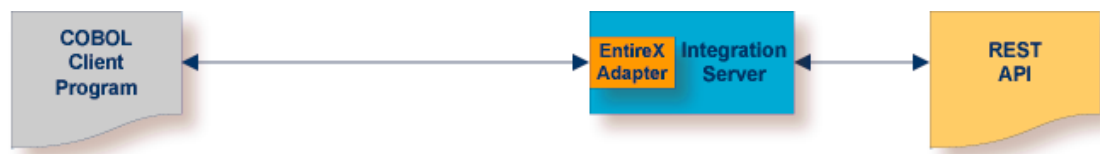
- You have a working Integration Server Service that you want to call from your REST API.

For Task 3:

- You can call the REST API at runtime using different methods:
 - For the EntireX RPC connection method you need EntireX Broker on one of the supported platforms: z/OS | Linux | Windows | BS2000.



- For the EntireX Direct RPC connection method there are no additional prerequisites.



What do I need to Install for this Scenario?

- You have Software AG Designer installed with EntireX plugins and Service Development plugins.
 - For EntireX plugins, see *Designer > EntireX Designer* under *EntireX Installation Packages* in the General Installation documentation.
 - For Service Development plugins, refer to the Integration Server documentation.
- You have an Integration Server with EntireX Adapter installed. See *Adapters > EntireX Adapter* under *EntireX Installation Packages* in the General Installation documentation.
- Optionally, for RPC Connection method (Task 3), you have an EntireX Broker installed.
 - For Linux and Windows, see *EntireX Broker* under *EntireX Installation Packages* in the General Installation documentation.
 - For z/OS, see *Installing EntireX Broker under z/OS* in the z/OS Installation documentation
- On the machine where the COBOL client program is executed, you have the Broker stubs installed. See *Step 3: Install the Broker Stubs and Bind the Broker Stub Executables* in the z/OS Installation documentation.

Task 1: Create the Consumer REST API Descriptor, Connections and Listeners

This section covers the following topics:

- [Introduction](#)
- [Creating a Consumer REST API Descriptor](#)
- [Extracting the Interface of the webMethods IS Service](#)

- [Creating Listener Objects in Integration Server](#)
- [Testing the Extraction Results \(Optional\)](#)

Introduction

After you have created the consumer REST API descriptor, follow the instructions for extracting a webMethods Integration Server (IS) service under *Using the IDL Extractor for Integration Server*.

This process creates the following EntireX metafiles:

- an IDL file, containing definitions of the interface between client and server; see *Software AG IDL File* in the IDL Editor documentation in the IDL Editor documentation
- a webMethods IS Connection
- a webMethods IS Listener, depending on connection type. See *Listeners* in the EntireX Adapter documentation.

Connection types are described under [Creating Listener Objects in Integration Server](#).

Creating a Consumer REST API Descriptor

➤ To create a consumer REST API Descriptor

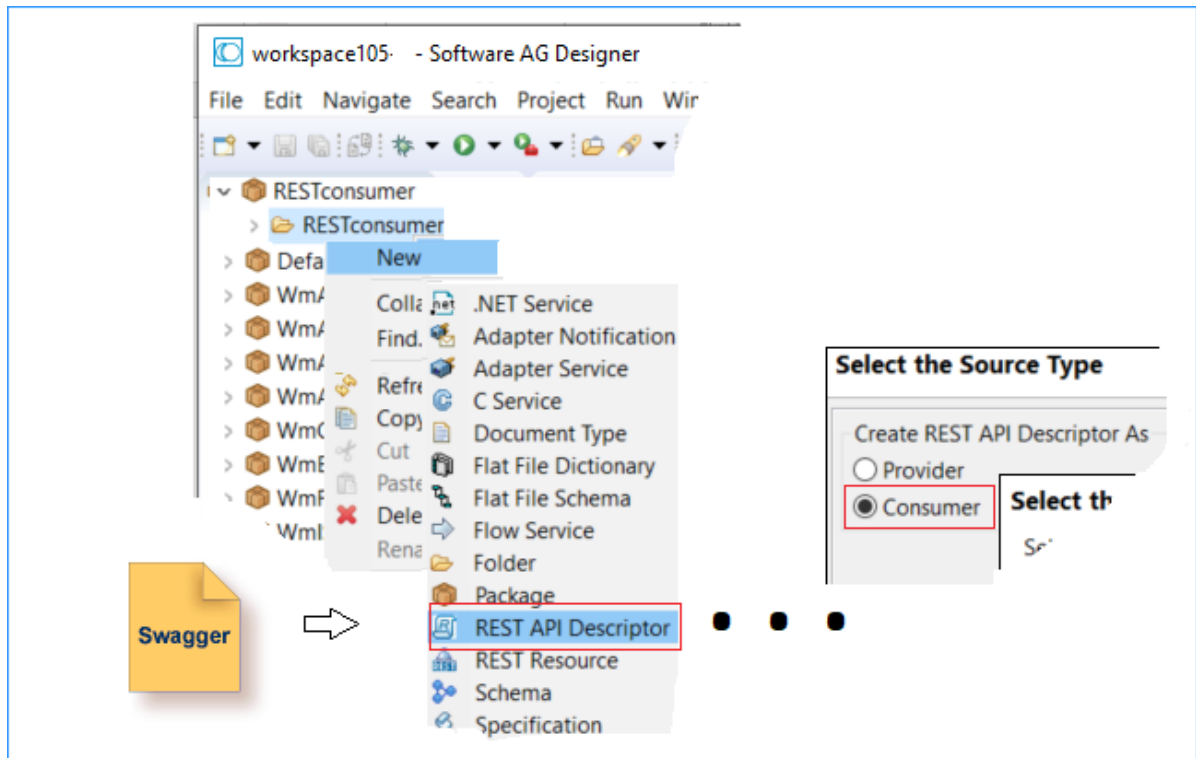
- 1 Switch to the **Service Development** perspective. Create a package and folder in Designer where the API descriptor is placed into.
- 2 Invoke the wizard to create a new REST API Descriptor in Eclipse. From the **File** menu, choose **New > REST API Descriptor**.

Or:

Choose **New** from the toolbar or context menu of a view showing resources.

Or:

Press **Ctrl-N** to start the selection of the New Wizards.



- 3 Follow the instructions on the screen. Provide the description of the REST API (Swagger file) and create the REST API descriptor for a **Consumer**.

For more information on REST API descriptors see *Service Development Help* in the webMethods Integration Server documentation.

Extracting the Interface of the webMethods IS Service

First create a project in Designer. This project is the container into which all extracted and generated EntireX artifacts are placed. Name this project *COBOL placeOrder client*.

» To start the IDL Extractor for Integration Server

- 1 Switch to the EntireX perspective.
- 2 Invoke the IDL Extractor for webMethods IS, which is a New Wizard in Eclipse. From the **File** menu, choose **New**. Select IDL Extractor for webMethods IS in the following page.

Or:

Choose **New** from the toolbar or context menu of a view showing resources.

Or:

Press **Ctrl-N** to start the selection of the New Wizards.

- 3 If you are using the wizard for the first time without any predefined Integration Server connections, enter the TCP/IP address of the webMethods IS to extract from. Otherwise select the connection to the Integration Server.
- 4 Press **Next**.

➤ **To extract the IDL from the selected package**

- 1 Under **Source (Integration Server)**, select the package you want to extract from.

Extract IDL from the selected package

Select the package (source), specify the container and file name (target) and choose the mapping.

Source (Integration Server)
 Packages on Integration Server localhost:5555

- OrderManagement
- WmEntireX
- WmFlatFile
- WmIExtDC

Target (Eclipse Workspace)
 Container: COBOL placeOrder client Browse...
 IDL File Name: OrderManagement

Optimize extracted IDL for usage with: COBOL

Map Integration Server data type to IDL

- ☐ alphanumeric with variable length
- ☒ alphanumeric with fixed length of 256

Used if no constraints available

☒ Create Listener Objects in Integration Server

? < Back Next > Finish Cancel

- 2 For **Target (Eclipse Workspace)**, specify project *COBOL placeOrder client*. Extraction results will be placed in this container. You can keep the **IDL File Name** as it has been derived from the package name by default. Since COBOL is your desired endpoint, select **Optimize extracted IDL for usage with: COBOL**.



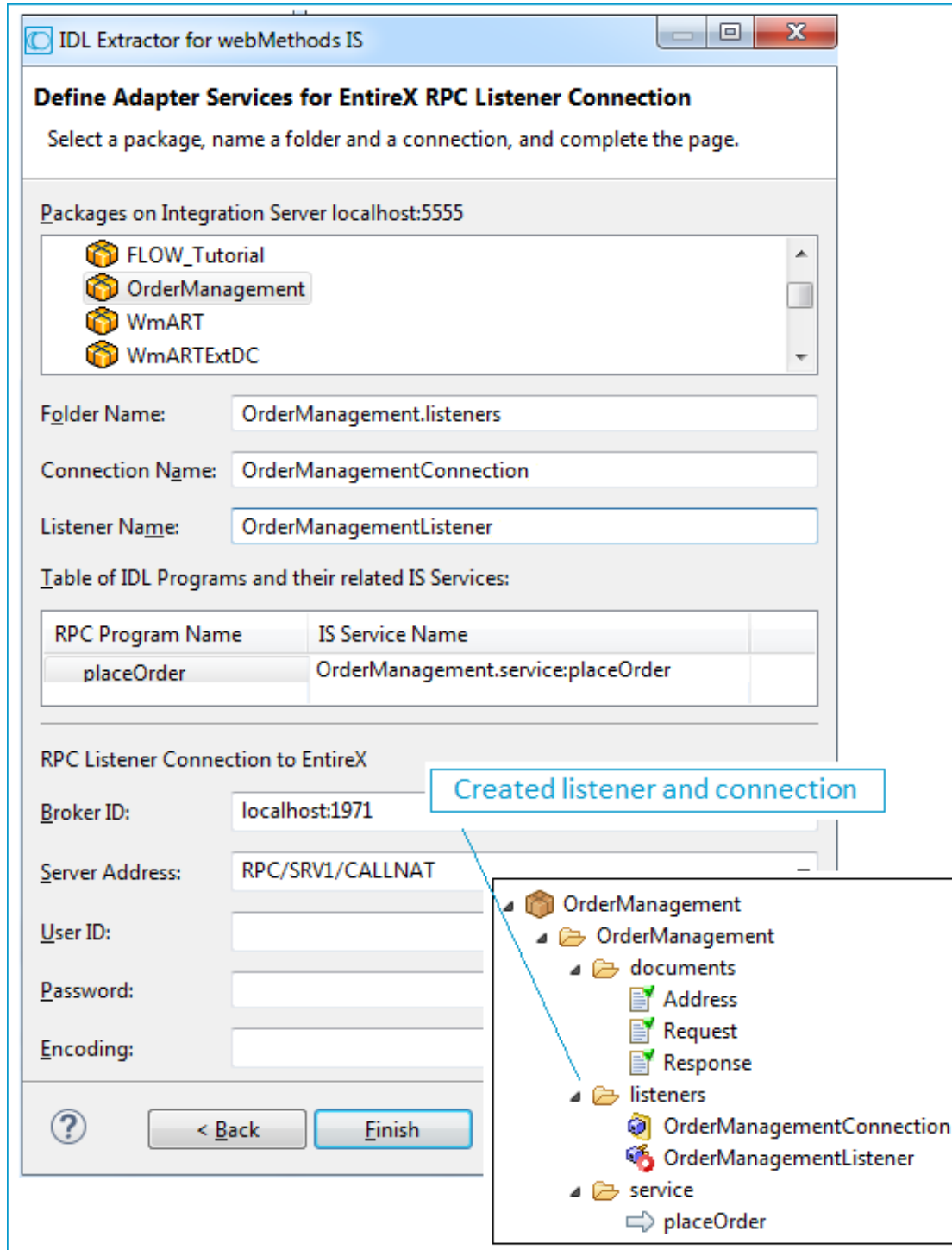
Note: COBOL only supports fixed-length data items, whereas webMethods IS has variable length types, often without any length restrictions (constraints). Parameters with no specified constraints will take the default fixed length. You can specify this default in field alphanumeric with fixed length. .

- 3 A listener is required for calling webMethods IS from an RPC client, so check **Create Listener Objects in Integration Server**.
- 4 Press **Next**.

Creating Listener Objects in Integration Server

➤ To create Listener Objects in Integration Server

- 1 Select a Listener connection type used for inbound connection to the Integration Server.
 - An EntireX RPC Listener connection is the standard way and is always available.
 - If your interface contains only In parameters, an EntireX Reliable RPC Listener connection is available.
 - If it is enabled by the license for the webMethods Integration Server:
 - an EntireX Direct RPC Listener connection is available
 - if your interfaces contain only In parameters, an EntireX Direct Reliable RPC Listener connection is available



- On this page, you set the properties for the listener objects to create; these objects will wait for the incoming COBOL requests. Select the package and specify the folder name into which the listener is generated. If the folder does not exist, it will be created automatically.

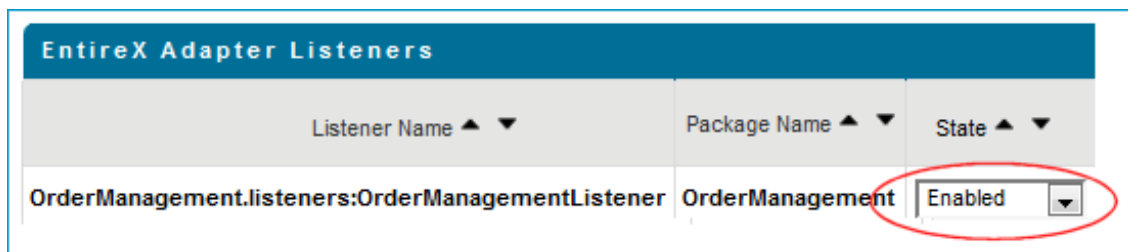
- 3 Keep the defaults given for **RPC Listener Connection to EntireX**. The **Broker ID** is a TCP/IP or SSL/TLS address and **Server Address** is an EntireX-specific namespace to locate a target server (here: *OrderManagementListener*).
- 4 When creating a connection, a package dependency is added such that the selected package depends on webMethods EntireX (the package WmEntireX) with the version currently used.
- 5 Press **Finish**. The listener and the connection appear in the package navigator. For the package navigator, switch to the service development perspective by choosing **Window > Perspective > Open Perspective > Service Development**.

Testing the Extraction Results (Optional)

You can test the results of the extraction and the server back end, using the EntireX IDL Tester.

➤ To test the extraction results

- 1 For EntireX RPC (see [Task 3](#) in the [Introduction](#)) and an RPC Listener Connection, make sure EntireX Broker is running.
- 2 Enable the EntireX Adapter Listener.



- 3 Activate the EntireX perspective by choosing **Window > Perspective > Open Perspective > EntireX**, and from the context menu of the IDL file, choose **IDL Tester**.
- 4 This step depends on the method used to call the server (see [Task 3](#) in the [Introduction](#)).
 - For *EntireX RPC*, enter the same TCP/IP address for **Broker** that you supplied when you created the RPC Listener Connection to EntireX (localhost:1971).
 - For *Direct RPC*, that is, you generated a Direct RPC Listener Connection, enter the TCP/IP address from the Integration Server (localhost:1971).
- 5 For **Server**, enter the values you specified for the server address (RPC/SRV1/CALLNAT).

The screenshot shows a graphical user interface for a COBOL application. At the top, there are two input fields: 'Broker:' with the value 'localhost:1971' and 'Server:' with the value 'RPC/SRV1/CALLNAT'. Both fields are circled in red. Below these fields are five buttons: 'Call' (highlighted with a dashed border), 'Ping', 'Open Conversation', 'Reset', and 'Exit'. Below the buttons is a 'Messages' section with a scrollable text area. Below the messages is a table displaying the results of the REST API call.

2 OrderItemID (N5)	45
2 OrderQuantity (N10)	18
2 OrderCustomerID (I4)	234
2 OrderItemName (A99)	tennis racquet
2 OrderUnitPrice (A20)	79.89 GBP
2 OrderTotal (A20)	1438.02 GBP
3 Name (A99)	The All England Lawn Tennis Club (Championships) Limited
3 Street (A99)	Church Road
3 City (A99)	Wimbledon London
3 ZipCode (A20)	SW 19 5AE
3 CountryCode (A20)	UK
2 OrderShippingDate (A256)	Your order will be shipped on Wed, Jul 27, 2016.
2 OrderConfirmation (A256)	Your order is confirmed.
2 OrderConfirmationFlag (L)	true

Task 2: Generate Client Interface Objects and Build Client Application

This section covers the following topics:

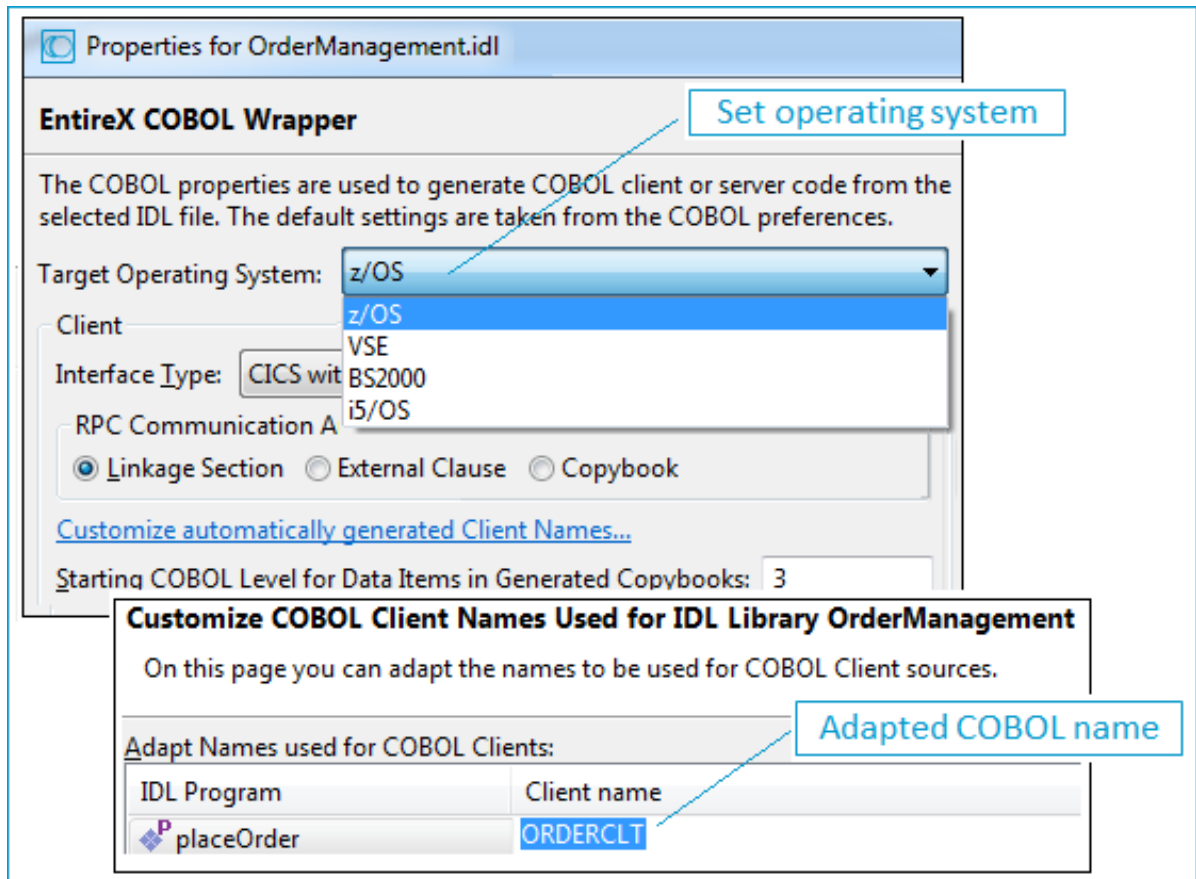
- [Setting Generation Options \(Properties\) for the COBOL Wrapper](#)
- [Generating COBOL Code](#)

■ Building the Client Application

Setting Generation Options (Properties) for the COBOL Wrapper

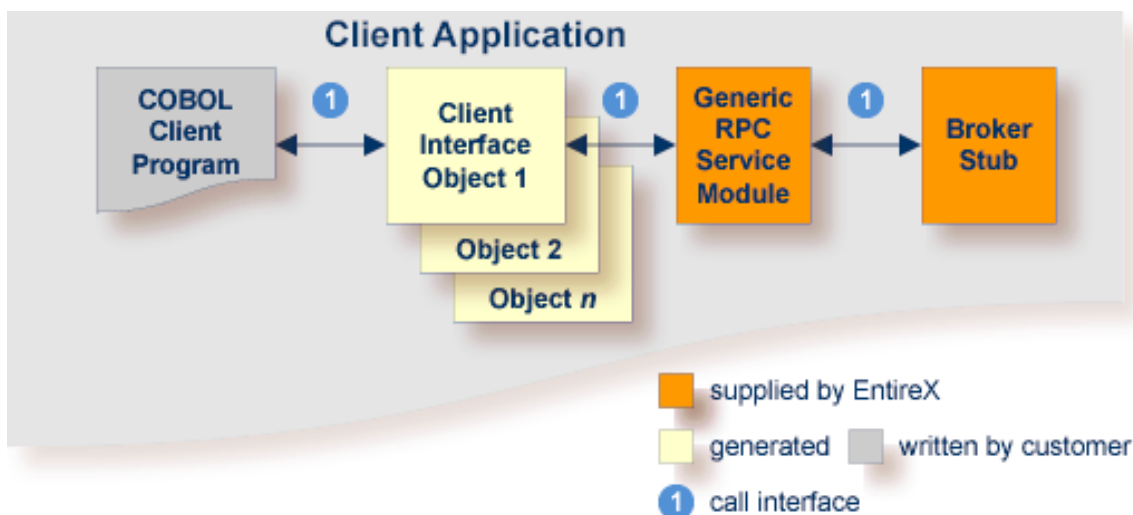
➤ To set generation options (properties) for the COBOL Wrapper

- 1 To generate suitable COBOL code for your environment, first set the correct target operating system. Make sure the EntireX perspective is active (**Window > Perspective > Open Perspective > EntireX**) and call the properties of the COBOL Wrapper.



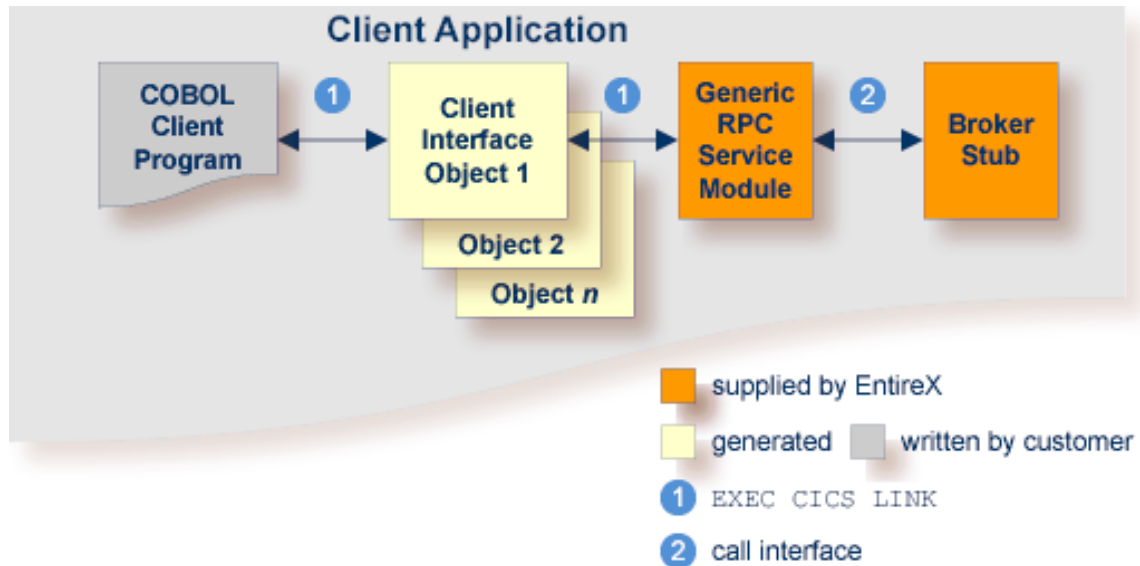
- 2 From the context menu of the IDL file, choose **Properties > EntireX COBOL Wrapper**. On the COBOL Wrapper **Properties** page, set the **Target Operating System**
- 3 Adapt the name of the generated COBOL program: Choose **Customize automatically generated Client Names**. Take ORDERCLT as the COBOL program's name.
- 4 Set the appropriate interface type for the generated client interface objects to your application. The options available for CICS are Standard Linkage Calling Convention and DFHCOM-MAREA Calling Convention. These are described below.

■ Standard Linkage Calling Convention



The COBOL Wrapper can be used with a call interface, even in CICS. This means you can build a client application where the COBOL client program, every generated client interface object, the generic RPC services module and the broker stub are linked together, similar to the batch scenario. See *Using the COBOL Wrapper for Batch (z/OS, BS2000, z/VSE and IBM i)*. Using a call interface within CICS may be useful in the following situations:

- The restriction of the COMMAREA length (about 31 KB) prevents you from using the scenario *Using the COBOL Wrapper for CICS with DFHCOMMAREA Calling Convention (z/OS and z/VSE)* (see above).
 - You do *not* require a distributed program link (CICS DPL) to your client interface object(s).
 - You prefer a call interface instead of EXEC CICS LINK to your client interface objects.
- ## ■ DFHCOMMAREA Calling Convention



In this scenario, the generic RPC services module and the broker stub are linked together to a CICS program. The COBOL client program, every generated client interface object and the generic RPC services module together with the broker stub are installed each as separate individual CICS programs. Use the COBOL Wrapper with this calling convention in the following situations:

- You want to have an EXEC CICS LINK DFHCOMMAREA interface to your client interface object(s).
- The restriction of the COMMAREA length suits your purposes. Because the RPC communication area is also transferred in the COMMAREA, the effective length that can be used for IDL data is shorter than the CICS COMMAREA length. Nearly 31 KB can be used for IDL data.
- You wish to separate the generic RPC service module and the broker stub from the client interface object(s).
- You require a program link to the client interface object(s).

Generating COBOL Code

➤ To generate COBOL code

- For COBOL code generation, make sure the EntireX Perspective is active: Choose **Window > Perspective > Open Perspective > EntireX**. From context menu of the IDL file, choose **COBOL > Generate RPC Client**. This generates the COBOL client *ORDERCLT* in folder *client* and a copybook in folder *include*:

The screenshot displays a COBOL development environment with the following components:

- COBOL client:** A window showing the source code of the COBOL program. It includes a header section with 'CBL LIB QUOTE' and 'IDENTIFICATION DIVISION. PROGRAM-ID. ORDERCLT.'.
- Copybook:** A window showing the COBOL RPC Interface Copybook. It contains metadata such as 'ID Library = OrderManagement' and 'ID Program = placeOrder'. It also lists the 'Copybook Name = ORDERCLT' and 'Target Platform = all'.
- Package Explorer:** A window showing the project structure, including 'COBOL placeOrder client', 'src', 'JRE System Library [JavaSE-1.8]', 'client', 'include', and 'OrderManagement.idl'.
- Constraints available:** A table listing the data types and lengths for the COBOL parameters. The table is as follows:

Parameter	Constraint
03 REQUEST.	PIC 9(5).
04 ORDERITEMID	PIC 9(10).
04 ORDERQUANTITY	PIC 9(10).
04 ORDERCUSTOMERID	PIC S9(9) BINARY.
03 RESPONSE.	
04 ORDERITEMNAME	PIC X(99).
04 ORDERUNITPRICE	PIC X(20).
04 ORDERTOTAL	PIC X(20).
04 ORDERCUSTOMERADDRESS.	
05 NAME	PIC X(99).
05 STREET	PIC X(99).
05 CITY	PIC X(99).
05 ZIPCODE	PIC X(20).
05 COUNTRYCODE	PIC X(20).
04 ORDERSHIPPINGDATE	PIC X(256).
04 ORDERCONFIRMATION	PIC X(256).
04 ORDERCONFIRMATIONFLAG	PIC X.
- No constraints available:** A label pointing to the '04 ORDERCUSTOMERADDRESS.' parameter, indicating that no constraints are available for this parameter.

The copybook is the COBOL programmer's API. It is for use in the COBOL application.

As you would expect, the length of generated COBOL parameters exactly matches the lengths extracted. COBOL parameters always have a fixed length defined at compile time. They are not altered at runtime to reflect actual string length as in modern programming languages. This means that if you choose your default length (see [Extracting the Interface of the webMethods IS Service](#)) you have to choose the maximum possible data size to ensure all data can be transmitted. However,

this may needlessly increase the message size if no constraints are set for fields of shorter data length. Choose your defaults and constraints carefully.

Building the Client Application

Depending on the chosen interface type of your client interface objects (CICS with Standard Linkage Calling Convention or CICS with DFHCOMMAREA Calling Convention under *Setting Generation Options (Properties) for the COBOL Wrapper*, refer to

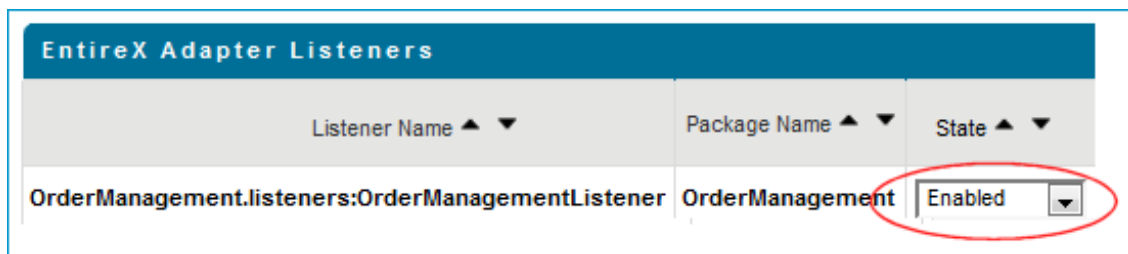
- *Using the COBOL Wrapper for CICS with Call Interfaces (z/OS and z/VSE)*
- *Using the COBOL Wrapper for CICS with DFHCOMMAREA Calling Convention (z/OS and z/VSE)*

Task 3: Execute the Call from COBOL to REST

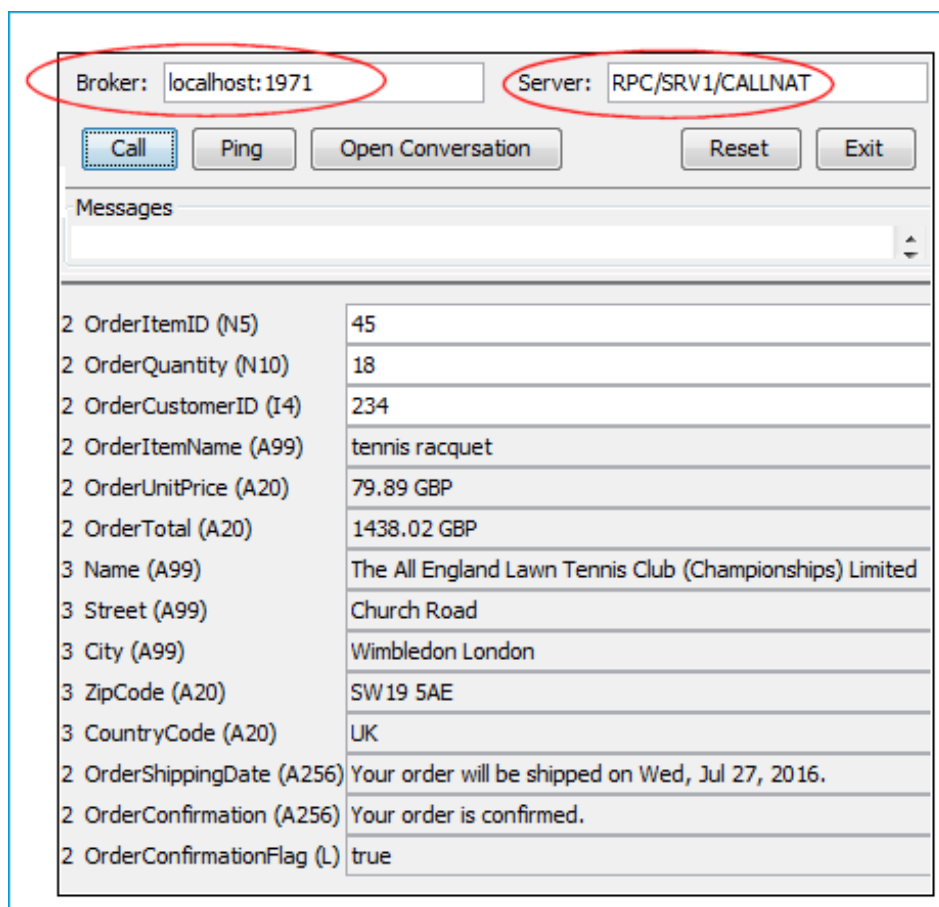
Use the EntireX IDL Tester to execute the call.

➤ To execute the call

- 1 For EntireX RPC (see [Task 3](#) in the [Introduction](#)) and an RPC Listener Connection, make sure EntireX Broker is running.
- 2 Enable the EntireX Adapter Listener.



- 3 Activate the EntireX perspective by choosing **Window > Perspective > Open Perspective > EntireX**, and from the context menu of the IDL file, choose **IDL Tester**.
- 4 This step depends on the method used to call the server (see [Task 3](#) in the [Introduction](#)).
 - For *EntireX RPC*, enter the same TCP/IP address for **Broker** that you supplied when you created the RPC Listener Connection to EntireX (localhost:1971).
 - For *Direct RPC*, that is, you generated a Direct RPC Listener Connection, enter the TCP/IP address from the Integration Server (localhost:1971).
- 5 For **Server**, enter the values you specified for the server address (RPC/SRV1/CALLNAT).



Broker: localhost:1971 Server: RPC/SRV1/CALLNAT

Call Ping Open Conversation Reset Exit

Messages

2 OrderItemID (N5)	45
2 OrderQuantity (N10)	18
2 OrderCustomerID (I4)	234
2 OrderItemName (A99)	tennis racquet
2 OrderUnitPrice (A20)	79.89 GBP
2 OrderTotal (A20)	1438.02 GBP
3 Name (A99)	The All England Lawn Tennis Club (Championships) Limited
3 Street (A99)	Church Road
3 City (A99)	Wimbledon London
3 ZipCode (A20)	SW 19 5AE
3 CountryCode (A20)	UK
2 OrderShippingDate (A256)	Your order will be shipped on Wed, Jul 27, 2016.
2 OrderConfirmation (A256)	Your order is confirmed.
2 OrderConfirmationFlag (L)	true

4

Calling REST from COBOL under z/OS Batch

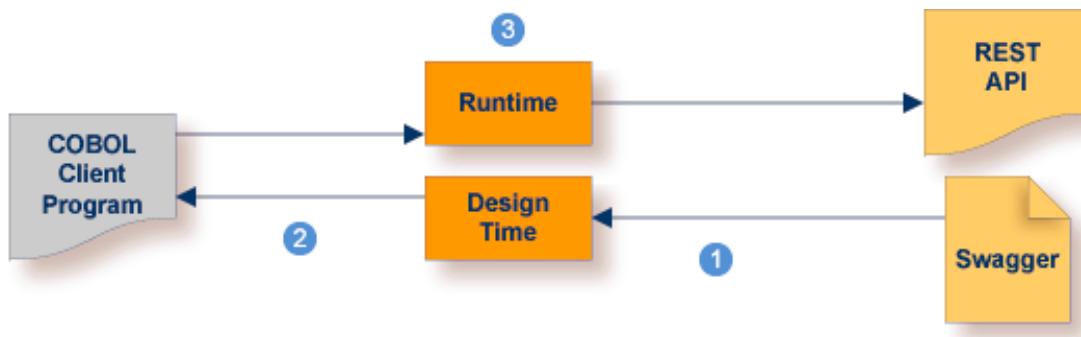
■ Introduction	24
■ What do I need to Install for this Scenario?	25
■ Task 1: Create the Consumer REST API Descriptor, Connections and Listeners	25
■ Task 2: Generate Client Interface Objects and Build Client Application	32
■ Task 3: Execute the Call from COBOL to REST	36

This scenario uses the following tools of the Designer:

- from the **Service Development** perspective: New REST API Descriptor
- from the **EntireX** perspective: *IDL Extractor for Integration Server* and *COBOL Wrapper*

Introduction

To call the REST API from COBOL, take an existing description of a REST API **1** and generate the integration logic **2** to call it from a COBOL application **3**, as shown below.



- 1** Create the consumer REST API descriptor in Integration Server. See *Service Development Help* in the webMethods Integration Server documentation. Then generate Integration Server connections and listeners. See *Using the IDL Extractor for Integration Server*.
- 2** Generate client interface objects and build COBOL client application.
- 3** Execute the call from the COBOL client to the REST API.

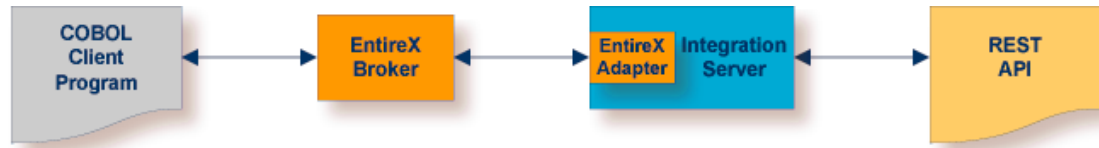
This scenario makes the following important assumptions:

For Task 1:

- You have a working Integration Server Service that you want to call from your REST API.

For Task 3:

- You can call the REST API at runtime using different methods:
 - For the EntireX RPC connection method you need EntireX Broker on one of the supported platforms: z/OS | Linux | Windows | BS2000.



- For the EntireX Direct RPC connection method there are no additional prerequisites.



What do I need to Install for this Scenario?

- You have Software AG Designer installed with EntireX plugins and Service Development plugins.
 - For EntireX plugins, see *Designer > EntireX Designer* under *EntireX Installation Packages* in the General Installation documentation.
 - For Service Development plugins, refer to the Integration Server documentation.
- You have an Integration Server with EntireX Adapter installed. See *Adapters > EntireX Adapter* under *EntireX Installation Packages* in the General Installation documentation.
- Optionally, for RPC Connection method (Task 3), you have an EntireX Broker installed.
 - For Linux and Windows, see *EntireX Broker* under *EntireX Installation Packages* in the General Installation documentation.
 - For z/OS, see *Installing EntireX Broker under z/OS* in the z/OS Installation documentation
- On the machine where the COBOL client program is executed, you have the Broker stubs installed. See *Step 3: Install the Broker Stubs and Bind the Broker Stub Executables* in the z/OS Installation documentation.

Task 1: Create the Consumer REST API Descriptor, Connections and Listeners

This section covers the following topics:

- [Introduction](#)
- [Creating a Consumer REST API Descriptor](#)
- [Extracting the Interface of the webMethods IS Service](#)

- [Creating Listener Objects in Integration Server](#)
- [Testing the Extraction Results \(Optional\)](#)

Introduction

After you have created the consumer REST API descriptor, follow the instructions for extracting a webMethods Integration Server (IS) service under *Using the IDL Extractor for Integration Server*.

This process creates the following EntireX metafiles:

- an IDL file, containing definitions of the interface between client and server; see *Software AG IDL File* in the IDL Editor documentation in the IDL Editor documentation
- a webMethods IS Connection
- a webMethods IS Listener, depending on connection type. See *Listeners* in the EntireX Adapter documentation.

Connection types are described under [Creating Listener Objects in Integration Server](#).

Creating a Consumer REST API Descriptor

➤ To create a consumer REST API Descriptor

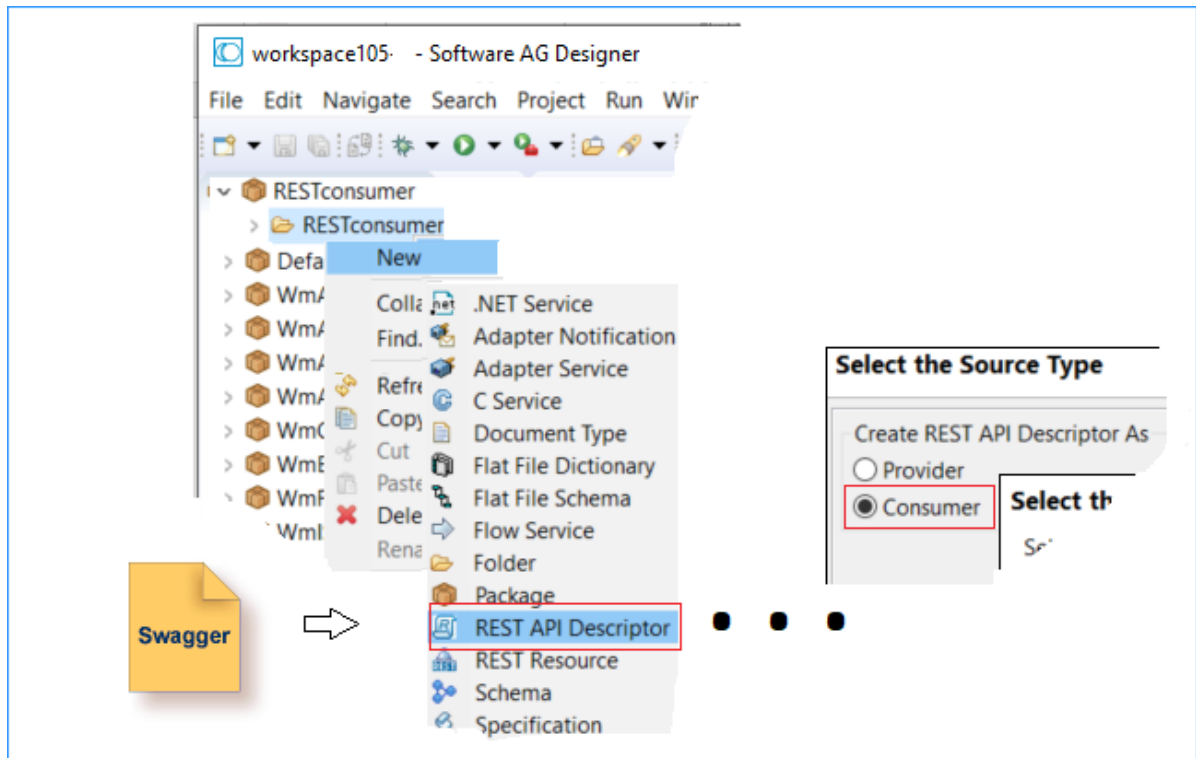
- 1 Switch to the **Service Development** perspective. Create a package and folder in Designer where the API descriptor is placed into.
- 2 Invoke the wizard to create a new REST API Descriptor in Eclipse. From the **File** menu, choose **New > REST API Descriptor**.

Or:

Choose **New** from the toolbar or context menu of a view showing resources.

Or:

Press **Ctrl-N** to start the selection of the New Wizards.



- 3 Follow the instructions on the screen. Provide the description of the REST API (Swagger file) and create the REST API descriptor for a **Consumer**.

For more information on REST API descriptors see *Service Development Help* in the webMethods Integration Server documentation.

Extracting the Interface of the webMethods IS Service

First create a project in Designer. This project is the container into which all extracted and generated EntireX artifacts are placed. Name this project *COBOL placeOrder client*.

» To start the IDL Extractor for Integration Server

- 1 Switch to the EntireX perspective.
- 2 Invoke the IDL Extractor for webMethods IS, which is a New Wizard in Eclipse. From the **File** menu, choose **New**. Select IDL Extractor for webMethods IS in the following page.

Or:

Choose **New** from the toolbar or context menu of a view showing resources.

Or:

Press **Ctrl-N** to start the selection of the New Wizards.

- 3 If you are using the wizard for the first time without any predefined Integration Server connections, enter the TCP/IP address of the webMethods IS to extract from. Otherwise select the connection to the Integration Server.
- 4 Press **Next**.

➤ **To extract the IDL from the selected package**

- 1 Under **Source (Integration Server)**, select the package you want to extract from.

Extract IDL from the selected package

Select the package (source), specify the container and file name (target) and choose the mapping.

Source (Integration Server)
 Packages on Integration Server localhost:5555

- OrderManagement
- WmEntireX
- WmFlatFile
- WmIExtDC

Target (Eclipse Workspace)
 Container: COBOL placeOrder client Browse...
 IDL File Name: OrderManagement

Optimize extracted IDL for usage with: COBOL

Map Integration Server data type to IDL

- ☐ alphanumeric with variable length
- ☒ alphanumeric with fixed length of 256

Used if no constraints available

☒ Create Listener Objects in Integration Server

? < Back Next > Finish Cancel

- 2 For **Target (Eclipse Workspace)**, specify project *COBOL placeOrder client*. Extraction results will be placed in this container. You can keep the **IDL File Name** as it has been derived from the package name by default. Since COBOL is your desired endpoint, select **Optimize extracted IDL for usage with: COBOL**.



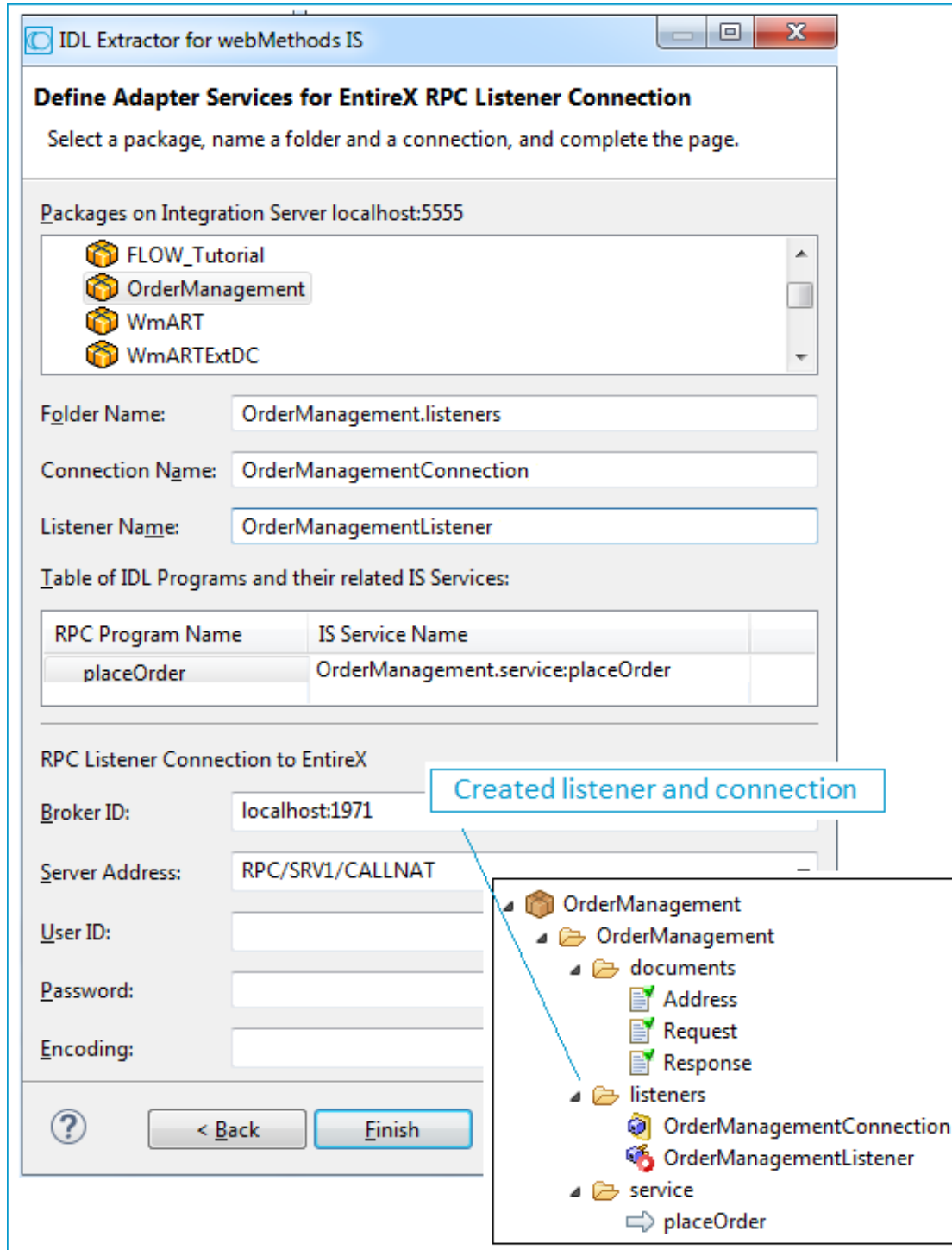
Note: COBOL only supports fixed-length data items, whereas webMethods IS has variable length types, often without any length restrictions (constraints). Parameters with no specified constraints will take the default fixed length. You can specify this default in field alphanumeric with fixed length. .

- 3 A listener is required for calling webMethods IS from an RPC client, so check **Create Listener Objects in Integration Server**.
- 4 Press **Next**.

Creating Listener Objects in Integration Server

➤ To create Listener Objects in Integration Server

- 1 Select a Listener connection type used for inbound connection to the Integration Server.
 - An EntireX RPC Listener connection is the standard way and is always available.
 - If your interface contains only In parameters, an EntireX Reliable RPC Listener connection is available.
 - If it is enabled by the license for the webMethods Integration Server:
 - an EntireX Direct RPC Listener connection is available
 - if your interfaces contain only In parameters, an EntireX Direct Reliable RPC Listener connection is available



- On this page, you set the properties for the listener objects to create; these objects will wait for the incoming COBOL requests. Select the package and specify the folder name into which the listener is generated. If the folder does not exist, it will be created automatically.

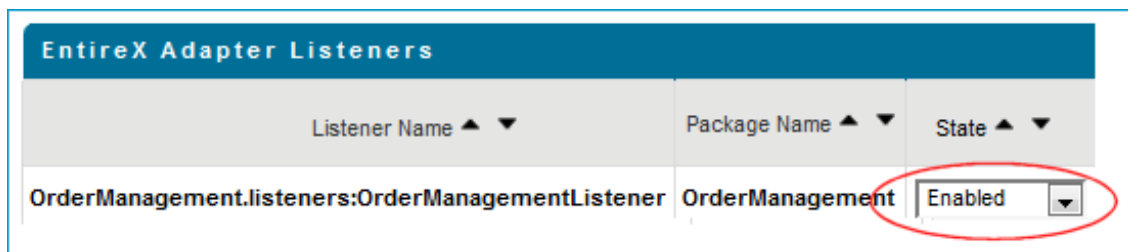
- 3 Keep the defaults given for **RPC Listener Connection to EntireX**. The **Broker ID** is a TCP/IP or SSL/TLS address and **Server Address** is an EntireX-specific namespace to locate a target server (here: *OrderManagementListener*).
- 4 When creating a connection, a package dependency is added such that the selected package depends on webMethods EntireX (the package WmEntireX) with the version currently used.
- 5 Press **Finish**. The listener and the connection appear in the package navigator. For the package navigator, switch to the service development perspective by choosing **Window > Perspective > Open Perspective > Service Development**.

Testing the Extraction Results (Optional)

You can test the results of the extraction and the server back end, using the EntireX IDL Tester.

➤ To test the extraction results

- 1 For EntireX RPC (see [Task 3](#) in the [Introduction](#)) and an RPC Listener Connection, make sure EntireX Broker is running.
- 2 Enable the EntireX Adapter Listener.



- 3 Activate the EntireX perspective by choosing **Window > Perspective > Open Perspective > EntireX**, and from the context menu of the IDL file, choose **IDL Tester**.
- 4 This step depends on the method used to call the server (see [Task 3](#) in the [Introduction](#)).
 - For *EntireX RPC*, enter the same TCP/IP address for **Broker** that you supplied when you created the RPC Listener Connection to EntireX (localhost:1971).
 - For *Direct RPC*, that is, you generated a Direct RPC Listener Connection, enter the TCP/IP address from the Integration Server (localhost:1971).
- 5 For **Server**, enter the values you specified for the server address (RPC/SRV1/CALLNAT).

Broker: localhost:1971 Server: RPC/SRV1/CALLNAT

Call Ping Open Conversation Reset Exit

Messages

2 OrderItemID (N5)	45
2 OrderQuantity (N10)	18
2 OrderCustomerID (I4)	234
2 OrderItemName (A99)	tennis racquet
2 OrderUnitPrice (A20)	79.89 GBP
2 OrderTotal (A20)	1438.02 GBP
3 Name (A99)	The All England Lawn Tennis Club (Championships) Limited
3 Street (A99)	Church Road
3 City (A99)	Wimbledon London
3 ZipCode (A20)	SW 19 5AE
3 CountryCode (A20)	UK
2 OrderShippingDate (A256)	Your order will be shipped on Wed, Jul 27, 2016.
2 OrderConfirmation (A256)	Your order is confirmed.
2 OrderConfirmationFlag (L)	true

Task 2: Generate Client Interface Objects and Build Client Application

This section covers the following topics:

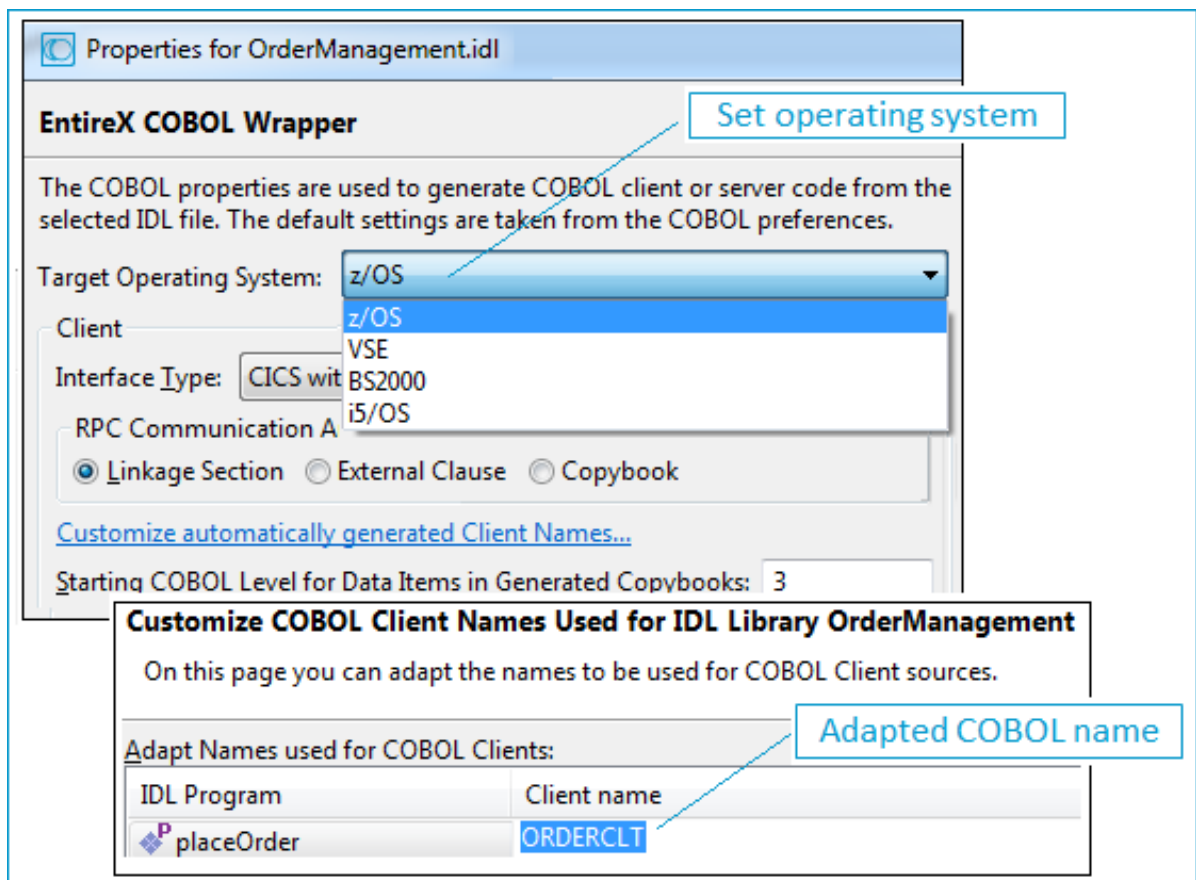
- [Setting Generation Options \(Properties\) for the COBOL Wrapper](#)
- [Generating COBOL Code](#)

■ Building the Client Application

Setting Generation Options (Properties) for the COBOL Wrapper

➤ To set generation options (properties) for the COBOL Wrapper

- 1 To generate suitable COBOL code for your environment, first set the correct target operating system. Make sure the EntireX perspective is active (**Window > Perspective > Open Perspective > EntireX**) and call the properties of the COBOL Wrapper.



- 2 From the context menu of the IDL file, choose **Properties > EntireX COBOL Wrapper**. On the COBOL Wrapper **Properties** page, set the **Target Operating System**
- 3 Adapt the name of the generated COBOL program: Choose **Customize automatically generated Client Names**. Take ORDERCLT as the COBOL program's name.

Generating COBOL Code

➤ To generate COBOL code

- For COBOL code generation, make sure the EntireX Perspective is active: Choose **Window > Perspective > Open Perspective > EntireX**. From context menu of the IDL file, choose **COBOL > Generate RPC Client**. This generates the COBOL client *ORDERCLT* in folder *client* and a copybook in folder *include*:

The screenshot shows the Eclipse IDE with the COBOL client *ORDERCLT* generated. The main editor displays the COBOL code for *ORDERCLT*, which includes a copybook reference. The Package Explorer on the right shows the project structure with folders *src*, *client*, and *include*. The *client* folder contains *COBSRVI*, *ORDERCLT*, and *include*. The *include* folder contains *ERXCOMM* and *ORDERCLT*. The *ORDERCLT* file is highlighted in the Package Explorer. The main editor shows the COBOL code for *ORDERCLT*, which includes a copybook reference. The copybook is named *ORDERCLT* and is located in the *include* folder. The code also includes a table of constraints for the data fields.

```

CBL LIB QUOTE
IDENTIFICATION DIVISION.
PROGRAM-ID. ORDERCLT.

*****
*
* COBOL RPC Client Interface Object
*
* IDL L: ORDERCLT
* IDL P: *****
*
* Program: COBOL RPC Interface Copybook
* Target:
* String: IDL Library = OrderManagement
* Inter: IDL Program = placeOrder
* RPC C:
* RPC P:
* Copybook Name = ORDERCLT
* Target Platform = all
* Interface Type = all
* String Literal = QUOTE
*
* In the
* ei
*
* Template Source = cobol_include.tpl $Revision: 1.55 $
*
* or
*
* Template Source = cobol_linkage.tpl
* 03 REQUEST.
* 04 ORDERITEMID PIC 9(5).
* 04 ORDERQUANTITY PIC 9(10).
* 04 ORDERCUSTOMERID PIC S9(9) BINARY.
* 03 RESPONSE.
* 04 ORDERITEMNAME PIC X(99).
* 04 ORDERUNITPRICE PIC X(20).
* 04 ORDERTOTAL PIC X(20).
* 04 ORDERCUSTOMERADDRESS.
* 05 NAME PIC X(99).
* 05 STREET PIC X(99).
* 05 CITY PIC X(20).
* 05 ZIPCODE PIC X(20).
* 05 COUNTRYCODE PIC X(256).
* 04 ORDERSHIPPINGDATE PIC X(256).
* 04 ORDERCONFIRMATION PIC X(256).
* 04 ORDERCONFIRMATIONFLAG PIC X.

```

Constraints available

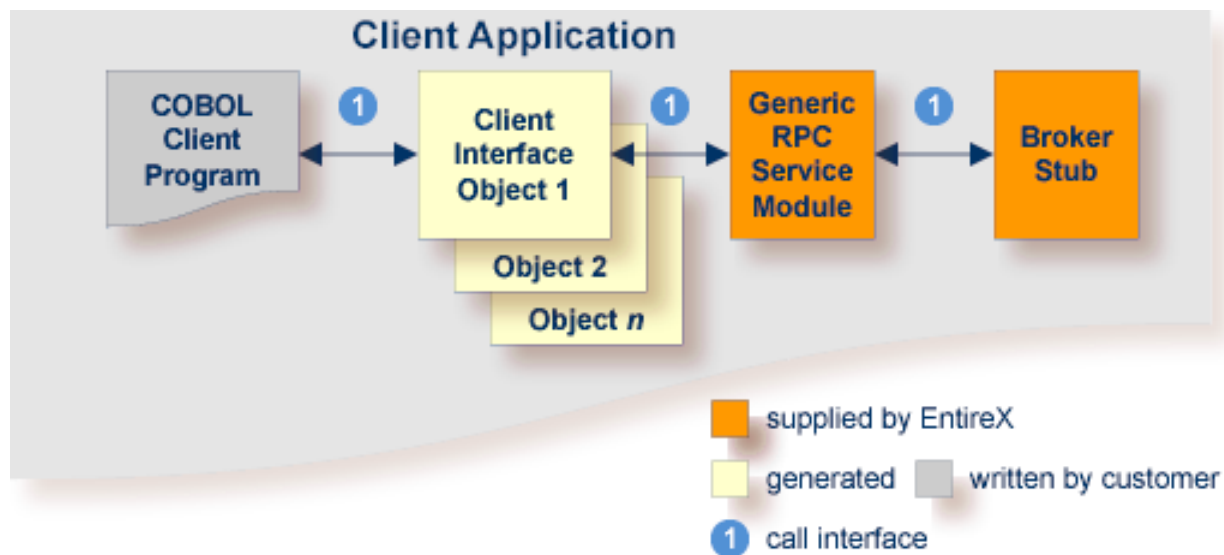
Field	Constraint
04 ORDERITEMID	PIC 9(5).
04 ORDERQUANTITY	PIC 9(10).
04 ORDERCUSTOMERID	PIC S9(9) BINARY.
04 ORDERITEMNAME	PIC X(99).
04 ORDERUNITPRICE	PIC X(20).
04 ORDERTOTAL	PIC X(20).
04 ORDERCUSTOMERADDRESS	
05 NAME	PIC X(99).
05 STREET	PIC X(99).
05 CITY	PIC X(20).
05 ZIPCODE	PIC X(20).
05 COUNTRYCODE	PIC X(256).
04 ORDERSHIPPINGDATE	PIC X(256).
04 ORDERCONFIRMATION	PIC X(256).
04 ORDERCONFIRMATIONFLAG	PIC X.

No constraints available

The copybook is the COBOL programmer's API. It is for use in the COBOL application.

As you would expect, the length of generated COBOL parameters exactly matches the lengths extracted. COBOL parameters always have a fixed length defined at compile time. They are not altered at runtime to reflect actual string length as in modern programming languages. This means that if you choose your default length (see [Extracting the Interface of the webMethods IS Service](#)) you have to choose the maximum possible data size to ensure all data can be transmitted. However, this may needlessly increase the message size if no constraints are set for fields of shorter data length. Choose your defaults and constraints carefully.

Building the Client Application



In this scenario, the COBOL client program, every generated client interface object, generic RPC services module and the broker stub are linked together to the client application.

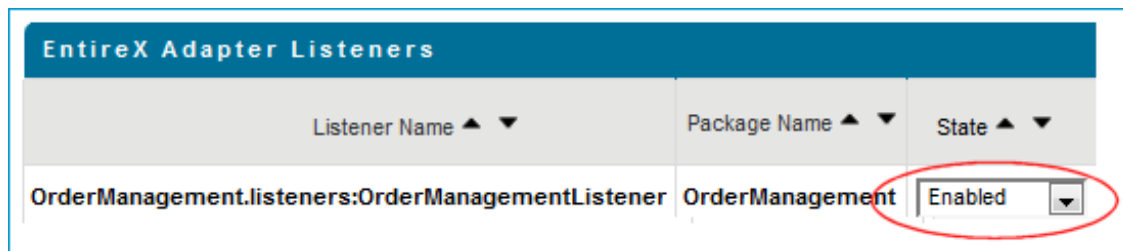
See *Using the COBOL Wrapper for Batch (z/OS, BS2000, z/VSE and IBM i)*.

Task 3: Execute the Call from COBOL to REST

Use the EntireX IDL Tester to execute the call.

➤ To execute the call

- 1 For EntireX RPC (see [Task 3](#) in the [Introduction](#)) and an RPC Listener Connection, make sure EntireX Broker is running.
- 2 Enable the EntireX Adapter Listener.



- 3 Activate the EntireX perspective by choosing **Window > Perspective > Open Perspective > EntireX**, and from the context menu of the IDL file, choose **IDL Tester**.

- 4 This step depends on the method used to call the server (see [Task 3](#) in the [Introduction](#)).
 - For *EntireX RPC*, enter the same TCP/IP address for **Broker** that you supplied when you created the RPC Listener Connection to EntireX (localhost:1971).
 - For *Direct RPC*, that is, you generated a Direct RPC Listener Connection, enter the TCP/IP address from the Integration Server (localhost:1971).
- 5 For **Server**, enter the values you specified for the server address (RPC/SRV1/CALLNAT).

The screenshot shows a REST client interface. At the top, there are two input fields: 'Broker: localhost:1971' and 'Server: RPC/SRV1/CALLNAT'. Both fields are circled in red. Below these fields are five buttons: 'Call' (highlighted with a blue border), 'Ping', 'Open Conversation', 'Reset', and 'Exit'. Below the buttons is a 'Messages' section with a scrollable list of data. The data is organized into two columns: a left column with field names and data types, and a right column with the corresponding values.

2 OrderItemID (N5)	45
2 OrderQuantity (N10)	18
2 OrderCustomerID (I4)	234
2 OrderItemName (A99)	tennis racquet
2 OrderUnitPrice (A20)	79.89 GBP
2 OrderTotal (A20)	1438.02 GBP
3 Name (A99)	The All England Lawn Tennis Club (Championships) Limited
3 Street (A99)	Church Road
3 City (A99)	Wimbledon London
3 ZipCode (A20)	SW19 5AE
3 CountryCode (A20)	UK
2 OrderShippingDate (A256)	Your order will be shipped on Wed, Jul 27, 2016.
2 OrderConfirmation (A256)	Your order is confirmed.
2 OrderConfirmationFlag (L)	true

5

Calling REST from COBOL under z/OS IMS

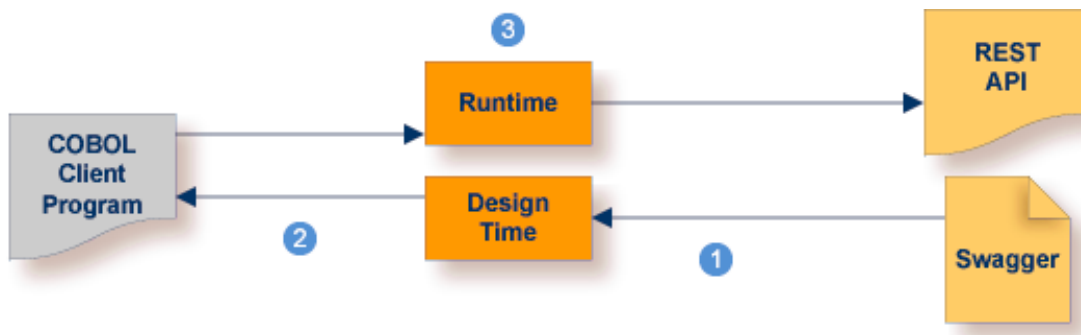
■ Introduction	40
■ What do I need to Install for this Scenario?	41
■ Task 1: Create the Consumer REST API Descriptor, Connections and Listeners	41
■ Task 2: Generate Client Interface Objects and Build Client Application	48
■ Task 3: Execute the Call from COBOL to REST	52

This scenario uses the following tools of the Designer:

- from the **Service Development** perspective: New REST API Descriptor
- from the **EntireX** perspective: *IDL Extractor for Integration Server* and *COBOL Wrapper*

Introduction

To call the REST API from COBOL, take an existing description of a REST API **1** and generate the integration logic **2** to call it from a COBOL application **3**, as shown below.



- 1** Create the consumer REST API descriptor in Integration Server. See *Service Development Help* in the webMethods Integration Server documentation. Then generate Integration Server connections and listeners. See *Using the IDL Extractor for Integration Server*.
- 2** Generate client interface objects and build COBOL client application.
- 3** Execute the call from the COBOL client to the REST API.

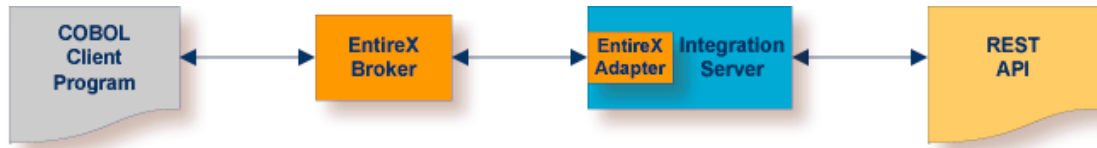
This scenario makes the following important assumptions:

For Task 1:

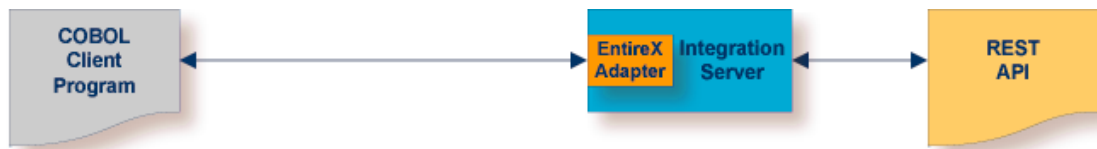
- You have a working Integration Server Service that you want to call from your REST API.

For Task 3:

- You can call the REST API at runtime using different methods:
 - For the EntireX RPC connection method you need EntireX Broker on one of the supported platforms: z/OS | Linux | Windows | BS2000.



- For the EntireX Direct RPC connection method there are no additional prerequisites.



What do I need to Install for this Scenario?

- You have Software AG Designer installed with EntireX plugins and Service Development plugins.
 - For EntireX plugins, see *Designer > EntireX Designer* under *EntireX Installation Packages* in the General Installation documentation.
 - For Service Development plugins, refer to the Integration Server documentation.
- You have an Integration Server with EntireX Adapter installed. See *Adapters > EntireX Adapter* under *EntireX Installation Packages* in the General Installation documentation.
- Optionally, for RPC Connection method (Task 3), you have an EntireX Broker installed.
 - For Linux and Windows, see *EntireX Broker* under *EntireX Installation Packages* in the General Installation documentation.
 - For z/OS, see *Installing EntireX Broker under z/OS* in the z/OS Installation documentation
- On the machine where the COBOL client program is executed, you have the Broker stubs installed. See *Step 3: Install the Broker Stubs and Bind the Broker Stub Executables* in the z/OS Installation documentation.

Task 1: Create the Consumer REST API Descriptor, Connections and Listeners

This section covers the following topics:

- [Introduction](#)
- [Creating a Consumer REST API Descriptor](#)
- [Extracting the Interface of the webMethods IS Service](#)

- [Creating Listener Objects in Integration Server](#)
- [Testing the Extraction Results \(Optional\)](#)

Introduction

After you have created the consumer REST API descriptor, follow the instructions for extracting a webMethods Integration Server (IS) service under *Using the IDL Extractor for Integration Server*.

This process creates the following EntireX metafiles:

- an IDL file, containing definitions of the interface between client and server; see *Software AG IDL File* in the IDL Editor documentation in the IDL Editor documentation
- a webMethods IS Connection
- a webMethods IS Listener, depending on connection type. See *Listeners* in the EntireX Adapter documentation.

Connection types are described under [Creating Listener Objects in Integration Server](#).

Creating a Consumer REST API Descriptor

➤ To create a consumer REST API Descriptor

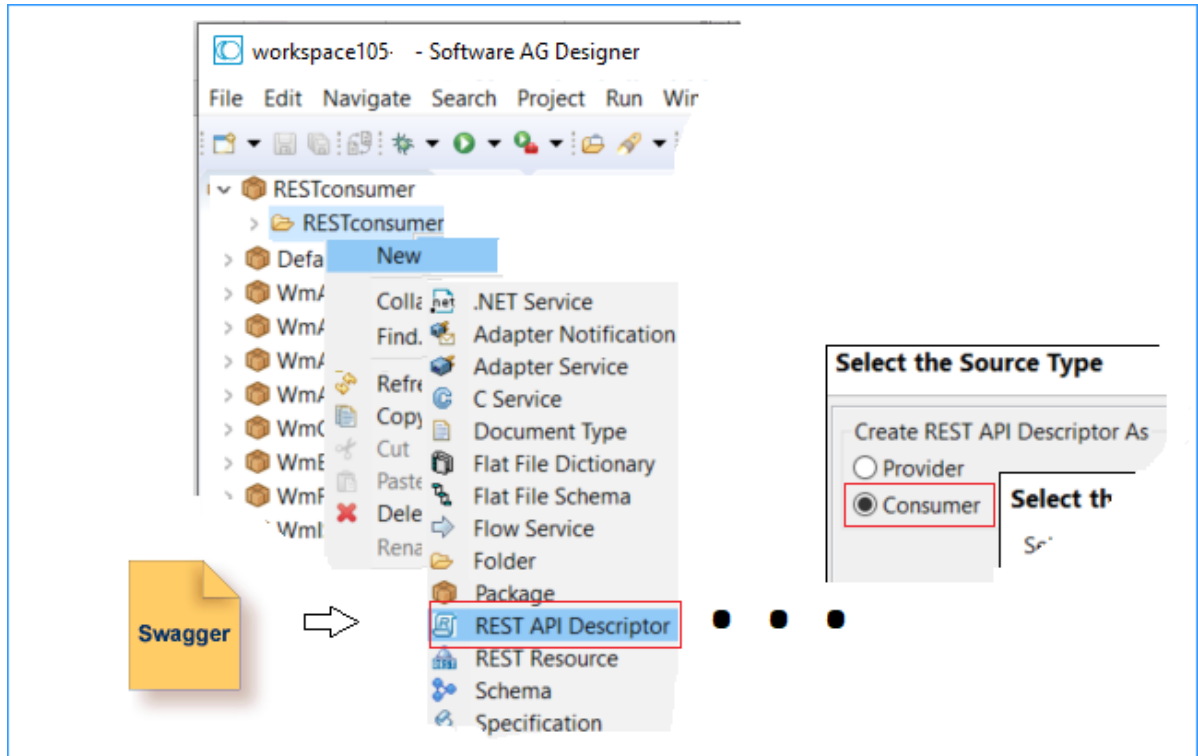
- 1 Switch to the **Service Development** perspective. Create a package and folder in Designer where the API descriptor is placed into.
- 2 Invoke the wizard to create a new REST API Descriptor in Eclipse. From the **File** menu, choose **New > REST API Descriptor**.

Or:

Choose **New** from the toolbar or context menu of a view showing resources.

Or:

Press **Ctrl-N** to start the selection of the New Wizards.



- 3 Follow the instructions on the screen. Provide the description of the REST API (Swagger file) and create the REST API descriptor for a **Consumer**.

For more information on REST API descriptors see *Service Development Help* in the webMethods Integration Server documentation.

Extracting the Interface of the webMethods IS Service

First create a project in Designer. This project is the container into which all extracted and generated EntireX artifacts are placed. Name this project *COBOL placeOrder client*.

» To start the IDL Extractor for Integration Server

- 1 Switch to the EntireX perspective.
- 2 Invoke the IDL Extractor for webMethods IS, which is a New Wizard in Eclipse. From the **File** menu, choose **New**. Select IDL Extractor for webMethods IS in the following page.

Or:

Choose **New** from the toolbar or context menu of a view showing resources.

Or:

Press **Ctrl-N** to start the selection of the New Wizards.

- 3 If you are using the wizard for the first time without any predefined Integration Server connections, enter the TCP/IP address of the webMethods IS to extract from. Otherwise select the connection to the Integration Server.
- 4 Press **Next**.

➤ **To extract the IDL from the selected package**

- 1 Under **Source (Integration Server)**, select the package you want to extract from.

Extract IDL from the selected package

Select the package (source), specify the container and file name (target) and choose the mapping.

Source (Integration Server)
 Packages on Integration Server localhost:5555

- OrderManagement
- WmEntireX
- WmFlatFile
- WmIExtDC

Target (Eclipse Workspace)
 Container: COBOL placeOrder client Browse...
 IDL File Name: OrderManagement

Optimize extracted IDL for usage with: COBOL

Map Integration Server data type to IDL

- ☐ alphanumeric with variable length
- ☒ alphanumeric with fixed length of 256

☒ Create Listener Objects in Integration Server

? < Back Next > Finish Cancel

- 2 For **Target (Eclipse Workspace)**, specify project *COBOL placeOrder client*. Extraction results will be placed in this container. You can keep the **IDL File Name** as it has been derived from the package name by default. Since COBOL is your desired endpoint, select **Optimize extracted IDL for usage with: COBOL**.



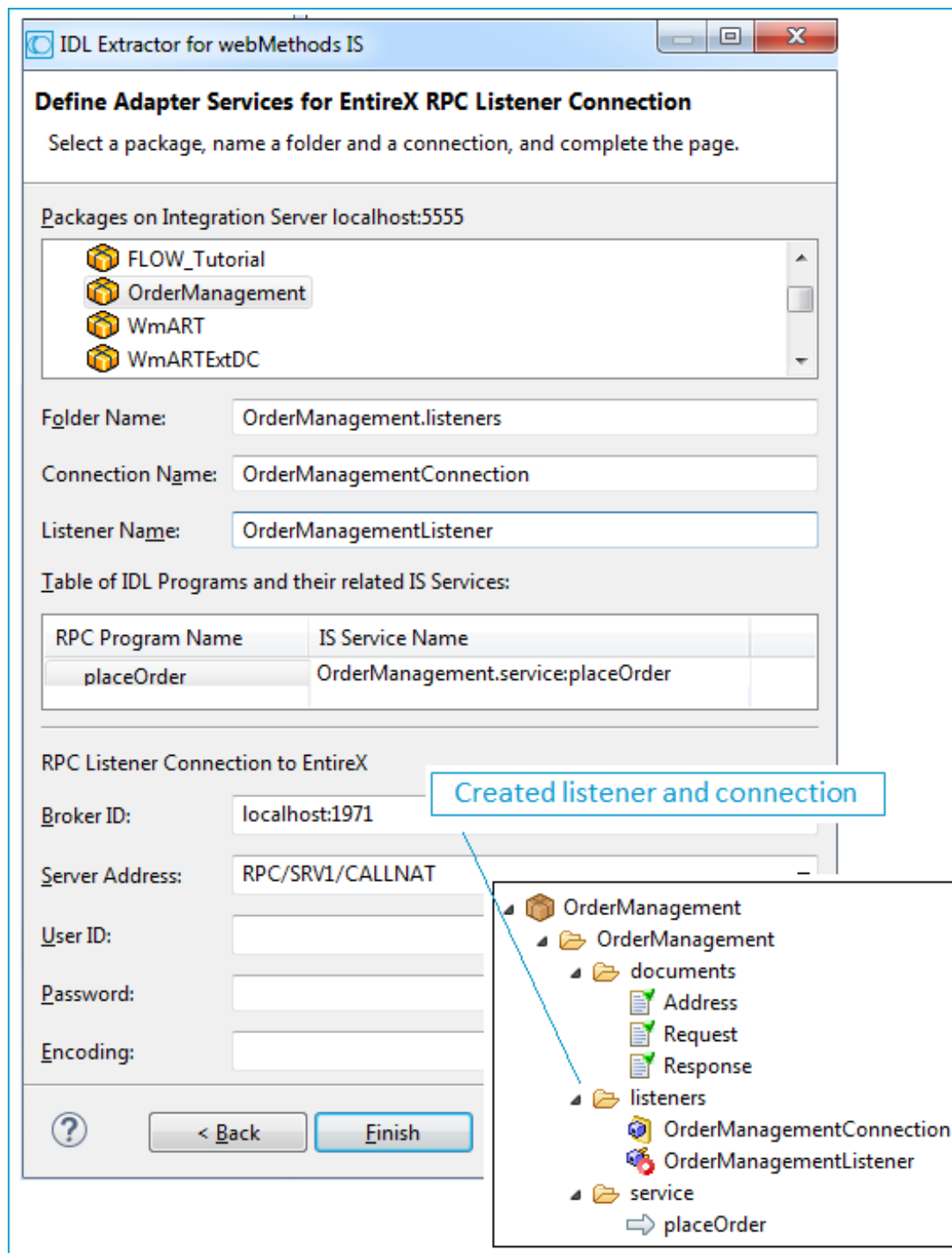
Note: COBOL only supports fixed-length data items, whereas webMethods IS has variable length types, often without any length restrictions (constraints). Parameters with no specified constraints will take the default fixed length. You can specify this default in field alphanumeric with fixed length. .

- 3 A listener is required for calling webMethods IS from an RPC client, so check **Create Listener Objects in Integration Server**.
- 4 Press **Next**.

Creating Listener Objects in Integration Server

➤ To create Listener Objects in Integration Server

- 1 Select a Listener connection type used for inbound connection to the Integration Server.
 - An EntireX RPC Listener connection is the standard way and is always available.
 - If your interface contains only In parameters, an EntireX Reliable RPC Listener connection is available.
 - If it is enabled by the license for the webMethods Integration Server:
 - an EntireX Direct RPC Listener connection is available
 - if your interfaces contain only In parameters, an EntireX Direct Reliable RPC Listener connection is available



- On this page, you set the properties for the listener objects to create; these objects will wait for the incoming COBOL requests. Select the package and specify the folder name into which the listener is generated. If the folder does not exist, it will be created automatically.

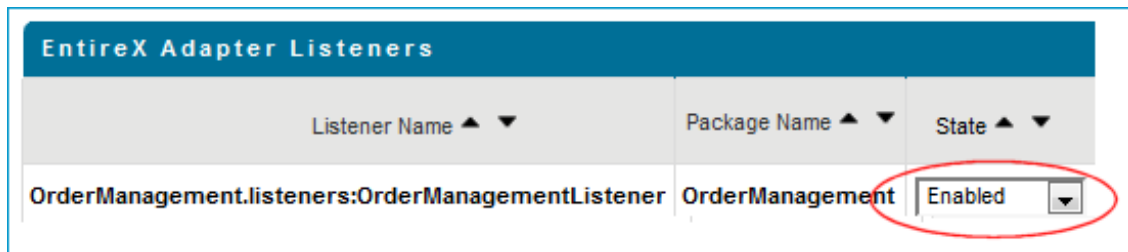
- 3 Keep the defaults given for **RPC Listener Connection to EntireX**. The **Broker ID** is a TCP/IP or SSL/TLS address and **Server Address** is an EntireX-specific namespace to locate a target server (here: *OrderManagementListener*).
- 4 When creating a connection, a package dependency is added such that the selected package depends on webMethods EntireX (the package WmEntireX) with the version currently used.
- 5 Press **Finish**. The listener and the connection appear in the package navigator. For the package navigator, switch to the service development perspective by choosing **Window > Perspective > Open Perspective > Service Development**.

Testing the Extraction Results (Optional)

You can test the results of the extraction and the server back end, using the EntireX IDL Tester.

➤ To test the extraction results

- 1 For EntireX RPC (see [Task 3](#) in the [Introduction](#)) and an RPC Listener Connection, make sure EntireX Broker is running.
- 2 Enable the EntireX Adapter Listener.



- 3 Activate the EntireX perspective by choosing **Window > Perspective > Open Perspective > EntireX**, and from the context menu of the IDL file, choose **IDL Tester**.
- 4 This step depends on the method used to call the server (see [Task 3](#) in the [Introduction](#)).
 - For *EntireX RPC*, enter the same TCP/IP address for **Broker** that you supplied when you created the RPC Listener Connection to EntireX (localhost:1971).
 - For *Direct RPC*, that is, you generated a Direct RPC Listener Connection, enter the TCP/IP address from the Integration Server (localhost:1971).
- 5 For **Server**, enter the values you specified for the server address (RPC/SRV1/CALLNAT).

The screenshot shows a graphical user interface for a COBOL application. At the top, there are two text input fields: "Broker:" with the value "localhost:1971" and "Server:" with the value "RPC/SRV1/CALLNAT". Both fields are circled in red. Below these fields are five buttons: "Call" (highlighted with a dashed border), "Ping", "Open Conversation", "Reset", and "Exit". Below the buttons is a "Messages" section with a scrollable text area. Below the messages is a table displaying the response data.

2 OrderItemID (N5)	45
2 OrderQuantity (N10)	18
2 OrderCustomerID (I4)	234
2 OrderItemName (A99)	tennis racquet
2 OrderUnitPrice (A20)	79.89 GBP
2 OrderTotal (A20)	1438.02 GBP
3 Name (A99)	The All England Lawn Tennis Club (Championships) Limited
3 Street (A99)	Church Road
3 City (A99)	Wimbledon London
3 ZipCode (A20)	SW 19 5AE
3 CountryCode (A20)	UK
2 OrderShippingDate (A256)	Your order will be shipped on Wed, Jul 27, 2016.
2 OrderConfirmation (A256)	Your order is confirmed.
2 OrderConfirmationFlag (L)	true

Task 2: Generate Client Interface Objects and Build Client Application

This section covers the following topics:

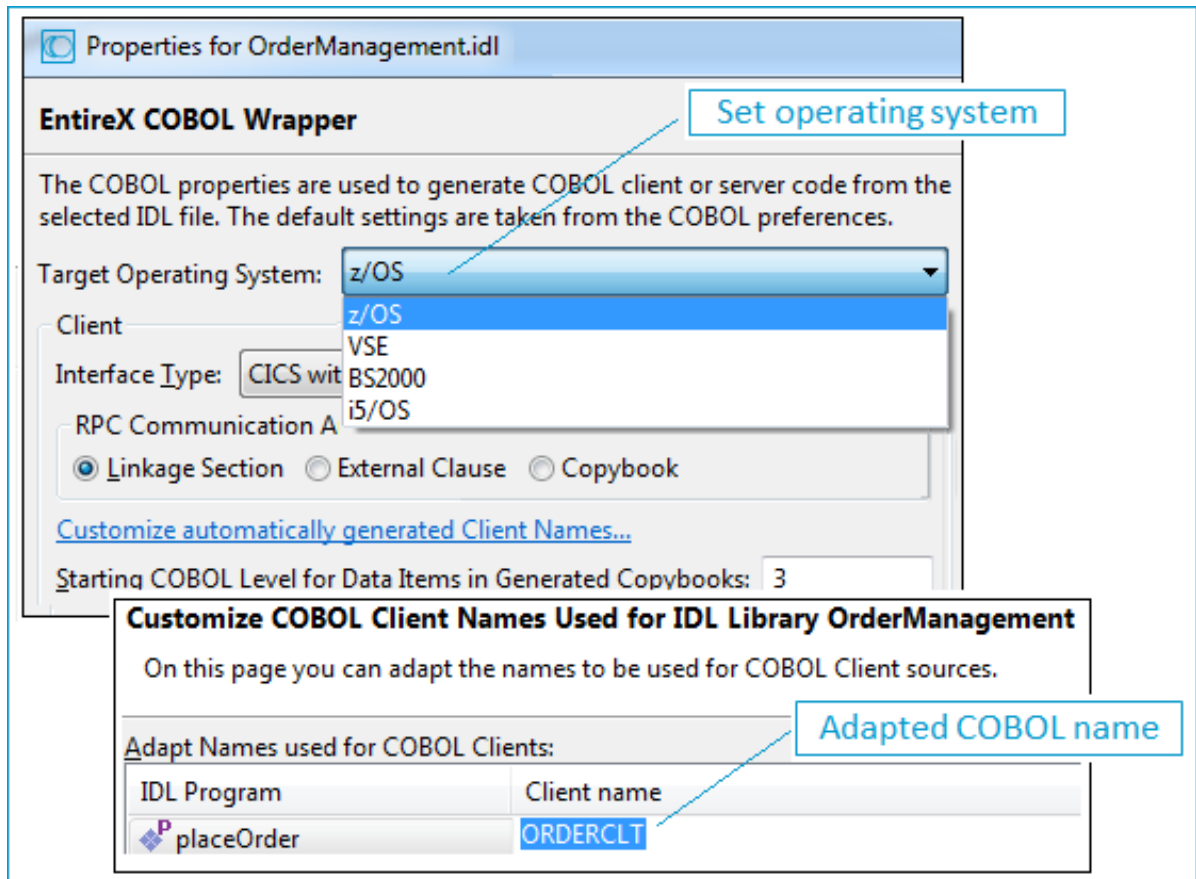
- [Setting Generation Options \(Properties\) for the COBOL Wrapper](#)
- [Generating COBOL Code](#)

■ Building the Client Application

Setting Generation Options (Properties) for the COBOL Wrapper

➤ To set generation options (properties) for the COBOL Wrapper

- 1 To generate suitable COBOL code for your environment, first set the correct target operating system. Make sure the EntireX perspective is active (**Window > Perspective > Open Perspective > EntireX**) and call the properties of the COBOL Wrapper.



- 2 From the context menu of the IDL file, choose **Properties > EntireX COBOL Wrapper**. On the COBOL Wrapper **Properties** page, set the **Target Operating System**
- 3 Adapt the name of the generated COBOL program: Choose **Customize automatically generated Client Names**. Take ORDERCLT as the COBOL program's name.

Generating COBOL Code

➤ To generate COBOL code

- For COBOL code generation, make sure the EntireX Perspective is active: Choose **Window > Perspective > Open Perspective > EntireX**. From context menu of the IDL file, choose **COBOL > Generate RPC Client**. This generates the COBOL client *ORDERCLT* in folder *client* and a copybook in folder *include*:

The screenshot shows the Eclipse IDE with the 'EntireX' perspective. The main editor displays the COBOL client code for 'ORDERCLT'. The code is structured as follows:

```

CBL LIB QUOTE
IDENTIFICATION DIVISION.
PROGRAM-ID. ORDERCLT.

*****
*
* COBOL RPC Client Interface Object
*
* IDL L: ORDERCLT
* IDL P: *****
*
* Program: COBOL RPC Interface Copybook
* Target:
* String: IDL Library = OrderManagement
* Inter: IDL Program = placeOrder
* RPC C:
* RPC P: Copybook Name = ORDERCLT
* Target Platform = all
* Interface Type = all
* String Literal = QUOTE
*
* In the
* ei
*
* Template Source = cobol_include.tpl $Revision: 1.55 $
*
* or
*
* Template Source = cobol_linkage.tpl
* 03 REQUEST.
* 04 ORDERITEMID PIC 9(5).
* 04 ORDERQUANTITY PIC 9(10).
* 04 ORDERCUSTOMERID PIC S9(9) BINARY.
* 03 RESPONSE.
* 04 ORDERITEMNAME PIC X(99).
* 04 ORDERUNITPRICE PIC X(20).
* 04 ORDERTOTAL PIC X(20).
* 04 ORDERCUSTOMERADDRESS.
* 05 NAME
* 05 STREET PIC X(99).
* 05 CITY PIC X(99).
* 05 ZIPCODE PIC X(20).
* 05 COUNTRYCODE PIC X(20).
* 04 ORDERSHIPPINGDATE PIC X(256).
* 04 ORDERCONFIRMATION PIC X(256).
* 04 ORDERCONFIRMATIONFLAG PIC X.

```

The Package Explorer on the right shows the project structure:

- COBOL placeOrder client
 - src
 - JRE System Library [JavaSE-1.8]
 - client
 - COBSRVI
 - ORDERCLT
 - include
 - ERXCOMM
 - ORDERCLT
 - OrderManagement.idl

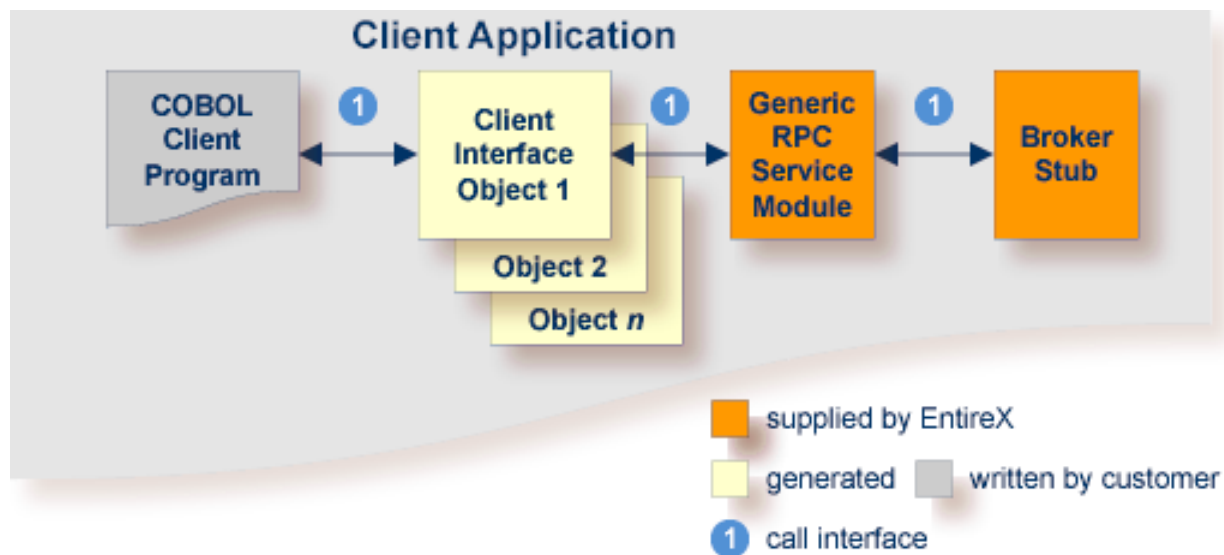
Callouts in the image highlight the following elements:

- COBOL client**: Points to the 'COBOL RPC Client Interface Object' section.
- Copybook**: Points to the 'Copybook Name = ORDERCLT' line.
- Constraints available**: Points to the '04 ORDERUNITPRICE PIC X(20)' line.
- No constraints available**: Points to the '04 ORDERSHIPPINGDATE PIC X(256)' line.

The copybook is the COBOL programmer's API. It is for use in the COBOL application.

As you would expect, the length of generated COBOL parameters exactly matches the lengths extracted. COBOL parameters always have a fixed length defined at compile time. They are not altered at runtime to reflect actual string length as in modern programming languages. This means that if you choose your default length (see [Extracting the Interface of the webMethods IS Service](#)) you have to choose the maximum possible data size to ensure all data can be transmitted. However, this may needlessly increase the message size if no constraints are set for fields of shorter data length. Choose your defaults and constraints carefully.

Building the Client Application



In this scenario, the COBOL client program, every generated client interface object, generic RPC services module and the broker stub are linked together to the client application.

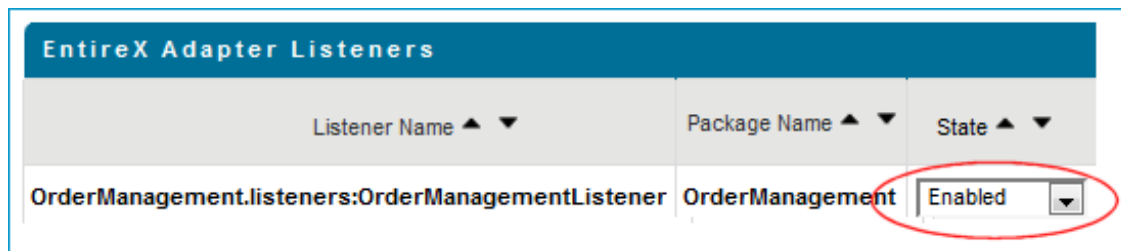
See *Using the COBOL Wrapper for IMS (z/OS)*.

Task 3: Execute the Call from COBOL to REST

Use the EntireX IDL Tester to execute the call.

➤ To execute the call

- 1 For EntireX RPC (see [Task 3](#) in the [Introduction](#)) and an RPC Listener Connection, make sure EntireX Broker is running.
- 2 Enable the EntireX Adapter Listener.



- 3 Activate the EntireX perspective by choosing **Window > Perspective > Open Perspective > EntireX**, and from the context menu of the IDL file, choose **IDL Tester**.

- 4 This step depends on the method used to call the server (see [Task 3](#) in the [Introduction](#)).
 - For *EntireX RPC*, enter the same TCP/IP address for **Broker** that you supplied when you created the RPC Listener Connection to EntireX (localhost:1971).
 - For *Direct RPC*, that is, you generated a Direct RPC Listener Connection, enter the TCP/IP address from the Integration Server (localhost:1971).
- 5 For **Server**, enter the values you specified for the server address (RPC/SRV1/CALLNAT).

The screenshot shows a REST client interface. At the top, there are two input fields: 'Broker: localhost:1971' and 'Server: RPC/SRV1/CALLNAT'. Both fields are circled in red. Below these fields are five buttons: 'Call' (highlighted with a blue border), 'Ping', 'Open Conversation', 'Reset', and 'Exit'. Below the buttons is a 'Messages' section with a scrollable list of data. The data is organized into two columns: a left column with field names and data types, and a right column with the corresponding values.

2 OrderItemID (N5)	45
2 OrderQuantity (N10)	18
2 OrderCustomerID (I4)	234
2 OrderItemName (A99)	tennis racquet
2 OrderUnitPrice (A20)	79.89 GBP
2 OrderTotal (A20)	1438.02 GBP
3 Name (A99)	The All England Lawn Tennis Club (Championships) Limited
3 Street (A99)	Church Road
3 City (A99)	Wimbledon London
3 ZipCode (A20)	SW19 5AE
3 CountryCode (A20)	UK
2 OrderShippingDate (A256)	Your order will be shipped on Wed, Jul 27, 2016.
2 OrderConfirmation (A256)	Your order is confirmed.
2 OrderConfirmationFlag (L)	true

6

Calling REST from COBOL under z/OS IDMS/DC

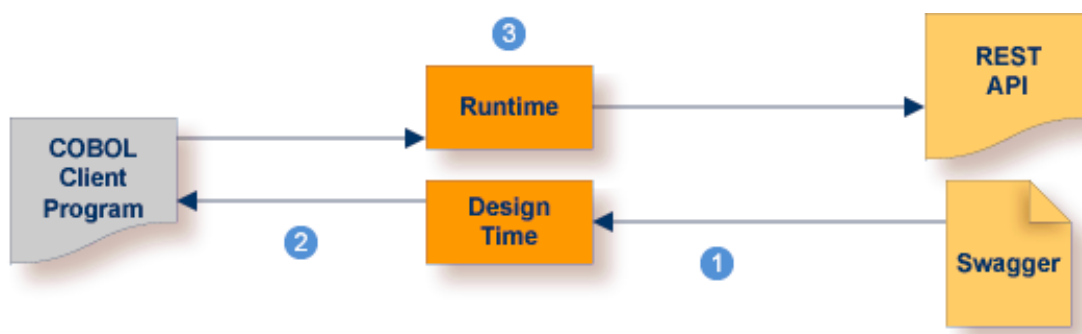
■ Introduction	56
■ What do I need to Install for this Scenario?	57
■ Task 1: Create the Consumer REST API Descriptor, Connections and Listeners	57
■ Task 2: Generate Client Interface Objects and Build Client Application	64
■ Task 3: Execute the Call from COBOL to REST	68

This scenario uses the following tools of the Designer:

- from the **Service Development** perspective: New REST API Descriptor
- from the **EntireX** perspective: *IDL Extractor for Integration Server* and *COBOL Wrapper*

Introduction

To call the REST API from COBOL, take an existing description of a REST API **1** and generate the integration logic **2** to call it from a COBOL application **3**, as shown below.



- 1** Create the consumer REST API descriptor in Integration Server. See *Service Development Help* in the webMethods Integration Server documentation. Then generate Integration Server connections and listeners. See *Using the IDL Extractor for Integration Server*.
- 2** Generate client interface objects and build COBOL client application.
- 3** Execute the call from the COBOL client to the REST API.

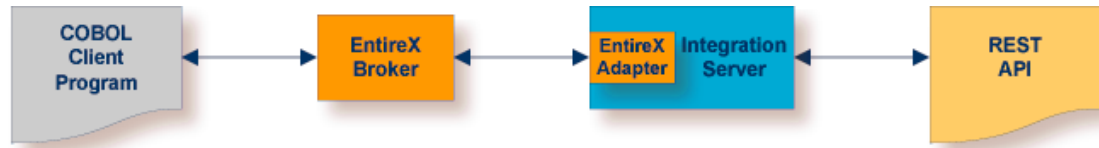
This scenario makes the following important assumptions:

For Task 1:

- You have a working Integration Server Service that you want to call from your REST API.

For Task 3:

- You can call the REST API at runtime using different methods:
 - For the EntireX RPC connection method you need EntireX Broker on one of the supported platforms: z/OS | Linux | Windows | BS2000.



- For the EntireX Direct RPC connection method there are no additional prerequisites.



What do I need to Install for this Scenario?

- You have Software AG Designer installed with EntireX plugins and Service Development plugins.
 - For EntireX plugins, see *Designer > EntireX Designer* under *EntireX Installation Packages* in the General Installation documentation.
 - For Service Development plugins, refer to the Integration Server documentation.
- You have an Integration Server with EntireX Adapter installed. See *Adapters > EntireX Adapter* under *EntireX Installation Packages* in the General Installation documentation.
- Optionally, for RPC Connection method (Task 3), you have an EntireX Broker installed.
 - For Linux and Windows, see *EntireX Broker* under *EntireX Installation Packages* in the General Installation documentation.
 - For z/OS, see *Installing EntireX Broker under z/OS* in the z/OS Installation documentation
- On the machine where the COBOL client program is executed, you have the Broker stubs installed. See *Step 3: Install the Broker Stubs and Bind the Broker Stub Executables* in the z/OS Installation documentation.

Task 1: Create the Consumer REST API Descriptor, Connections and Listeners

This section covers the following topics:

- [Introduction](#)
- [Creating a Consumer REST API Descriptor](#)
- [Extracting the Interface of the webMethods IS Service](#)

- [Creating Listener Objects in Integration Server](#)
- [Testing the Extraction Results \(Optional\)](#)

Introduction

After you have created the consumer REST API descriptor, follow the instructions for extracting a webMethods Integration Server (IS) service under *Using the IDL Extractor for Integration Server*.

This process creates the following EntireX metafiles:

- an IDL file, containing definitions of the interface between client and server; see *Software AG IDL File* in the IDL Editor documentation in the IDL Editor documentation
- a webMethods IS Connection
- a webMethods IS Listener, depending on connection type. See *Listeners* in the EntireX Adapter documentation.

Connection types are described under [Creating Listener Objects in Integration Server](#).

Creating a Consumer REST API Descriptor

➤ To create a consumer REST API Descriptor

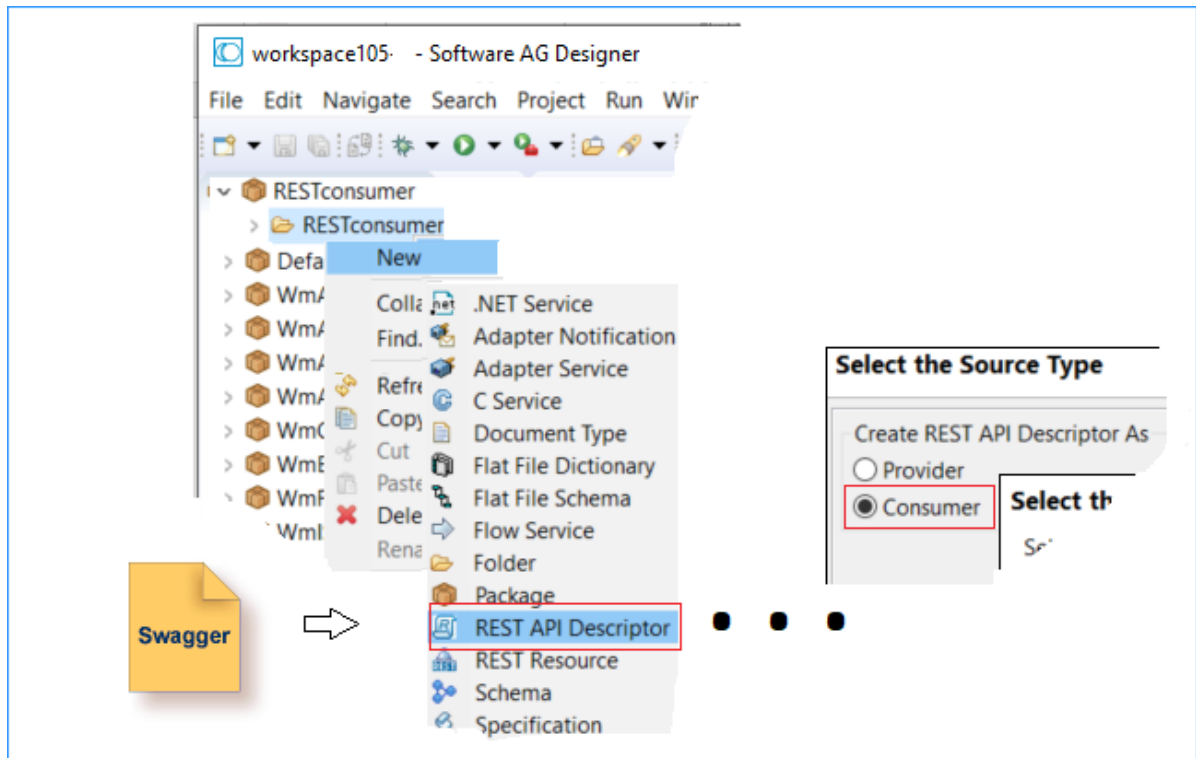
- 1 Switch to the **Service Development** perspective. Create a package and folder in Designer where the API descriptor is placed into.
- 2 Invoke the wizard to create a new REST API Descriptor in Eclipse. From the **File** menu, choose **New > REST API Descriptor**.

Or:

Choose **New** from the toolbar or context menu of a view showing resources.

Or:

Press **Ctrl-N** to start the selection of the New Wizards.



- 3 Follow the instructions on the screen. Provide the description of the REST API (Swagger file) and create the REST API descriptor for a **Consumer**.

For more information on REST API descriptors see *Service Development Help* in the webMethods Integration Server documentation.

Extracting the Interface of the webMethods IS Service

First create a project in Designer. This project is the container into which all extracted and generated EntireX artifacts are placed. Name this project *COBOL placeOrder client*.

» To start the IDL Extractor for Integration Server

- 1 Switch to the EntireX perspective.
- 2 Invoke the IDL Extractor for webMethods IS, which is a New Wizard in Eclipse. From the **File** menu, choose **New**. Select IDL Extractor for webMethods IS in the following page.

Or:

Choose **New** from the toolbar or context menu of a view showing resources.

Or:

Press **Ctrl-N** to start the selection of the New Wizards.

- 3 If you are using the wizard for the first time without any predefined Integration Server connections, enter the TCP/IP address of the webMethods IS to extract from. Otherwise select the connection to the Integration Server.
- 4 Press **Next**.

➤ **To extract the IDL from the selected package**

- 1 Under **Source (Integration Server)**, select the package you want to extract from.

Extract IDL from the selected package

Select the package (source), specify the container and file name (target) and choose the mapping.

Source (Integration Server)
 Packages on Integration Server localhost:5555

- OrderManagement
- WmEntireX
- WmFlatFile
- WmIExtDC

Target (Eclipse Workspace)
 Container: COBOL placeOrder client Browse...
 IDL File Name: OrderManagement

Optimize extracted IDL for usage with: COBOL

Map Integration Server data type to IDL

- ☐ alphanumeric with variable length
- ☒ alphanumeric with fixed length of 256

Used if no constraints available

☒ Create Listener Objects in Integration Server

? < Back Next > Finish Cancel

- 2 For **Target (Eclipse Workspace)**, specify project *COBOL placeOrder client*. Extraction results will be placed in this container. You can keep the **IDL File Name** as it has been derived from the package name by default. Since COBOL is your desired endpoint, select **Optimize extracted IDL for usage with: COBOL**.



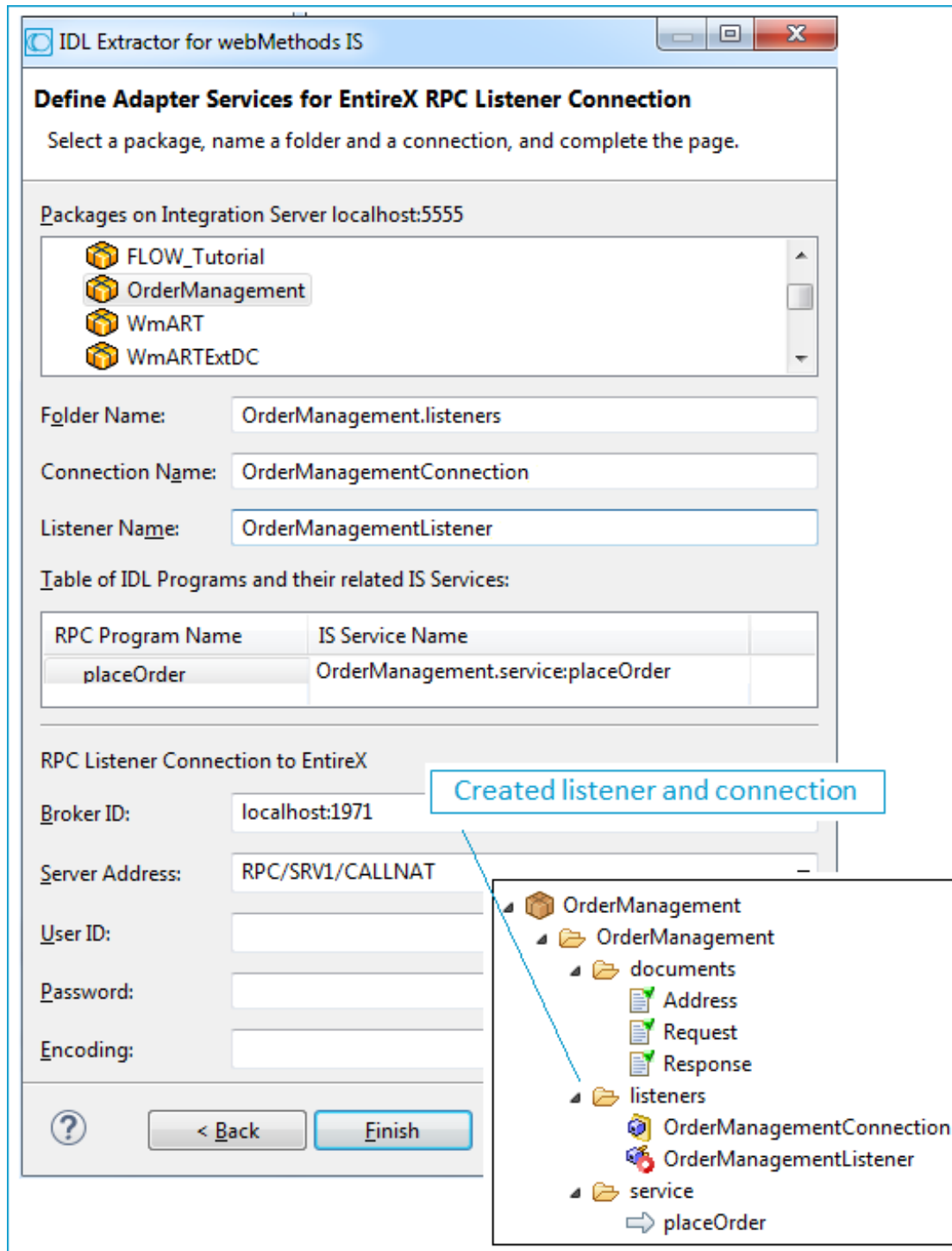
Note: COBOL only supports fixed-length data items, whereas webMethods IS has variable length types, often without any length restrictions (constraints). Parameters with no specified constraints will take the default fixed length. You can specify this default in field alphanumeric with fixed length. .

- 3 A listener is required for calling webMethods IS from an RPC client, so check **Create Listener Objects in Integration Server**.
- 4 Press **Next**.

Creating Listener Objects in Integration Server

➤ To create Listener Objects in Integration Server

- 1 Select a Listener connection type used for inbound connection to the Integration Server.
 - An EntireX RPC Listener connection is the standard way and is always available.
 - If your interface contains only In parameters, an EntireX Reliable RPC Listener connection is available.
 - If it is enabled by the license for the webMethods Integration Server:
 - an EntireX Direct RPC Listener connection is available
 - if your interfaces contain only In parameters, an EntireX Direct Reliable RPC Listener connection is available



- On this page, you set the properties for the listener objects to create; these objects will wait for the incoming COBOL requests. Select the package and specify the folder name into which the listener is generated. If the folder does not exist, it will be created automatically.

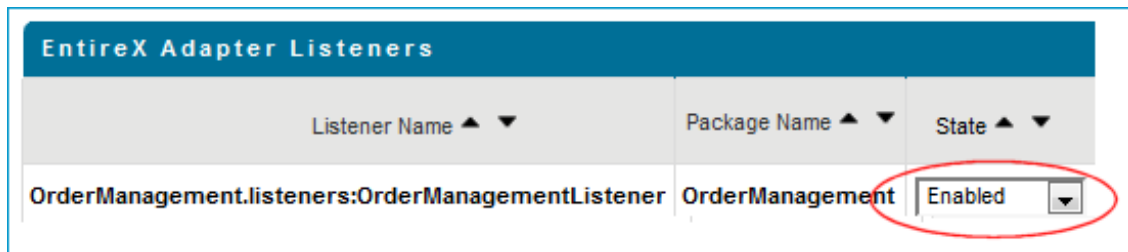
- 3 Keep the defaults given for **RPC Listener Connection to EntireX**. The **Broker ID** is a TCP/IP or SSL/TLS address and **Server Address** is an EntireX-specific namespace to locate a target server (here: *OrderManagementListener*).
- 4 When creating a connection, a package dependency is added such that the selected package depends on webMethods EntireX (the package WmEntireX) with the version currently used.
- 5 Press **Finish**. The listener and the connection appear in the package navigator. For the package navigator, switch to the service development perspective by choosing **Window > Perspective > Open Perspective > Service Development**.

Testing the Extraction Results (Optional)

You can test the results of the extraction and the server back end, using the EntireX IDL Tester.

➤ To test the extraction results

- 1 For EntireX RPC (see [Task 3](#) in the [Introduction](#)) and an RPC Listener Connection, make sure EntireX Broker is running.
- 2 Enable the EntireX Adapter Listener.



- 3 Activate the EntireX perspective by choosing **Window > Perspective > Open Perspective > EntireX**, and from the context menu of the IDL file, choose **IDL Tester**.
- 4 This step depends on the method used to call the server (see [Task 3](#) in the [Introduction](#)).
 - For *EntireX RPC*, enter the same TCP/IP address for **Broker** that you supplied when you created the RPC Listener Connection to EntireX (localhost:1971).
 - For *Direct RPC*, that is, you generated a Direct RPC Listener Connection, enter the TCP/IP address from the Integration Server (localhost:1971).
- 5 For **Server**, enter the values you specified for the server address (RPC/SRV1/CALLNAT).

Broker: localhost:1971 Server: RPC/SRV1/CALLNAT

Call Ping Open Conversation Reset Exit

Messages

2 OrderItemID (N5)	45
2 OrderQuantity (N10)	18
2 OrderCustomerID (I4)	234
2 OrderItemName (A99)	tennis racquet
2 OrderUnitPrice (A20)	79.89 GBP
2 OrderTotal (A20)	1438.02 GBP
3 Name (A99)	The All England Lawn Tennis Club (Championships) Limited
3 Street (A99)	Church Road
3 City (A99)	Wimbledon London
3 ZipCode (A20)	SW 19 5AE
3 CountryCode (A20)	UK
2 OrderShippingDate (A256)	Your order will be shipped on Wed, Jul 27, 2016.
2 OrderConfirmation (A256)	Your order is confirmed.
2 OrderConfirmationFlag (L)	true

Task 2: Generate Client Interface Objects and Build Client Application

This section covers the following topics:

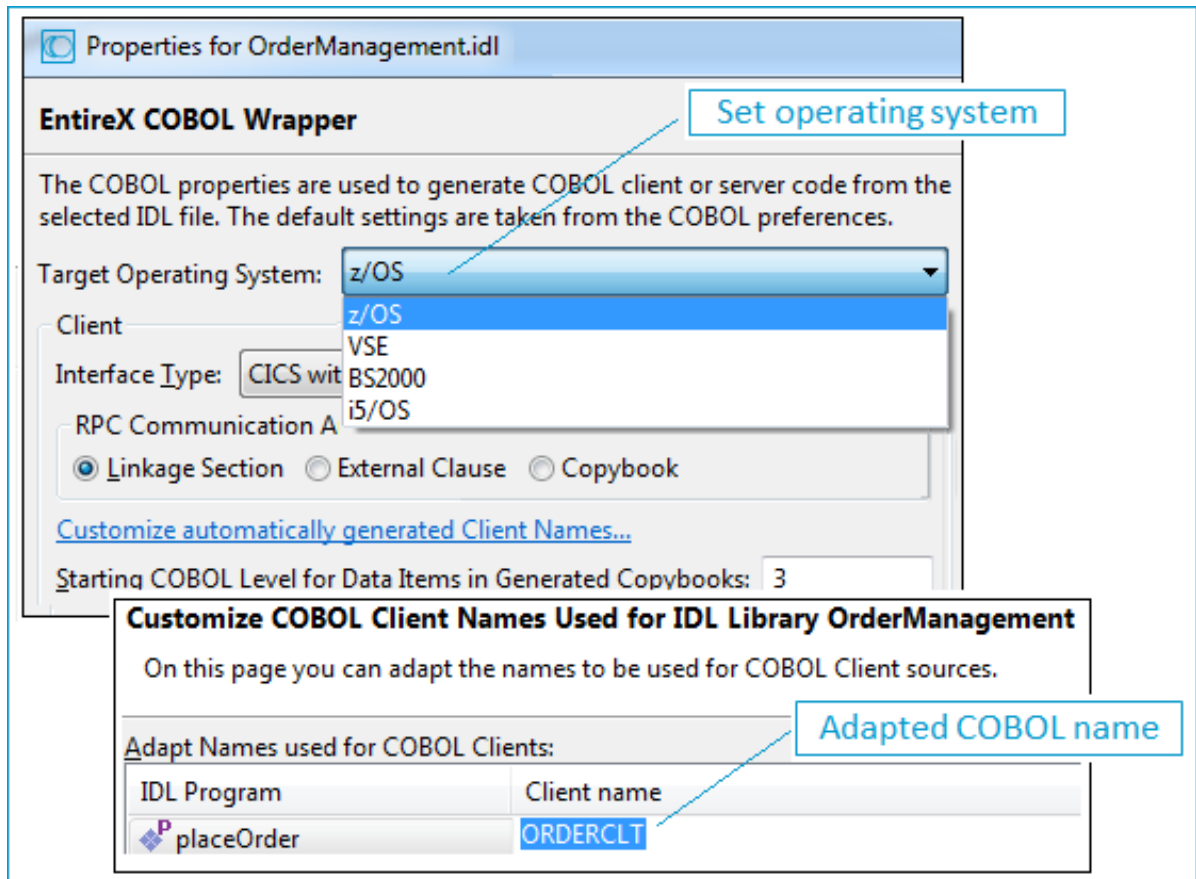
- Setting Generation Options (Properties) for the COBOL Wrapper
- Generating COBOL Code

■ Building the Client Application

Setting Generation Options (Properties) for the COBOL Wrapper

➤ To set generation options (properties) for the COBOL Wrapper

- 1 To generate suitable COBOL code for your environment, first set the correct target operating system. Make sure the EntireX perspective is active (**Window > Perspective > Open Perspective > EntireX**) and call the properties of the COBOL Wrapper.



- 2 From the context menu of the IDL file, choose **Properties > EntireX COBOL Wrapper**. On the COBOL Wrapper **Properties** page, set the **Target Operating System**
- 3 Adapt the name of the generated COBOL program: Choose **Customize automatically generated Client Names**. Take ORDERCLT as the COBOL program's name.

Generating COBOL Code

➤ To generate COBOL code

- For COBOL code generation, make sure the EntireX Perspective is active: Choose **Window > Perspective > Open Perspective > EntireX**. From context menu of the IDL file, choose **COBOL > Generate RPC Client**. This generates the COBOL client *ORDERCLT* in folder *client* and a copybook in folder *include*:

The screenshot shows the Eclipse IDE with the COBOL client *ORDERCLT* generated in the *client* folder and a copybook in the *include* folder. The main editor displays the COBOL code for *ORDERCLT*, which includes a copybook reference. The Package Explorer on the right shows the project structure with folders *src*, *client*, and *include*. The *client* folder contains *COBSRVI*, *ORDERCLT*, and *include*. The *include* folder contains *ERXCOMM* and *ORDERCLT*. The *ORDERCLT* file is highlighted in the Package Explorer. The main editor shows the COBOL code for *ORDERCLT*, which includes a copybook reference. The copybook is named *ORDERCLT* and is located in the *include* folder. The code also includes a table of constraints for the data fields.

```

CBL LIB QUOTE
IDENTIFICATION DIVISION.
PROGRAM-ID. ORDERCLT.

*****
*
* COBOL RPC Client Interface Object
*
* IDL L: ORDERCLT
* IDL P: *****
*
* Program: COBOL RPC Interface Copybook
* Target:
* String: IDL Library = OrderManagement
* Inter: IDL Program = placeOrder
* RPC C:
* RPC P:
* Copybook Name = ORDERCLT
* Target Platform = all
* Interface Type = all
* String Literal = QUOTE
*
* In the
* ei
*
* Template Source = cobol_include.tpl $Revision: 1.55 $
*
* or
*
* Template Source = cobol_linkage.tpl
* 03 REQUEST.
* 04 ORDERITEMID PIC 9(5).
* 04 ORDERQUANTITY PIC 9(10).
* 04 ORDERCUSTOMERID PIC S9(9) BINARY.
* 03 RESPONSE.
* 04 ORDERITEMNAME PIC X(99).
* 04 ORDERUNITPRICE PIC X(20).
* 04 ORDERTOTAL PIC X(20).
* 04 ORDERCUSTOMERADDRESS.
* 05 NAME
* 05 STREET PIC X(99).
* 05 CITY PIC X(99).
* 05 ZIPCODE PIC X(20).
* 05 COUNTRYCODE PIC X(20).
* 04 ORDERSHIPPINGDATE PIC X(256).
* 04 ORDERCONFIRMATION PIC X(256).
* 04 ORDERCONFIRMATIONFLAG PIC X.

```

Constraints available

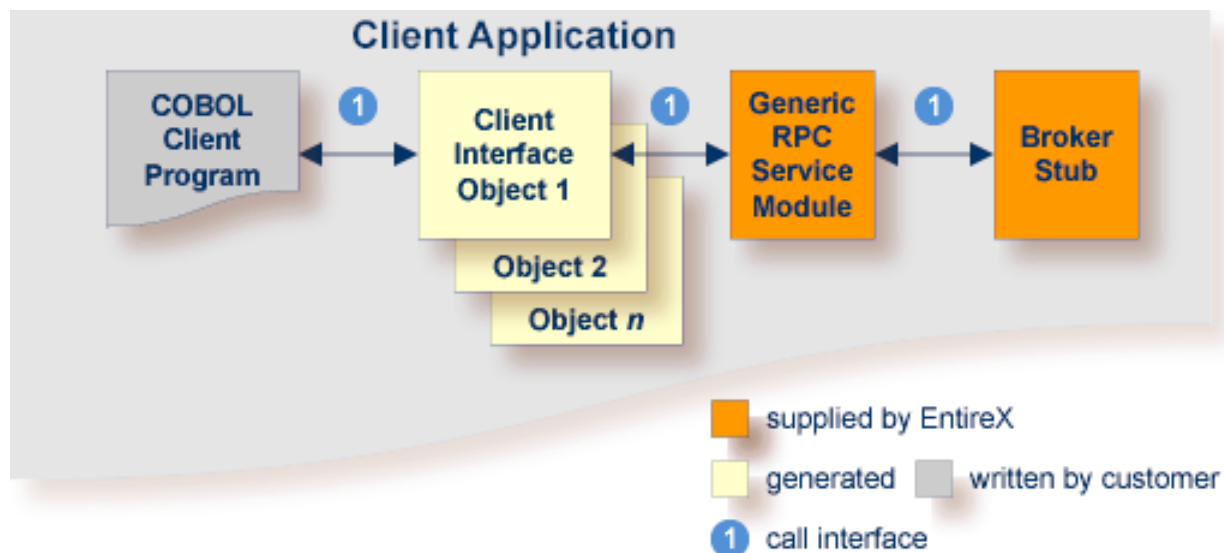
04 ORDERITEMID	PIC 9(5).
04 ORDERQUANTITY	PIC 9(10).
04 ORDERCUSTOMERID	PIC S9(9) BINARY.
04 ORDERITEMNAME	PIC X(99).
04 ORDERUNITPRICE	PIC X(20).
04 ORDERTOTAL	PIC X(20).
04 ORDERCUSTOMERADDRESS.	
05 NAME	
05 STREET	PIC X(99).
05 CITY	PIC X(99).
05 ZIPCODE	PIC X(20).
05 COUNTRYCODE	PIC X(20).
04 ORDERSHIPPINGDATE	PIC X(256).
04 ORDERCONFIRMATION	PIC X(256).
04 ORDERCONFIRMATIONFLAG	PIC X.

No constraints available

The copybook is the COBOL programmer's API. It is for use in the COBOL application.

As you would expect, the length of generated COBOL parameters exactly matches the lengths extracted. COBOL parameters always have a fixed length defined at compile time. They are not altered at runtime to reflect actual string length as in modern programming languages. This means that if you choose your default length (see [Extracting the Interface of the webMethods IS Service](#)) you have to choose the maximum possible data size to ensure all data can be transmitted. However, this may needlessly increase the message size if no constraints are set for fields of shorter data length. Choose your defaults and constraints carefully.

Building the Client Application



In this scenario, the COBOL client program, every generated client interface object, generic RPC services module and the broker stub are linked together to the client application.

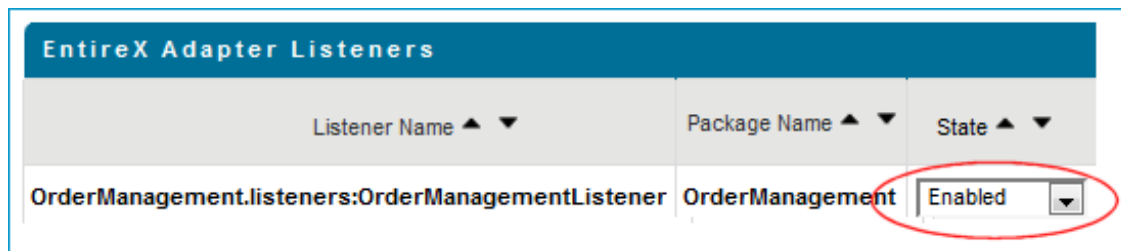
See *Using the COBOL Wrapper for IDMS/DC with Call Interfaces (z/OS)*.

Task 3: Execute the Call from COBOL to REST

Use the EntireX IDL Tester to execute the call.

➤ To execute the call

- 1 For EntireX RPC (see [Task 3](#) in the [Introduction](#)) and an RPC Listener Connection, make sure EntireX Broker is running.
- 2 Enable the EntireX Adapter Listener.



- 3 Activate the EntireX perspective by choosing **Window > Perspective > Open Perspective > EntireX**, and from the context menu of the IDL file, choose **IDL Tester**.

- 4 This step depends on the method used to call the server (see [Task 3](#) in the [Introduction](#)).
 - For *EntireX RPC*, enter the same TCP/IP address for **Broker** that you supplied when you created the RPC Listener Connection to EntireX (localhost:1971).
 - For *Direct RPC*, that is, you generated a Direct RPC Listener Connection, enter the TCP/IP address from the Integration Server (localhost:1971).
- 5 For **Server**, enter the values you specified for the server address (RPC/SRV1/CALLNAT).

The screenshot shows a software interface with two input fields at the top: "Broker: localhost:1971" and "Server: RPC/SRV1/CALLNAT". Both fields are circled in red. Below these fields are five buttons: "Call" (highlighted with a dashed border), "Ping", "Open Conversation", "Reset", and "Exit". Below the buttons is a "Messages" section containing a list of data items and their values.

2 OrderItemID (N5)	45
2 OrderQuantity (N10)	18
2 OrderCustomerID (I4)	234
2 OrderItemName (A99)	tennis racquet
2 OrderUnitPrice (A20)	79.89 GBP
2 OrderTotal (A20)	1438.02 GBP
3 Name (A99)	The All England Lawn Tennis Club (Championships) Limited
3 Street (A99)	Church Road
3 City (A99)	Wimbledon London
3 ZipCode (A20)	SW19 5AE
3 CountryCode (A20)	UK
2 OrderShippingDate (A256)	Your order will be shipped on Wed, Jul 27, 2016.
2 OrderConfirmation (A256)	Your order is confirmed.
2 OrderConfirmationFlag (L)	true

7

Calling REST from COBOL under BS2000

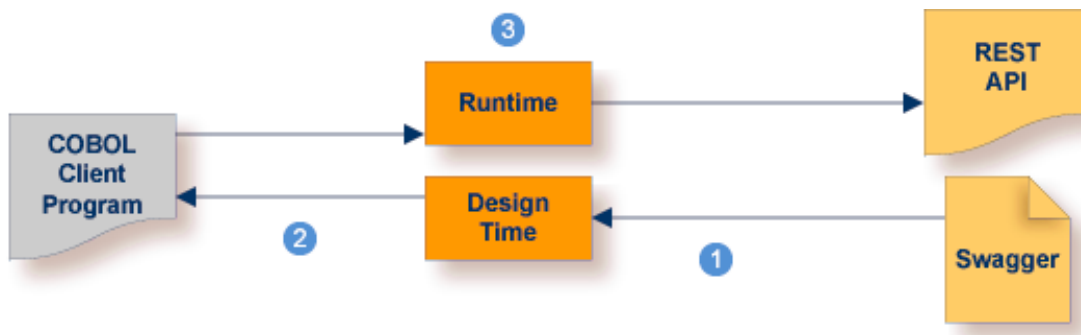
■ Introduction	72
■ What do I need to Install for this Scenario?	73
■ Task 1: Create the Consumer REST API Descriptor, Connections and Listeners	73
■ Task 2: Generate Client Interface Objects and Build Client Application	80
■ Task 3: Execute the Call from COBOL to REST	84

This scenario uses the following tools of the Designer:

- from the **Service Development** perspective: New REST API Descriptor
- from the **EntireX** perspective: *IDL Extractor for Integration Server* and *COBOL Wrapper*

Introduction

To call the REST API from COBOL, take an existing description of a REST API **1** and generate the integration logic **2** to call it from a COBOL application **3**, as shown below.



- 1** Create the consumer REST API descriptor in Integration Server. See *Service Development Help* in the webMethods Integration Server documentation. Then generate Integration Server connections and listeners. See *Using the IDL Extractor for Integration Server*.
- 2** Generate client interface objects and build COBOL client application.
- 3** Execute the call from the COBOL client to the REST API.

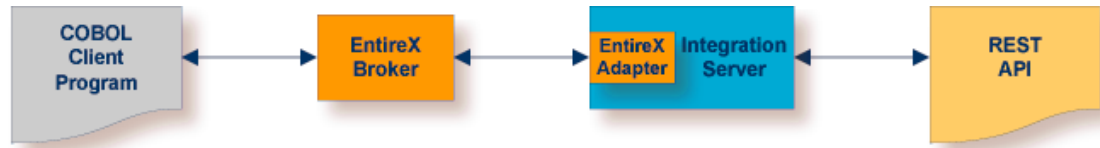
This scenario makes the following important assumptions:

For Task 1:

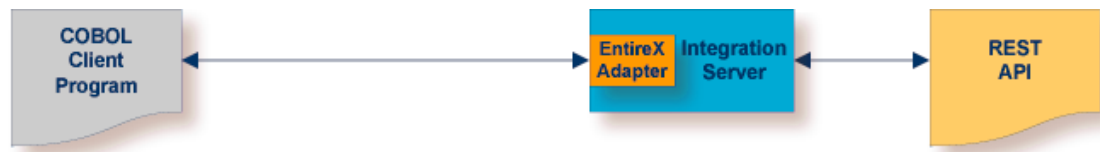
- You have a working Integration Server Service that you want to call from your REST API.

For Task 3:

- You can call the REST API at runtime using different methods:
 - For the EntireX RPC connection method you need EntireX Broker on one of the supported platforms: z/OS | Linux | Windows | BS2000.



- For the EntireX Direct RPC connection method there are no additional prerequisites.



What do I need to Install for this Scenario?

- You have Software AG Designer installed with EntireX plugins and Service Development plugins.
 - For EntireX plugins, see *Designer > EntireX Designer* under *EntireX Installation Packages* in the General Installation documentation.
 - For Service Development plugins, refer to the Integration Server documentation.
- You have an Integration Server with EntireX Adapter installed. See *Adapters > EntireX Adapter* under *EntireX Installation Packages* in the General Installation documentation.
- Optionally, for RPC Connection method (Task 3), you have an EntireX Broker installed.
 - For Linux and Windows, see *EntireX Broker* under *EntireX Installation Packages* in the General Installation documentation.
- On the machine where the COBOL client program is executed, you have the Broker stubs installed. See *Administering Broker Stubs* in the BS2000 Administration documentation.

Task 1: Create the Consumer REST API Descriptor, Connections and Listeners

This section covers the following topics:

- [Introduction](#)
- [Creating a Consumer REST API Descriptor](#)
- [Extracting the Interface of the webMethods IS Service](#)
- [Creating Listener Objects in Integration Server](#)

- [Testing the Extraction Results \(Optional\)](#)

Introduction

After you have created the consumer REST API descriptor, follow the instructions for extracting a webMethods Integration Server (IS) service under *Using the IDL Extractor for Integration Server*.

This process creates the following EntireX metafiles:

- an IDL file, containing definitions of the interface between client and server; see *Software AG IDL File* in the IDL Editor documentation in the IDL Editor documentation
- a webMethods IS Connection
- a webMethods IS Listener, depending on connection type. See *Listeners* in the EntireX Adapter documentation.

Connection types are described under [Creating Listener Objects in Integration Server](#).

Creating a Consumer REST API Descriptor

➤ To create a consumer REST API Descriptor

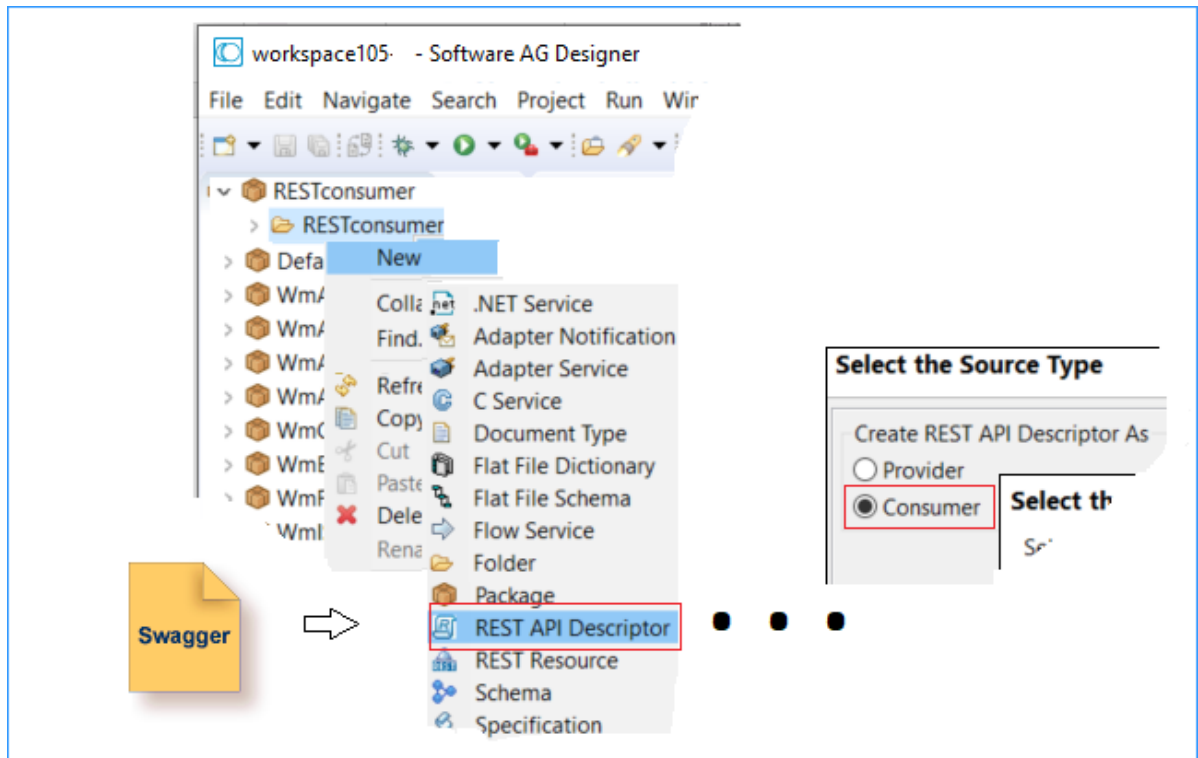
- 1 Switch to the **Service Development** perspective. Create a package and folder in Designer where the API descriptor is placed into.
- 2 Invoke the wizard to create a new REST API Descriptor in Eclipse. From the **File** menu, choose **New > REST API Descriptor**.

Or:

Choose **New** from the toolbar or context menu of a view showing resources.

Or:

Press **Ctrl-N** to start the selection of the New Wizards.



- 3 Follow the instructions on the screen. Provide the description of the REST API (Swagger file) and create the REST API descriptor for a **Consumer**.

For more information on REST API descriptors see *Service Development Help* in the webMethods Integration Server documentation.

Extracting the Interface of the webMethods IS Service

First create a project in Designer. This project is the container into which all extracted and generated EntireX artifacts are placed. Name this project *COBOL placeOrder client*.

» To start the IDL Extractor for Integration Server

- 1 Switch to the EntireX perspective.
- 2 Invoke the IDL Extractor for webMethods IS, which is a New Wizard in Eclipse. From the **File** menu, choose **New**. Select IDL Extractor for webMethods IS in the following page.

Or:

Choose **New** from the toolbar or context menu of a view showing resources.

Or:

Press **Ctrl-N** to start the selection of the New Wizards.

- 3 If you are using the wizard for the first time without any predefined Integration Server connections, enter the TCP/IP address of the webMethods IS to extract from. Otherwise select the connection to the Integration Server.
- 4 Press **Next**.

➤ **To extract the IDL from the selected package**

- 1 Under **Source (Integration Server)**, select the package you want to extract from.

Extract IDL from the selected package

Select the package (source), specify the container and file name (target) and choose the mapping.

Source (Integration Server)
 Packages on Integration Server localhost:5555

- OrderManagement
- WmEntireX
- WmFlatFile
- WmIExtDC

Target (Eclipse Workspace)
 Container: COBOL placeOrder client Browse...
 IDL File Name: OrderManagement

Optimize extracted IDL for usage with: COBOL

Map Integration Server data type to IDL

- ☐ alphanumeric with variable length
- ☒ alphanumeric with fixed length of 256

Used if no constraints available

☒ Create Listener Objects in Integration Server

? < Back Next > Finish Cancel

- 2 For **Target (Eclipse Workspace)**, specify project *COBOL placeOrder client*. Extraction results will be placed in this container. You can keep the **IDL File Name** as it has been derived from the package name by default. Since COBOL is your desired endpoint, select **Optimize extracted IDL for usage with: COBOL**.



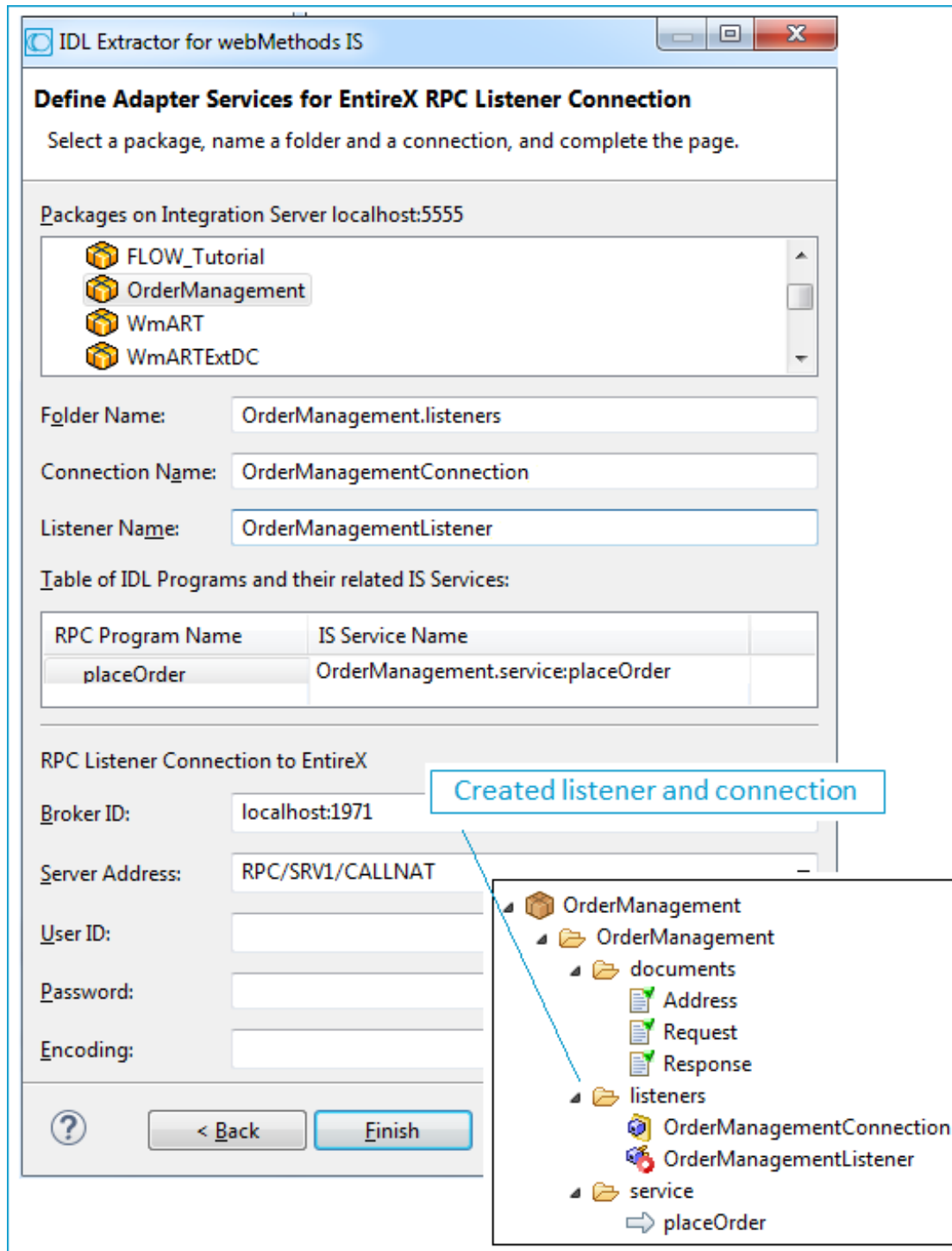
Note: COBOL only supports fixed-length data items, whereas webMethods IS has variable length types, often without any length restrictions (constraints). Parameters with no specified constraints will take the default fixed length. You can specify this default in field alphanumeric with fixed length. .

- 3 A listener is required for calling webMethods IS from an RPC client, so check **Create Listener Objects in Integration Server**.
- 4 Press **Next**.

Creating Listener Objects in Integration Server

➤ To create Listener Objects in Integration Server

- 1 Select a Listener connection type used for inbound connection to the Integration Server.
 - An EntireX RPC Listener connection is the standard way and is always available.
 - If your interface contains only In parameters, an EntireX Reliable RPC Listener connection is available.
 - If it is enabled by the license for the webMethods Integration Server:
 - an EntireX Direct RPC Listener connection is available
 - if your interfaces contain only In parameters, an EntireX Direct Reliable RPC Listener connection is available



- On this page, you set the properties for the listener objects to create; these objects will wait for the incoming COBOL requests. Select the package and specify the folder name into which the listener is generated. If the folder does not exist, it will be created automatically.

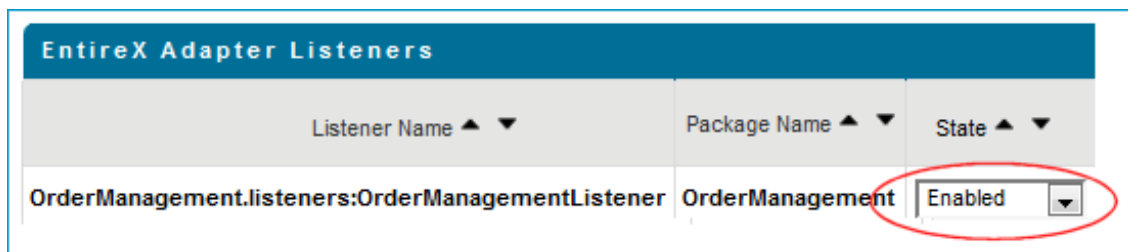
- 3 Keep the defaults given for **RPC Listener Connection to EntireX**. The **Broker ID** is a TCP/IP or SSL/TLS address and **Server Address** is an EntireX-specific namespace to locate a target server (here: *OrderManagementListener*).
- 4 When creating a connection, a package dependency is added such that the selected package depends on webMethods EntireX (the package WmEntireX) with the version currently used.
- 5 Press **Finish**. The listener and the connection appear in the package navigator. For the package navigator, switch to the service development perspective by choosing **Window > Perspective > Open Perspective > Service Development**.

Testing the Extraction Results (Optional)

You can test the results of the extraction and the server back end, using the EntireX IDL Tester.

➤ To test the extraction results

- 1 For EntireX RPC (see [Task 3](#) in the [Introduction](#)) and an RPC Listener Connection, make sure EntireX Broker is running.
- 2 Enable the EntireX Adapter Listener.



- 3 Activate the EntireX perspective by choosing **Window > Perspective > Open Perspective > EntireX**, and from the context menu of the IDL file, choose **IDL Tester**.
- 4 This step depends on the method used to call the server (see [Task 3](#) in the [Introduction](#)).
 - For *EntireX RPC*, enter the same TCP/IP address for **Broker** that you supplied when you created the RPC Listener Connection to EntireX (localhost:1971).
 - For *Direct RPC*, that is, you generated a Direct RPC Listener Connection, enter the TCP/IP address from the Integration Server (localhost:1971).
- 5 For **Server**, enter the values you specified for the server address (RPC/SRV1/CALLNAT).

The screenshot shows a graphical user interface for a COBOL application. At the top, there are two text input fields: "Broker:" with the value "localhost:1971" and "Server:" with the value "RPC/SRV1/CALLNAT". Both fields are circled in red. Below these fields are five buttons: "Call" (highlighted with a dashed border), "Ping", "Open Conversation", "Reset", and "Exit". Below the buttons is a "Messages" section with a scrollable text area. Below the messages is a table displaying the results of a REST call.

2 OrderItemID (N5)	45
2 OrderQuantity (N10)	18
2 OrderCustomerID (I4)	234
2 OrderItemName (A99)	tennis racquet
2 OrderUnitPrice (A20)	79.89 GBP
2 OrderTotal (A20)	1438.02 GBP
3 Name (A99)	The All England Lawn Tennis Club (Championships) Limited
3 Street (A99)	Church Road
3 City (A99)	Wimbledon London
3 ZipCode (A20)	SW 19 5AE
3 CountryCode (A20)	UK
2 OrderShippingDate (A256)	Your order will be shipped on Wed, Jul 27, 2016.
2 OrderConfirmation (A256)	Your order is confirmed.
2 OrderConfirmationFlag (L)	true

Task 2: Generate Client Interface Objects and Build Client Application

This section covers the following topics:

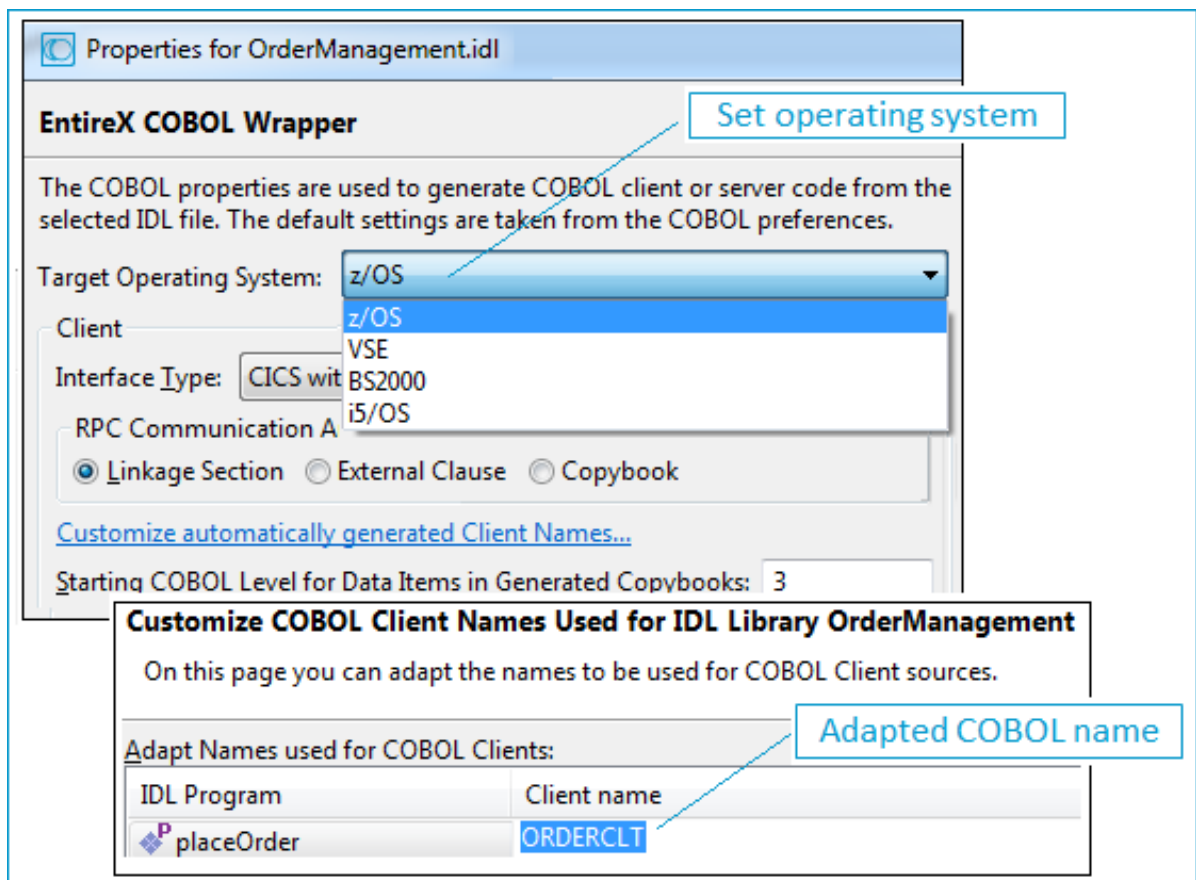
- [Setting Generation Options \(Properties\) for the COBOL Wrapper](#)
- [Generating COBOL Code](#)

■ Building the Client Application

Setting Generation Options (Properties) for the COBOL Wrapper

➤ To set generation options (properties) for the COBOL Wrapper

- 1 To generate suitable COBOL code for your environment, first set the correct target operating system. Make sure the EntireX perspective is active (**Window > Perspective > Open Perspective > EntireX**) and call the properties of the COBOL Wrapper.



- 2 From the context menu of the IDL file, choose **Properties > EntireX COBOL Wrapper**. On the COBOL Wrapper **Properties** page, set the **Target Operating System**
- 3 Adapt the name of the generated COBOL program: Choose **Customize automatically generated Client Names**. Take ORDERCLT as the COBOL program's name.

Generating COBOL Code

➤ To generate COBOL code

- For COBOL code generation, make sure the EntireX Perspective is active: Choose **Window > Perspective > Open Perspective > EntireX**. From context menu of the IDL file, choose **COBOL > Generate RPC Client**. This generates the COBOL client *ORDERCLT* in folder *client* and a copybook in folder *include*:

The screenshot shows the Eclipse IDE with the COBOL client *ORDERCLT* and its copybook. The main editor displays the COBOL code for *ORDERCLT*, which includes a COBOL RPC Client Interface Object and a COBOL RPC Interface Copybook. The Package Explorer on the right shows the project structure with folders *src*, *client*, and *include*. The *client* folder contains *COBSRVI*, *ORDERCLT*, and *include*. The *include* folder contains *ERXCOMM* and *ORDERCLT*. The *ORDERCLT* file is highlighted in the Package Explorer. The *COPYBOOK* file is also highlighted in the Package Explorer. The *COPYBOOK* file is highlighted in the Package Explorer.

COBOL client

```

CBL LIB QUOTE
IDENTIFICATION DIVISION.
PROGRAM-ID. ORDERCLT.

*****
*
* COBOL RPC Client Interface Object
*
* IDL L: ORDERCLT
* IDL P: *****
*
* Program: COBOL RPC Interface Copybook
* Target:
* String: IDL Library = OrderManagement
* Inter: IDL Program = placeOrder
* RPC C:
* RPC P:
* Copybook Name = ORDERCLT
* Target Platform = all
* Interface Type = all
* String Literal = QUOTE
*
* In the
* ei
*
* Template Source = cobol_include.tpl $Revision: 1.55 $
*
* or
*
* Template Source = cobol_linkage.tpl
* 03 REQUEST.
* 04 ORDERITEMID PIC 9(5).
* 04 ORDERQUANTITY PIC 9(10).
* 04 ORDERCUSTOMERID PIC S9(9) BINARY.
* 03 RESPONSE.
* 04 ORDERITEMNAME PIC X(99).
* 04 ORDERUNITPRICE PIC X(20).
* 04 ORDERTOTAL PIC X(20).
* 04 ORDERCUSTOMERADDRESS.
* 05 NAME
* 05 STREET PIC X(99).
* 05 CITY PIC X(99).
* 05 ZIPCODE PIC X(20).
* 05 COUNTRYCODE PIC X(20).
* 04 ORDERSHIPPINGDATE PIC X(256).
* 04 ORDERCONFIRMATION PIC X(256).
* 04 ORDERCONFIRMATIONFLAG PIC X.

```

Copybook

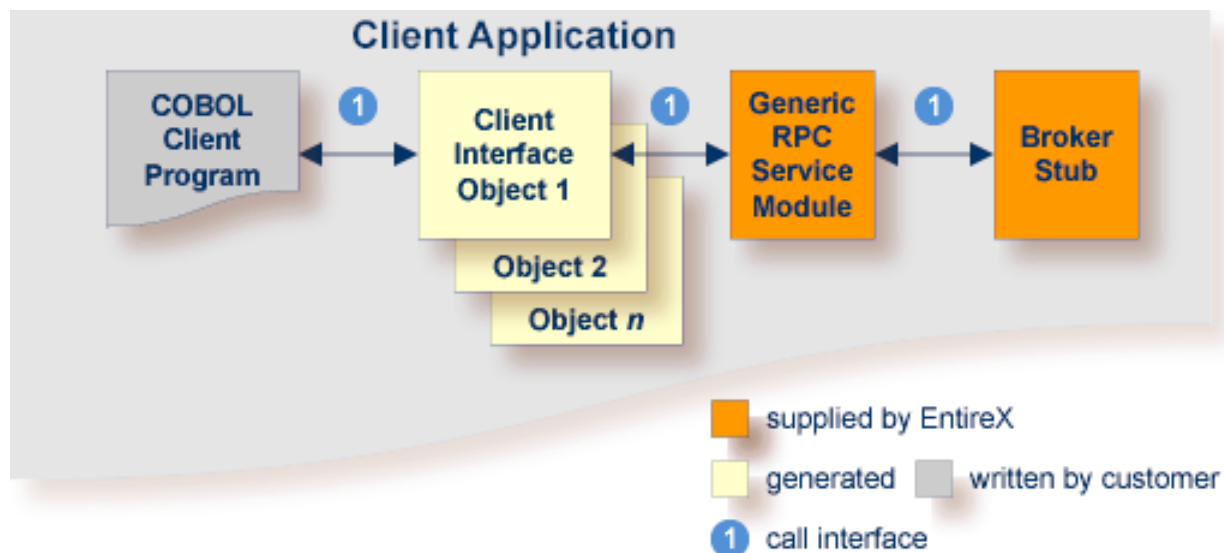
Constraints available

No constraints available

The copybook is the COBOL programmer's API. It is for use in the COBOL application.

As you would expect, the length of generated COBOL parameters exactly matches the lengths extracted. COBOL parameters always have a fixed length defined at compile time. They are not altered at runtime to reflect actual string length as in modern programming languages. This means that if you choose your default length (see [Extracting the Interface of the webMethods IS Service](#)) you have to choose the maximum possible data size to ensure all data can be transmitted. However, this may needlessly increase the message size if no constraints are set for fields of shorter data length. Choose your defaults and constraints carefully.

Building the Client Application



In this scenario, the COBOL client program, every generated client interface object, generic RPC services module and the broker stub are linked together to the client application.

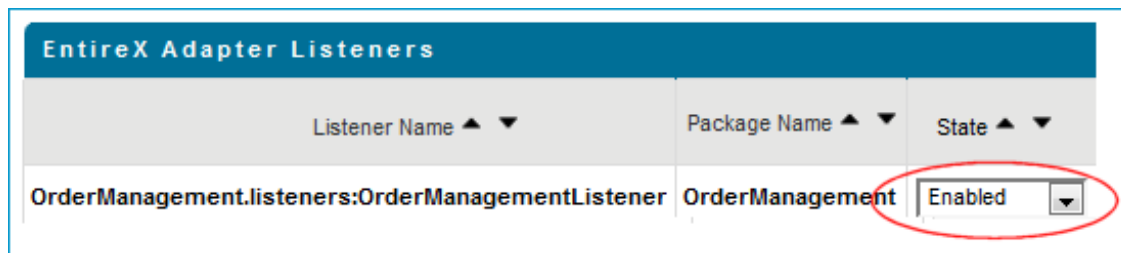
See *Using the COBOL Wrapper for Batch (z/OS, BS2000, z/VSE and IBM i)*.

Task 3: Execute the Call from COBOL to REST

Use the EntireX IDL Tester to execute the call.

➤ To execute the call

- 1 For EntireX RPC (see [Task 3](#) in the [Introduction](#)) and an RPC Listener Connection, make sure EntireX Broker is running.
- 2 Enable the EntireX Adapter Listener.



- 3 Activate the EntireX perspective by choosing **Window > Perspective > Open Perspective > EntireX**, and from the context menu of the IDL file, choose **IDL Tester**.

- 4 This step depends on the method used to call the server (see [Task 3](#) in the [Introduction](#)).
 - For *EntireX RPC*, enter the same TCP/IP address for **Broker** that you supplied when you created the RPC Listener Connection to EntireX (localhost:1971).
 - For *Direct RPC*, that is, you generated a Direct RPC Listener Connection, enter the TCP/IP address from the Integration Server (localhost:1971).
- 5 For **Server**, enter the values you specified for the server address (RPC/SRV1/CALLNAT).

Broker: localhost:1971 Server: RPC/SRV1/CALLNAT

Call Ping Open Conversation Reset Exit

Messages

2 OrderItemID (N5)	45
2 OrderQuantity (N10)	18
2 OrderCustomerID (I4)	234
2 OrderItemName (A99)	tennis racquet
2 OrderUnitPrice (A20)	79.89 GBP
2 OrderTotal (A20)	1438.02 GBP
3 Name (A99)	The All England Lawn Tennis Club (Championships) Limited
3 Street (A99)	Church Road
3 City (A99)	Wimbledon London
3 ZipCode (A20)	SW19 5AE
3 CountryCode (A20)	UK
2 OrderShippingDate (A256)	Your order will be shipped on Wed, Jul 27, 2016.
2 OrderConfirmation (A256)	Your order is confirmed.
2 OrderConfirmationFlag (L)	true

8

Calling REST from COBOL under z/VSE CICS

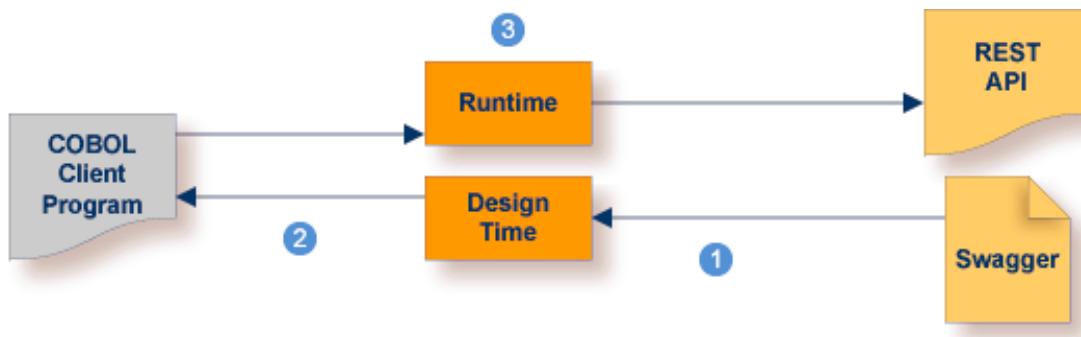
■ Introduction	88
■ What do I need to Install for this Scenario?	89
■ Task 1: Create the Consumer REST API Descriptor, Connections and Listeners	89
■ Task 2: Generate Client Interface Objects and Build Client Application	96
■ Task 3: Execute the Call from COBOL to REST	101

This scenario uses the following tools of the Designer:

- from the **Service Development** perspective: New REST API Descriptor
- from the **EntireX** perspective: *IDL Extractor for Integration Server* and *COBOL Wrapper*

Introduction

To call the REST API from COBOL, take an existing description of a REST API **1** and generate the integration logic **2** to call it from a COBOL application **3**, as shown below.



- 1** Create the consumer REST API descriptor in Integration Server. See *Service Development Help* in the webMethods Integration Server documentation. Then generate Integration Server connections and listeners. See *Using the IDL Extractor for Integration Server*.
- 2** Generate client interface objects and build COBOL client application.
- 3** Execute the call from the COBOL client to the REST API.

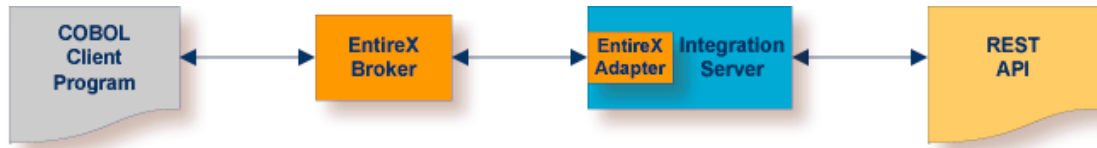
This scenario makes the following important assumptions:

For Task 1:

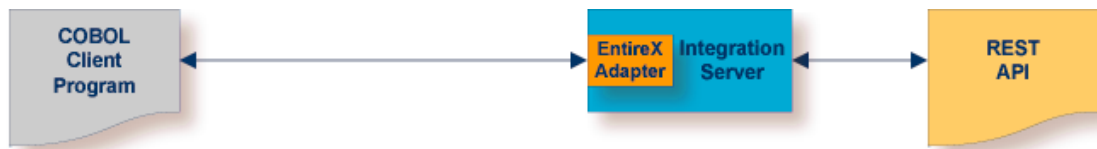
- You have a working Integration Server Service that you want to call from your REST API.

For Task 3:

- You can call the REST API at runtime using different methods:
 - For the EntireX RPC connection method you need EntireX Broker on one of the supported platforms: z/OS | Linux | Windows | BS2000.



- For the EntireX Direct RPC connection method there are no additional prerequisites.



What do I need to Install for this Scenario?

- You have Software AG Designer installed with EntireX plugins and Service Development plugins.
 - For EntireX plugins, see *Designer > EntireX Designer* under *EntireX Installation Packages* in the General Installation documentation.
 - For Service Development plugins, refer to the Integration Server documentation.
- You have an Integration Server with EntireX Adapter installed. See *Adapters > EntireX Adapter* under *EntireX Installation Packages* in the General Installation documentation.
- Optionally, for RPC Connection method (Task 3), you have an EntireX Broker installed.
 - For Linux and Windows, see *EntireX Broker* under *EntireX Installation Packages* in the General Installation documentation.
- On the machine where the COBOL client program is executed, you have the Broker stubs installed. See *Administering Broker Stubs under z/VSE* in the z/VSE Administration documentation.

Task 1: Create the Consumer REST API Descriptor, Connections and Listeners

This section covers the following topics:

- [Introduction](#)
- [Creating a Consumer REST API Descriptor](#)
- [Extracting the Interface of the webMethods IS Service](#)
- [Creating Listener Objects in Integration Server](#)

- [Testing the Extraction Results \(Optional\)](#)

Introduction

After you have created the consumer REST API descriptor, follow the instructions for extracting a webMethods Integration Server (IS) service under *Using the IDL Extractor for Integration Server*.

This process creates the following EntireX metafiles:

- an IDL file, containing definitions of the interface between client and server; see *Software AG IDL File* in the IDL Editor documentation in the IDL Editor documentation
- a webMethods IS Connection
- a webMethods IS Listener, depending on connection type. See *Listeners* in the EntireX Adapter documentation.

Connection types are described under [Creating Listener Objects in Integration Server](#).

Creating a Consumer REST API Descriptor

➤ To create a consumer REST API Descriptor

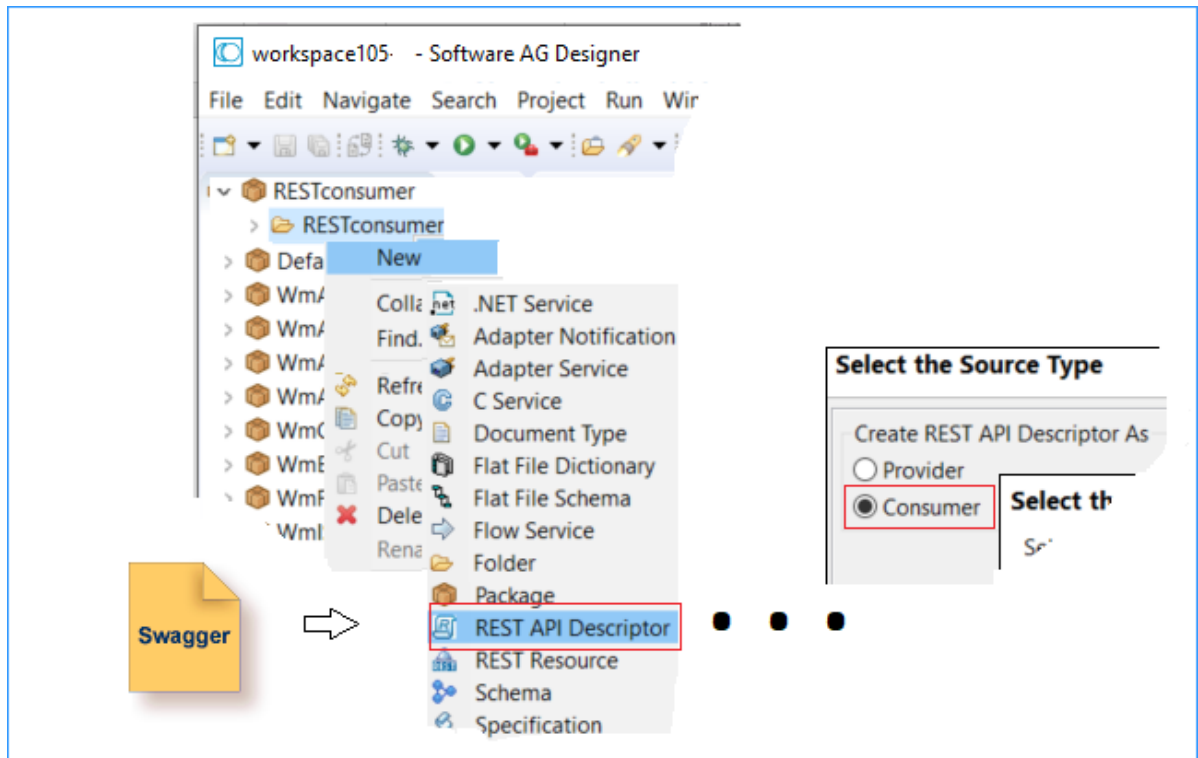
- 1 Switch to the **Service Development** perspective. Create a package and folder in Designer where the API descriptor is placed into.
- 2 Invoke the wizard to create a new REST API Descriptor in Eclipse. From the **File** menu, choose **New > REST API Descriptor**.

Or:

Choose **New** from the toolbar or context menu of a view showing resources.

Or:

Press **Ctrl-N** to start the selection of the New Wizards.



- 3 Follow the instructions on the screen. Provide the description of the REST API (Swagger file) and create the REST API descriptor for a **Consumer**.

For more information on REST API descriptors see *Service Development Help* in the webMethods Integration Server documentation.

Extracting the Interface of the webMethods IS Service

First create a project in Designer. This project is the container into which all extracted and generated EntireX artifacts are placed. Name this project *COBOL placeOrder client*.

» To start the IDL Extractor for Integration Server

- 1 Switch to the EntireX perspective.
- 2 Invoke the IDL Extractor for webMethods IS, which is a New Wizard in Eclipse. From the **File** menu, choose **New**. Select IDL Extractor for webMethods IS in the following page.

Or:

Choose **New** from the toolbar or context menu of a view showing resources.

Or:

Press **Ctrl-N** to start the selection of the New Wizards.

- 3 If you are using the wizard for the first time without any predefined Integration Server connections, enter the TCP/IP address of the webMethods IS to extract from. Otherwise select the connection to the Integration Server.
- 4 Press **Next**.

➤ **To extract the IDL from the selected package**

- 1 Under **Source (Integration Server)**, select the package you want to extract from.

Extract IDL from the selected package

Select the package (source), specify the container and file name (target) and choose the mapping.

Source (Integration Server)
 Packages on Integration Server localhost:5555

- OrderManagement
- WmEntireX
- WmFlatFile
- WmIExtDC

Target (Eclipse Workspace)
 Container: COBOL placeOrder client Browse...
 IDL File Name: OrderManagement

Optimize extracted IDL for usage with: COBOL

Map Integration Server data type to IDL

- ☐ alphanumeric with variable length
- ☒ alphanumeric with fixed length of 256

Used if no constraints available

☒ Create Listener Objects in Integration Server

? < Back Next > Finish Cancel

- 2 For **Target (Eclipse Workspace)**, specify project *COBOL placeOrder client*. Extraction results will be placed in this container. You can keep the **IDL File Name** as it has been derived from the package name by default. Since COBOL is your desired endpoint, select **Optimize extracted IDL for usage with: COBOL**.



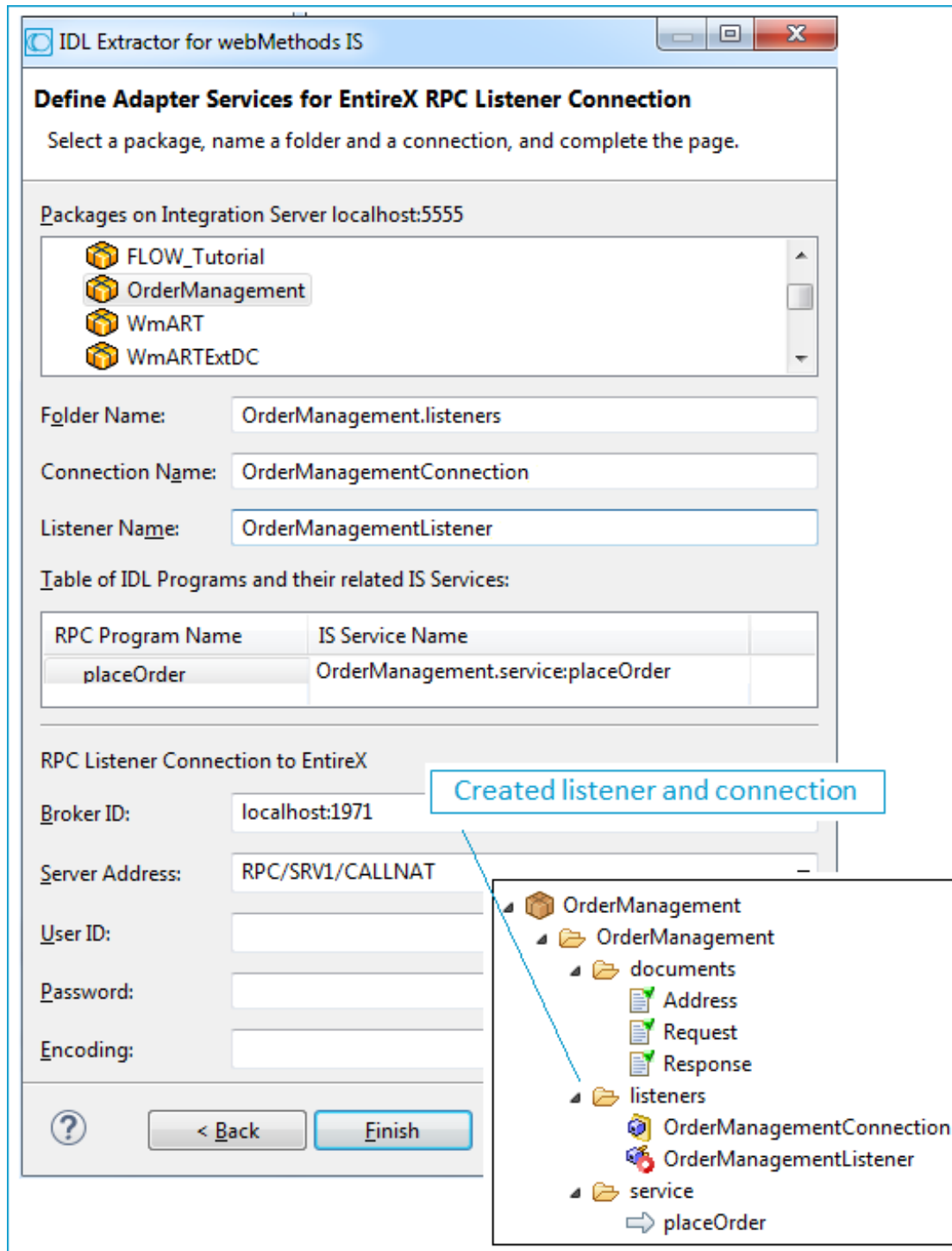
Note: COBOL only supports fixed-length data items, whereas webMethods IS has variable length types, often without any length restrictions (constraints). Parameters with no specified constraints will take the default fixed length. You can specify this default in field alphanumeric with fixed length. .

- 3 A listener is required for calling webMethods IS from an RPC client, so check **Create Listener Objects in Integration Server**.
- 4 Press **Next**.

Creating Listener Objects in Integration Server

➤ To create Listener Objects in Integration Server

- 1 Select a Listener connection type used for inbound connection to the Integration Server.
 - An EntireX RPC Listener connection is the standard way and is always available.
 - If your interface contains only In parameters, an EntireX Reliable RPC Listener connection is available.
 - If it is enabled by the license for the webMethods Integration Server:
 - an EntireX Direct RPC Listener connection is available
 - if your interfaces contain only In parameters, an EntireX Direct Reliable RPC Listener connection is available



- On this page, you set the properties for the listener objects to create; these objects will wait for the incoming COBOL requests. Select the package and specify the folder name into which the listener is generated. If the folder does not exist, it will be created automatically.

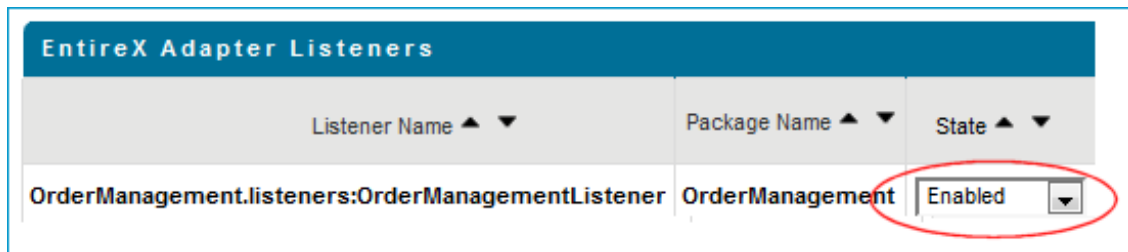
- 3 Keep the defaults given for **RPC Listener Connection to EntireX**. The **Broker ID** is a TCP/IP or SSL/TLS address and **Server Address** is an EntireX-specific namespace to locate a target server (here: *OrderManagementListener*).
- 4 When creating a connection, a package dependency is added such that the selected package depends on webMethods EntireX (the package WmEntireX) with the version currently used.
- 5 Press **Finish**. The listener and the connection appear in the package navigator. For the package navigator, switch to the service development perspective by choosing **Window > Perspective > Open Perspective > Service Development**.

Testing the Extraction Results (Optional)

You can test the results of the extraction and the server back end, using the EntireX IDL Tester.

➤ To test the extraction results

- 1 For EntireX RPC (see [Task 3](#) in the [Introduction](#)) and an RPC Listener Connection, make sure EntireX Broker is running.
- 2 Enable the EntireX Adapter Listener.



- 3 Activate the EntireX perspective by choosing **Window > Perspective > Open Perspective > EntireX**, and from the context menu of the IDL file, choose **IDL Tester**.
- 4 This step depends on the method used to call the server (see [Task 3](#) in the [Introduction](#)).
 - For *EntireX RPC*, enter the same TCP/IP address for **Broker** that you supplied when you created the RPC Listener Connection to EntireX (localhost:1971).
 - For *Direct RPC*, that is, you generated a Direct RPC Listener Connection, enter the TCP/IP address from the Integration Server (localhost:1971).
- 5 For **Server**, enter the values you specified for the server address (RPC/SRV1/CALLNAT).

The screenshot shows a COBOL application interface with the following elements:

- Broker:** localhost:1971 (circled in red)
- Server:** RPC/SRV1/CALLNAT (circled in red)
- Buttons:** Call, Ping, Open Conversation, Reset, Exit
- Messages:** A scrollable area for displaying messages.
- Response Data:** A table of data returned from the REST API call.

2 OrderItemID (N5)	45
2 OrderQuantity (N10)	18
2 OrderCustomerID (I4)	234
2 OrderItemName (A99)	tennis racquet
2 OrderUnitPrice (A20)	79.89 GBP
2 OrderTotal (A20)	1438.02 GBP
3 Name (A99)	The All England Lawn Tennis Club (Championships) Limited
3 Street (A99)	Church Road
3 City (A99)	Wimbledon London
3 ZipCode (A20)	SW 19 5AE
3 CountryCode (A20)	UK
2 OrderShippingDate (A256)	Your order will be shipped on Wed, Jul 27, 2016.
2 OrderConfirmation (A256)	Your order is confirmed.
2 OrderConfirmationFlag (L)	true

Task 2: Generate Client Interface Objects and Build Client Application

This section covers the following topics:

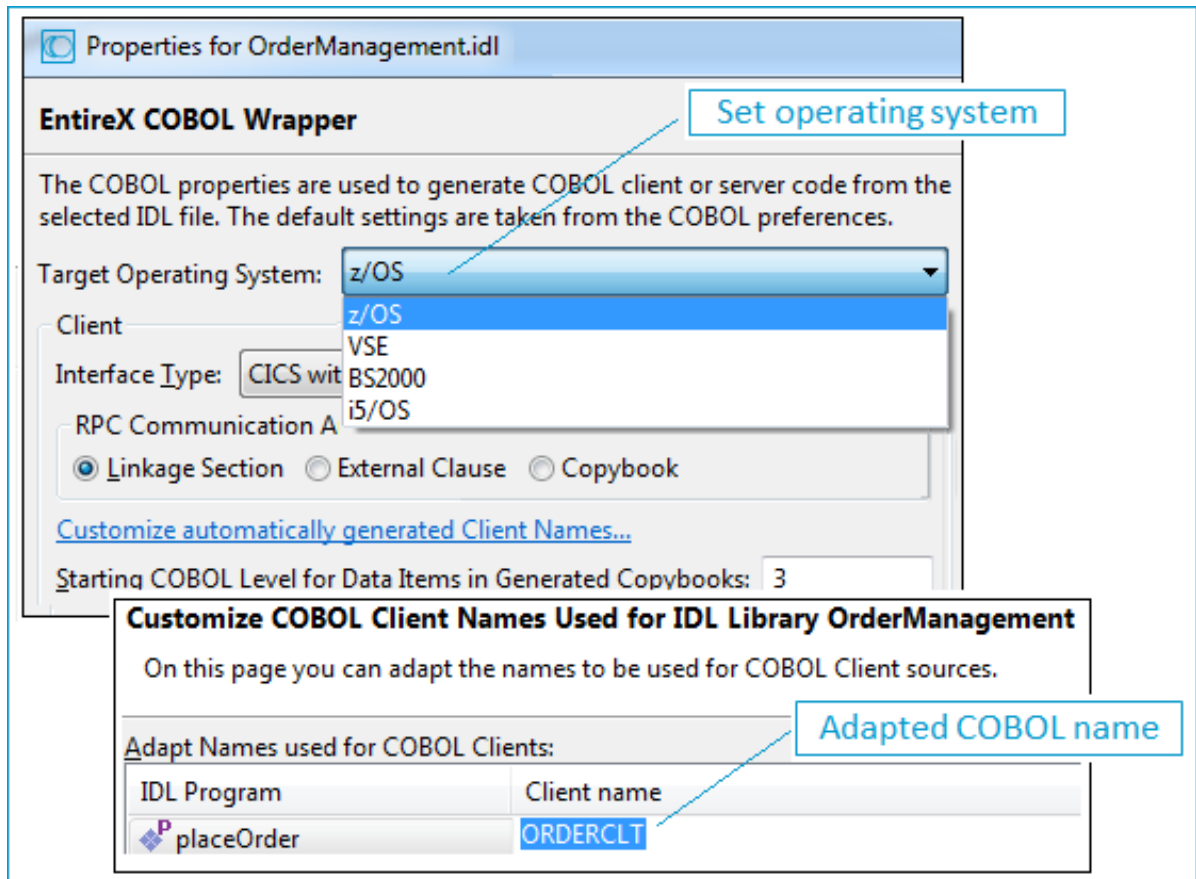
- [Setting Generation Options \(Properties\) for the COBOL Wrapper](#)
- [Generating COBOL Code](#)

■ Building the Client Application

Setting Generation Options (Properties) for the COBOL Wrapper

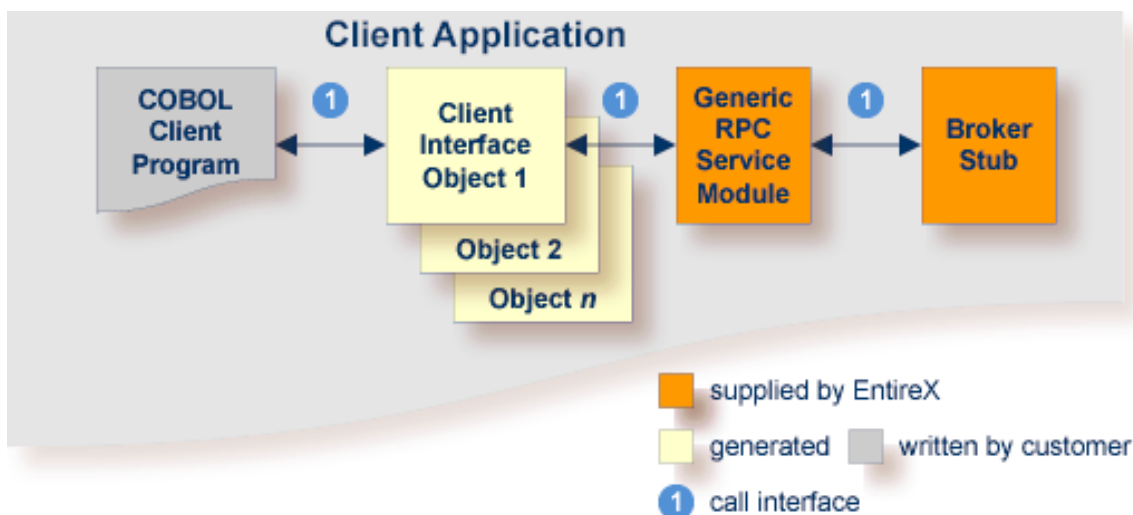
➤ To set generation options (properties) for the COBOL Wrapper

- 1 To generate suitable COBOL code for your environment, first set the correct target operating system. Make sure the EntireX perspective is active (**Window > Perspective > Open Perspective > EntireX**) and call the properties of the COBOL Wrapper.



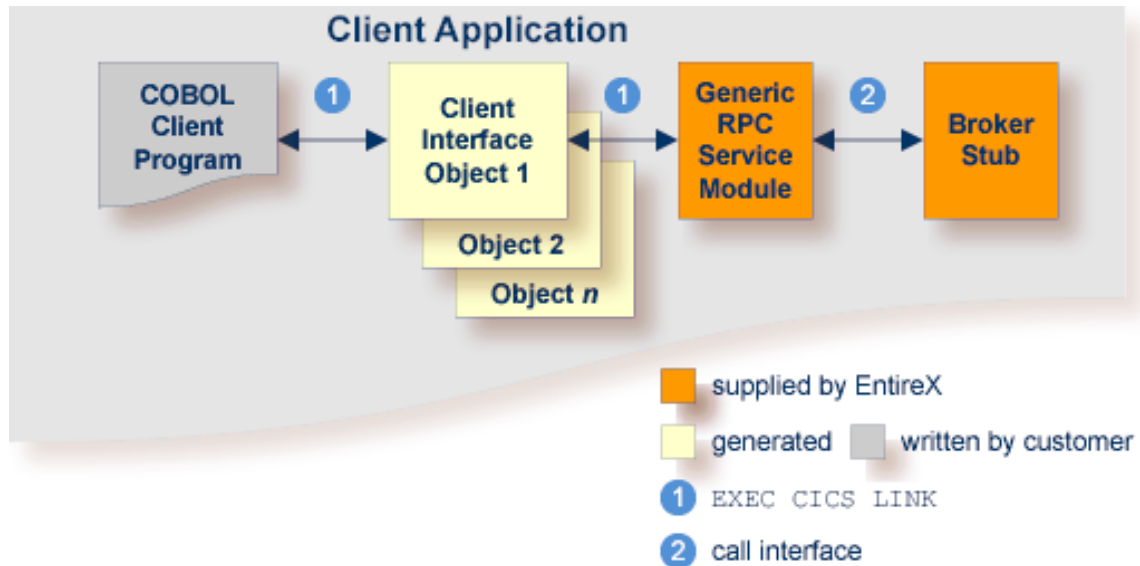
- 2 From the context menu of the IDL file, choose **Properties > EntireX COBOL Wrapper**. On the COBOL Wrapper **Properties** page, set the **Target Operating System**
- 3 Adapt the name of the generated COBOL program: Choose **Customize automatically generated Client Names**. Take ORDERCLT as the COBOL program's name.
- 4 Set the appropriate interface type for the generated client interface objects to your application. The options available for CICS are Standard Linkage Calling Convention and DFHCOM-MAREA Calling Convention. These are described below.

■ Standard Linkage Calling Convention



The COBOL Wrapper can be used with a call interface, even in CICS. This means you can build a client application where the COBOL client program, every generated client interface object, the generic RPC services module and the broker stub are linked together, similar to the batch scenario. See *Using the COBOL Wrapper for Batch (z/OS, BS2000, z/VSE and IBM i)*. Using a call interface within CICS may be useful in the following situations:

- The restriction of the COMMAREA length (about 31 KB) prevents you from using the scenario *Using the COBOL Wrapper for CICS with DFHCOMMAREA Calling Convention (z/OS and z/VSE)* (see above).
 - You do *not* require a distributed program link (CICS DPL) to your client interface object(s).
 - You prefer a call interface instead of EXEC CICS LINK to your client interface objects.
- ## ■ DFHCOMMAREA Calling Convention



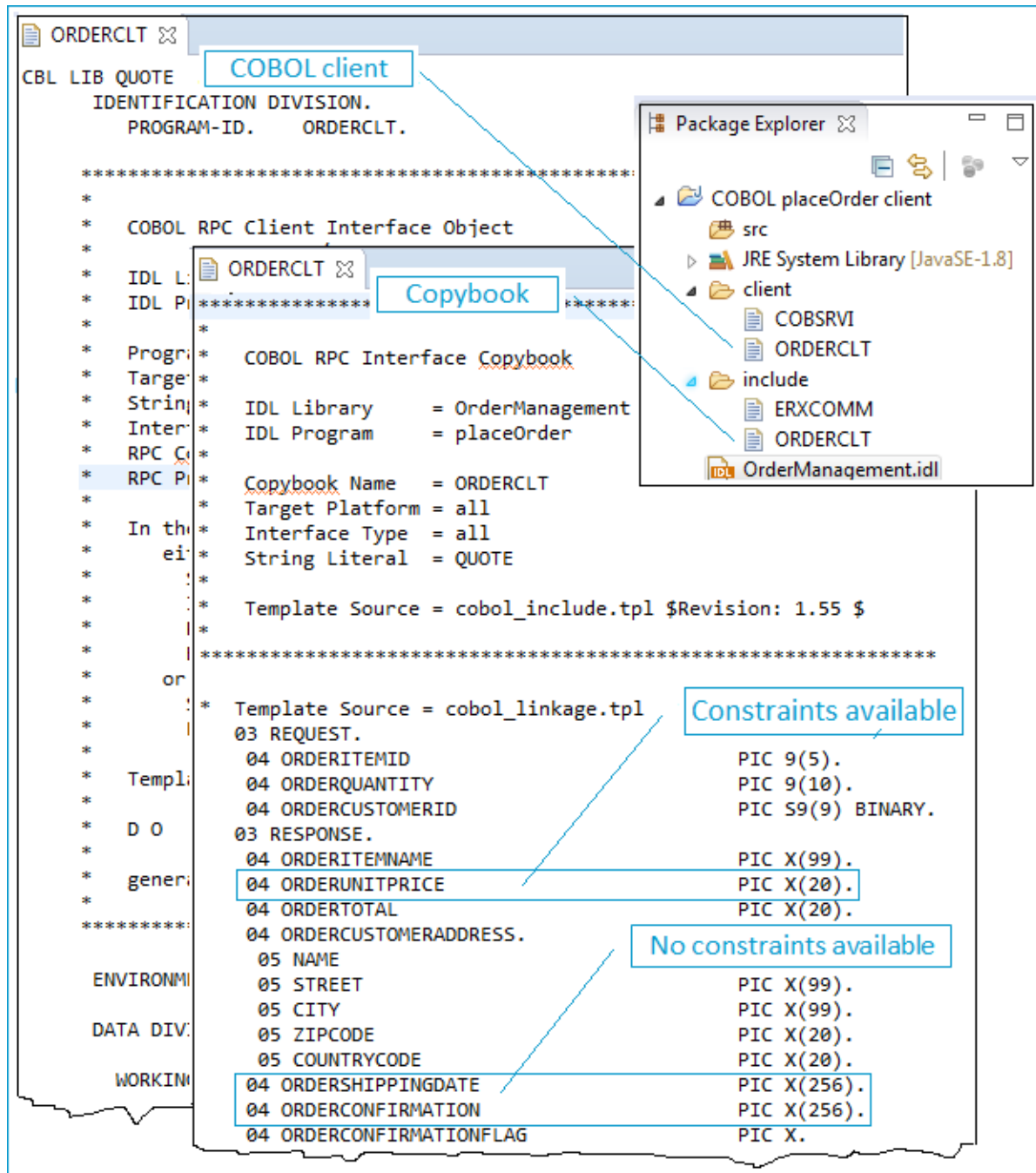
In this scenario, the generic RPC services module and the broker stub are linked together to a CICS program. The COBOL client program, every generated client interface object and the generic RPC services module together with the broker stub are installed each as separate individual CICS programs. Use the COBOL Wrapper with this calling convention in the following situations:

- You want to have an EXEC CICS LINK DFHCOMMAREA interface to your client interface object(s).
- The restriction of the COMMAREA length suits your purposes. Because the RPC communication area is also transferred in the COMMAREA, the effective length that can be used for IDL data is shorter than the CICS COMMAREA length. Nearly 31 KB can be used for IDL data.
- You wish to separate the generic RPC service module and the broker stub from the client interface object(s).
- You require a program link to the client interface object(s).

Generating COBOL Code

➤ To generate COBOL code

- For COBOL code generation, make sure the EntireX Perspective is active: Choose **Window > Perspective > Open Perspective > EntireX**. From context menu of the IDL file, choose **COBOL > Generate RPC Client**. This generates the COBOL client *ORDERCLT* in folder *client* and a copybook in folder *include*:



The copybook is the COBOL programmer's API. It is for use in the COBOL application.

As you would expect, the length of generated COBOL parameters exactly matches the lengths extracted. COBOL parameters always have a fixed length defined at compile time. They are not altered at runtime to reflect actual string length as in modern programming languages. This means that if you choose your default length (see [Extracting the Interface of the webMethods IS Service](#)) you have to choose the maximum possible data size to ensure all data can be transmitted. However,

this may needlessly increase the message size if no constraints are set for fields of shorter data length. Choose your defaults and constraints carefully.

Building the Client Application

Depending on the chosen interface type of your client interface objects (CICS with Standard Linkage Calling Convention or CICS with DFHCOMMAREA Calling Convention under *Setting Generation Options (Properties) for the COBOL Wrapper*, refer to

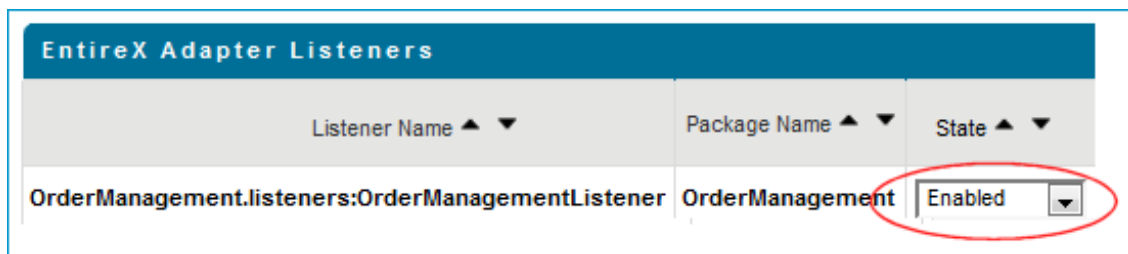
- *Using the COBOL Wrapper for CICS with Call Interfaces (z/OS and z/VSE)*
- *Using the COBOL Wrapper for CICS with DFHCOMMAREA Calling Convention (z/OS and z/VSE)*

Task 3: Execute the Call from COBOL to REST

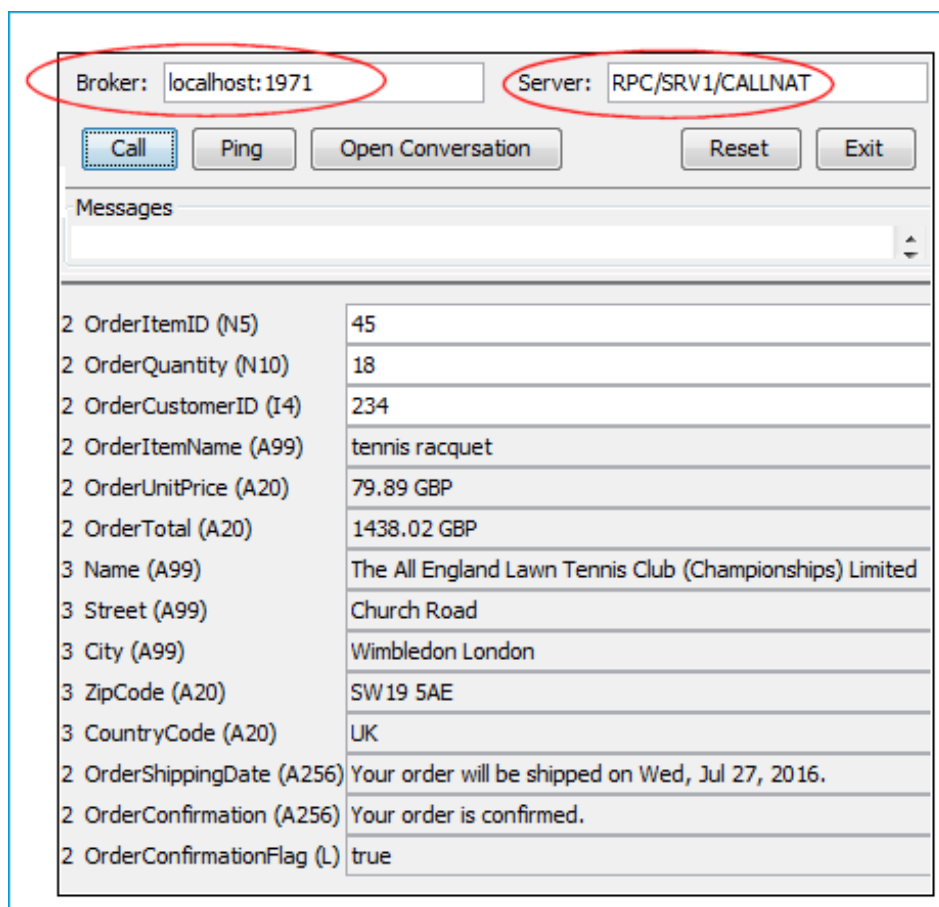
Use the EntireX IDL Tester to execute the call.

> To execute the call

- 1 For EntireX RPC (see [Task 3](#) in the [Introduction](#)) and an RPC Listener Connection, make sure EntireX Broker is running.
- 2 Enable the EntireX Adapter Listener.



- 3 Activate the EntireX perspective by choosing **Window > Perspective > Open Perspective > EntireX**, and from the context menu of the IDL file, choose **IDL Tester**.
- 4 This step depends on the method used to call the server (see [Task 3](#) in the [Introduction](#)).
 - For *EntireX RPC*, enter the same TCP/IP address for **Broker** that you supplied when you created the RPC Listener Connection to EntireX (localhost:1971).
 - For *Direct RPC*, that is, you generated a Direct RPC Listener Connection, enter the TCP/IP address from the Integration Server (localhost:1971).
- 5 For **Server**, enter the values you specified for the server address (RPC/SRV1/CALLNAT).



Broker: localhost:1971 Server: RPC/SRV1/CALLNAT

Call Ping Open Conversation Reset Exit

Messages

2 OrderItemID (N5)	45
2 OrderQuantity (N10)	18
2 OrderCustomerID (I4)	234
2 OrderItemName (A99)	tennis racquet
2 OrderUnitPrice (A20)	79.89 GBP
2 OrderTotal (A20)	1438.02 GBP
3 Name (A99)	The All England Lawn Tennis Club (Championships) Limited
3 Street (A99)	Church Road
3 City (A99)	Wimbledon London
3 ZipCode (A20)	SW 19 5AE
3 CountryCode (A20)	UK
2 OrderShippingDate (A256)	Your order will be shipped on Wed, Jul 27, 2016.
2 OrderConfirmation (A256)	Your order is confirmed.
2 OrderConfirmationFlag (L)	true

9

Calling REST from COBOL under z/VSE Batch

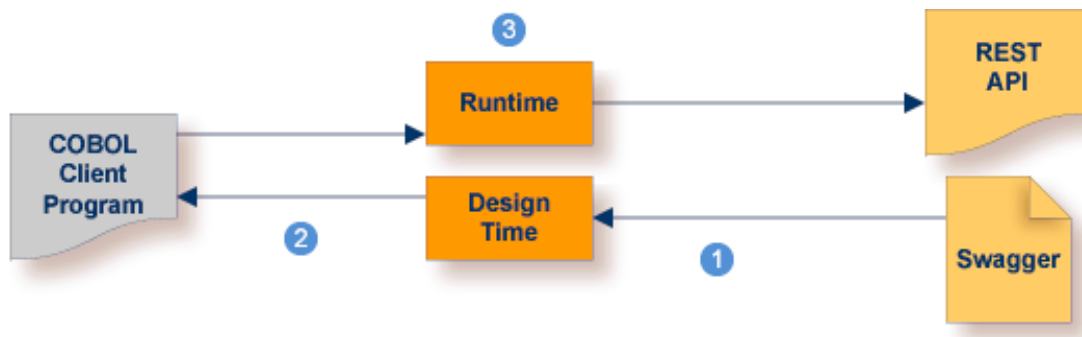
■ Introduction	104
■ What do I need to Install for this Scenario?	105
■ Task 1: Create the Consumer REST API Descriptor, Connections and Listeners	105
■ Task 2: Generate Client Interface Objects and Build Client Application	112
■ Task 3: Execute the Call from COBOL to REST	116

This scenario uses the following tools of the Designer:

- from the **Service Development** perspective: New REST API Descriptor
- from the **EntireX** perspective: *IDL Extractor for Integration Server* and *COBOL Wrapper*

Introduction

To call the REST API from COBOL, take an existing description of a REST API **1** and generate the integration logic **2** to call it from a COBOL application **3**, as shown below.



- 1** Create the consumer REST API descriptor in Integration Server. See *Service Development Help* in the webMethods Integration Server documentation. Then generate Integration Server connections and listeners. See *Using the IDL Extractor for Integration Server*.
- 2** Generate client interface objects and build COBOL client application.
- 3** Execute the call from the COBOL client to the REST API.

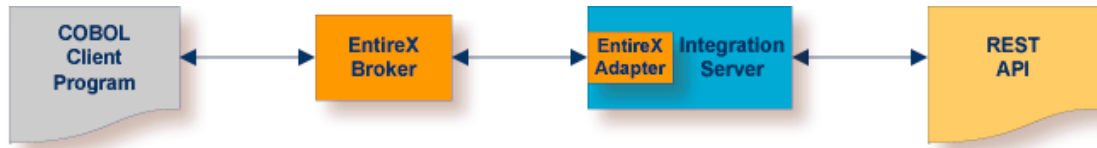
This scenario makes the following important assumptions:

For Task 1:

- You have a working Integration Server Service that you want to call from your REST API.

For Task 3:

- You can call the REST API at runtime using different methods:
 - For the EntireX RPC connection method you need EntireX Broker on one of the supported platforms: z/OS | Linux | Windows | BS2000.



- For the EntireX Direct RPC connection method there are no additional prerequisites.



What do I need to Install for this Scenario?

- You have Software AG Designer installed with EntireX plugins and Service Development plugins.
 - For EntireX plugins, see *Designer > EntireX Designer* under *EntireX Installation Packages* in the General Installation documentation.
 - For Service Development plugins, refer to the Integration Server documentation.
- You have an Integration Server with EntireX Adapter installed. See *Adapters > EntireX Adapter* under *EntireX Installation Packages* in the General Installation documentation.
- Optionally, for RPC Connection method (Task 3), you have an EntireX Broker installed.
 - For Linux and Windows, see *EntireX Broker* under *EntireX Installation Packages* in the General Installation documentation.
- On the machine where the COBOL client program is executed, you have the Broker stubs installed. See *Administering Broker Stubs under z/VSE* in the z/VSE Administration documentation.

Task 1: Create the Consumer REST API Descriptor, Connections and Listeners

This section covers the following topics:

- [Introduction](#)
- [Creating a Consumer REST API Descriptor](#)
- [Extracting the Interface of the webMethods IS Service](#)
- [Creating Listener Objects in Integration Server](#)

- [Testing the Extraction Results \(Optional\)](#)

Introduction

After you have created the consumer REST API descriptor, follow the instructions for extracting a webMethods Integration Server (IS) service under *Using the IDL Extractor for Integration Server*.

This process creates the following EntireX metafiles:

- an IDL file, containing definitions of the interface between client and server; see *Software AG IDL File* in the IDL Editor documentation in the IDL Editor documentation
- a webMethods IS Connection
- a webMethods IS Listener, depending on connection type. See *Listeners* in the EntireX Adapter documentation.

Connection types are described under [Creating Listener Objects in Integration Server](#).

Creating a Consumer REST API Descriptor

➤ To create a consumer REST API Descriptor

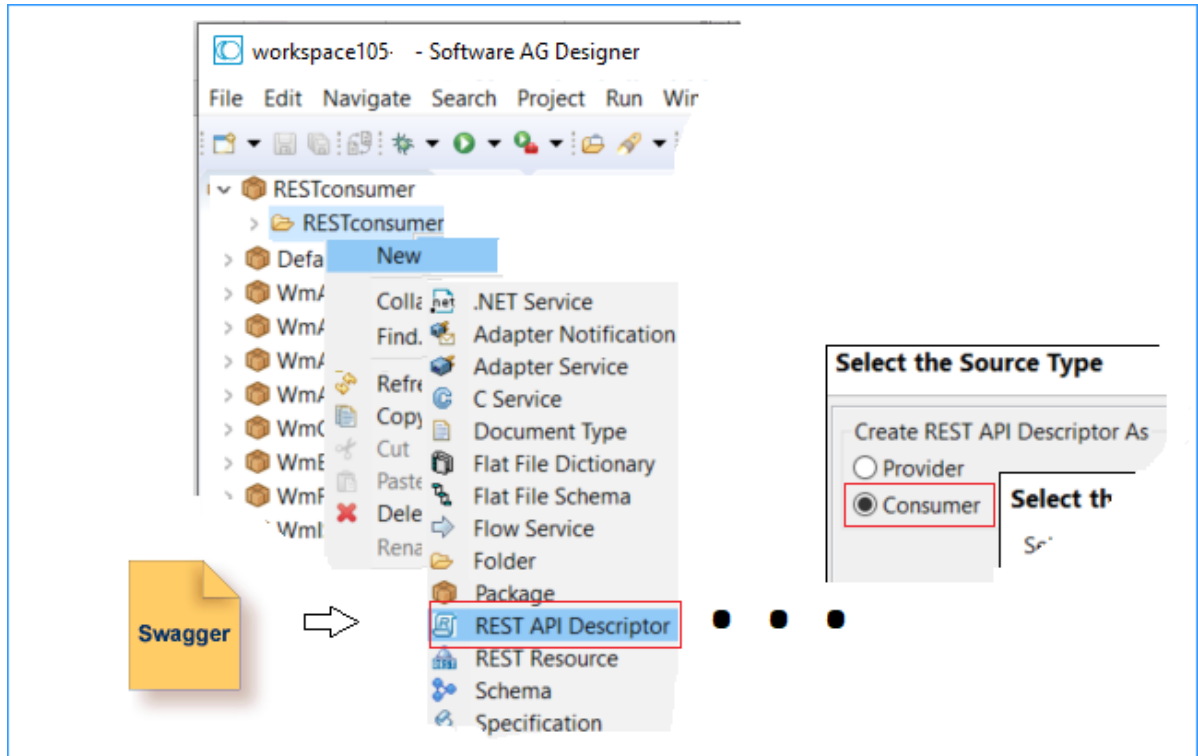
- 1 Switch to the **Service Development** perspective. Create a package and folder in Designer where the API descriptor is placed into.
- 2 Invoke the wizard to create a new REST API Descriptor in Eclipse. From the **File** menu, choose **New > REST API Descriptor**.

Or:

Choose **New** from the toolbar or context menu of a view showing resources.

Or:

Press **Ctrl-N** to start the selection of the New Wizards.



- 3 Follow the instructions on the screen. Provide the description of the REST API (Swagger file) and create the REST API descriptor for a **Consumer**.

For more information on REST API descriptors see *Service Development Help* in the webMethods Integration Server documentation.

Extracting the Interface of the webMethods IS Service

First create a project in Designer. This project is the container into which all extracted and generated EntireX artifacts are placed. Name this project *COBOL placeOrder client*.

» To start the IDL Extractor for Integration Server

- 1 Switch to the EntireX perspective.
- 2 Invoke the IDL Extractor for webMethods IS, which is a New Wizard in Eclipse. From the **File** menu, choose **New**. Select IDL Extractor for webMethods IS in the following page.

Or:

Choose **New** from the toolbar or context menu of a view showing resources.

Or:

Press **Ctrl-N** to start the selection of the New Wizards.

- 3 If you are using the wizard for the first time without any predefined Integration Server connections, enter the TCP/IP address of the webMethods IS to extract from. Otherwise select the connection to the Integration Server.
- 4 Press **Next**.

➤ **To extract the IDL from the selected package**

- 1 Under **Source (Integration Server)**, select the package you want to extract from.

Extract IDL from the selected package

Select the package (source), specify the container and file name (target) and choose the mapping.

Source (Integration Server)
 Packages on Integration Server localhost:5555

- OrderManagement
- WmEntireX
- WmFlatFile
- WmIExtDC

Target (Eclipse Workspace)
 Container: COBOL placeOrder client Browse...
 IDL File Name: OrderManagement

Optimize extracted IDL for usage with: COBOL

Map Integration Server data type to IDL

- ☐ alphanumeric with variable length
- ☒ alphanumeric with fixed length of 256

Used if no constraints available

☒ Create Listener Objects in Integration Server

? < Back Next > Finish Cancel

- 2 For **Target (Eclipse Workspace)**, specify project *COBOL placeOrder client*. Extraction results will be placed in this container. You can keep the **IDL File Name** as it has been derived from the package name by default. Since COBOL is your desired endpoint, select **Optimize extracted IDL for usage with: COBOL**.



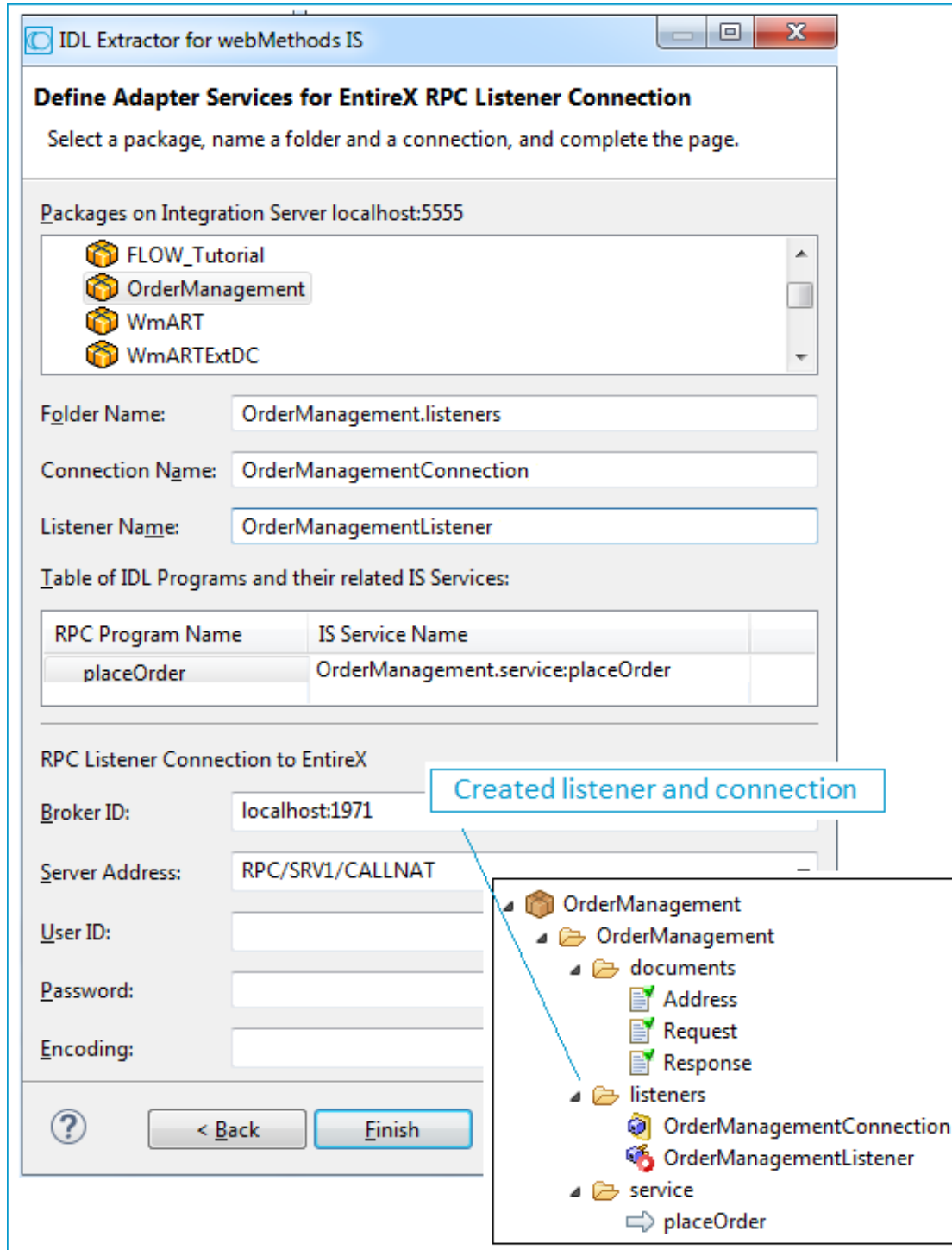
Note: COBOL only supports fixed-length data items, whereas webMethods IS has variable length types, often without any length restrictions (constraints). Parameters with no specified constraints will take the default fixed length. You can specify this default in field alphanumeric with fixed length. .

- 3 A listener is required for calling webMethods IS from an RPC client, so check **Create Listener Objects in Integration Server**.
- 4 Press **Next**.

Creating Listener Objects in Integration Server

➤ To create Listener Objects in Integration Server

- 1 Select a Listener connection type used for inbound connection to the Integration Server.
 - An EntireX RPC Listener connection is the standard way and is always available.
 - If your interface contains only In parameters, an EntireX Reliable RPC Listener connection is available.
 - If it is enabled by the license for the webMethods Integration Server:
 - an EntireX Direct RPC Listener connection is available
 - if your interfaces contain only In parameters, an EntireX Direct Reliable RPC Listener connection is available



- On this page, you set the properties for the listener objects to create; these objects will wait for the incoming COBOL requests. Select the package and specify the folder name into which the listener is generated. If the folder does not exist, it will be created automatically.

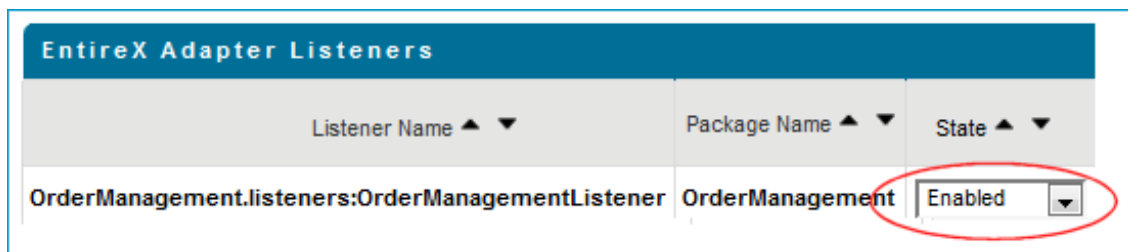
- 3 Keep the defaults given for **RPC Listener Connection to EntireX**. The **Broker ID** is a TCP/IP or SSL/TLS address and **Server Address** is an EntireX-specific namespace to locate a target server (here: *OrderManagementListener*).
- 4 When creating a connection, a package dependency is added such that the selected package depends on webMethods EntireX (the package WmEntireX) with the version currently used.
- 5 Press **Finish**. The listener and the connection appear in the package navigator. For the package navigator, switch to the service development perspective by choosing **Window > Perspective > Open Perspective > Service Development**.

Testing the Extraction Results (Optional)

You can test the results of the extraction and the server back end, using the EntireX IDL Tester.

➤ To test the extraction results

- 1 For EntireX RPC (see [Task 3](#) in the [Introduction](#)) and an RPC Listener Connection, make sure EntireX Broker is running.
- 2 Enable the EntireX Adapter Listener.



- 3 Activate the EntireX perspective by choosing **Window > Perspective > Open Perspective > EntireX**, and from the context menu of the IDL file, choose **IDL Tester**.
- 4 This step depends on the method used to call the server (see [Task 3](#) in the [Introduction](#)).
 - For *EntireX RPC*, enter the same TCP/IP address for **Broker** that you supplied when you created the RPC Listener Connection to EntireX (localhost:1971).
 - For *Direct RPC*, that is, you generated a Direct RPC Listener Connection, enter the TCP/IP address from the Integration Server (localhost:1971).
- 5 For **Server**, enter the values you specified for the server address (RPC/SRV1/CALLNAT).

Broker: localhost:1971 Server: RPC/SRV1/CALLNAT

Call Ping Open Conversation Reset Exit

Messages

2 OrderItemID (N5)	45
2 OrderQuantity (N10)	18
2 OrderCustomerID (I4)	234
2 OrderItemName (A99)	tennis racquet
2 OrderUnitPrice (A20)	79.89 GBP
2 OrderTotal (A20)	1438.02 GBP
3 Name (A99)	The All England Lawn Tennis Club (Championships) Limited
3 Street (A99)	Church Road
3 City (A99)	Wimbledon London
3 ZipCode (A20)	SW 19 5AE
3 CountryCode (A20)	UK
2 OrderShippingDate (A256)	Your order will be shipped on Wed, Jul 27, 2016.
2 OrderConfirmation (A256)	Your order is confirmed.
2 OrderConfirmationFlag (L)	true

Task 2: Generate Client Interface Objects and Build Client Application

This section covers the following topics:

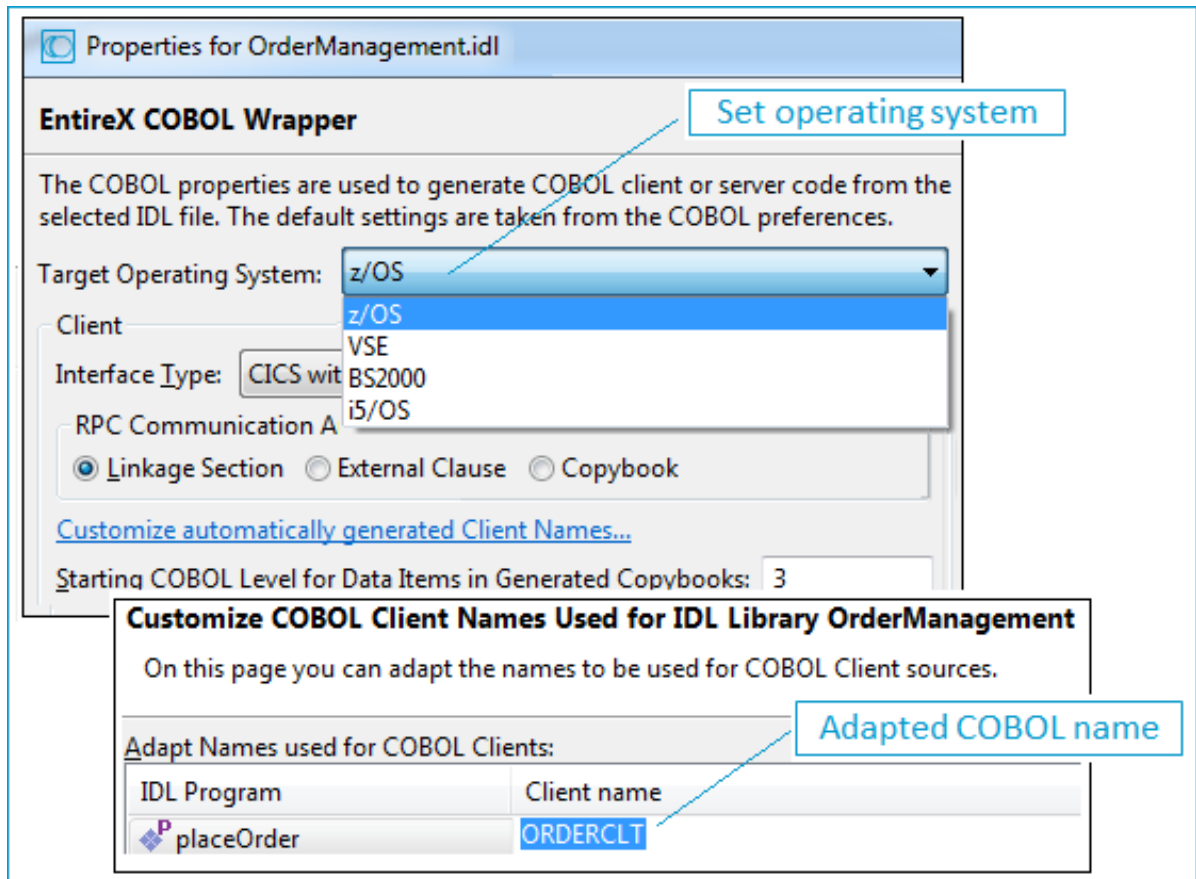
- [Setting Generation Options \(Properties\) for the COBOL Wrapper](#)
- [Generating COBOL Code](#)

■ Building the Client Application

Setting Generation Options (Properties) for the COBOL Wrapper

➤ To set generation options (properties) for the COBOL Wrapper

- 1 To generate suitable COBOL code for your environment, first set the correct target operating system. Make sure the EntireX perspective is active (**Window > Perspective > Open Perspective > EntireX**) and call the properties of the COBOL Wrapper.



- 2 From the context menu of the IDL file, choose **Properties > EntireX COBOL Wrapper**. On the COBOL Wrapper **Properties** page, set the **Target Operating System**
- 3 Adapt the name of the generated COBOL program: Choose **Customize automatically generated Client Names**. Take ORDERCLT as the COBOL program's name.

Generating COBOL Code

➤ To generate COBOL code

- For COBOL code generation, make sure the EntireX Perspective is active: Choose **Window > Perspective > Open Perspective > EntireX**. From context menu of the IDL file, choose **COBOL > Generate RPC Client**. This generates the COBOL client *ORDERCLT* in folder *client* and a copybook in folder *include*:

The screenshot shows the Eclipse IDE with the COBOL client generation process. The main editor displays the COBOL code for *ORDERCLT*. The code is structured as follows:

```

CBL LIB QUOTE
IDENTIFICATION DIVISION.
PROGRAM-ID. ORDERCLT.

*****
*
* COBOL RPC Client Interface Object
*
* IDL L: ORDERCLT
* IDL P: *****
*
* Program: COBOL RPC Interface Copybook
* Target:
* String: IDL Library = OrderManagement
* Inter: IDL Program = placeOrder
* RPC C:
* RPC P:
* Copybook Name = ORDERCLT
* Target Platform = all
* Interface Type = all
* String Literal = QUOTE
*
* In the
* ei
*
* Template Source = cobol_include.tpl $Revision: 1.55 $
*
* or
*
* Template Source = cobol_linkage.tpl
* 03 REQUEST.
* 04 ORDERITEMID PIC 9(5).
* 04 ORDERQUANTITY PIC 9(10).
* 04 ORDERCUSTOMERID PIC S9(9) BINARY.
* 03 RESPONSE.
* 04 ORDERITEMNAME PIC X(99).
* 04 ORDERUNITPRICE PIC X(20).
* 04 ORDERTOTAL PIC X(20).
* 04 ORDERCUSTOMERADDRESS.
* 05 NAME
* 05 STREET PIC X(99).
* 05 CITY PIC X(99).
* 05 ZIPCODE PIC X(20).
* 05 COUNTRYCODE PIC X(20).
* 04 ORDERSHIPPINGDATE PIC X(256).
* 04 ORDERCONFIRMATION PIC X(256).
* 04 ORDERCONFIRMATIONFLAG PIC X.

```

The Package Explorer on the right shows the project structure:

- COBOL placeOrder client
 - src
 - JRE System Library [JavaSE-1.8]
 - client
 - COBSRVI
 - ORDERCLT
 - include
 - ERXCOMM
 - ORDERCLT
 - OrderManagement.idl

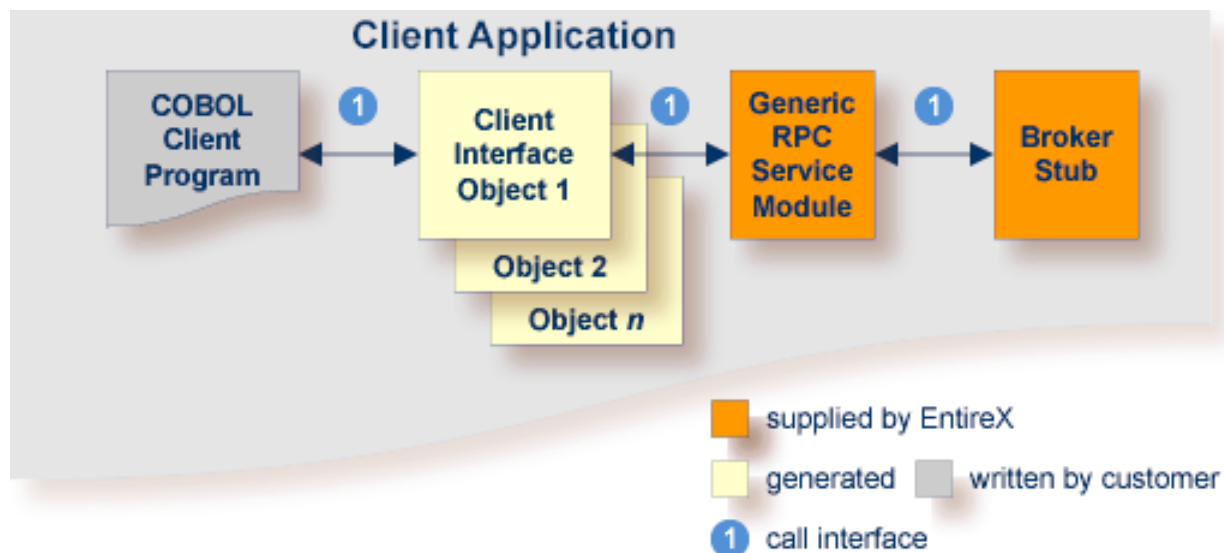
The 'Constraints available' table is as follows:

Constraint	Value
03 REQUEST.	
04 ORDERITEMID	PIC 9(5).
04 ORDERQUANTITY	PIC 9(10).
04 ORDERCUSTOMERID	PIC S9(9) BINARY.
03 RESPONSE.	
04 ORDERITEMNAME	PIC X(99).
04 ORDERUNITPRICE	PIC X(20).
04 ORDERTOTAL	PIC X(20).
04 ORDERCUSTOMERADDRESS.	
05 NAME	
05 STREET	PIC X(99).
05 CITY	PIC X(99).
05 ZIPCODE	PIC X(20).
05 COUNTRYCODE	PIC X(20).
04 ORDERSHIPPINGDATE	PIC X(256).
04 ORDERCONFIRMATION	PIC X(256).
04 ORDERCONFIRMATIONFLAG	PIC X.

The copybook is the COBOL programmer's API. It is for use in the COBOL application.

As you would expect, the length of generated COBOL parameters exactly matches the lengths extracted. COBOL parameters always have a fixed length defined at compile time. They are not altered at runtime to reflect actual string length as in modern programming languages. This means that if you choose your default length (see [Extracting the Interface of the webMethods IS Service](#)) you have to choose the maximum possible data size to ensure all data can be transmitted. However, this may needlessly increase the message size if no constraints are set for fields of shorter data length. Choose your defaults and constraints carefully.

Building the Client Application



In this scenario, the COBOL client program, every generated client interface object, generic RPC services module and the broker stub are linked together to the client application.

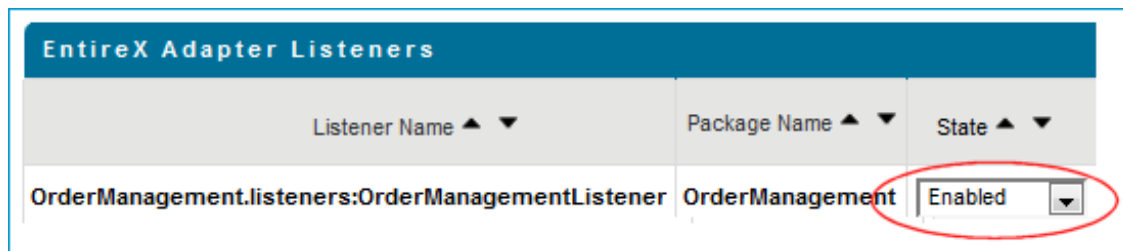
See *Using the COBOL Wrapper for Batch (z/OS, BS2000, z/VSE and IBM i)*.

Task 3: Execute the Call from COBOL to REST

Use the EntireX IDL Tester to execute the call.

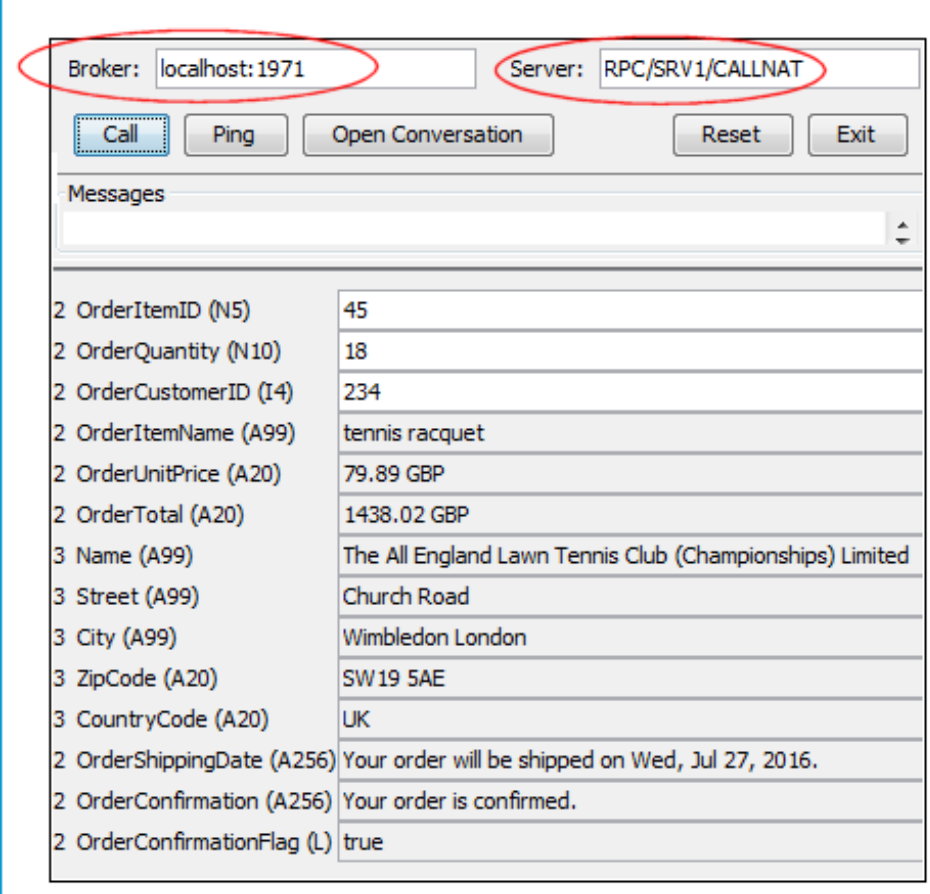
➤ To execute the call

- 1 For EntireX RPC (see [Task 3](#) in the [Introduction](#)) and an RPC Listener Connection, make sure EntireX Broker is running.
- 2 Enable the EntireX Adapter Listener.



- 3 Activate the EntireX perspective by choosing **Window > Perspective > Open Perspective > EntireX**, and from the context menu of the IDL file, choose **IDL Tester**.

- 4 This step depends on the method used to call the server (see [Task 3](#) in the [Introduction](#)).
 - For *EntireX RPC*, enter the same TCP/IP address for **Broker** that you supplied when you created the RPC Listener Connection to EntireX (localhost:1971).
 - For *Direct RPC*, that is, you generated a Direct RPC Listener Connection, enter the TCP/IP address from the Integration Server (localhost:1971).
- 5 For **Server**, enter the values you specified for the server address (RPC/SRV1/CALLNAT).



Broker: localhost:1971 Server: RPC/SRV1/CALLNAT

Call Ping Open Conversation Reset Exit

Messages

2 OrderItemID (N5)	45
2 OrderQuantity (N10)	18
2 OrderCustomerID (I4)	234
2 OrderItemName (A99)	tennis racquet
2 OrderUnitPrice (A20)	79.89 GBP
2 OrderTotal (A20)	1438.02 GBP
3 Name (A99)	The All England Lawn Tennis Club (Championships) Limited
3 Street (A99)	Church Road
3 City (A99)	Wimbledon London
3 ZipCode (A20)	SW19 5AE
3 CountryCode (A20)	UK
2 OrderShippingDate (A256)	Your order will be shipped on Wed, Jul 27, 2016.
2 OrderConfirmation (A256)	Your order is confirmed.
2 OrderConfirmationFlag (L)	true

10

Calling REST from COBOL under IBM i

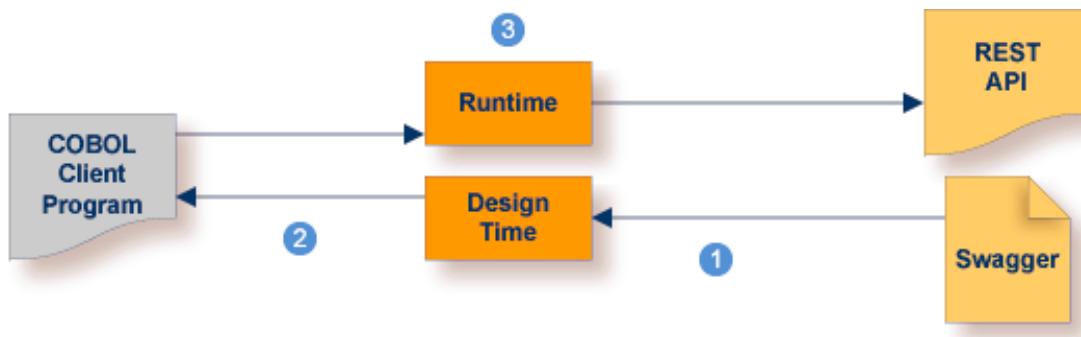
■ Introduction	120
■ What do I need to Install for this Scenario?	121
■ Task 1: Create the Consumer REST API Descriptor, Connections and Listeners	121
■ Task 2: Generate Client Interface Objects and Build Client Application	128
■ Task 3: Execute the Call from COBOL to REST	132

This scenario uses the following tools of the Designer:

- from the **Service Development** perspective: New REST API Descriptor
- from the **EntireX** perspective: *IDL Extractor for Integration Server* and *COBOL Wrapper*

Introduction

To call the REST API from COBOL, take an existing description of a REST API **1** and generate the integration logic **2** to call it from a COBOL application **3**, as shown below.



- 1** Create the consumer REST API descriptor in Integration Server. See *Service Development Help* in the webMethods Integration Server documentation. Then generate Integration Server connections and listeners. See *Using the IDL Extractor for Integration Server*.
- 2** Generate client interface objects and build COBOL client application.
- 3** Execute the call from the COBOL client to the REST API.

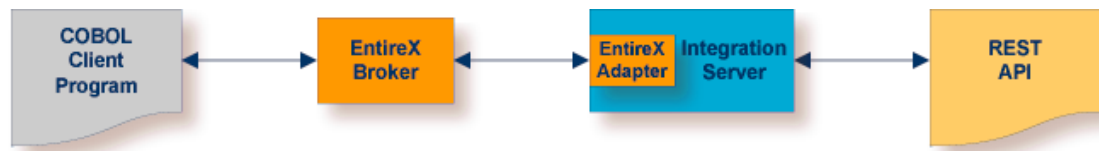
This scenario makes the following important assumptions:

For Task 1:

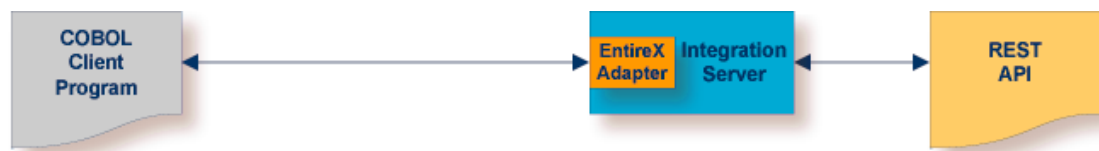
- You have a working Integration Server Service that you want to call from your REST API.

For Task 3:

- You can call the REST API at runtime using different methods:
 - For the EntireX RPC connection method you need EntireX Broker on one of the supported platforms: z/OS | Linux | Windows | BS2000.



- For the EntireX Direct RPC connection method there are no additional prerequisites.



What do I need to Install for this Scenario?

- You have Software AG Designer installed with EntireX plugins and Service Development plugins.
 - For EntireX plugins, see *Designer > EntireX Designer* under *EntireX Installation Packages* in the General Installation documentation.
 - For Service Development plugins, refer to the Integration Server documentation.
- You have an Integration Server with EntireX Adapter installed. See *Adapters > EntireX Adapter* under *EntireX Installation Packages* in the General Installation documentation.
- Optionally, for RPC Connection method (Task 3), you have an EntireX Broker installed.
 - For Linux and Windows, see *EntireX Broker* under *EntireX Installation Packages* in the General Installation documentation.
- On the machine where the COBOL client program is executed, you have the Broker stubs installed. See *Installing EntireX under IBM i*.

Task 1: Create the Consumer REST API Descriptor, Connections and Listeners

This section covers the following topics:

- [Introduction](#)
- [Creating a Consumer REST API Descriptor](#)
- [Extracting the Interface of the webMethods IS Service](#)
- [Creating Listener Objects in Integration Server](#)

- [Testing the Extraction Results \(Optional\)](#)

Introduction

After you have created the consumer REST API descriptor, follow the instructions for extracting a webMethods Integration Server (IS) service under *Using the IDL Extractor for Integration Server*.

This process creates the following EntireX metafiles:

- an IDL file, containing definitions of the interface between client and server; see *Software AG IDL File* in the IDL Editor documentation in the IDL Editor documentation
- a webMethods IS Connection
- a webMethods IS Listener, depending on connection type. See *Listeners* in the EntireX Adapter documentation.

Connection types are described under [Creating Listener Objects in Integration Server](#).

Creating a Consumer REST API Descriptor

➤ To create a consumer REST API Descriptor

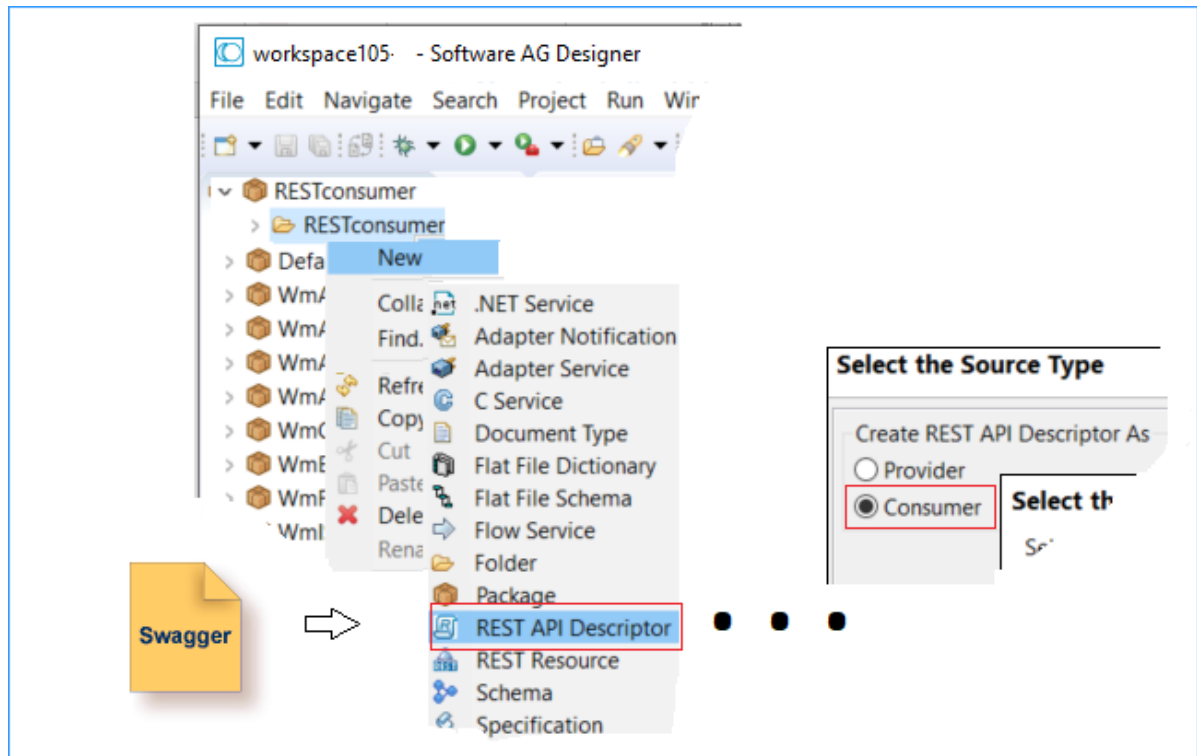
- 1 Switch to the **Service Development** perspective. Create a package and folder in Designer where the API descriptor is placed into.
- 2 Invoke the wizard to create a new REST API Descriptor in Eclipse. From the **File** menu, choose **New > REST API Descriptor**.

Or:

Choose **New** from the toolbar or context menu of a view showing resources.

Or:

Press **Ctrl-N** to start the selection of the New Wizards.



- 3 Follow the instructions on the screen. Provide the description of the REST API (Swagger file) and create the REST API descriptor for a **Consumer**.

For more information on REST API descriptors see *Service Development Help* in the webMethods Integration Server documentation.

Extracting the Interface of the webMethods IS Service

First create a project in Designer. This project is the container into which all extracted and generated EntireX artifacts are placed. Name this project *COBOL placeOrder client*.

» To start the IDL Extractor for Integration Server

- 1 Switch to the EntireX perspective.
- 2 Invoke the IDL Extractor for webMethods IS, which is a New Wizard in Eclipse. From the **File** menu, choose **New**. Select IDL Extractor for webMethods IS in the following page.

Or:

Choose **New** from the toolbar or context menu of a view showing resources.

Or:

Press **Ctrl-N** to start the selection of the New Wizards.

- 3 If you are using the wizard for the first time without any predefined Integration Server connections, enter the TCP/IP address of the webMethods IS to extract from. Otherwise select the connection to the Integration Server.
- 4 Press **Next**.

➤ **To extract the IDL from the selected package**

- 1 Under **Source (Integration Server)**, select the package you want to extract from.

Extract IDL from the selected package

Select the package (source), specify the container and file name (target) and choose the mapping.

Source (Integration Server)
 Packages on Integration Server localhost:5555

- OrderManagement
- WmEntireX
- WmFlatFile
- WmIExtDC

Target (Eclipse Workspace)
 Container: COBOL placeOrder client Browse...
 IDL File Name: OrderManagement

Optimize extracted IDL for usage with: COBOL

Map Integration Server data type to IDL

- ☐ alphanumeric with variable length
- ☒ alphanumeric with fixed length of 256

Used if no constraints available

☒ Create Listener Objects in Integration Server

? < Back Next > Finish Cancel

- 2 For **Target (Eclipse Workspace)**, specify project *COBOL placeOrder client*. Extraction results will be placed in this container. You can keep the **IDL File Name** as it has been derived from the package name by default. Since COBOL is your desired endpoint, select **Optimize extracted IDL for usage with: COBOL**.



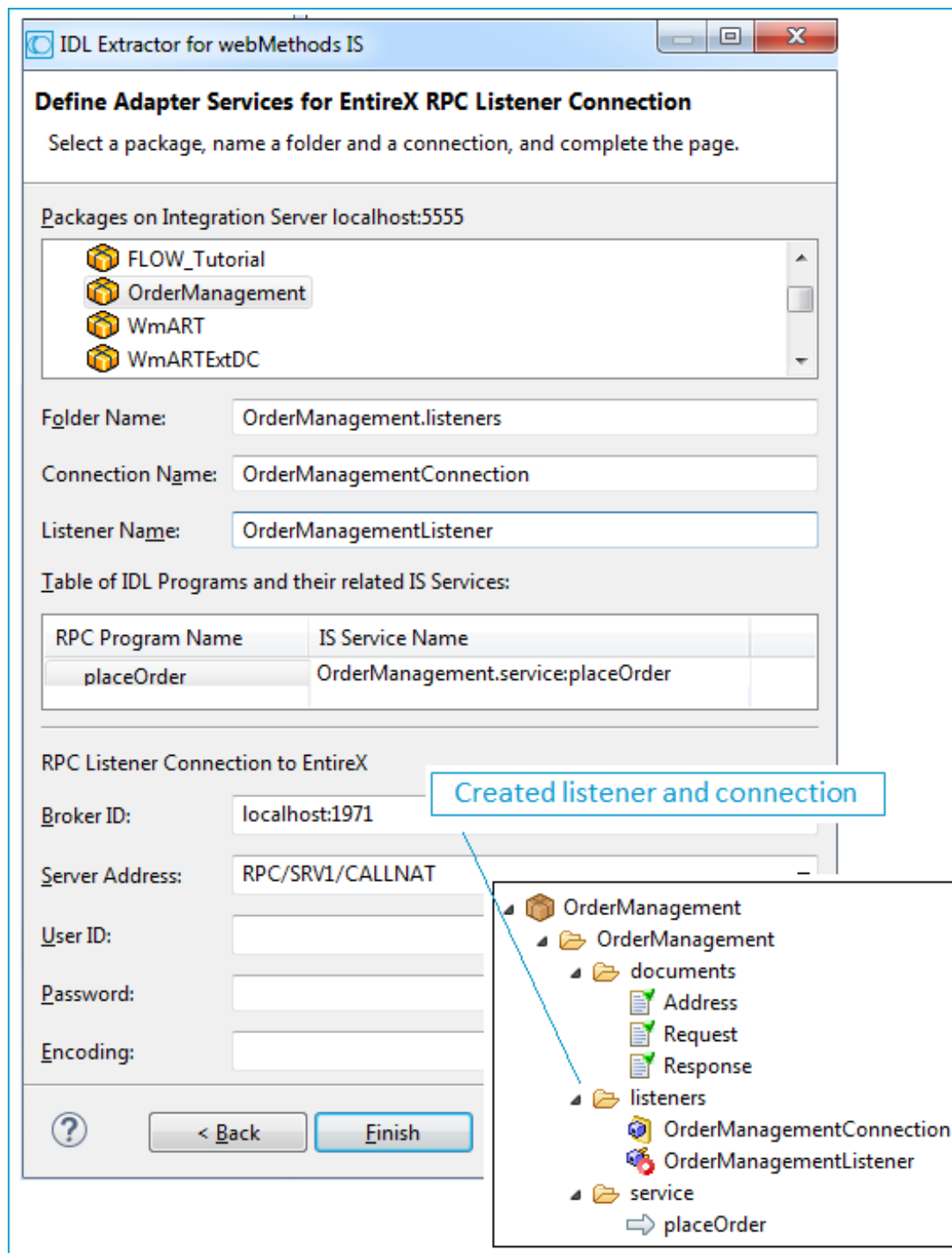
Note: COBOL only supports fixed-length data items, whereas webMethods IS has variable length types, often without any length restrictions (constraints). Parameters with no specified constraints will take the default fixed length. You can specify this default in field alphanumeric with fixed length. .

- 3 A listener is required for calling webMethods IS from an RPC client, so check **Create Listener Objects in Integration Server**.
- 4 Press **Next**.

Creating Listener Objects in Integration Server

➤ To create Listener Objects in Integration Server

- 1 Select a Listener connection type used for inbound connection to the Integration Server.
 - An EntireX RPC Listener connection is the standard way and is always available.
 - If your interface contains only In parameters, an EntireX Reliable RPC Listener connection is available.
 - If it is enabled by the license for the webMethods Integration Server:
 - an EntireX Direct RPC Listener connection is available
 - if your interfaces contain only In parameters, an EntireX Direct Reliable RPC Listener connection is available



- On this page, you set the properties for the listener objects to create; these objects will wait for the incoming COBOL requests. Select the package and specify the folder name into which the listener is generated. If the folder does not exist, it will be created automatically.

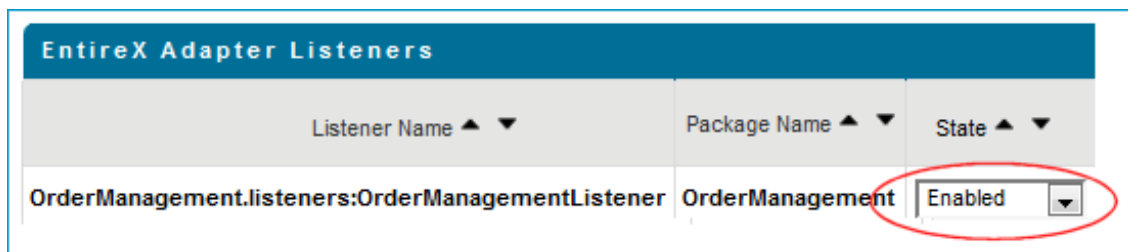
- 3 Keep the defaults given for **RPC Listener Connection to EntireX**. The **Broker ID** is a TCP/IP or SSL/TLS address and **Server Address** is an EntireX-specific namespace to locate a target server (here: *OrderManagementListener*).
- 4 When creating a connection, a package dependency is added such that the selected package depends on webMethods EntireX (the package WmEntireX) with the version currently used.
- 5 Press **Finish**. The listener and the connection appear in the package navigator. For the package navigator, switch to the service development perspective by choosing **Window > Perspective > Open Perspective > Service Development**.

Testing the Extraction Results (Optional)

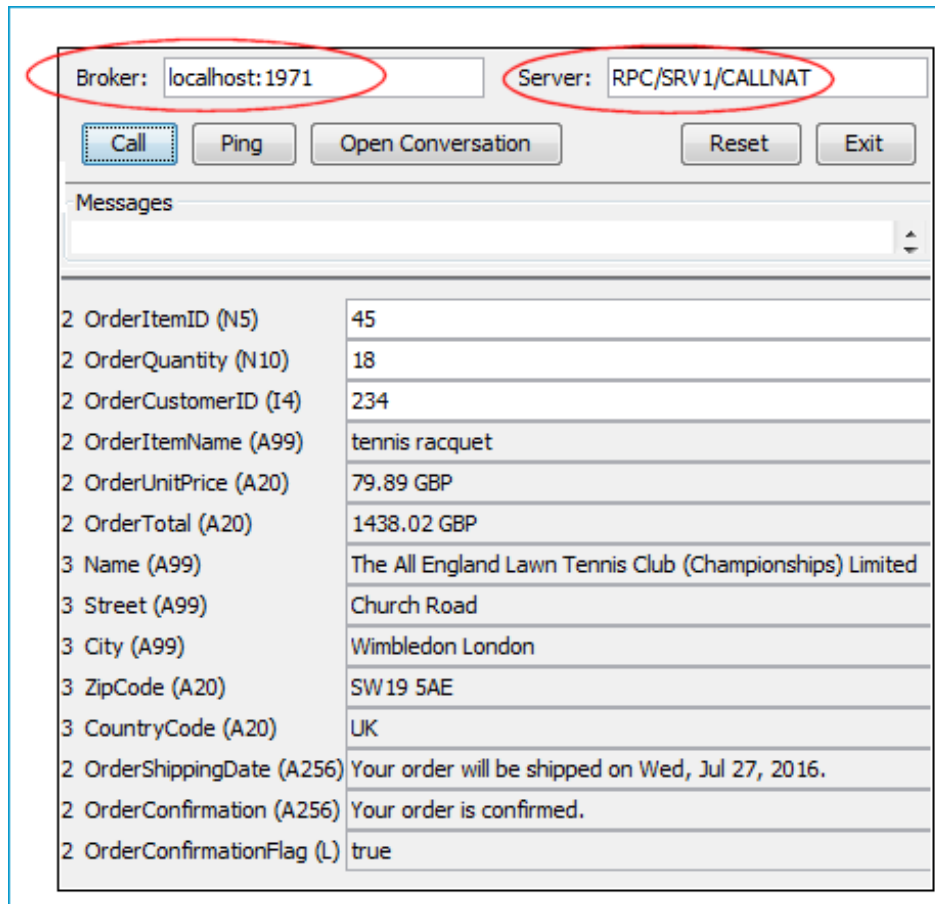
You can test the results of the extraction and the server back end, using the EntireX IDL Tester.

➤ To test the extraction results

- 1 For EntireX RPC (see [Task 3](#) in the [Introduction](#)) and an RPC Listener Connection, make sure EntireX Broker is running.
- 2 Enable the EntireX Adapter Listener.



- 3 Activate the EntireX perspective by choosing **Window > Perspective > Open Perspective > EntireX**, and from the context menu of the IDL file, choose **IDL Tester**.
- 4 This step depends on the method used to call the server (see [Task 3](#) in the [Introduction](#)).
 - For *EntireX RPC*, enter the same TCP/IP address for **Broker** that you supplied when you created the RPC Listener Connection to EntireX (localhost:1971).
 - For *Direct RPC*, that is, you generated a Direct RPC Listener Connection, enter the TCP/IP address from the Integration Server (localhost:1971).
- 5 For **Server**, enter the values you specified for the server address (RPC/SRV1/CALLNAT).



Broker: localhost:1971 Server: RPC/SRV1/CALLNAT

Call Ping Open Conversation Reset Exit

Messages

2 OrderItemID (N5)	45
2 OrderQuantity (N10)	18
2 OrderCustomerID (I4)	234
2 OrderItemName (A99)	tennis racquet
2 OrderUnitPrice (A20)	79.89 GBP
2 OrderTotal (A20)	1438.02 GBP
3 Name (A99)	The All England Lawn Tennis Club (Championships) Limited
3 Street (A99)	Church Road
3 City (A99)	Wimbledon London
3 ZipCode (A20)	SW 19 5AE
3 CountryCode (A20)	UK
2 OrderShippingDate (A256)	Your order will be shipped on Wed, Jul 27, 2016.
2 OrderConfirmation (A256)	Your order is confirmed.
2 OrderConfirmationFlag (L)	true

Task 2: Generate Client Interface Objects and Build Client Application

This section covers the following topics:

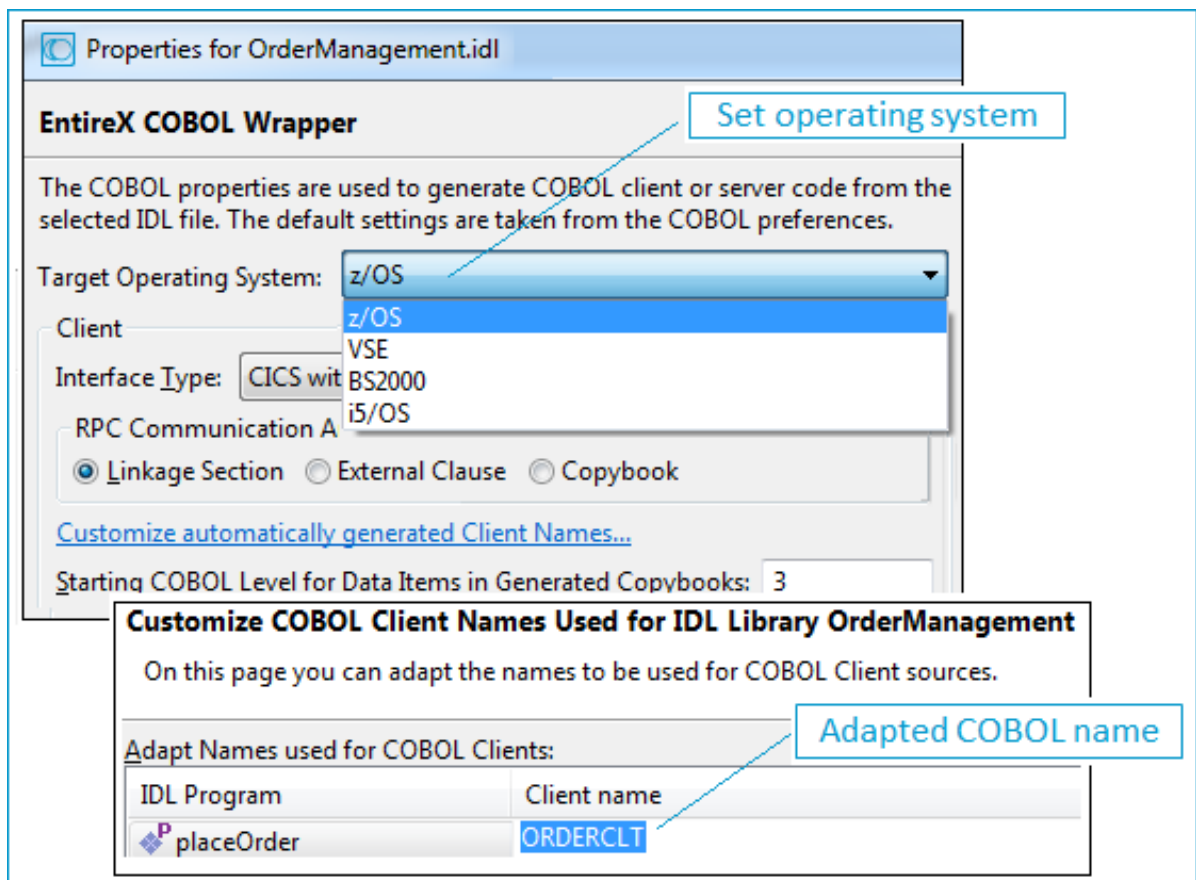
- [Setting Generation Options \(Properties\) for the COBOL Wrapper](#)
- [Generating COBOL Code](#)

■ Building the Client Application

Setting Generation Options (Properties) for the COBOL Wrapper

➤ To set generation options (properties) for the COBOL Wrapper

- 1 To generate suitable COBOL code for your environment, first set the correct target operating system. Make sure the EntireX perspective is active (**Window > Perspective > Open Perspective > EntireX**) and call the properties of the COBOL Wrapper.



- 2 From the context menu of the IDL file, choose **Properties > EntireX COBOL Wrapper**. On the COBOL Wrapper **Properties** page, set the **Target Operating System**
- 3 Adapt the name of the generated COBOL program: Choose **Customize automatically generated Client Names**. Take ORDERCLT as the COBOL program's name.

Generating COBOL Code

> **To generate COBOL code**

- For COBOL code generation, make sure the EntireX Perspective is active: Choose **Window > Perspective > Open Perspective > EntireX**. From context menu of the IDL file, choose **COBOL > Generate RPC Client**. This generates the COBOL client *ORDERCLT* in folder *client* and a copybook in folder *include*:

The screenshot displays the COBOL client development environment. The main window shows the source code for the ORDERCLT program, which is a COBOL RPC Client Interface Object. The code includes a header section with the program name and a data section with various fields and their constraints. The constraints are listed as follows:

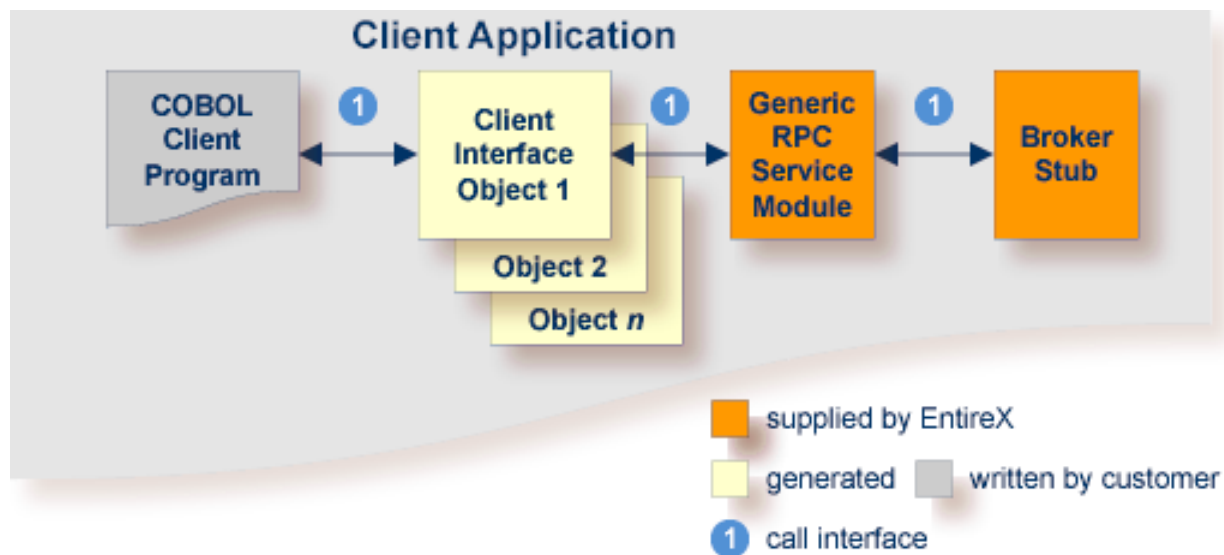
Field	Constraint
03 REQUEST.	
04 ORDERITEMID	PIC 9(5).
04 ORDERQUANTITY	PIC 9(10).
04 ORDERCUSTOMERID	PIC S9(9) BINARY.
03 RESPONSE.	
04 ORDERITEMNAME	PIC X(99).
04 ORDERUNITPRICE	PIC X(20).
04 ORDERTOTAL	PIC X(20).
04 ORDERCUSTOMERADDRESS.	
05 NAME	PIC X(99).
05 STREET	PIC X(99).
05 CITY	PIC X(20).
05 ZIPCODE	PIC X(20).
05 COUNTRYCODE	PIC X(256).
04 ORDERSHIPPINGDATE	PIC X(256).
04 ORDERCONFIRMATION	PIC X.
04 ORDERCONFIRMATIONFLAG	

The Package Explorer on the right shows the project structure, including the source files and the generated IDL file (OrderManagement.idl). The IDL file is highlighted, indicating it is the current view.

The copybook is the COBOL programmer's API. It is for use in the COBOL application.

As you would expect, the length of generated COBOL parameters exactly matches the lengths extracted. COBOL parameters always have a fixed length defined at compile time. They are not altered at runtime to reflect actual string length as in modern programming languages. This means that if you choose your default length (see [Extracting the Interface of the webMethods IS Service](#)) you have to choose the maximum possible data size to ensure all data can be transmitted. However, this may needlessly increase the message size if no constraints are set for fields of shorter data length. Choose your defaults and constraints carefully.

Building the Client Application



In this scenario, the COBOL client program, every generated client interface object, generic RPC services module and the broker stub are linked together to the client application.

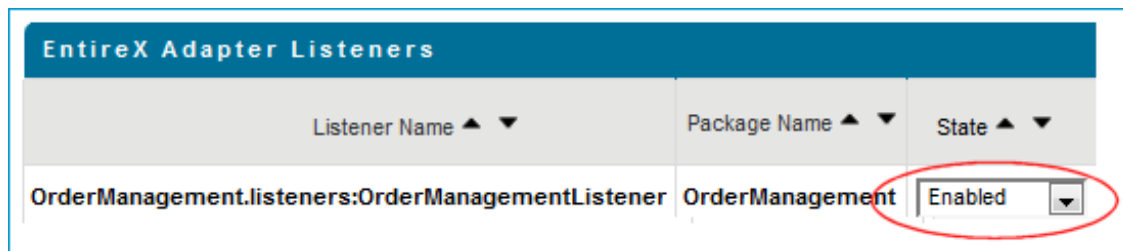
See *Using the COBOL Wrapper for Batch (z/OS, BS2000, z/VSE and IBM i)*.

Task 3: Execute the Call from COBOL to REST

Use the EntireX IDL Tester to execute the call.

➤ To execute the call

- 1 For EntireX RPC (see [Task 3](#) in the [Introduction](#)) and an RPC Listener Connection, make sure EntireX Broker is running.
- 2 Enable the EntireX Adapter Listener.



- 3 Activate the EntireX perspective by choosing **Window > Perspective > Open Perspective > EntireX**, and from the context menu of the IDL file, choose **IDL Tester**.

- 4 This step depends on the method used to call the server (see [Task 3](#) in the [Introduction](#)).
 - For *EntireX RPC*, enter the same TCP/IP address for **Broker** that you supplied when you created the RPC Listener Connection to EntireX (localhost:1971).
 - For *Direct RPC*, that is, you generated a Direct RPC Listener Connection, enter the TCP/IP address from the Integration Server (localhost:1971).
- 5 For **Server**, enter the values you specified for the server address (RPC/SRV1/CALLNAT).

The screenshot shows a REST client interface. At the top, there are two input fields: "Broker:" with the value "localhost:1971" and "Server:" with the value "RPC/SRV1/CALLNAT". Both fields are circled in red. Below these fields are five buttons: "Call" (highlighted with a dashed border), "Ping", "Open Conversation", "Reset", and "Exit". Below the buttons is a "Messages" section with a scrollable list of data. The data is organized into two columns: a list of field names with their data types in parentheses, and a list of corresponding values.

2 OrderItemID (N5)	45
2 OrderQuantity (N10)	18
2 OrderCustomerID (I4)	234
2 OrderItemName (A99)	tennis racquet
2 OrderUnitPrice (A20)	79.89 GBP
2 OrderTotal (A20)	1438.02 GBP
3 Name (A99)	The All England Lawn Tennis Club (Championships) Limited
3 Street (A99)	Church Road
3 City (A99)	Wimbledon London
3 ZipCode (A20)	SW19 5AE
3 CountryCode (A20)	UK
2 OrderShippingDate (A256)	Your order will be shipped on Wed, Jul 27, 2016.
2 OrderConfirmation (A256)	Your order is confirmed.
2 OrderConfirmationFlag (L)	true

