

webMethods EntireX

Calling COBOL from REST

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-COB2REST-109-20230403

Table of Contents

Calling COBOL from REST	vii
1 About this Documentation	1
Document Conventions	2
Online Information and Support	2
Data Protection	3
2 Calling COBOL on z/OS from REST	5
3 Calling COBOL on z/OS CICS from REST	7
4 Calling COBOL Channel Container (Zero Footprint using CICS IPIC) on z/OS CICS from REST	9
Introduction	11
What do I need to Install for this Scenario?	12
Task 1: Extract the Interface of a COBOL Server	13
Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server	25
Task 3: Execute the Call from the REST Client to COBOL	31
5 Calling COBOL Channel Container (Minimal Footprint using CICS Socket Listener) on z/OS CICS from REST	33
Introduction	35
What do I need to Install for this Scenario?	37
Task 1: Extract the Interface of a COBOL Server	38
Task 2: Generate the Connection and Adapter Services in Integration Server	50
Task 3: Execute the Call from the REST Client to COBOL	56
6 Calling COBOL Channel Container on z/OS CICS from REST	57
Introduction	59
What do I need to Install for this Scenario?	61
Task 1: Extract the Interface of a COBOL Server	62
Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server	74
Task 3: Execute the Call from the REST Client to COBOL	80
7 Calling COBOL DFHCOMMAREA (Zero Footprint using CICS ECI) on z/OS CICS from REST	81
Introduction	82
What do I need to Install for this Scenario?	84
Task 1: Extract the Interface of a COBOL Server	85
Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server	97
Task 3: Execute the Call from the REST Client to COBOL	103
8 Calling COBOL DFHCOMMAREA (Zero Footprint using CICS IPIC) on z/OS CICS from REST	105
Introduction	106
What do I need to Install for this Scenario?	107
Task 1: Extract the Interface of a COBOL Server	108
Task 2: Generate the Connection and Adapter Services in Integration Server	121

Task 3: Execute the Call from the REST Client to COBOL	127
9 Calling COBOL DFHCOMMAREA (Minimal Footprint Using CICS Socket Listener)	
from REST	129
Introduction	130
What do I need to Install for this Scenario?	132
Task 1: Extract the Interface of a COBOL Server	133
Task 2: Generate the REST Resources, Connection and Adapter Services in	
Integration Server	146
Task 3: Execute the Call from the REST Client to COBOL	152
10 Calling COBOL DFHCOMMAREA on z/OS CICS from REST	153
Introduction	154
What do I need to Install for this Scenario?	155
Task 1: Extract the Interface of a COBOL Server	157
Task 2: Generate the REST Resources, Connection and Adapter Services in	
Integration Server	170
Task 3: Execute the Call from the REST Client to COBOL	176
11 Calling COBOL Large Buffer on z/OS CICS from REST	177
Introduction	179
What do I need to Install for this Scenario?	181
Task 1: Extract the Interface of a COBOL Server	182
Task 2: Generate the REST Resources, Connection and Adapter Services in	
Integration Server	195
Task 3: Execute the Call from the REST Client to COBOL	201
12 Calling COBOL Large Buffer (Minimal Footprint using CICS Socket Listener) on	
z/OS CICS from REST	203
Introduction	205
What do I need to Install for this Scenario?	207
Task 1: Extract the Interface of a COBOL Server	208
Task 2: Generate the REST Resources, Connection and Adapter Services in	
Integration Server	221
Task 3: Execute the Call from the REST Client to COBOL	227
13 Calling COBOL on z/OS Batch from REST	229
Introduction	231
What do I need to Install for this Scenario?	233
Task 1: Extract the Interface of a COBOL Server	234
Task 2: Generate the REST Resources, Connection and Adapter Services in	
Integration Server	246
Task 3: Execute the Call from the REST Client to COBOL	252
14 Calling COBOL on z/OS IMS from REST	253
15 Calling COBOL on z/OS IMS MPP (Zero Footprint using IMS Connect) from	
REST	255
Introduction	257
What do I need to Install for this Scenario?	258
Task 1: Extract the Interface of a COBOL Server	259

Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server	271
Task 3: Execute the Call from the REST Client to COBOL	277
16 Calling COBOL on z/OS IMS BMP (Batch) from REST	279
Introduction	281
What do I need to Install for this Scenario?	283
Task 1: Extract the Interface of a COBOL Server	284
Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server	296
Task 3: Execute the Call from the REST Client to COBOL	302
17 Calling COBOL on BS2000 from REST	303
Introduction	305
What do I need to Install for this Scenario?	306
Task 1: Extract the Interface of a COBOL Server	307
Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server	319
Task 3: Execute the Call from the REST Client to COBOL	325
18 Calling COBOL on z/VSE from REST	327
19 Calling COBOL on z/VSE CICS from REST	329
20 Calling COBOL DFHCOMMAREA (Zero Footprint using CICS ECI) on z/VSE CICS from REST	331
Introduction	332
What do I need to Install for this Scenario?	334
Task 1: Extract the Interface of a COBOL Server	335
Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server	348
Task 3: Execute the Call from the REST Client to COBOL	354
21 Calling COBOL DFHCOMMAREA (Minimal Footprint Using CICS Socket Listener) on z/VSE CICS from REST	355
Introduction	356
What do I need to Install for this Scenario?	358
Task 1: Extract the Interface of a COBOL Server	359
Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server	372
Task 3: Execute the Call from the REST Client to COBOL	378
22 Calling COBOL Channel Container (Minimal Footprint Using CICS Socket Listener) on z/VSE CICS from REST	379
Introduction	381
What do I need to Install for this Scenario?	383
Task 1: Extract the Interface of a COBOL Server	384
Task 2: Generate the Connection and Adapter Services in Integration Server	396
Task 3: Execute the Call from the REST Client to COBOL	402
23 Calling COBOL Large Buffer (Minimal Footprint Using CICS Socket Listener) from REST	403
Introduction	405

What do I need to Install for this Scenario?	407
Task 1: Extract the Interface of a COBOL Server	408
Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server	421
Task 3: Execute the Call from the REST Client to COBOL	427
24 Calling COBOL on IBM i from REST	429
Introduction	431
What do I need to Install for this Scenario?	433
Task 1: Extract the Interface of a COBOL Server	434
Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server	446
Task 3: Execute the Call from the REST Client to COBOL	452

Calling COBOL from REST

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:

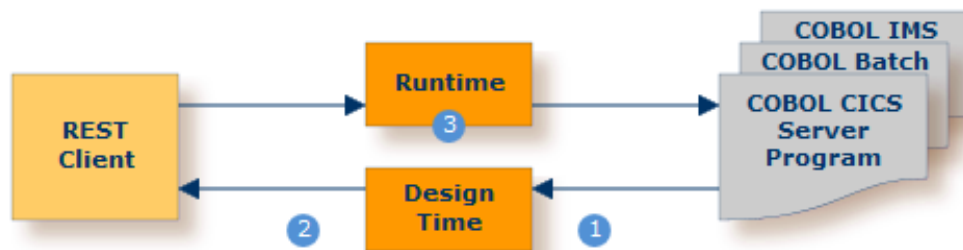
- Download products, updates and fixes.
- Search the Knowledge Center for technical information and tips.
- Subscribe to early warnings and critical alerts.
- Open and update support incidents.
- Add product feature requests.

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 COBOL on z/OS from REST

Under z/OS, a COBOL server can be called in different environments:



- ❶ Extract the interface of the COBOL server program. See *Using the IDL Extractor for COBOL - Overview*.
- ❷ Generate REST resources, connection and adapter services in Integration Server.
- ❸ Execute the call from the REST client to the COBOL server program.

Continue with the appropriate scenario:

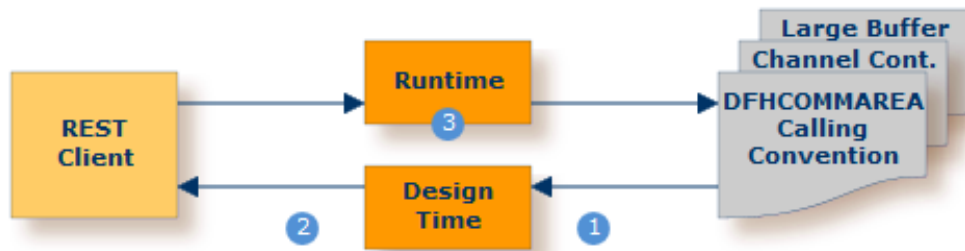
- **CICS** - Choose your CICS calling convention:
 - **Channel Container**
 - **Channel Container**
 - **DFHCOMMAREA** (zero footprint using CICS IPIC)
 - **DFHCOMMAREA** (zero footprint using CICS ECI)
 - **DFHCOMMAREA** (using RPC Server for CICS)
 - **Large Buffer**
- **Batch**

- **IMS** - Choose your IMS environment:
 - **MPP** - online transactions (zero footprint using IMS Connect)
 - **BMP** - batch processing (using RPC Server for IMS)

3

Calling COBOL on z/OS CICS from REST

There are different styles (interface types) for calling a COBOL server.



- ❶ Extract the interface of the COBOL server program. See *Using the IDL Extractor for COBOL - Overview*.
- ❷ Generate REST resources, connection and adapter services in Integration Server.
- ❸ Execute the call from the REST client to the COBOL server program.

➤ To determine the appropriate scenario

- 1 First it is important to know the interface type of your COBOL server. Analyzing the technique used to access the interface with COBOL and CICS statements is the safest way to determine this. The following interface types are available and are described in the IDL Extractor for COBOL documentation:
 - CICS with *DFHCOMMAREA Calling Convention*
 - CICS with *Channel Container Calling Convention*
 - CICS with *DFHCOMMAREA Large Buffer Interface*

There is no clear and easy indication how to identify the interface type of a CICS COBOL server without COBOL and CICS knowledge. Below are some criteria that might help to determine the interface type. If you are unsure, consult a CICS COBOL specialist.

- The payload size of the CICS COBOL server is greater than 32 KB:
 - In this case it is *not* a DFHCOMMAREA interface, because the DFHCOMMAREA is limited to 32 KB.
 - It could be a large buffer or channel container interface, which are only limited by the storage (memory) available to them.
- The CICS COBOL server is located in a remote CICS region:
 - In this case it is *not* a large buffer interface, because large buffer programs must reside on the same CICS region as the caller, that is, the *RPC Server for CICS*.
 - It could be a DFHCOMMAREA or channel container interface, which can reside in a remote CICS region.



Note: The most used interface type is the DFHCOMMAREA interface. Large buffer and channel container interfaces are used much less frequently.

- 2 Second, for the DFHCOMMAREA interface only, you need to decide for the zero footprint or the RPC Server for CICS approach. Here are some guidelines:

Use the zero footprint approach if:

- you are already using and familiar with ECI
- you do not want to install any additional resources in your CICS environment

Use the RPC Server for CICS if:

- you are already using (or planning on using) EntireX Broker for extended messaging

When you are sure which interface type and approach you are using, continue with the appropriate scenario:

- **Channel Container** (zero footprint using CICS IPIC)
- **Channel Container** (minimal footprint using CICS Socket Listener)
- **Channel Container**
- **DFHCOMMAREA** (zero footprint using CICS IPIC)
- **DFHCOMMAREA** (zero footprint using CICS ECI)
- **DFHCOMMAREA** (using RPC Server for CICS)
- **Large Buffer**
- **Large Buffer** (minimal footprint using CICS Socket Listener)

4

Calling COBOL Channel Container (Zero Footprint using CICS IPIC) on z/OS CICS from REST

■ Introduction	11
■ What do I need to Install for this Scenario?	12
■ Task 1: Extract the Interface of a COBOL Server	13
■ Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server	25
■ Task 3: Execute the Call from the REST Client to COBOL	31

This scenario uses the tools IDL Extractor for COBOL and Integration Server Wrapper of the Designer.

Introduction



- ❶ Extract the interface of the COBOL server program. See *Using the IDL Extractor for COBOL - Overview*.
- ❷ Generate REST resources, connection and adapter services in Integration Server.
- ❸ Execute the call from the REST client to the COBOL server program.

This scenario makes the following important assumptions:

For Task 1:

- You have a working COBOL Channel Container server. For illustration and examples on such a server, see *CICS with Channel Container Calling Convention* under *Introduction to the IDL Extractor for COBOL*.
- You have access to the related COBOL sources and copybooks as files on your local machine. The minimum requirement is the `DATA DIVISION` of the interface.



For Tasks 2 and 3:

- To call the COBOL server program at runtime using the *EntireX CICS IPIC* connection method, you need to configure the CICS IPIC TCP/IP service within your CICS region. See *Preparing IBM CICS for IPIC* and *Connection Parameters for CICS IPIC Connections*.



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.

Task 1: Extract the Interface of a COBOL Server

- [Introduction](#)
- [Sample COBOL Server used in the Examples](#)
- [Extracting a COBOL Server - Fast-track Method](#)
- [Extracting a COBOL Server - Modern Method with User-defined Mapping](#)
- [Comparison Chart: Fast-track versus User-defined Mapping](#)
- [Testing the Extraction Results](#)

Introduction

Follow the instructions for extracting COBOL, see *Using the IDL Extractor for COBOL - Overview* and choose *Scenario I: Create New IDL and Server Mapping Files* if this is your first extraction. Learn how to invoke the IDL Extractor for COBOL, create a COBOL Extractor Environment and about the possibilities to select the COBOL source.

The extraction process creates the following EntireX metafiles:

- IDL file. A Software AG IDL file contains definitions of the interface between client and server. See *Software AG IDL File* in the IDL Editor documentation in the IDL Editor documentation.
- Server mapping file (optional). The mapping file is a Designer file with extension .cvm that contains COBOL-specific mapping information. See *Server Mapping Files for COBOL* in the Designer documentation.

Sample COBOL Server used in the Examples

The following COBOL server is used to illustrate the features of the IDL Extractor for COBOL. Imagine an existing COBOL server called `EMPLOYEE`. It implements the access logic for a `LIST` and `DETAILS` function to a database view `EMPLOYEE`:

```
IDENTIFICATION DIVISION.
PROGRAM-ID.      EMPLOYEE.
*****
*   Program Name      = EMPLOYEE
*   String Literal    = QUOTE
*****
01 DFHCOMMAREA.
02 OPERATION                                PIC X(1).
    88 OPERATION-LIST                        VALUE 'L'.
    88 OPERATION-DETAILS                    VALUE 'D'.
02 EMPLOYEE-ID                                PIC X(10).
02 EMPLOYEE-DETAILS.
    03 FIRSTNAME                            PIC X(20).
    03 SURNAME                              PIC X(20).
    03 DATE-BIRTH                          PIC X(12).
    03 DETAILS                             PIC X(100).
    03 FILLER REDEFINES DETAILS.
        04 ANNUAL-SALARY                    PACKED-DECIMAL PIC S9(9).
        04 VACATION                        PIC S9(2).
        04 LANGUAGE                        PIC X(3).
        04 FILLER                          PIC X(90).
02 EMPLOYEES-COUNT                          PIC 9(8) BINARY.
02 EMPLOYEES OCCURS 1 TO 99 DEPENDING ON EMPLOYEES-COUNT.
    03 IDENT                                PIC X(10).
    03 FIRSTNAME                            PIC X(20).
    03 SURNAME                              PIC X(20).
    03 DATE-BIRTH                          PIC X(12).
```

Two approaches for extracting the COBOL server are described below:

- **Fast-track**

Learn how EntireX helps you to connect the COBOL Server with quick results.

- **User-defined Mapping**

The following features are among those available with a user-defined mapping:

- **Shape the interface to your COBOL server**

You can minimize the interface by suppressing COBOL server fields or by providing constant values. Renaming interfaces allows you to specify a readable long name. These two features together - constants and renaming of interfaces - are most powerful when multiple interfaces are implemented in a single COBOL server. Each function of the COBOL server triggered by an operation code or function code can be modelled as a separate REST resource method/operation.

■ Select alternative mappings

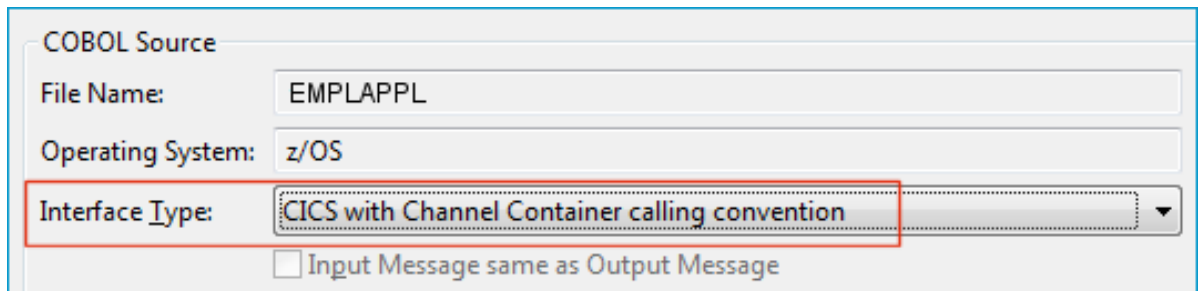
You can select alternative mappings for COBOL data items (REDEFINES, tables etc.) to enable scenarios with COBOL servers where data exchange is not fully described by the LINKAGE SECTION COBOL data items.

These approaches are described under *Fast-track Method* and *Modern Method* below.

Extracting a COBOL Server - Fast-track Method

➤ To extract the COBOL Server

- 1 Switch to the **EntireX** perspective.
- 2 Call the IDL Extractor for COBOL.
- 3 Set the correct interface type as described under *Step 4: Define the Extraction Settings and Start Extraction* under *Scenario I: Create New IDL and Server Mapping Files* in the IDL Extractor for COBOL documentation.



COBOL Source

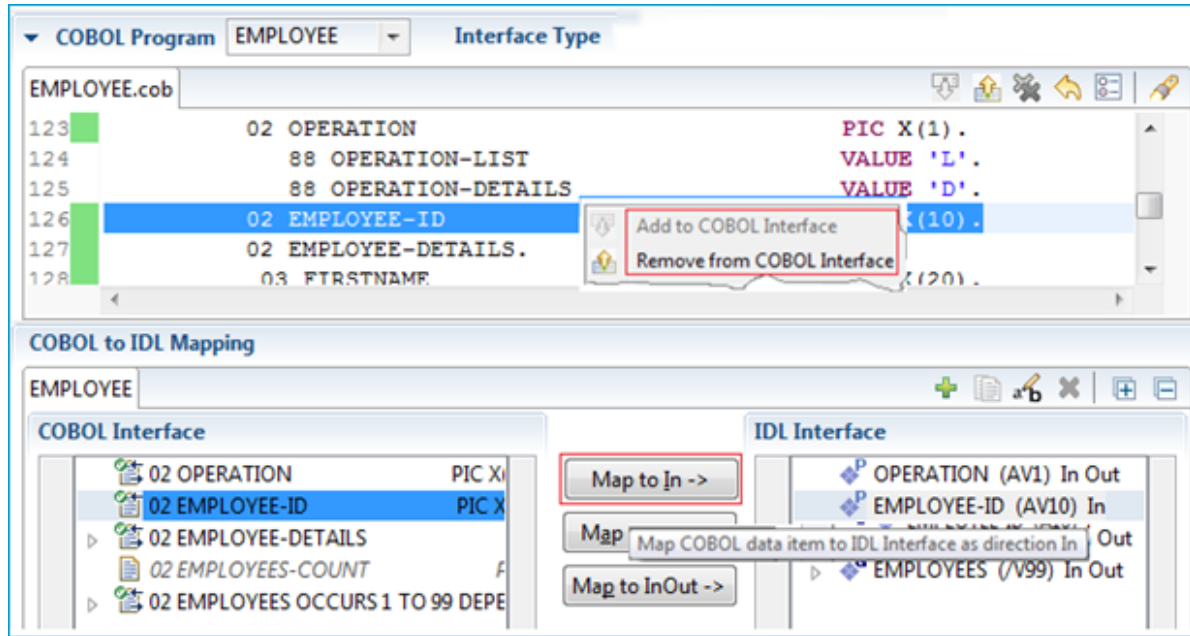
File Name: EMPLAPPL

Operating System: z/OS

Interface Type: CICS with Channel Container calling convention

☐ Input Message same as Output Message

- 4 Press **Next** to enter the COBOL Mapping Editor.



The purpose of the COBOL Mapping Editor is to map COBOL data items to IDL (Interface Definition Language) parameters. It consists of three main areas:

- The upper window pane is the COBOL program source.
- The window pane on the left-hand side is the COBOL interface, consisting of COBOL data items representing the parameters in your COBOL server program.
- The window pane on the right-hand side is the IDL interface consisting of IDL parameters.

For more information see *Mapping Editor User Interface* in the IDL Extractor for COBOL documentation.

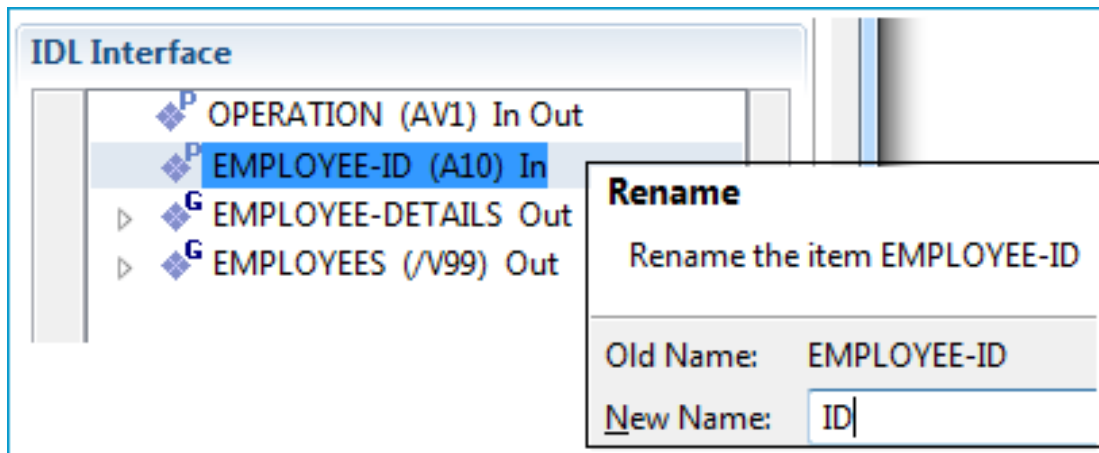
COBOL extraction is a two-step process:

- First, select the COBOL parameters using the **Add to** and **Remove from COBOL Interface** from the context menu on the COBOL data items in the upper window pane (COBOL source); in many situations the COBOL interface is detected by the Mapping Editor directly and you can omit this step.
- Then map the COBOL parameters to IDL directions using the **Map to In/Out/InOut** buttons. COBOL parameters do not have directions, they are always INOUT. This step can be left out for a quick test in non-production scenarios; for real production it is recommended to set IDL directions to reduce data transfer. You also have the option to assign a meaningful name to an IDL parameter if the original COBOL data item is not self-explanatory.

For our example:

- Set **Map to In** for COBOL data item `EMPLOYEE-ID` as IDL direction (see screenshot in previous step). For `EMPLOYEE-DETAILS` and `EMPLOYEE-DETAILS-DETAILS` perform the similar **Map to Out**.

- Use the function **Rename** from the context menu on the IDL parameters in the **IDL Interface** pane to rename COBOL parameter `EMPLOYEE-ID` to IDL parameter `ID`. Do the same for `EMPLOYEE-DETAILS`: Rename to `EMPLOYEE`.



- 5 Press **Finish**. The default extraction settings ensure you get useful results. The outcome is a Software AG IDL file (interface definition language), a Designer file with extension `.idl`:

```
library 'EMPLAPPL' is
program 'EMPLOYEE' is
  define data parameter
    1 OPERATION (AV1) In Out
    1 ID (AV10) In /* Original Name:EMPLOYEE-ID
    1 EMPLOYEE Out /* Original Name:EMPLOYEE-DETAILS
      2 FIRSTNAME (AV20)
      2 SURNAME (AV20)
      2 DATE-BIRTH (AV12)
      2 DETAILS (AV100)
    1 EMPLOYEES (/V99) Out
      2 IDENT (AV10)
      2 FIRSTNAME (AV20)
      2 SURNAME (AV20)
      2 DATE-BIRTH (AV12)
  end-define
```

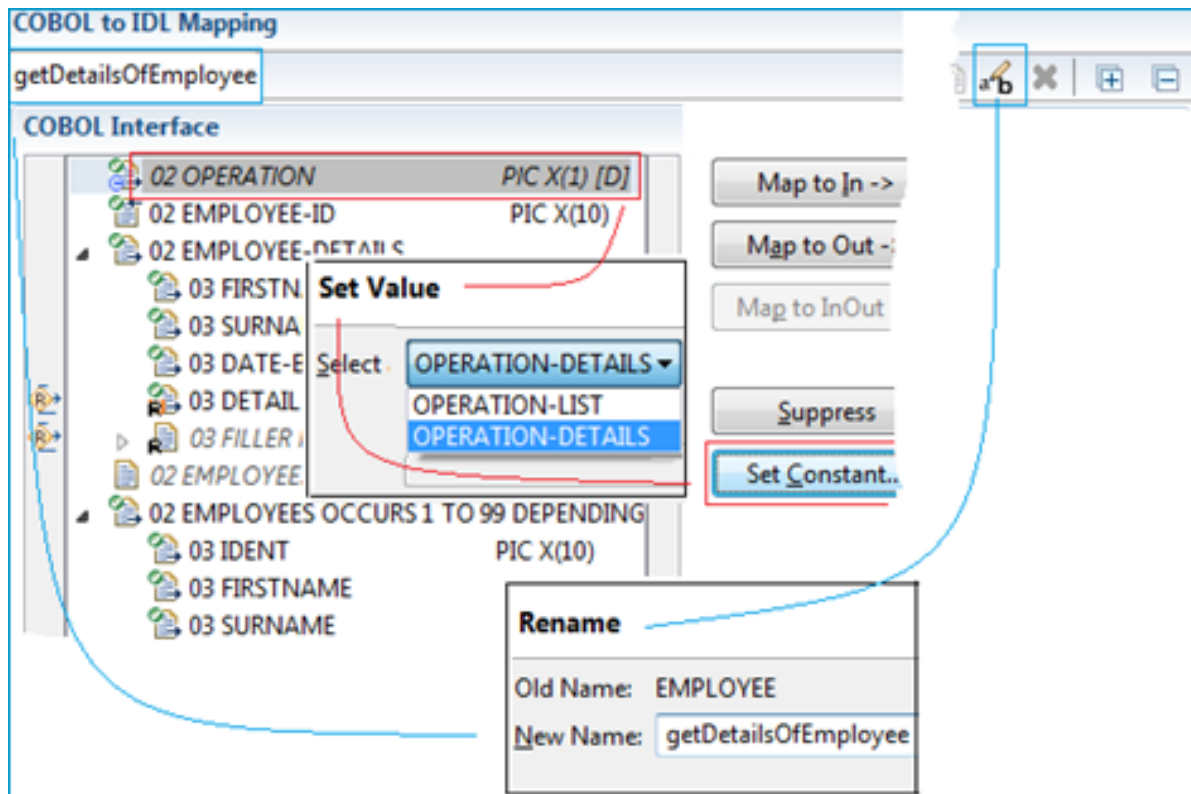
If required, a server mapping file with the same name as the IDL file but the extension `.cvm` may be generated too, for technical reasons. This file must always be stored together with the concomitant IDL file. For more information, see *Server Mapping Files for COBOL* in the Designer documentation. By default, the interface name `EMPLOYEE` is taken from the COBOL server program name.


Extracting a COBOL Server - Modern Method with User-defined Mapping

EntireX allows you to extract all implemented interfaces of the COBOL server separately. Instead of a large `EMPLOYEE` interface, separate interfaces `getListOfEmployees` and `getDetailsOfEmployee` are extracted. Each interface contains required parameters; obsolete parameters for an interface are suppressed, improving its usability.

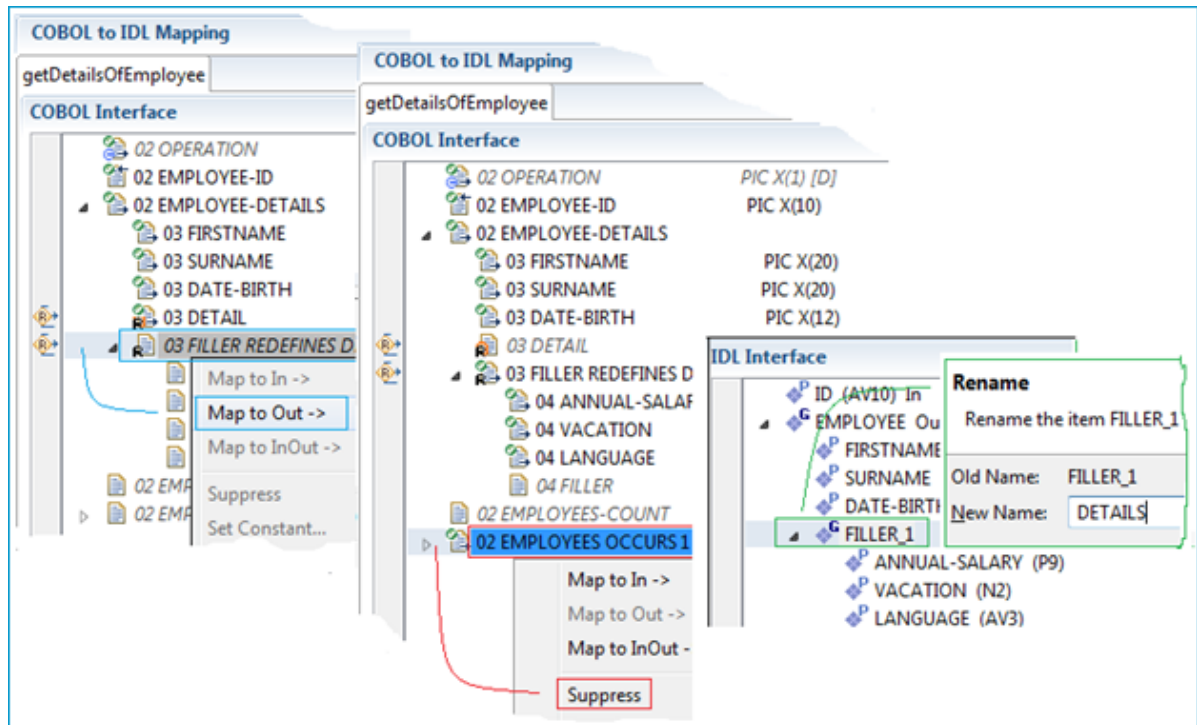
➤ To extract a COBOL server with a user-defined mapping

- 1 Shape `EMPLOYEE` to `getDetailsOfEmployee`:



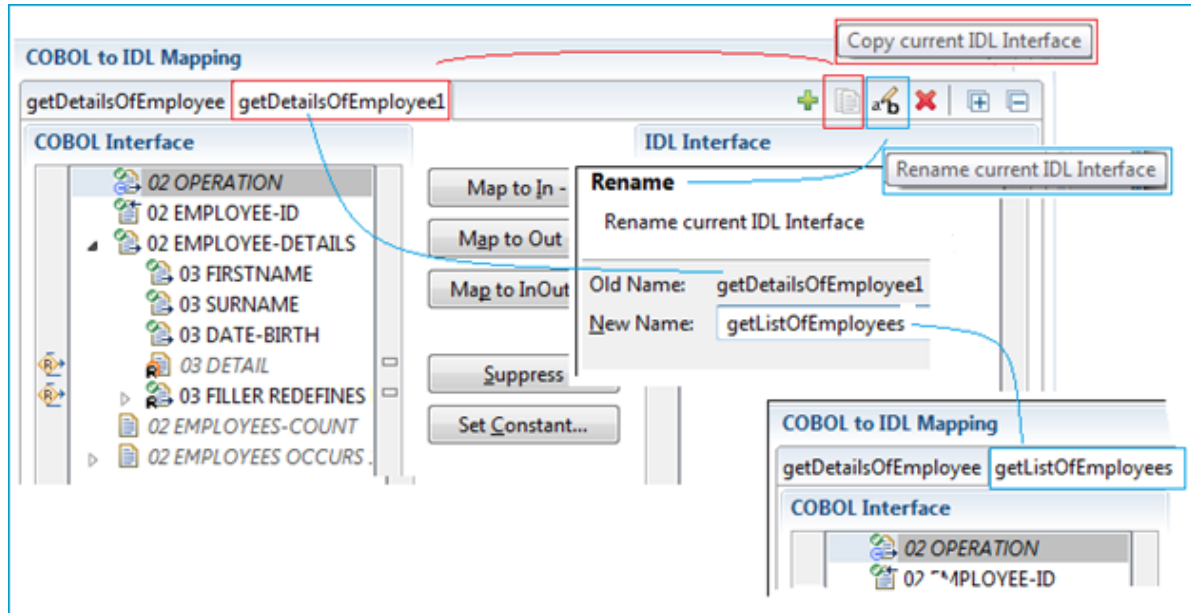
- Mark the `OPERATION` field in the COBOL **Interface** pane and press the **Set Constant** button. In the pop-up window, select `OPERATION-DETAILS` (red markers). The `OPERATION-DETAILS` value originates from COBOL level-88 data item (blue markers). If there is no COBOL level-88 enumeration type in the COBOL program defined, you can enter the function code manually in the pop-up window.
- By default the IDL interface gets the same name as your COBOL server: `EMPLOYEE`. Use the **Rename** button  from the toolbar of the COBOL Mapping Editor and change the IDL interface name to `getDetailsOfEmployee` (blue markers). This name is used as the IS service name later.

- 2 Continue shaping, making `getDetailsOfEmployee` easier to use:




- Use the context menu of the redefinition of the `DETAIL` field in the **COBOL Interface** pane, `FILLER REDEFINES DETAIL` and choose **Map to Out** (blue markers). A redefinition is a secondary layout (field types) of the same memory area. The redefinition contains more specific information (`ANNUAL-SALARY`, `VACATION`, etc.) than character buffer `DETAIL`. This is the reason why it is more useful in your interface.
- Use the context menu of `FILLER_1` in the IDL interface pane and rename `FILLER_1` to `DETAILS` (green markers) for a readable IS service field name later.
- The `DETAILS` function in the COBOL server does not make use of the `EMPLOYEES` field. Therefore you can leave it out of the interface: use the context menu of the `EMPLOYEES` field in the **COBOL Interface** pane and choose **Suppress** (red markers).


3 Shape `EMPLOYEE` to `getListOfEmployees` Function:



The COBOL Mapping Editor allows you to define multiple IDL interfaces to the same COBOL server:

- You create the additional IDL interface using the toolbar button **Copy current IDL interface**

 (red markers). A new tab `getDetailsOfEmployee1` is created. The previously extracted `getDetailsOfEmployee` interface still exists in the first tab. Once you reactivate the first tab, you will see the interface of `getDetailsOfEmployee` again.

- Use the **Rename** button  from the toolbar of the COBOL Mapping Editor and change the IDL interface name to `getListOfEmployees` (blue markers). This name is used as the IS service name later.

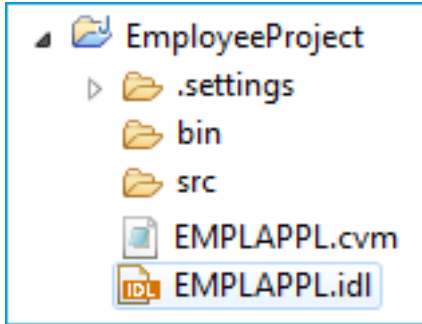
- For the `LIST` function you need the `OPERATION-LIST` value in the `OPERATION` field: Mark the `OPERATION` field in the **COBOL Interface** pane and press the **Set Constant** button. In the pop-up window, select `OPERATION-LIST` (similar to Step 1. above *Shape `EMPLOYEE` to `getDetailsOfEmployee`*; red markers).

- Use the context menu of the `EMPLOYEES` field in the **COBOL Interface** pane, `EMPLOYEES OCCURS 1 TO 99` and choose **Map to Out** (Similar to Step 2. above *Continue shaping, making `getDetailsOfEmployee` easier to use*; blue markers).

- Because the `EMPLOYEE` and `EMPLOYEE-DETAIL` fields are not used in the COBOL server `LIST` function, you leave them out in the IDL interface: use the context menu of `EMPLOYEE-DETAIL` field in the **COBOL Interface** pane and choose **Suppress** (similar to Step 2. above *Continue shaping, making `getDetailsOfEmployee` easier to use*; red markers). Do the same for the `EMPLOYEE-ID` field.

- 4 Press **Finish** to retrieve the extraction result in the form of a Software AG IDL file. At the same time, a server mapping file for COBOL (Designer file with extension `.cvm`) is created.

See *Server Mapping Files for COBOL* in the Designer documentation. The *Software AG IDL File* in the IDL Editor documentation describes the interfaces from the client point of view, while the server mapping file contains the mapping to the real COBOL server. Both of these files must be kept together and in sync, otherwise a call to the COBOL server may fail.



To summarize: We created two IDL interfaces. In the IDL file, these resulted in two IDL programs: `getListOfEmployees` and `getDetailsOfEmployee`. Both IDL programs were given readable names (Steps 1 and 3). Meaningful fields were kept, while superfluous fields were suppressed (Steps 1 and 3). The program `getDetailsOfEmployee` contains the redefined fields of parameter `DETAILS` (see below) mapped during extraction (Step 1).



Note: At runtime, the RPC client generated with the extracted interface will send data for the redesigned interfaces, while your COBOL server still expects `EMPLOYEE` data. The EntireX runtime transforms the incoming data stream from the RPC client, using the server mapping file.

```

program 'getDetailsOfEmployee' is
  define data parameter
    1 ID (AV10) In /* Original Name:EMPLOYEE-ID
    1 EMPLOYEE Out /* Original Name:EMPLOYEE-DETAILS
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
    2 DETAILS /* Original Name:FILLER_1
    3 ANNUAL-SALARY (P9)
    3 VACATION (N2)
    3 LANGUAGE (AV3)
  end-define
  program 'getListOfEmployees' is
    1 EMPLOYEES (/V99) Out
    2 IDENT (AV10)
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
  end-define
  
```

Comparison Chart: Fast-track versus User-defined Mapping

The following table highlights the differences when shaping an interface with a *user-defined mapping* compared with a *fast-path extraction*.

	Interface Shaping with User-defined Mapping	Fast-path Extraction
General	User-defined interface with dedicated mapping. For our example, the interfaces are small and tidy and more self-explanatory with long, readable interface names. They are easier to use with the hidden <code>OPERATION</code> field and the suppressed COBOL fields, which are not needed.	Automatic, quick result extraction with COBOL-like interfaces on IS. For our example, the IS interface <code>EMPLOYEE</code> matches exactly 1:1, field by field.
IS service(s)	Multiple small and handy IS services; each <code>OPERATION</code> code is mapped to a separate IS service.	One big IS service.
IS service name	Readable long name.	Short Subprogram name; up to 8 characters.
IS fields	Usage of Suppress and Set Constant reduces the message length. This keeps focus on relevant data items and keeps the client's interface clean. It may also improve performance.	The IS fields and COBOL server parameters match 1:1. As COBOL layout descriptions are sometimes used for many different purposes, irrelevant data items appear and clutter up the IS interface.
<code>OPERATION</code> parameter	Suppressed: the <code>OPERATION</code> parameter does not exist in the IS service as an IS field.	The <code>OPERATION</code> parameter exists in the IS service as an IS field and needs to be filled-in by the client endpoint.
<code>OPERATION</code> code	The <code>OPERATION</code> code is provided internally by EntireX runtime in the <code>OPERATION</code> field.	The <code>OPERATION</code> code needs to be specified in the IS service in the <code>OPERATION</code> field by the client endpoint.
<code>REDEFINE</code> parameters	Either the parameter that is redefined or one of its redefinitions is available as an IS field.	Only the parameter that is redefined is available as an IS field. Redefinitions of the parameter are not available as IS fields.

A user-defined mapping enables you also to define alternative mappings for COBOL data items (`REDEFINES`, tables etc.). These enable scenarios with COBOL servers where data exchange is not fully described by the `LINKAGE SECTION` COBOL data items. For more information refer to the IDL Extractor for COBOL documentation:

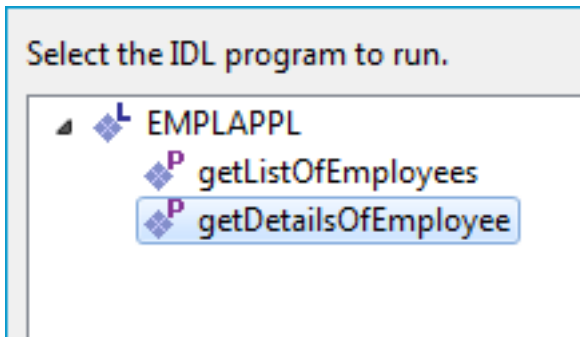
- *Mapping Editor IDL Interface Mapping Functions*
- *User-defined Mapping*

Testing the Extraction Results

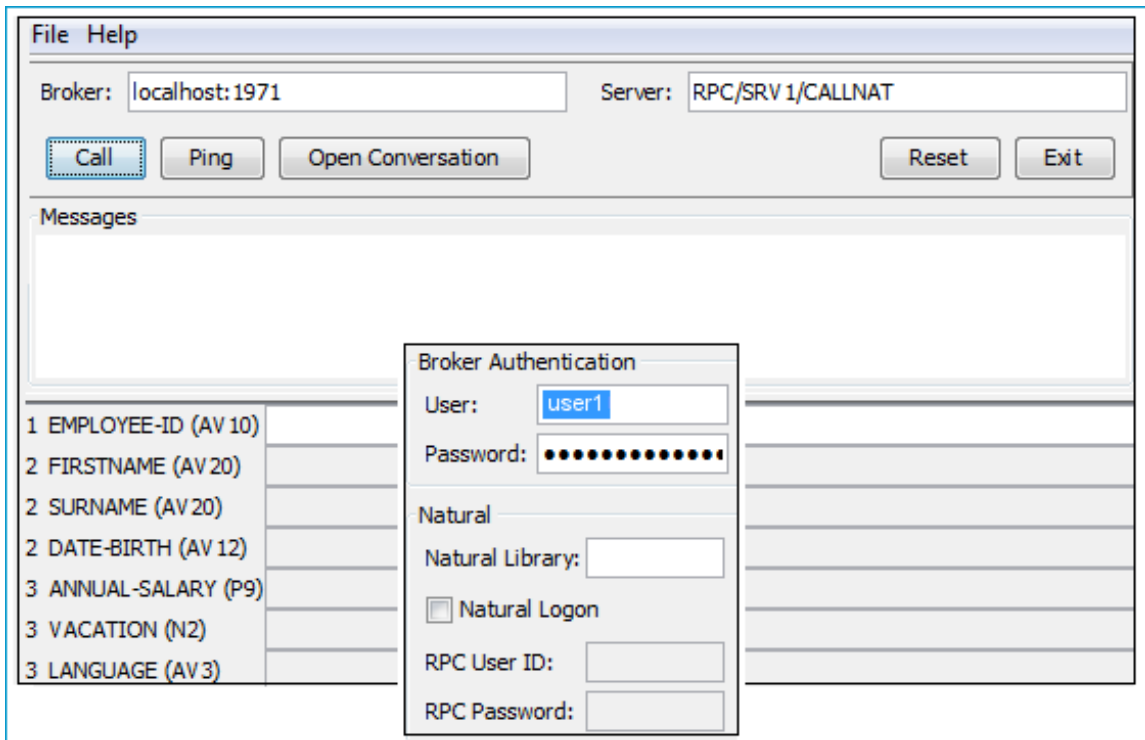
The following pictures use the extraction results described under *Extracting a COBOL Server - Modern Method with User-defined Mapping*.

> To test the extraction results (optional)

- 1 You can test the results of the extraction operation and the REST server back end, using the EntireX IDL Tester. From the context menu of the IDL file in the Designer, choose **Software AG IDL Tester**.



- 2 Select `getDetailsOfEmployee`.



Note that the broker and server parameters contain the explicit route to call the server program, and you can optionally ping the connection from this client. With the **File > Options** dialog, you can set:

- **User** and **Password** for *broker* authentication
- **RPC User ID** and **RPC Password** for *server* authentication

See *EntireX IDL Tester* in the Designer documentation.

- 3 Check the Integration Server log, the EntireX Adapter log or the RPC logs. Applies to all connection methods.

Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server

This section describes your first steps to create a new Integration Server connection. This is described in more detail under *Using the Integration Server Wrapper*, for example working with existing Integration Server connections. This section covers the following topics:

- [Step 1: Start the Integration Server Wrapper Wizard](#)
- [Step 2: Create a New Integration Server Connection](#)
- [Step 3: Select the Connection Type](#)
- [Step 4: Define Adapter Services for a CICS IPIC Connection](#)
- [Step 5: Create or Update a REST Resource](#)
- [Step 6: Edit the REST Resource](#)

Step 1: Start the Integration Server Wrapper Wizard

➤ To start the Integration Server Wrapper wizard

- 1 In the context menu of a Software AG IDL file, choose **Integration Server > Generate web-Methods IS Connection**.

This starts the wizard with a list of existing Integration Server Wrapper connections.



Note: If the selected IDL file is not valid because of a syntax error, an error dialog comes up and the wizard does not start.

- 2 Continue with *Step 2: Create a New Integration Server Connection*.

Step 2: Create a New Integration Server Connection

Define New Integration Server Connection
Add host:port, user and password for a new Integration Server

Server:

User:

Password:

☐ Use secure connection

➤ To create a new Integration Server connection

- 1 Define the new Integration Server connection on the wizard page.



Notes:

1. The only required field is **Server**. Enter the hostname of the Integration Server including an optional port number. If no port number is specified, port number defaults to "5555". The **Integration Server Authentication** can be passed with the **User** and **Password** fields.
 2. Optional settings are for secure connections. The **Truststore for HTTPS** contains all signed certificates and must be a valid truststore.
 3. The check box **Verify host name** checks that the hostname is entered in the stored certificate.
 4. When the Integration Server has **Client Authentication** enabled, you can specify your **Keystore** file and keystore **Password**.
 5. For managing Integration Server connections, see [Preferences](#).
- 2 Choose **Next** and continue with *Step 3: Select the Connection Type*.

Step 3: Select the Connection Type

☒ Create a new EntireX Adapter connection of type **CICS IPIC Connection**

☐ Update Adapter Services or Listeners to an existing Integration Server connection:

List of Connections

Connection Name	Package Name	Connection Type	Enabled
Employees:EmployeesConnection	Employees	EntireX RPC Connection	Yes

Total: 1

☒ Map Software AG IDL data types to String

☒ Create or Update REST resource

> To create a new connection

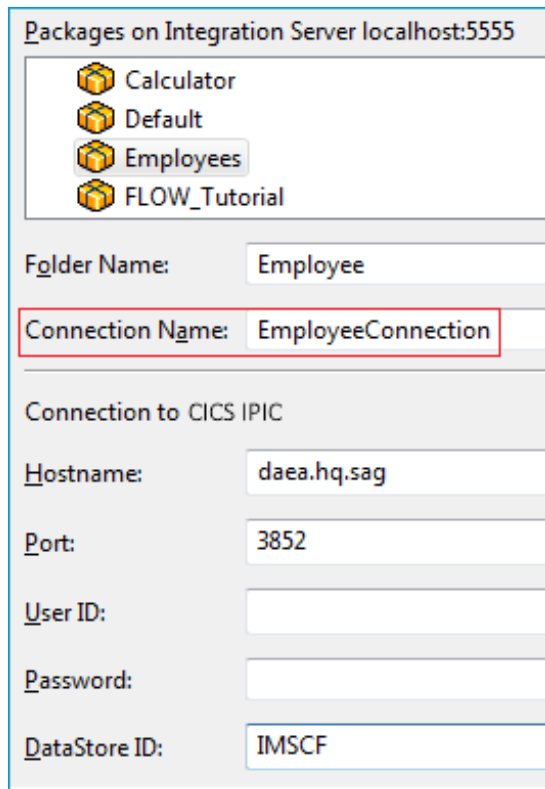
- 1 Select a connection type from the drop down list. Connection types are described under *EntireX Adapter Connections* in the EntireX Adapter documentation and *Introduction to the Integration Server Wrapper*.



Note: The list of connection types is filtered: connection types that require a license are only shown if a corresponding license file is available. Reliable RPC connections are only shown if all IDL programs contain only IN parameters. Also, if a server mapping file is available, only those connection types that support the interface type specified in the server mapping file are shown.

- 2 Check the box **Create or Update REST resource**.
- 3 Click **Next** and continue with *Step 4: Define Adapter Services for a CICS IPIC Connection*.

Step 4: Define Adapter Services for a CICS IPIC Connection



Packages on Integration Server localhost:5555

- Calculator
- Default
- Employees
- FLOW_Tutorial

Folder Name: Employee

Connection Name: EmployeeConnection

Connection to CICS IPIC

Hostname: daea.hq.sag

Port: 3852

User ID:

Password:

DataStore ID: IMSCF

➤ To create a connection and related adapter services

- 1 Select a package for the created objects.
- 2 Define a folder name. If the folder does not exist, it will be created.
- 3 Define a connection name.
- 4 Define the parameters of the connection type. For details, see *Connection Parameters for CICS IPIC Connections* in the EntireX Adapter documentation.

As a result, the folder will contain the connection and the adapter services (one for each IDL program). The name of a service is the same as the respective IDL program.

The default settings for the adapter services are:

- the **Default** package; if not available, the first package
- the IDL library name for the **Folder Name**
- the IDL library name with the suffix "Connection" for the **Connection Name**



Note: 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.

Step 5: Create or Update a REST Resource

> To create or update a REST resource

Make sure checkbox **Create or Update REST resource** is checked.

- 1 Select a package.

Packages on Integration Server

- Default
- Employees
- WmAdmin
- WmART

Folder Name: Employees

REST Resource Name: employees

☒ Update existing Resource

☒ Create or Update simplified REST services for this REST resource

- 2 Define a folder name. If the folder does not exist, it will be created.
- 3 Define the REST Resource name.
- 4 Check **Update existing Resource** if you want to add additional services to an existing resource.
- 5 Check **Create or Update simplified REST services for this REST resource** if you want to generate additional services with a simplified REST API. This allows you to use a simple HTTP-GET method without a payload.

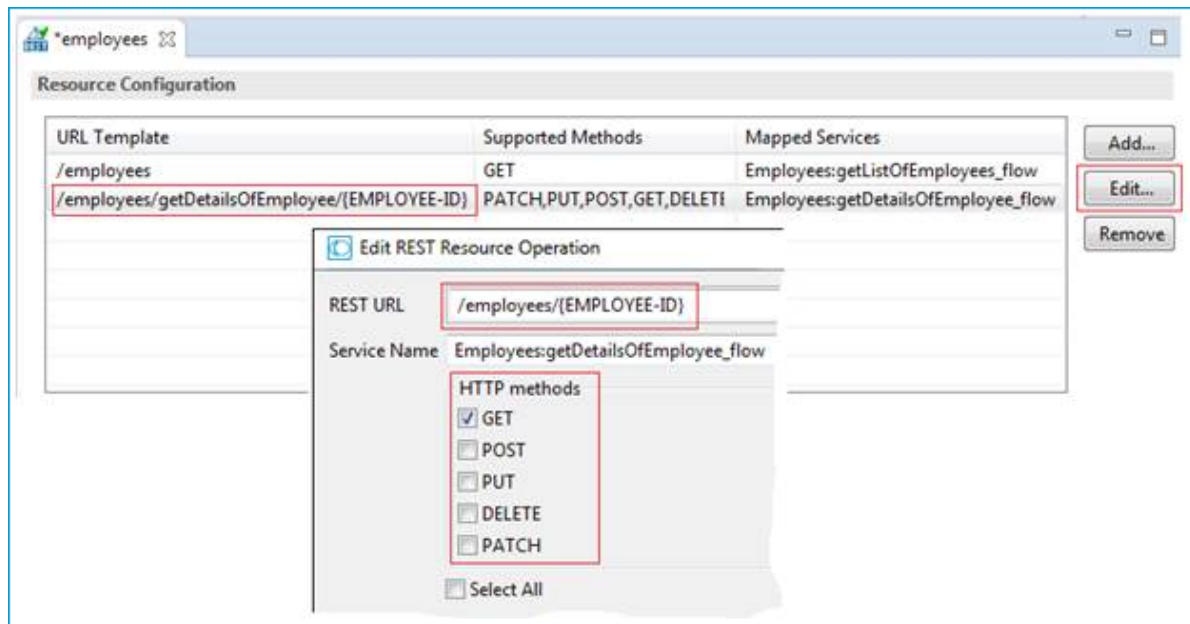


Note: This applies only to APIs where input signature parameters do not contain arrays.

Step 6: Edit the REST Resource

> To edit the REST resource

- 1 Switch to the **Service Development** perspective of Software AG Designer.



- 2 Refresh the package where you generated the REST Resource in the previous step.
- 3 Open the REST Resource.
- 4 Edit the REST URLs and HTTP Methods to match the functionality of your application.

For more information see:

- *REST Developer's Guide* in the webMethods Integration Server documentation for information on using the generated REST resources
- *REST Developer's Guide* in the webMethods Integration Server documentation

Task 3: Execute the Call from the REST Client to COBOL

From the **Service Development** perspective, refresh the package where the connection service was written, select the adapter service and use the service test to **Run Service**. This invokes the adapter service through the connector service.

The screenshot shows the 'Employees' package in the Service Development perspective. The 'getListOfEmployees' service is selected, and the 'Run Service' button is highlighted. Below the service list, a table displays the results of the service test.

Name	Value
outRec	
EMPLOYEES	
EMPLOYEES[0]	
IDENT	11100102
FIRSTNAME	EDGAR
SURNAME	SCHINDLER
DATE-BIRTH	Dec 4, 1962
EMPLOYEES[1]	
EMPLOYEES[2]	
EMPLOYEES[3]	
EMPLOYEES[4]	
errorCode	00000000
errorFlag	false

In case of error or unexpected results:

- Check the Integration Server log, the EntireX Adapter log or the RPC logs.
- Use the IDL Tester as described under [Task 1](#) above.

5

Calling COBOL Channel Container (Minimal Footprint using CICS Socket Listener) on z/OS CICS from REST

■ Introduction	35
■ What do I need to Install for this Scenario?	37
■ Task 1: Extract the Interface of a COBOL Server	38
■ Task 2: Generate the Connection and Adapter Services in Integration Server	50
■ Task 3: Execute the Call from the REST Client to COBOL	56

This scenario uses the tools IDL Extractor for COBOL and Integration Server Wrapper of the Designer.

Introduction



- ❶ Extract the interface of the COBOL server program. See *Using the IDL Extractor for COBOL - Overview*.
- ❷ Generate REST resources, connection and adapter services in Integration Server.
- ❸ Execute the call from the REST client to the COBOL server program.

This scenario makes the following important assumptions:

For Task 1:

- You have a working COBOL Channel Container server. For illustration and examples on such a server, see *CICS with Channel Container Calling Convention* under *Introduction to the IDL Extractor for COBOL*.
- You have access to the related COBOL sources and copybooks as files on your local machine. The minimum requirement is the `DATA DIVISION` of the interface.



For Tasks 2 and 3:

- You have a REST client.
- To call the COBOL server program at runtime using the *EntireX CICS Socket Listener* connection method, you need to install the CICS Socket Listener within your CICS region: see *Preparing for CICS Socket Listener*. For configuration, see *Connection Parameters for CICS Socket Listener Connections*.



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.
- The CICS Socket Listener, which is installed
 - together with the RPC Server for CICS, see *Installing the RPC Server for CICS*, or
 - separately, see *EntireX CICS® Socket Listener* in the z/OS Installation documentation

Task 1: Extract the Interface of a COBOL Server

- [Introduction](#)
- [Sample COBOL Server used in the Examples](#)
- [Extracting a COBOL Server - Fast-track Method](#)
- [Extracting a COBOL Server - Modern Method with User-defined Mapping](#)
- [Comparison Chart: Fast-track versus User-defined Mapping](#)
- [Testing the Extraction Results](#)

Introduction

Follow the instructions for extracting COBOL, see *Using the IDL Extractor for COBOL - Overview* and choose *Scenario I: Create New IDL and Server Mapping Files* if this is your first extraction. Learn how to invoke the IDL Extractor for COBOL, create a COBOL Extractor Environment and about the possibilities to select the COBOL source.

The extraction process creates the following EntireX metafiles:

- IDL file. A Software AG IDL file contains definitions of the interface between client and server. See *Software AG IDL File* in the IDL Editor documentation in the IDL Editor documentation.
- Server mapping file (optional). The mapping file is a Designer file with extension `.cvm` that contains COBOL-specific mapping information. See *Server Mapping Files for COBOL* in the Designer documentation.

Sample COBOL Server used in the Examples

The following COBOL server is used to illustrate the features of the IDL Extractor for COBOL. Imagine an existing COBOL server called `EMPLOYEE`. It implements the access logic for a `LIST` and `DETAILS` function to a database view `EMPLOYEE`:

```

IDENTIFICATION DIVISION.
    PROGRAM-ID.      EMPLOYEE.
*****
*   Program Name    = EMPLOYEE
*   String Literal  = QUOTE
*****
01 DFHCOMMAREA.
    02 OPERATION                                PIC X(1).
        88 OPERATION-LIST                      VALUE 'L'.
        88 OPERATION-DETAILS                  VALUE 'D'.
    02 EMPLOYEE-ID                                PIC X(10).
    02 EMPLOYEE-DETAILS.
        03 FIRSTNAME                            PIC X(20).
        03 SURNAME                              PIC X(20).
        03 DATE-BIRTH                          PIC X(12).
        03 DETAILS                             PIC X(100).
        03 FILLER REDEFINES DETAILS.
            04 ANNUAL-SALARY          PACKED-DECIMAL PIC S9(9).
            04 VACATION                PIC S9(2).
            04 LANGUAGE                PIC X(3).
            04 FILLER                  PIC X(90).
    02 EMPLOYEES-COUNT                          PIC 9(8) BINARY.
    02 EMPLOYEES OCCURS 1 TO 99 DEPENDING ON EMPLOYEES-COUNT.
        03 IDENT                          PIC X(10).
        03 FIRSTNAME                      PIC X(20).
        03 SURNAME                        PIC X(20).
        03 DATE-BIRTH                    PIC X(12).

```

Two approaches for extracting the COBOL server are described below:

- **Fast-track**

Learn how EntireX helps you to connect the COBOL Server with quick results.

- **User-defined Mapping**

The following features are among those available with a user-defined mapping:

- **Shape the interface to your COBOL server**

You can minimize the interface by suppressing COBOL server fields or by providing constant values. Renaming interfaces allows you to specify a readable long name. These two features together - constants and renaming of interfaces - are most powerful when multiple interfaces are implemented in a single COBOL server. Each function of the COBOL server triggered by an operation code or function code can be modelled as a separate REST resource method/operation.

■ **Select alternative mappings**

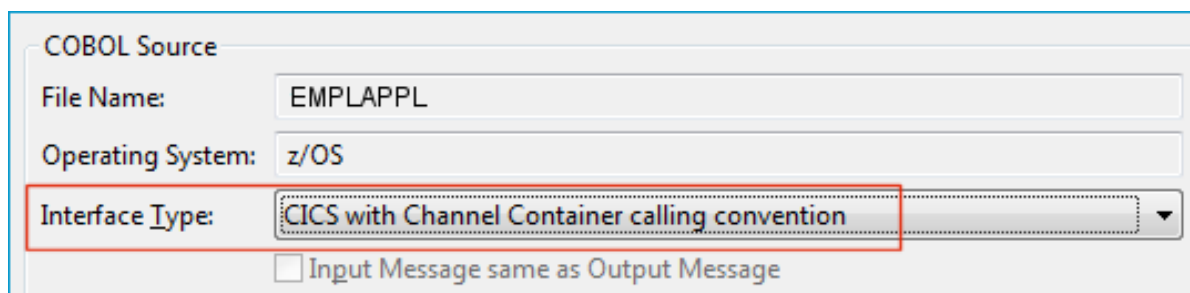
You can select alternative mappings for COBOL data items (REDEFINES, tables etc.) to enable scenarios with COBOL servers where data exchange is not fully described by the LINKAGE SECTION COBOL data items.

These approaches are described under *Fast-track Method* and *Modern Method* below.

Extracting a COBOL Server - Fast-track Method

➤ **To extract the COBOL Server**

- 1 Switch to the **EntireX** perspective.
- 2 Call the IDL Extractor for COBOL.
- 3 Set the correct interface type as described under *Step 4: Define the Extraction Settings and Start Extraction* under *Scenario I: Create New IDL and Server Mapping Files* in the IDL Extractor for COBOL documentation.



COBOL Source

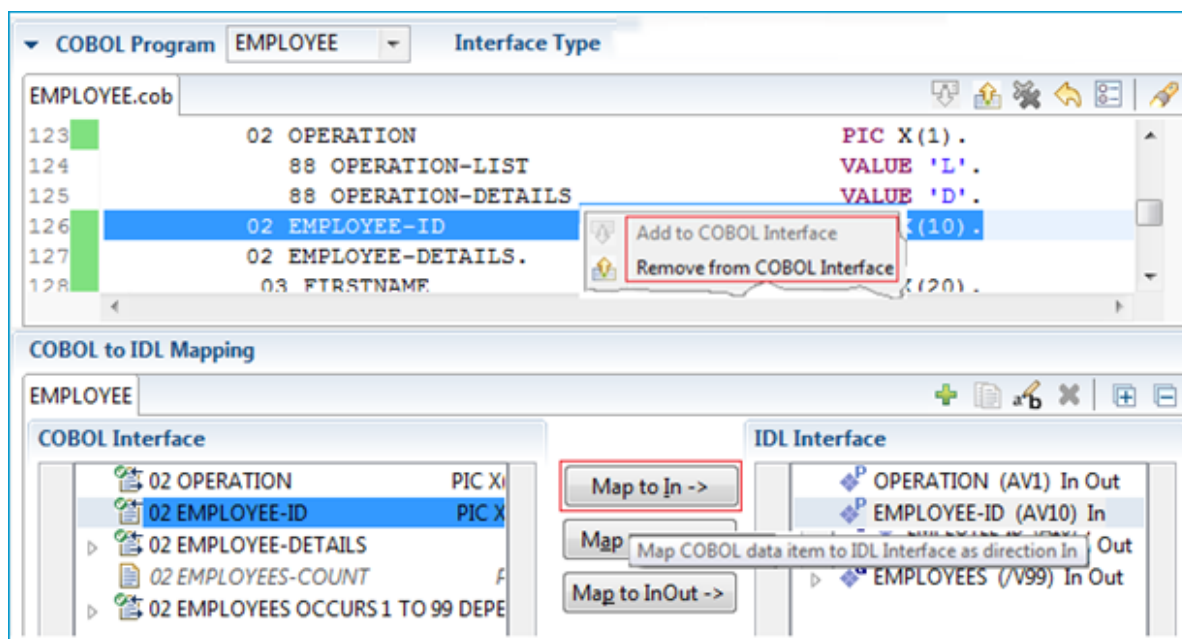
File Name: EMPLAPPL

Operating System: z/OS

Interface Type: CICS with Channel Container calling convention

☐ Input Message same as Output Message

- 4 Press **Next** to enter the COBOL Mapping Editor.



The purpose of the COBOL Mapping Editor is to map COBOL data items to IDL (Interface Definition Language) parameters. It consists of three main areas:

- The upper window pane is the COBOL program source.
- The window pane on the left-hand side is the COBOL interface, consisting of COBOL data items representing the parameters in your COBOL server program.
- The window pane on the right-hand side is the IDL interface consisting of IDL parameters.

For more information see *Mapping Editor User Interface* in the IDL Extractor for COBOL documentation.

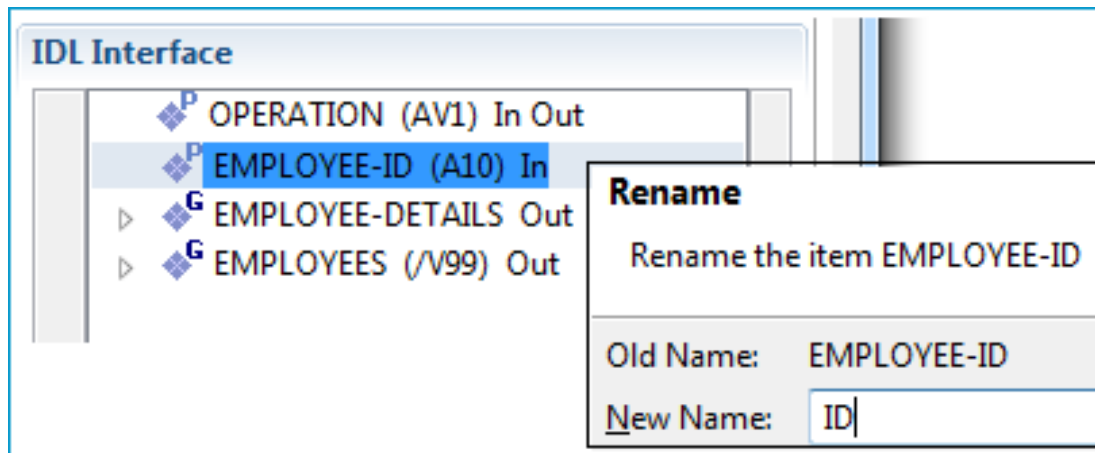
COBOL extraction is a two-step process:

- First, select the COBOL parameters using the **Add to** and **Remove from COBOL Interface** from the context menu on the COBOL data items in the upper window pane (COBOL source); in many situations the COBOL interface is detected by the Mapping Editor directly and you can omit this step.
- Then map the COBOL parameters to IDL directions using the **Map to In/Out/InOut** buttons. COBOL parameters do not have directions, they are always INOUT. This step can be left out for a quick test in non-production scenarios; for real production it is recommended to set IDL directions to reduce data transfer. You also have the option to assign a meaningful name to an IDL parameter if the original COBOL data item is not self-explanatory.

For our example:

- Set **Map to In** for COBOL data item `EMPLOYEE-ID` as IDL direction (see screenshot in previous step). For `EMPLOYEES-DETAILS` and `EMPLOYEES` perform the similar **Map to Out**.

- Use the function **Rename** from the context menu on the IDL parameters in the **IDL Interface** pane to rename COBOL parameter `EMPLOYEE- ID` to IDL parameter `ID`. Do the same for `EMPLOYEE- DETAILS`: Rename to `EMPLOYEE`.



- 5 Press **Finish**. The default extraction settings ensure you get useful results. The outcome is a Software AG IDL file (interface definition language), a Designer file with extension .idl:

```
library 'EMPLAPPL' is
program 'EMPLOYEE' is
define data parameter
  1 OPERATION (AV1) In Out
  1 ID (AV10) In /* Original Name:EMPLOYEE-ID
  1 EMPLOYEE Out /* Original Name:EMPLOYEE-DETAILS
  2 FIRSTNAME (AV20)
  2 SURNAME (AV20)
  2 DATE-BIRTH (AV12)
  2 DETAILS (AV100)
  1 EMPLOYEES (/V99) Out
  2 IDENT (AV10)
  2 FIRSTNAME (AV20)
  2 SURNAME (AV20)
  2 DATE-BIRTH (AV12)
end-define
```

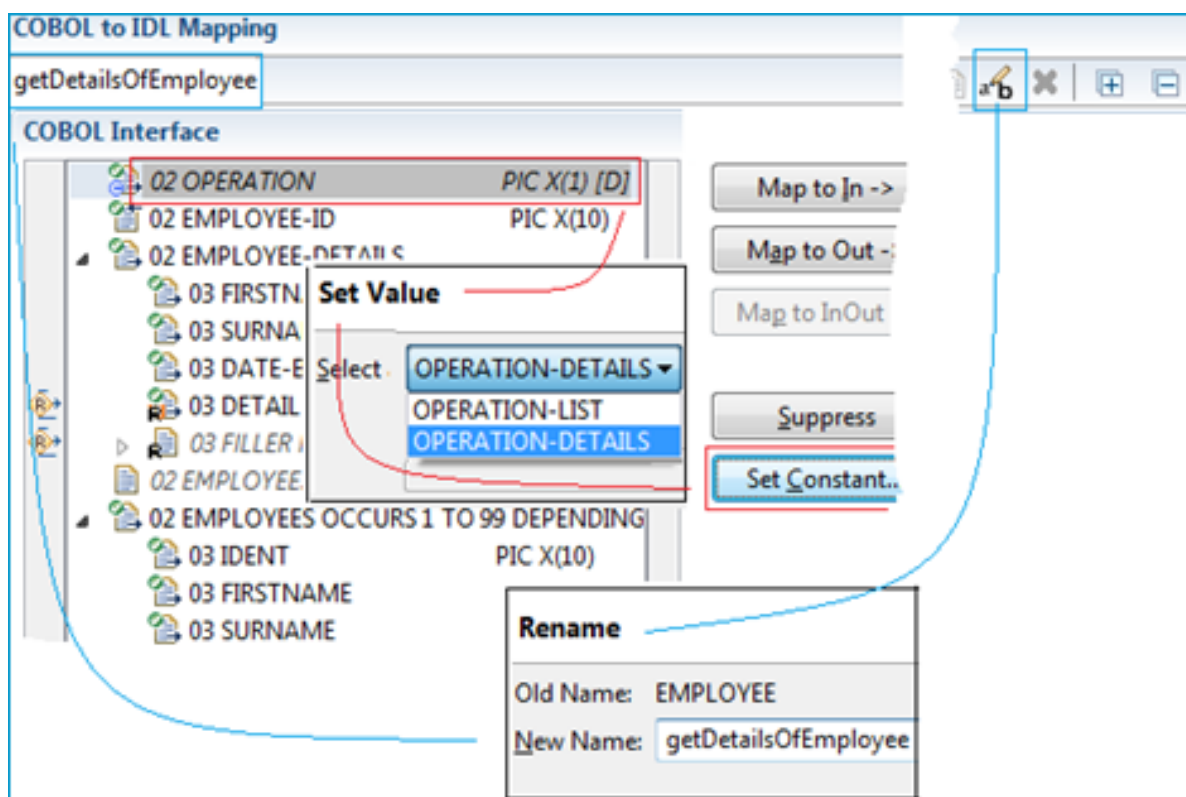
If required, a server mapping file with the same name as the IDL file but the extension .cvm may be generated too, for technical reasons. This file must always be stored together with the concomitant IDL file. For more information, see *Server Mapping Files for COBOL* in the Designer documentation. By default, the interface name `EMPLOYEE` is taken from the COBOL server program name.


Extracting a COBOL Server - Modern Method with User-defined Mapping

EntireX allows you to extract all implemented interfaces of the COBOL server separately. Instead of a large `EMPLOYEE` interface, separate interfaces `getListOfEmployees` and `getDetailsOfEmployee` are extracted. Each interface contains required parameters; obsolete parameters for an interface are suppressed, improving its usability.

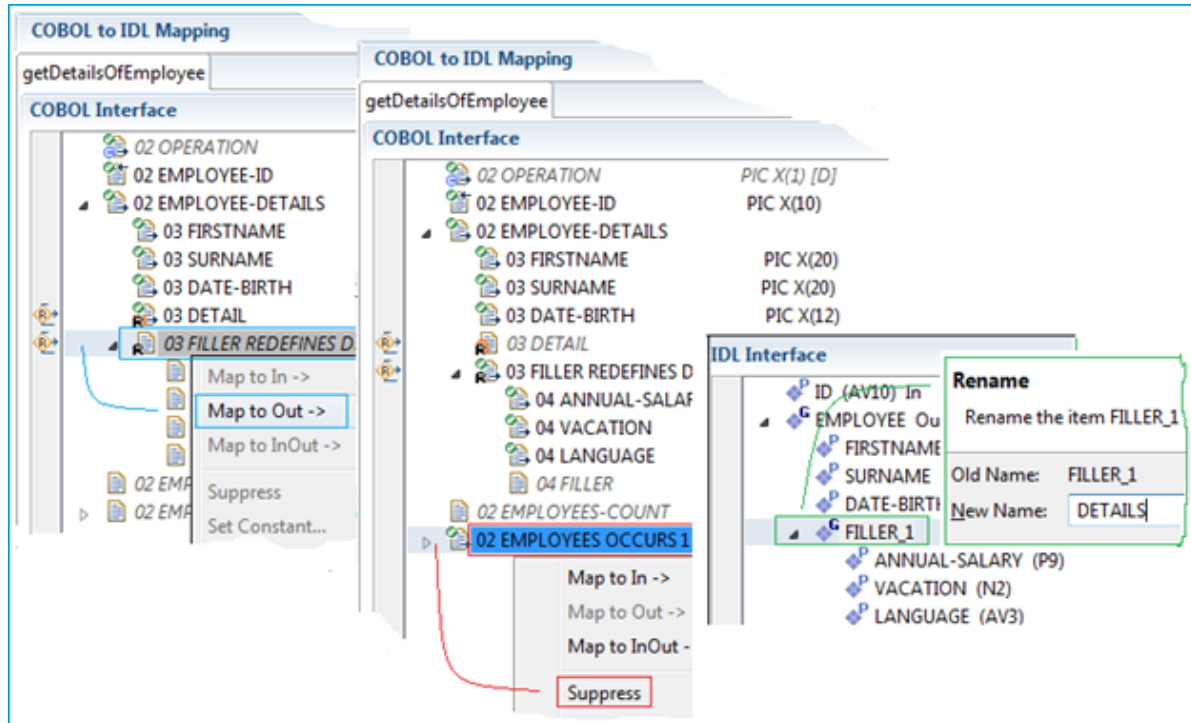
> To extract a COBOL server with a user-defined mapping

- 1 Shape `EMPLOYEE` to `getDetailsOfEmployee`:



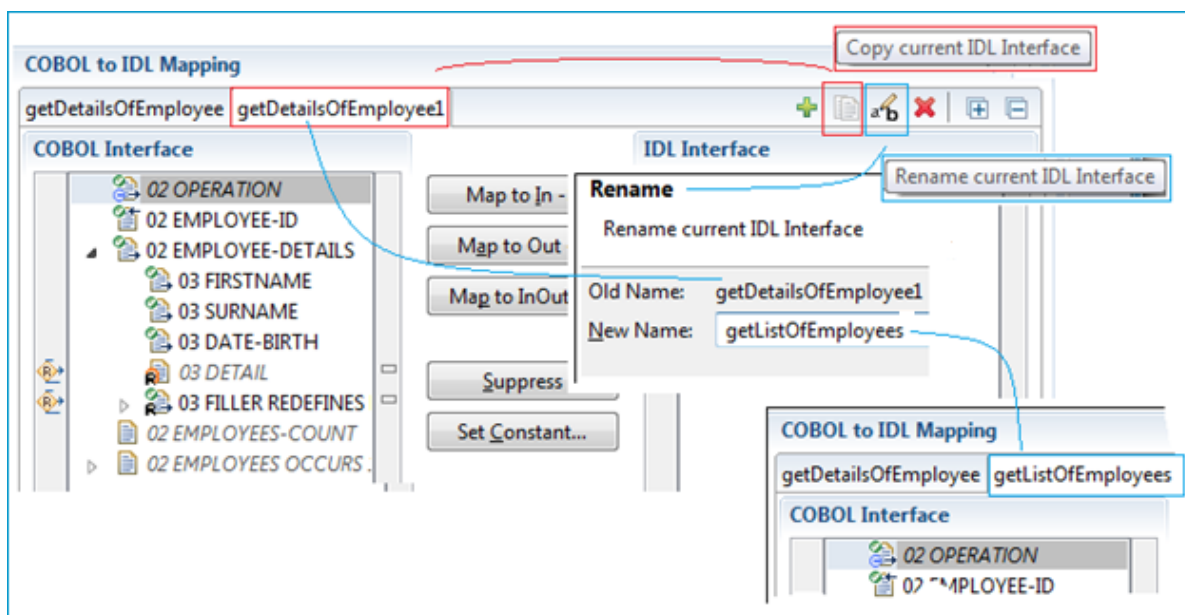
- Mark the `OPERATION` field in the COBOL **Interface** pane and press the **Set Constant** button. In the pop-up window, select `OPERATION-DETAILS` (red markers). The `OPERATION-DETAILS` value originates from COBOL level-88 data item (blue markers). If there is no COBOL level-88 enumeration type in the COBOL program defined, you can enter the function code manually in the pop-up window.
- By default the IDL interface gets the same name as your COBOL server: `EMPLOYEE`. Use the **Rename** button  from the toolbar of the COBOL Mapping Editor and change the IDL interface name to `getDetailsOfEmployee` (blue markers). This name is used as the IS service name later.

- 2 Continue shaping, making `getDetailsOfEmployee` easier to use:




- Use the context menu of the redefinition of the `DETAIL` field in the **COBOL Interface** pane, `FILLER REDEFINES DETAIL` and choose **Map to Out** (blue markers). A redefinition is a secondary layout (field types) of the same memory area. The redefinition contains more specific information (`ANNUAL-SALARY`, `VACATION`, etc.) than character buffer `DETAIL`. This is the reason why it is more useful in your interface.
- Use the context menu of `FILLER_1` in the IDL interface pane and rename `FILLER_1` to `DETAILS` (green markers) for a readable IS service field name later.
- The `DETAILS` function in the COBOL server does not make use of the `EMPLOYEES` field. Therefore you can leave it out of the interface: use the context menu of the `EMPLOYEES` field in the **COBOL Interface** pane and choose **Suppress** (red markers).


3 Shape `EMPLOYEE` to `getListOfEmployees` Function:



The COBOL Mapping Editor allows you to define multiple IDL interfaces to the same COBOL server:

- You create the additional IDL interface using the toolbar button **Copy current IDL interface**

 (red markers). A new tab `getDetailsOfEmployee1` is created. The previously extracted `getDetailsOfEmployee` interface still exists in the first tab. Once you reactivate the first tab, you will see the interface of `getDetailsOfEmployee` again.

- Use the **Rename** button  from the toolbar of the COBOL Mapping Editor and change the IDL interface name to `getListOfEmployees` (blue markers). This name is used as the IS service name later.

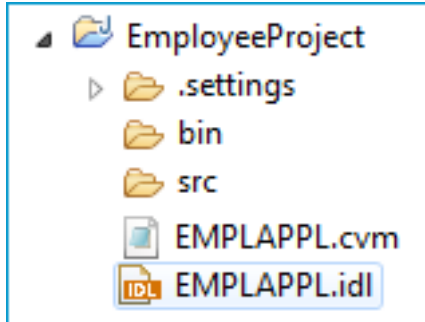
- For the `LIST` function you need the `OPERATION-LIST` value in the `OPERATION` field: Mark the `OPERATION` field in the **COBOL Interface** pane and press the **Set Constant** button. In the pop-up window, select `OPERATION-LIST` (similar to Step 1. above *Shape `EMPLOYEE` to `getDetailsOfEmployee`*; red markers).

- Use the context menu of the `EMPLOYEES` field in the **COBOL Interface** pane, `EMPLOYEES OCCURS 1 TO 99` and choose **Map to Out** (Similar to Step 2. above *Continue shaping, making `getDetailsOfEmployee` easier to use*; blue markers).

- Because the `EMPLOYEE` and `EMPLOYEE-DETAIL` fields are not used in the COBOL server `LIST` function, you leave them out in the IDL interface: use the context menu of `EMPLOYEE-DETAIL` field in the **COBOL Interface** pane and choose **Suppress** (similar to Step 2. above *Continue shaping, making `getDetailsOfEmployee` easier to use*; red markers). Do the same for the `EMPLOYEE-ID` field.

- 4 Press **Finish** to retrieve the extraction result in the form of a Software AG IDL file. At the same time, a server mapping file for COBOL (Designer file with extension `.cvm`) is created.

See *Server Mapping Files for COBOL* in the Designer documentation. The *Software AG IDL File* in the IDL Editor documentation describes the interfaces from the client point of view, while the server mapping file contains the mapping to the real COBOL server. Both of these files must be kept together and in sync, otherwise a call to the COBOL server may fail.



To summarize: We created two IDL interfaces. In the IDL file, these resulted in two IDL programs: `getListOfEmployees` and `getDetailsOfEmployee`. Both IDL programs were given readable names (Steps 1 and 3). Meaningful fields were kept, while superfluous fields were suppressed (Steps 1 and 3). The program `getDetailsOfEmployee` contains the redefined fields of parameter `DETAILS` (see below) mapped during extraction (Step 1).



Note: At runtime, the RPC client generated with the extracted interface will send data for the redesigned interfaces, while your COBOL server still expects `EMPLOYEE` data. The EntireX runtime transforms the incoming data stream from the RPC client, using the server mapping file.

```
program 'getDetailsOfEmployee' is
  define data parameter
    1 ID (AV10) In /* Original Name:EMPLOYEE-ID
    1 EMPLOYEE Out /* Original Name:EMPLOYEE-DETAILS
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
    2 DETAILS /* Original Name:FILLER_1
    3 ANNUAL-SALARY (P9)
    3 VACATION (N2)
    3 LANGUAGE (AV3)
  end-define
  program 'getListOfEmployees' is
    1 EMPLOYEES (/V99) Out
    2 IDENT (AV10)
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
  end-define
```


Comparison Chart: Fast-track versus User-defined Mapping

The following table highlights the differences when shaping an interface with a *user-defined mapping* compared with a *fast-path extraction*.

	Interface Shaping with User-defined Mapping	Fast-path Extraction
General	User-defined interface with dedicated mapping. For our example, the interfaces are small and tidy and more self-explanatory with long, readable interface names. They are easier to use with the hidden <code>OPERATION</code> field and the suppressed COBOL fields, which are not needed.	Automatic, quick result extraction with COBOL-like interfaces on IS. For our example, the IS interface <code>EMPLOYEE</code> matches exactly 1:1, field by field.
IS service(s)	Multiple small and handy IS services; each <code>OPERATION</code> code is mapped to a separate IS service.	One big IS service.
IS service name	Readable long name.	Short Subprogram name; up to 8 characters.
IS fields	Usage of Suppress and Set Constant reduces the message length. This keeps focus on relevant data items and keeps the client's interface clean. It may also improve performance.	The IS fields and COBOL server parameters match 1:1. As COBOL layout descriptions are sometimes used for many different purposes, irrelevant data items appear and clutter up the IS interface.
<code>OPERATION</code> parameter	Suppressed: the <code>OPERATION</code> parameter does not exist in the IS service as an IS field.	The <code>OPERATION</code> parameter exists in the IS service as an IS field and needs to be filled-in by the client endpoint.
<code>OPERATION</code> code	The <code>OPERATION</code> code is provided internally by EntireX runtime in the <code>OPERATION</code> field.	The <code>OPERATION</code> code needs to be specified in the IS service in the <code>OPERATION</code> field by the client endpoint.
<code>REDEFINE</code> parameters	Either the parameter that is redefined or one of its redefinitions is available as an IS field.	Only the parameter that is redefined is available as an IS field. Redefinitions of the parameter are not available as IS fields.

A user-defined mapping enables you also to define alternative mappings for COBOL data items (`REDEFINES`, tables etc.). These enable scenarios with COBOL servers where data exchange is not fully described by the `LINKAGE SECTION` COBOL data items. For more information refer to the IDL Extractor for COBOL documentation:

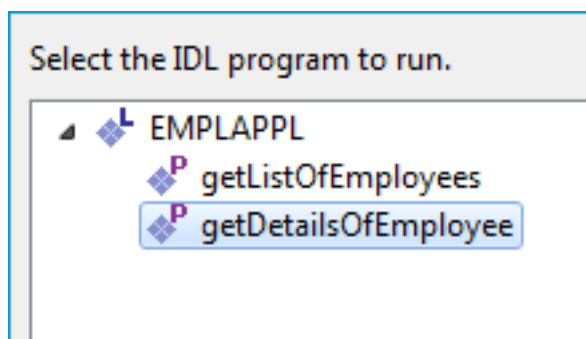
- *Mapping Editor IDL Interface Mapping Functions*
- *User-defined Mapping*

Testing the Extraction Results

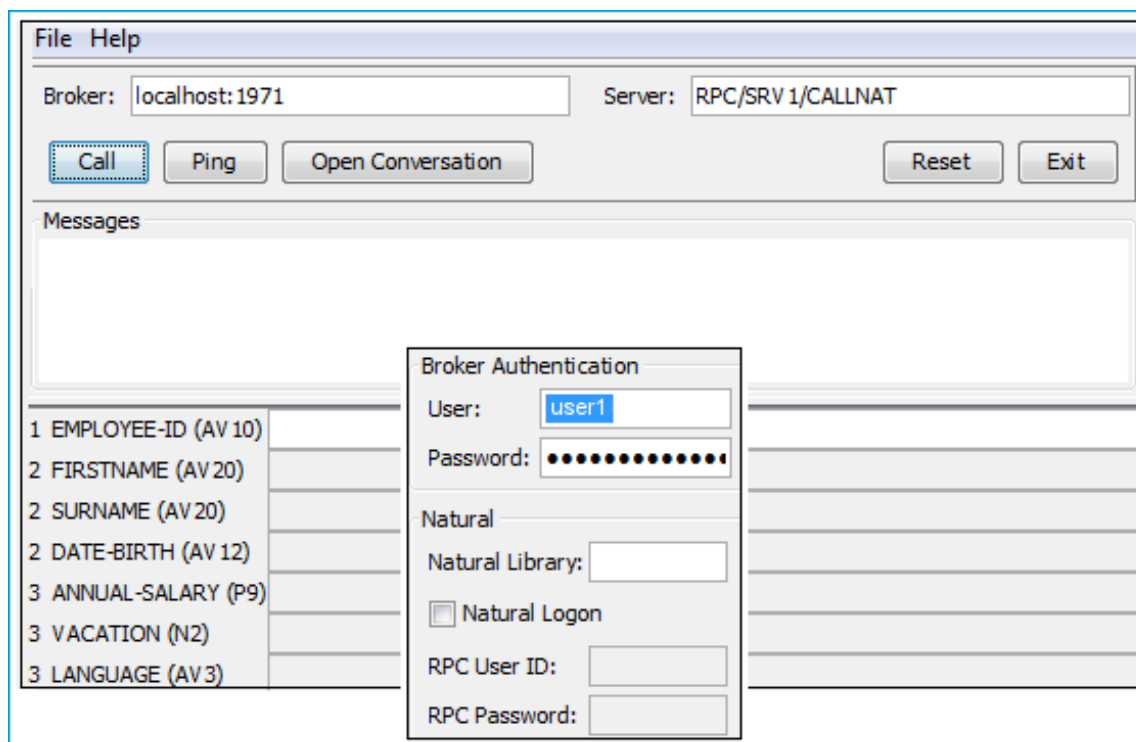
The following pictures use the extraction results described under [Extracting a COBOL Server - Modern Method with User-defined Mapping](#).

➤ To test the extraction results (optional)

- 1 You can test the results of the extraction operation and the REST server back end, using the EntireX IDL Tester. From the context menu of the IDL file in the Designer, choose **Software AG IDL Tester**.



- 2 Select `getDetailsOfEmployee`.



Note that the broker and server parameters contain the explicit route to call the server program, and you can optionally ping the connection from this client. With the **File > Options** dialog, you can set:

- **User** and **Password** for *broker* authentication
- **RPC User ID** and **RPC Password** for *server* authentication

See *EntireX IDL Tester* in the Designer documentation.

- 3 Check the Integration Server log, the EntireX Adapter log or the RPC logs. Applies to all connection methods.

Task 2: Generate the Connection and Adapter Services in Integration Server

This section describes your first steps to create a new Integration Server connection. This is described in more detail under *Using the Integration Server Wrapper*, for example working with existing Integration Server connections. This section covers the following topics:

- [Step 1: Start the Integration Server Wrapper Wizard](#)
- [Step 2: Create a New Integration Server Connection](#)
- [Step 3: Select the Connection Type](#)
- [Step 4: Define Adapter Services for a CICS Socket Listener Connection](#)
- [Step 5: Create or Update a REST Resource](#)
- [Step 6: Edit the REST Resource](#)

Step 1: Start the Integration Server Wrapper Wizard

➤ To start the Integration Server Wrapper wizard

- 1 In the context menu of a Software AG IDL file, choose **Integration Server > Generate web-Methods IS Connection**.

This starts the wizard with a list of existing Integration Server Wrapper connections.



Note: If the selected IDL file is not valid because of a syntax error, an error dialog comes up and the wizard does not start.

- 2 Continue with *Step 2: Create a New Integration Server Connection*.

Step 2: Create a New Integration Server Connection

Define New Integration Server Connection
Add host:port, user and password for a new Integration Server

Server:

User:

Password:

☐ Use secure connection

➤ To create a new Integration Server connection

- 1 Define the new Integration Server connection on the wizard page.



Notes:

1. The only required field is **Server**. Enter the hostname of the Integration Server including an optional port number. If no port number is specified, port number defaults to "5555". The **Integration Server Authentication** can be passed with the **User** and **Password** fields.
2. Optional settings are for secure connections. The **Truststore for HTTPS** contains all signed certificates and must be a valid truststore.
3. The check box **Verify host name** checks that the hostname is entered in the stored certificate.
4. When the Integration Server has **Client Authentication** enabled, you can specify your **Keystore** file and keystore **Password**.
5. For managing Integration Server connections, see [Preferences](#).

- 2 Choose **Next** and continue with *Step 3: Select the Connection Type*.

Step 3: Select the Connection Type

☒ Create a new EntireX Adapter connection of type **CICS Socket Listener Connection**

☐ Update Adapter Services or Listeners to an existing Integration Server connection:

List of Connections

Connection Name	Package Name	Connection Type	Enabled
Employees:EmployeesConnection	Employees	EntireX RPC Connection	Yes

Total: 1

☒ Map Software AG IDL data types to String

☒ Create or Update REST resource

> To create a new connection

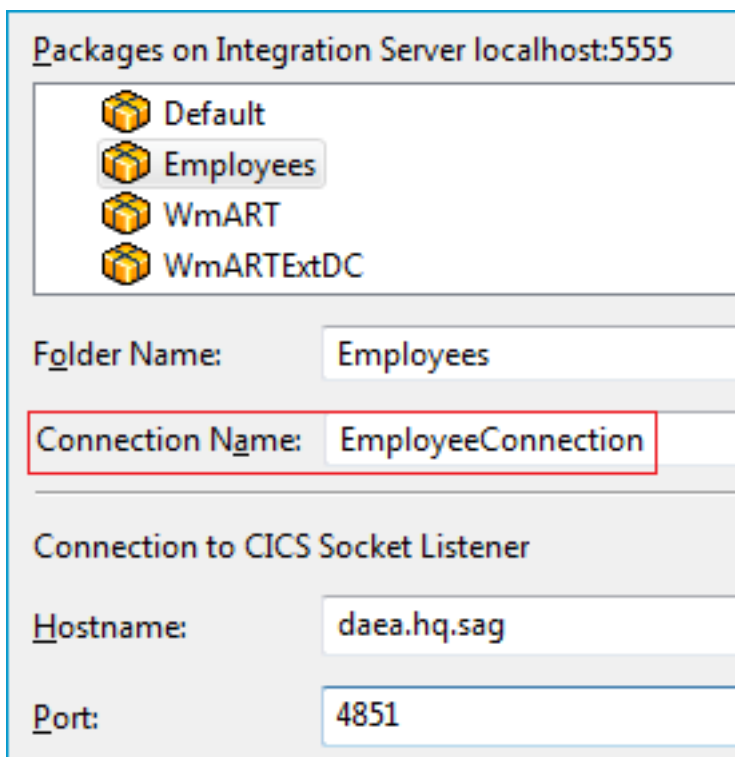
- 1 Select a connection type from the drop down list. Connection types are described under *EntireX Adapter Connections* in the EntireX Adapter documentation and *Introduction to the Integration Server Wrapper*.



Note: The list of connection types is filtered: connection types that require a license are only shown if a corresponding license file is available. Reliable RPC connections are only shown if all IDL programs contain only IN parameters. Also, if a server mapping file is available, only those connection types that support the interface type specified in the server mapping file are shown.

- 2 Check the box **Create or Update REST resource**.
- 3 Click **Next** and continue with *Step 4: Define Adapter Services for a CICS Socket Listener Connection*.

Step 4: Define Adapter Services for a CICS Socket Listener Connection



Packages on Integration Server localhost:5555

- Default
- Employees
- WmART
- WmARTEExtDC

Folder Name: Employees

Connection Name: EmployeeConnection

Connection to CICS Socket Listener

Hostname: daea.hq.sag

Port: 4851

➤ To create a connection and related adapter services

- 1 Select a package for the created objects.
- 2 Define a folder name. If the folder does not exist, it will be created.
- 3 Define a connection name.
- 4 Define the parameters of the connection type. For details, see *Connection Parameters for CICS Socket Listener Connections* in the EntireX Adapter documentation.

As a result, the folder will contain the connection and the adapter services (one for each IDL program). The name of a service is the same as the respective IDL program.

The default settings for the adapter services are:

- the **Default** package; if not available, the first package
- the IDL library name for the **Folder Name**
- the IDL library name with the suffix "Connection" for the **Connection Name**



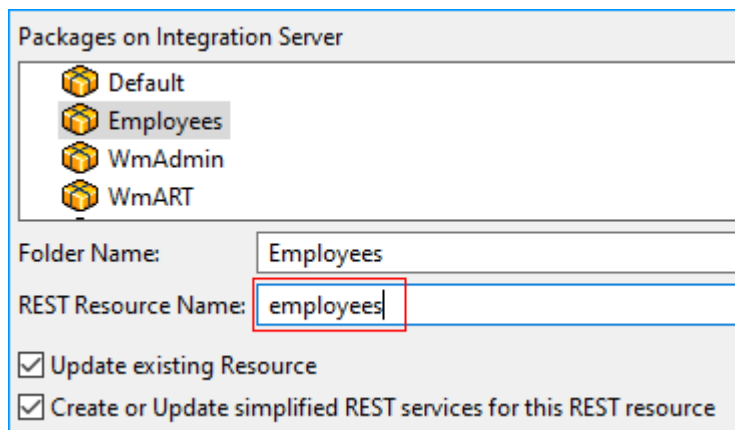
Note: When creating a connection, a package dependency is added such that the selected package depends on webMethods EntireX (the package `WmEntireX`) with the version currently used.

Step 5: Create or Update a REST Resource

➤ To create or update a REST resource

Make sure checkbox **Create or Update REST resource** is checked.

- 1 Select a package.



Packages on Integration Server

- Default
- Employees
- WmAdmin
- WmART

Folder Name: Employees

REST Resource Name: employees

☒ Update existing Resource

☒ Create or Update simplified REST services for this REST resource

- 2 Define a folder name. If the folder does not exist, it will be created.
- 3 Define the REST Resource name.
- 4 Check **Update existing Resource** if you want to add additional services to an existing resource.
- 5 Check **Create or Update simplified REST services for this REST resource** if you want to generate additional services with a simplified REST API. This allows you to use a simple HTTP-GET method without a payload.



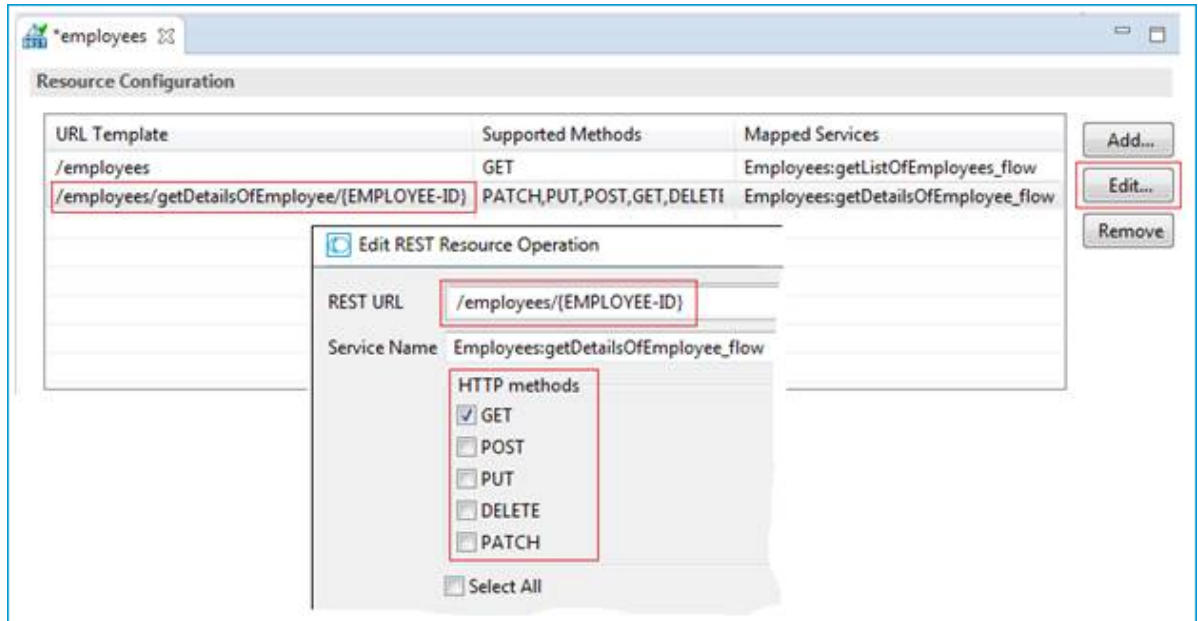
Note: This applies only to APIs where input signature parameters do not contain arrays.

;

Step 6: Edit the REST Resource

➤ To edit the REST resource

- 1 Switch to the **Service Development** perspective of Software AG Designer.



- 2 Refresh the package where you generated the REST Resource in the previous step.
- 3 Open the REST Resource.
- 4 Edit the REST URLs and HTTP Methods to match the functionality of your application.

For more information see:

- *REST Developer's Guide* in the webMethods Integration Server documentation for information on using the generated REST resources
- *REST Developer's Guide* in the webMethods Integration Server documentation

Task 3: Execute the Call from the REST Client to COBOL

Use an appropriate REST client, specify URL, Method and Headers and invoke your REST resource:

URL http://localhost:5555/restv2/employees	URL http://localhost:5555/restv2/employees/111000105
Method GET	Method GET
Headers Accept application/json	Headers Accept application/json
200 OK	200 OK
<pre>{ "errorCode": "00000000", "errorFlag": "false", "response": { "EMPLOYEES-LIST": { "EMPLOYEES": [{ "IDENT": "11100102", "FIRSTNAME": "Edgar", "SURNAME": "Schindler", "DATE-BIRTH": "Dec 4, 1962" }, { "IDENT": "11100105", "FIRSTNAME": "Christian", "SURNAME": "Schirm", "DATE-BIRTH": "Mar 15, 1961" }, { "IDENT": "11100106", "FIRSTNAME": "Rainer", "SURNAME": "Schmitt", "DATE-BIRTH": "Feb 13, 1955" }, { "IDENT": "11100107", "FIRSTNAME": "Helga", "SURNAME": "Schmidt", "DATE-BIRTH": "May 26, 1961" }] } } }</pre>	<pre>{ "errorCode": "00000000", "errorFlag": "false", "response": { "EMPLOYEE-DETAILS": { "FIRSTNAME": "Christian", "SURNAME": "Schirm", "DATE-BIRTH": "Mar 15, 1961", "DETAIL": { "ANNUAL-SALARY": "68000", "VACATION": "21", "LANGUAGE": "GER" } } } }</pre>

6

Calling COBOL Channel Container on z/OS CICS from REST

■ Introduction	59
■ What do I need to Install for this Scenario?	61
■ Task 1: Extract the Interface of a COBOL Server	62
■ Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server	74
■ Task 3: Execute the Call from the REST Client to COBOL	80

This scenario uses the tools IDL Extractor for COBOL and Integration Server Wrapper of the Designer.

Introduction



- ❶ Extract the interface of the COBOL server program. See *Using the IDL Extractor for COBOL - Overview*.
- ❷ Generate REST resources, connection and adapter services in Integration Server.
- ❸ Execute the call from the REST client to the COBOL server program.

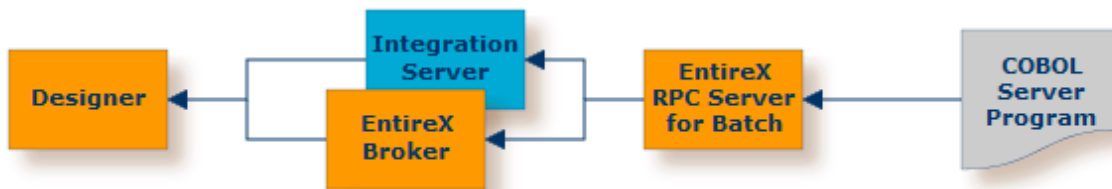
This scenario makes the following important assumptions:

For Task 1:

- You have a working COBOL Channel Container server. For illustration and examples on such a server, see *CICS with Channel Container Calling Convention* under *Introduction to the IDL Extractor for COBOL*.
- You have access to the related COBOL sources and copybooks. The minimum requirement is the `DATA DIVISION` of the interface. The sources and copybooks must be stored either
 - *locally*, that is, on the same machine where the Designer is running



- or *remotely* in a PDS or CA Librarian data set and accessed via the *RPC Server for Batch*, and either EntireX Broker or Integration Server



For Tasks 2 and 3:

- You have a REST client.
- You can call the COBOL server program 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
 - the *RPC Server for CICS*



- For the *EntireX Direct RPC* connection method you need:
 - the *RPC Server for CICS*



See also *Direct RPC* in the EntireX Adapter documentation.

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.
- You have an RPC Server for CICS installed. See *Installing the RPC Server for CICS* in the z/OS Installation documentation.
- Optionally 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.
- Optionally, for remote extraction only (Task 1), you have an RPC Server for Batch installed. See *Installing the RPC Server for Batch* in the z/OS Installation documentation.

Task 1: Extract the Interface of a COBOL Server

- [Introduction](#)
- [Sample COBOL Server used in the Examples](#)
- [Extracting a COBOL Server - Fast-track Method](#)
- [Extracting a COBOL Server - Modern Method with User-defined Mapping](#)
- [Comparison Chart: Fast-track versus User-defined Mapping](#)
- [Testing the Extraction Results](#)

Introduction

Follow the instructions for extracting COBOL, see *Using the IDL Extractor for COBOL - Overview* and choose *Scenario I: Create New IDL and Server Mapping Files* if this is your first extraction. Learn how to invoke the IDL Extractor for COBOL, create a COBOL Extractor Environment and about the possibilities to select the COBOL source.

The extraction process creates the following EntireX metafiles:

- IDL file. A Software AG IDL file contains definitions of the interface between client and server. See *Software AG IDL File* in the IDL Editor documentation in the IDL Editor documentation.
- Server mapping file (optional). The mapping file is a Designer file with extension .cvm that contains COBOL-specific mapping information. See *Server Mapping Files for COBOL* in the Designer documentation.

Sample COBOL Server used in the Examples

The following COBOL server is used to illustrate the features of the IDL Extractor for COBOL. Imagine an existing COBOL server called `EMPLOYEE`. It implements the access logic for a `LIST` and `DETAILS` function to a database view `EMPLOYEE`:

```

IDENTIFICATION DIVISION.
    PROGRAM-ID.      EMPLOYEE.
*****
*   Program Name    = EMPLOYEE
*   String Literal  = QUOTE
*****
01 DFHCOMMAREA.
    02 OPERATION                                PIC X(1).
        88 OPERATION-LIST                      VALUE 'L'.
        88 OPERATION-DETAILS                  VALUE 'D'.
    02 EMPLOYEE-ID                             PIC X(10).
    02 EMPLOYEE-DETAILS.
        03 FIRSTNAME                          PIC X(20).
        03 SURNAME                           PIC X(20).
        03 DATE-BIRTH                        PIC X(12).
        03 DETAILS                          PIC X(100).
        03 FILLER REDEFINES DETAILS.
            04 ANNUAL-SALARY                 PACKED-DECIMAL PIC S9(9).
            04 VACATION                      PIC S9(2).
            04 LANGUAGE                      PIC X(3).
            04 FILLER                      PIC X(90).
    02 EMPLOYEES-COUNT                         PIC 9(8) BINARY.
    02 EMPLOYEES OCCURS 1 TO 99 DEPENDING ON EMPLOYEES-COUNT.
        03 IDENT                            PIC X(10).
        03 FIRSTNAME                        PIC X(20).
        03 SURNAME                         PIC X(20).
        03 DATE-BIRTH                      PIC X(12).

```

Two approaches for extracting the COBOL server are described below:

- **Fast-track**

Learn how EntireX helps you to connect the COBOL Server with quick results.

- **User-defined Mapping**

The following features are among those available with a user-defined mapping:

- **Shape the interface to your COBOL server**

You can minimize the interface by suppressing COBOL server fields or by providing constant values. Renaming interfaces allows you to specify a readable long name. These two features together - constants and renaming of interfaces - are most powerful when multiple interfaces are implemented in a single COBOL server. Each function of the COBOL server triggered by an operation code or function code can be modelled as a separate REST resource method/operation.

■ **Select alternative mappings**

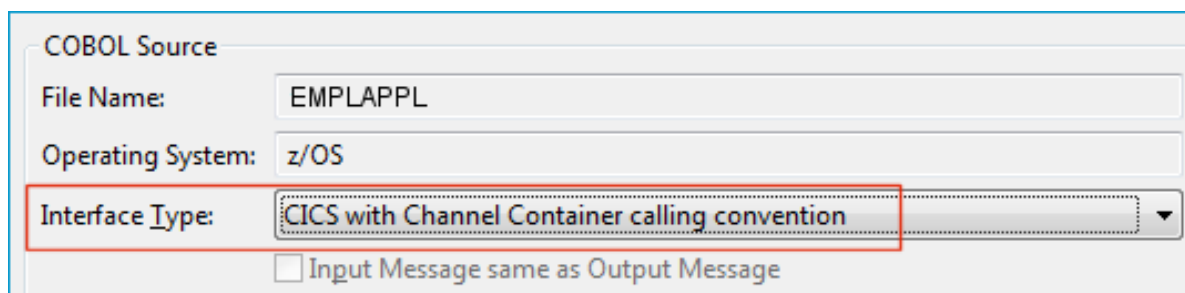
You can select alternative mappings for COBOL data items (REDEFINES, tables etc.) to enable scenarios with COBOL servers where data exchange is not fully described by the LINKAGE SECTION COBOL data items.

These approaches are described under *Fast-track Method* and *Modern Method* below.

Extracting a COBOL Server - Fast-track Method

➤ **To extract the COBOL Server**

- 1 Switch to the **EntireX** perspective.
- 2 Call the IDL Extractor for COBOL.
- 3 Set the correct interface type as described under *Step 4: Define the Extraction Settings and Start Extraction* under *Scenario I: Create New IDL and Server Mapping Files* in the IDL Extractor for COBOL documentation.



COBOL Source

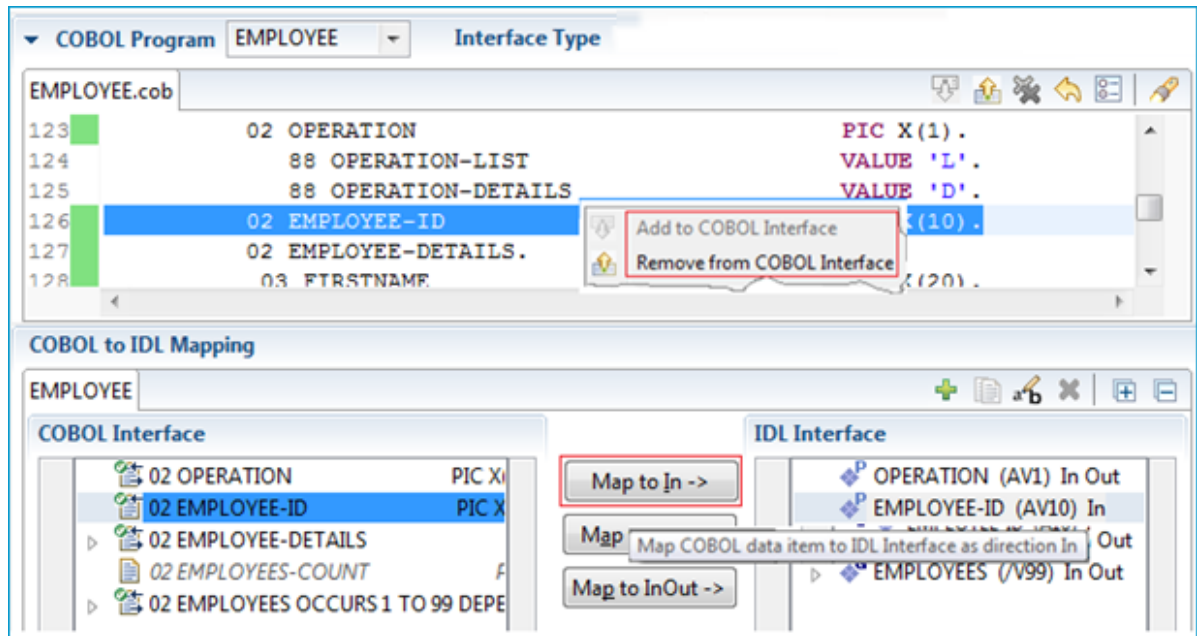
File Name: EMPLAPPL

Operating System: z/OS

Interface Type: CICS with Channel Container calling convention

☐ Input Message same as Output Message

- 4 Press **Next** to enter the COBOL Mapping Editor.



The purpose of the COBOL Mapping Editor is to map COBOL data items to IDL (Interface Definition Language) parameters. It consists of three main areas:

- The upper window pane is the COBOL program source.
- The window pane on the left-hand side is the COBOL interface, consisting of COBOL data items representing the parameters in your COBOL server program.
- The window pane on the right-hand side is the IDL interface consisting of IDL parameters.

For more information see *Mapping Editor User Interface* in the IDL Extractor for COBOL documentation.

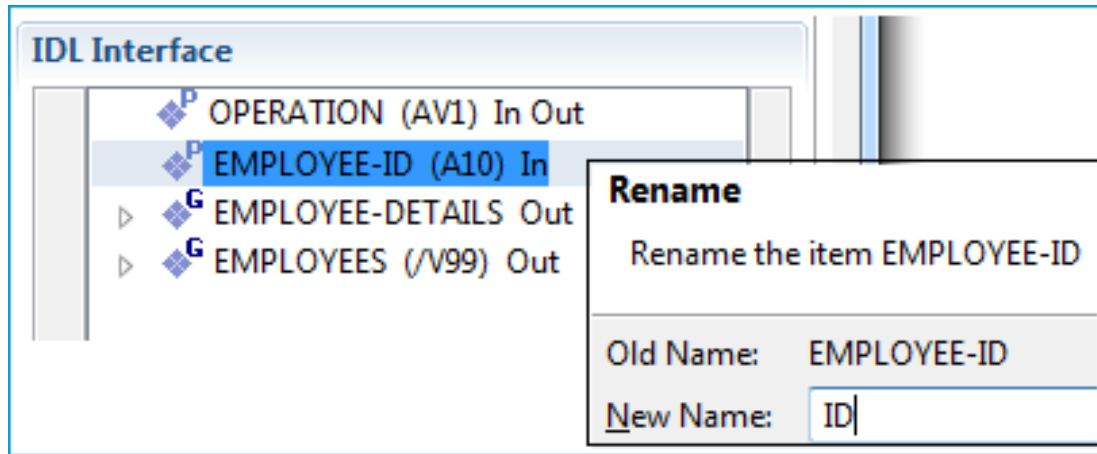
COBOL extraction is a two-step process:

- First, select the COBOL parameters using the **Add to** and **Remove from COBOL Interface** from the context menu on the COBOL data items in the upper window pane (COBOL source); in many situations the COBOL interface is detected by the Mapping Editor directly and you can omit this step.
- Then map the COBOL parameters to IDL directions using the **Map to In/Out/InOut** buttons. COBOL parameters do not have directions, they are always INOUT. This step can be left out for a quick test in non-production scenarios; for real production it is recommended to set IDL directions to reduce data transfer. You also have the option to assign a meaningful name to an IDL parameter if the original COBOL data item is not self-explanatory.

For our example:

- Set **Map to In** for COBOL data item `EMPLOYEE-ID` as IDL direction (see screenshot in previous step). For `EMPLOYEES-DETAILS` and `EMPLOYEES` perform the similar **Map to Out**.

- Use the function **Rename** from the context menu on the IDL parameters in the **IDL Interface** pane to rename COBOL parameter `EMPLOYEES-ID` to IDL parameter `ID`. Do the same for `EMPLOYEE-DETAILS`: Rename to `EMPLOYEE`.



- 5 Press **Finish**. The default extraction settings ensure you get useful results. The outcome is a Software AG IDL file (interface definition language), a Designer file with extension `.idl`:

```
library 'EMPLAPPL' is
program 'EMPLOYEE' is
define data parameter
  1 OPERATION (AV1) In Out
  1 ID (AV10) In /* Original Name:EMPLOYEE-ID
  1 EMPLOYEE Out /* Original Name:EMPLOYEE-DETAILS
  2 FIRSTNAME (AV20)
  2 SURNAME (AV20)
  2 DATE-BIRTH (AV12)
  2 DETAILS (AV100)
  1 EMPLOYEES (/V99) Out
  2 IDENT (AV10)
  2 FIRSTNAME (AV20)
  2 SURNAME (AV20)
  2 DATE-BIRTH (AV12)
end-define
```

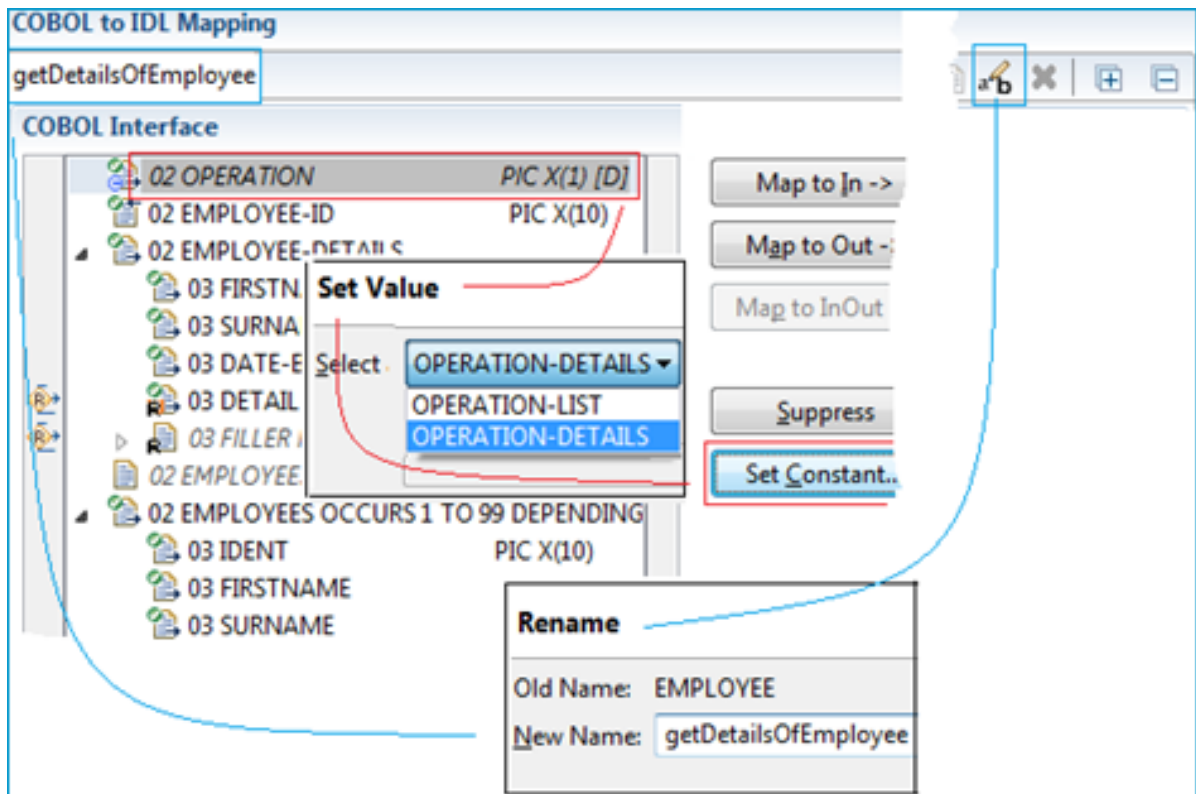
If required, a server mapping file with the same name as the IDL file but the extension `.cvm` may be generated too, for technical reasons. This file must always be stored together with the concomitant IDL file. For more information, see *Server Mapping Files for COBOL* in the Designer documentation. By default, the interface name `EMPLOYEE` is taken from the COBOL server program name.


Extracting a COBOL Server - Modern Method with User-defined Mapping

EntireX allows you to extract all implemented interfaces of the COBOL server separately. Instead of a large `EMPLOYEE` interface, separate interfaces `getListOfEmployees` and `getDetailsOfEmployee` are extracted. Each interface contains required parameters; obsolete parameters for an interface are suppressed, improving its usability.

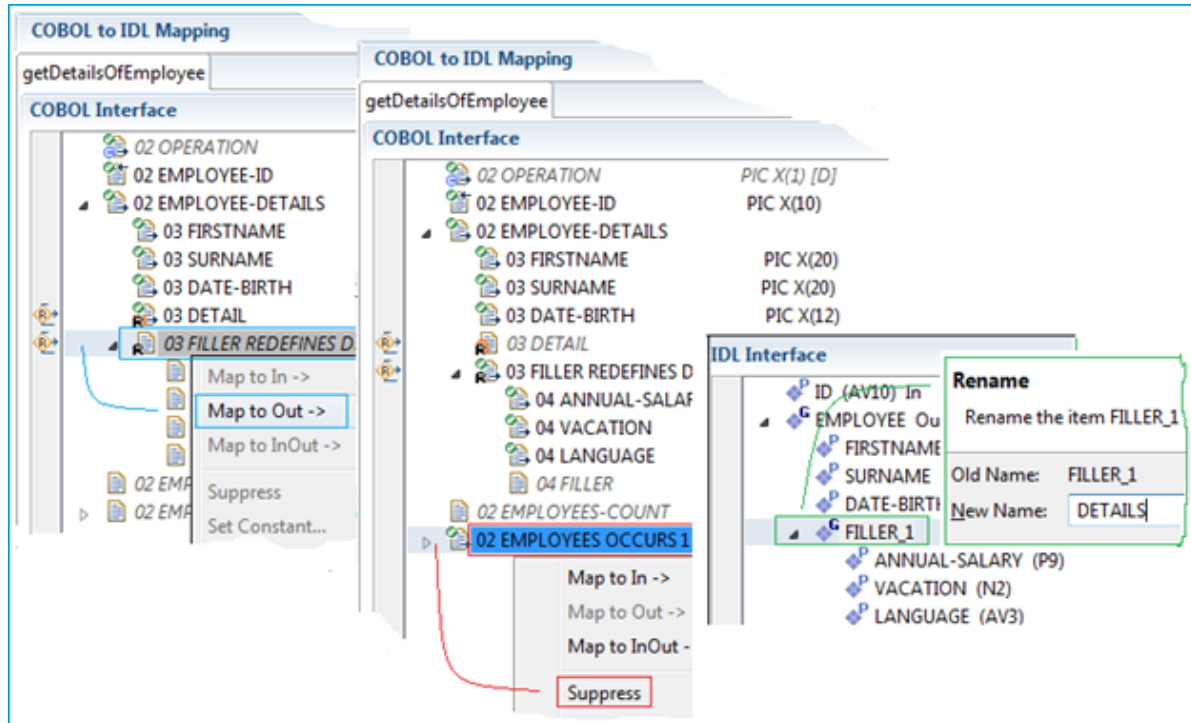
> To extract a COBOL server with a user-defined mapping

- 1 Shape `EMPLOYEE` to `getDetailsOfEmployee`:



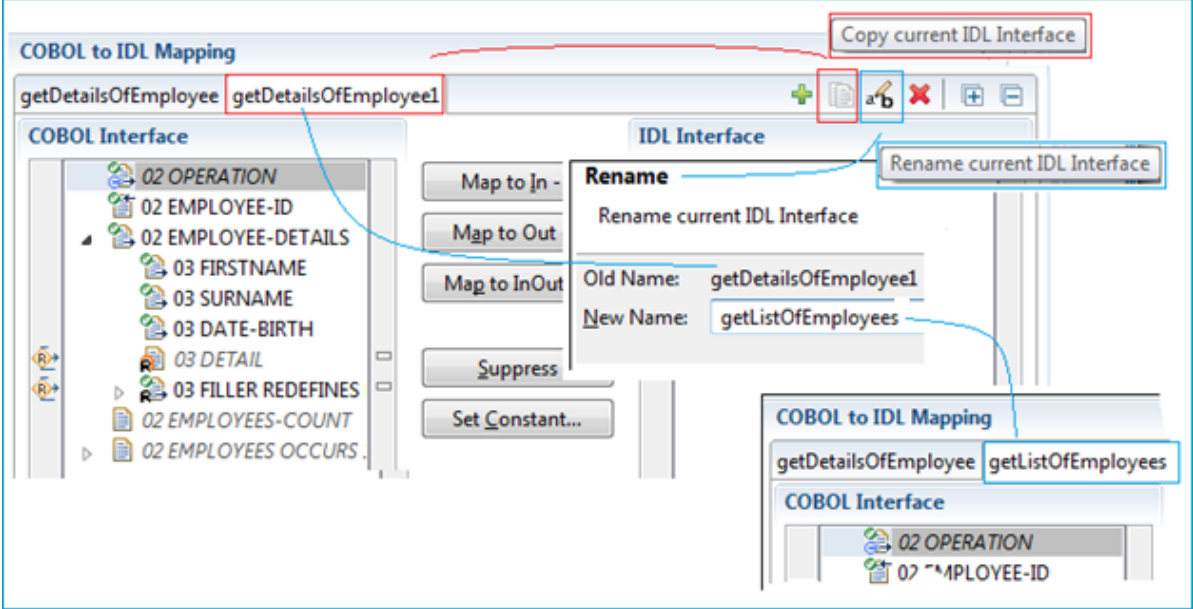
- Mark the `OPERATION` field in the COBOL **Interface** pane and press the **Set Constant** button. In the pop-up window, select `OPERATION-DETAILS` (red markers). The `OPERATION-DETAILS` value originates from COBOL level-88 data item (blue markers). If there is no COBOL level-88 enumeration type in the COBOL program defined, you can enter the function code manually in the pop-up window.
- By default the IDL interface gets the same name as your COBOL server: `EMPLOYEE`. Use the **Rename** button  from the toolbar of the COBOL Mapping Editor and change the IDL interface name to `getDetailsOfEmployee` (blue markers). This name is used as the IS service name later.



- 2 Continue shaping, making `getDetailsOfEmployee` easier to use:



- Use the context menu of the redefinition of the `DETAIL` field in the **COBOL Interface** pane, `FILLER REDEFINES DETAIL` and choose **Map to Out** (blue markers). A redefinition is a secondary layout (field types) of the same memory area. The redefinition contains more specific information (`ANNUAL-SALARY`, `VACATION`, etc.) than character buffer `DETAIL`. This is the reason why it is more useful in your interface.
- Use the context menu of `FILLER_1` in the IDL interface pane and rename `FILLER_1` to `DETAILS` (green markers) for a readable IS service field name later.
- The `DETAILS` function in the COBOL server does not make use of the `EMPLOYEES` field. Therefore you can leave it out of the interface: use the context menu of the `EMPLOYEES` field in the **COBOL Interface** pane and choose **Suppress** (red markers).

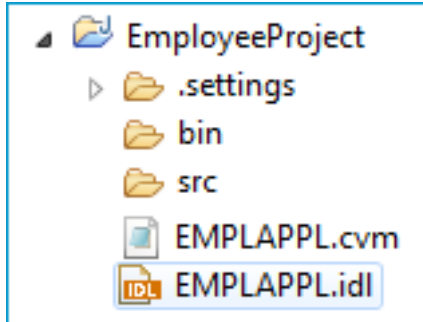
3 Shape `EMPLOYEE` to `getListOfEmployees` Function:



- You create the additional IDL interface using the toolbar button **Copy current IDL interface** . (red markers). A new tab `getDetailsOfEmployee1` is created. The previously extracted `getDetailsOfEmployee` interface still exists in the first tab. Once you reactivate the first tab, you will see the interface of `getDetailsOfEmployee` again.
- Use the **Rename** button  from the toolbar of the COBOL Mapping Editor and change the IDL interface name to **`getListOfEmployees`** (blue markers). This name is used as the IS service name later.
- For the LIST function you need the OPERATION-LIST value in the OPERATION field: Mark the OPERATION field in the **COBOL Interface** pane and press the **Set Constant** button. In the pop-up window, select OPERATION-LIST (similar to Step 1. above *Shape EMPLOYEE to getDetailsOfEmployee*; red markers).
- Use the context menu of the EMPLOYEES field in the **COBOL Interface** pane, EMPLOYEES OCCURS 1 TO 99 and choose **Map to Out** (Similar to Step 2. above *Continue shaping, making getDetailsOfEmployee easier to use*; blue markers).
- Because the EMPLOYEE and EMPLOYEE-DETAIL fields are not used in the COBOL server LIST function, you leave them out in the IDL interface: use the context menu of EMPLOYEE-DETAIL field in the **COBOL Interface** pane and choose **Suppress** (similar to Step 2. above *Continue shaping, making getDetailsOfEmployee easier to use*; red markers). Do the same for the EMPLOYEE-ID field.

- 4 Press **Finish** to retrieve the extraction result in the form of a Software AG IDL file. At the same time, a server mapping file for COBOL (Designer file with extension .cvm) is created.

See *Server Mapping Files for COBOL* in the Designer documentation. The *Software AG IDL File* in the IDL Editor documentation describes the interfaces from the client point of view, while the server mapping file contains the mapping to the real COBOL server. Both of these files must be kept together and in sync, otherwise a call to the COBOL server may fail.



To summarize: We created two IDL interfaces. In the IDL file, these resulted in two IDL programs: `getListOfEmployees` and `getDetailsOfEmployee`. Both IDL programs were given readable names (Steps 1 and 3). Meaningful fields were kept, while superfluous fields were suppressed (Steps 1 and 3). The program `getDetailsOfEmployee` contains the redefined fields of parameter `DETAILS` (see below) mapped during extraction (Step 1).



Note: At runtime, the RPC client generated with the extracted interface will send data for the redesigned interfaces, while your COBOL server still expects `EMPLOYEE` data. The EntireX runtime transforms the incoming data stream from the RPC client, using the server mapping file.

```

program 'getDetailsOfEmployee' is
  define data parameter
    1 ID (AV10) In /* Original Name:EMPLOYEE-ID
    1 EMPLOYEE Out /* Original Name:EMPLOYEE-DETAILS
      2 FIRSTNAME (AV20)
      2 SURNAME (AV20)
      2 DATE-BIRTH (AV12)
      2 DETAILS /* Original Name:FILLER_1
        3 ANNUAL-SALARY (P9)
        3 VACATION (N2)
        3 LANGUAGE (AV3)
  end-define
  program 'getListOfEmployees' is
    1 EMPLOYEES (/V99) Out
    2 IDENT (AV10)
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
  end-define

```


Comparison Chart: Fast-track versus User-defined Mapping

The following table highlights the differences when shaping an interface with a *user-defined mapping* compared with a *fast-path extraction*.

	Interface Shaping with User-defined Mapping	Fast-path Extraction
General	User-defined interface with dedicated mapping. For our example, the interfaces are small and tidy and more self-explanatory with long, readable interface names. They are easier to use with the hidden <code>OPERATION</code> field and the suppressed COBOL fields, which are not needed.	Automatic, quick result extraction with COBOL-like interfaces on IS. For our example, the IS interface <code>EMPLOYEE</code> matches exactly 1:1, field by field.
IS service(s)	Multiple small and handy IS services; each <code>OPERATION</code> code is mapped to a separate IS service.	One big IS service.
IS service name	Readable long name.	Short Subprogram name; up to 8 characters.
IS fields	Usage of Suppress and Set Constant reduces the message length. This keeps focus on relevant data items and keeps the client's interface clean. It may also improve performance.	The IS fields and COBOL server parameters match 1:1. As COBOL layout descriptions are sometimes used for many different purposes, irrelevant data items appear and clutter up the IS interface.
<code>OPERATION</code> parameter	Suppressed: the <code>OPERATION</code> parameter does not exist in the IS service as an IS field.	The <code>OPERATION</code> parameter exists in the IS service as an IS field and needs to be filled-in by the client endpoint.
<code>OPERATION</code> code	The <code>OPERATION</code> code is provided internally by EntireX runtime in the <code>OPERATION</code> field.	The <code>OPERATION</code> code needs to be specified in the IS service in the <code>OPERATION</code> field by the client endpoint.
<code>REDEFINE</code> parameters	Either the parameter that is redefined or one of its redefinitions is available as an IS field.	Only the parameter that is redefined is available as an IS field. Redefinitions of the parameter are not available as IS fields.

A user-defined mapping enables you also to define alternative mappings for COBOL data items (`REDEFINES`, tables etc.). These enable scenarios with COBOL servers where data exchange is not fully described by the `LINKAGE SECTION` COBOL data items. For more information refer to the IDL Extractor for COBOL documentation:

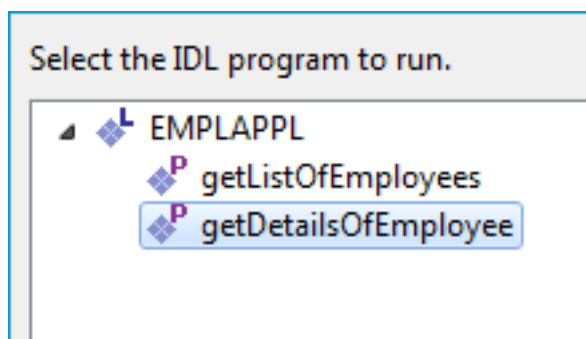
- *Mapping Editor IDL Interface Mapping Functions*
- *User-defined Mapping*

Testing the Extraction Results

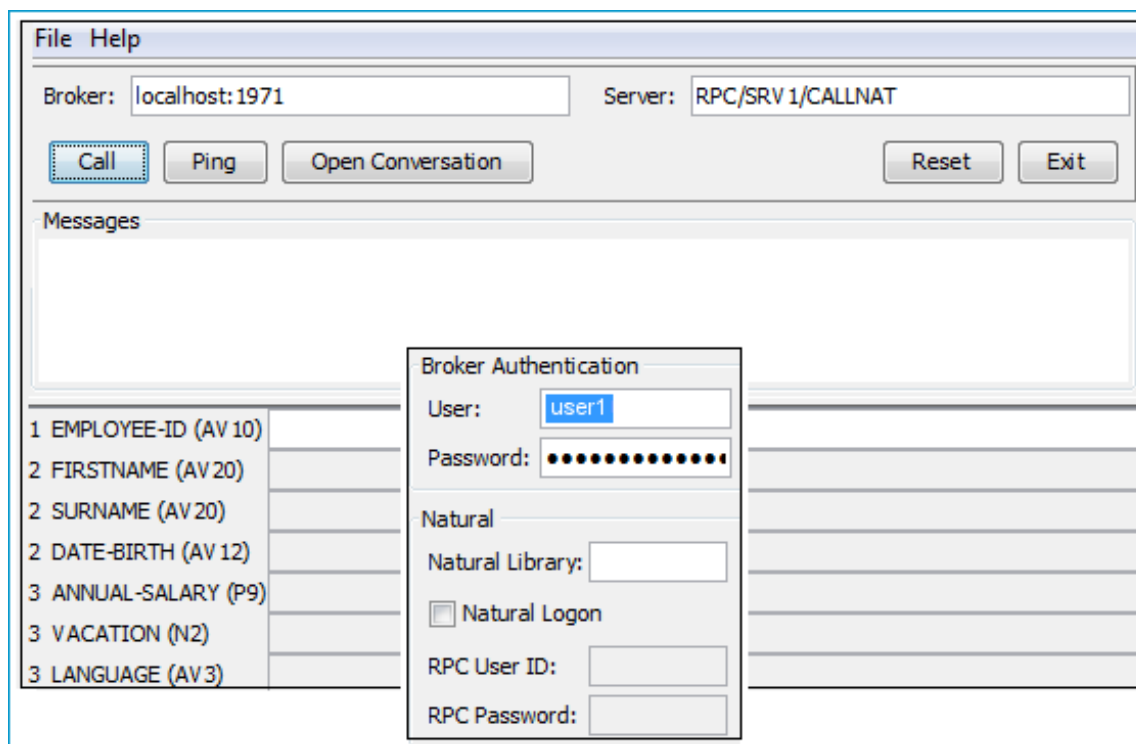
The following pictures use the extraction results described under *Extracting a COBOL Server - Modern Method with User-defined Mapping*.

> To test the extraction results (optional)

- 1 You can test the results of the extraction operation and the REST server back end, using the EntireX IDL Tester. From the context menu of the IDL file in the Designer, choose **Software AG IDL Tester**.



- 2 Select `getDetailsOfEmployee`.



Note that the broker and server parameters contain the explicit route to call the server program, and you can optionally ping the connection from this client. With the **File > Options** dialog, you can set:

- **User** and **Password** for *broker* authentication
- **RPC User ID** and **RPC Password** for *server* authentication

See *EntireX IDL Tester* in the Designer documentation.

- 3 Check the Integration Server log, the EntireX Adapter log or the RPC logs. Applies to all connection methods.

Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server

This section describes your first steps to create a new Integration Server connection. This is described in more detail under *Using the Integration Server Wrapper*, for example working with existing Integration Server connections. This section covers the following topics:

- [Step 1: Start the Integration Server Wrapper Wizard](#)
- [Step 2: Create a New Integration Server Connection](#)
- [Step 3: Select the Connection Type](#)
- [Step 4: Define Adapter Services for an RPC Connection](#)
- [Step 5: Create or Update a REST Resource](#)
- [Step 6: Edit the REST Resource](#)

Step 1: Start the Integration Server Wrapper Wizard

➤ To start the Integration Server Wrapper wizard

- 1 In the context menu of a Software AG IDL file, choose **Integration Server > Generate web-Methods IS Connection**.

This starts the wizard with a list of existing Integration Server Wrapper connections.



Note: If the selected IDL file is not valid because of a syntax error, an error dialog comes up and the wizard does not start.

- 2 Continue with *Step 2: Create a New Integration Server Connection*.

Step 2: Create a New Integration Server Connection

Define New Integration Server Connection

Add host:port, user and password for a new Integration Server

Server:

User:

Password:

☐ Use secure connection

➤ To create a new Integration Server connection

- 1 Define the new Integration Server connection on the wizard page.

**Notes:**

1. The only required field is **Server**. Enter the hostname of the Integration Server including an optional port number. If no port number is specified, port number defaults to "5555". The **Integration Server Authentication** can be passed with the **User** and **Password** fields.
2. Optional settings are for secure connections. The **Truststore for HTTPS** contains all signed certificates and must be a valid truststore.
3. The check box **Verify host name** checks that the hostname is entered in the stored certificate.
4. When the Integration Server has **Client Authentication** enabled, you can specify your **Keystore** file and keystore **Password**.
5. For managing Integration Server connections, see [Preferences](#).

- 2 Choose **Next** and continue with *Step 3: Select the Connection Type*.

Step 3: Select the Connection Type

☒ Create a new EntireX Adapter connection of type: **EntireX RPC Connection**

☐ Update Adapter Services or Listeners to an existing Integration Server connection:

List of Connections

Connection Name	Package Name	Connection Type	Enabled
Employees:EmployeesConnection	Employees	EntireX RPC Connection	Yes

Total: 1

☒ Map Software AG IDL data types to String

☒ Create or Update REST resource

> To create a new connection

- 1 Select a connection type from the drop down list. Connection types are described under *EntireX Adapter Connections* in the EntireX Adapter documentation and *Introduction to the Integration Server Wrapper*.



Note: The list of connection types is filtered: connection types that require a license are only shown if a corresponding license file is available. Reliable RPC connections are only shown if all IDL programs contain only IN parameters. Also, if a server mapping file is available, only those connection types that support the interface type specified in the server mapping file are shown.

- 2 Check the box **Create or Update REST resource**.
- 3 Click **Next** and continue with *Step 4: Define Adapter Services for an RPC Connection*.

Step 4: Define Adapter Services for an RPC Connection

Packages on Integration Server localhost:5555

- Default
- Employees**
- WmART
- WmARTEstDC

Folder Name:

Connection Name:

RPC Connection to EntireX

Broker ID:

Server Address:

➤ To create a connection and related adapter services

- 1 Select an existing package or create a new package for the created objects.
- 2 Define a folder name. If the folder does not exist, it will be created.
- 3 Define a connection name.
- 4 Define the parameters of the connection type. For details, see *Connection Parameters for RPC Connections* in the EntireX Adapter documentation.

As a result, the folder will contain the connection and the adapter services (one for each IDL program). The name of a service is the same as the respective IDL program.

The default settings for the adapter services are:

- the **Default** package; if not available, the first package
- the IDL library name for the **Folder Name**
- the IDL library name with the suffix "Connection" for the **Connection Name**



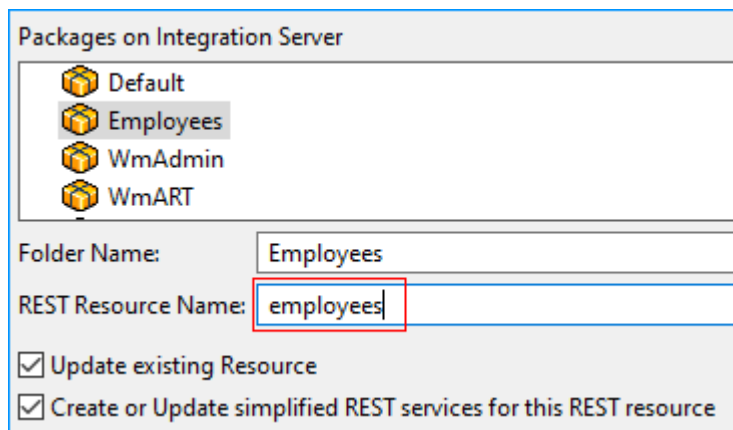
Note: When creating a connection, a package dependency is added such that the selected package depends on webMethods EntireX (the package `WmEntireX`) with the version currently used.

Step 5: Create or Update a REST Resource

➤ To create or update a REST resource

Make sure checkbox **Create or Update REST resource** is checked.

- 1 Select a package.



Packages on Integration Server

- Default
- Employees**
- WmAdmin
- WmART

Folder Name: Employees

REST Resource Name: employees

☒ Update existing Resource

☒ Create or Update simplified REST services for this REST resource

- 2 Define a folder name. If the folder does not exist, it will be created.
- 3 Define the REST Resource name.
- 4 Check **Update existing Resource** if you want to add additional services to an existing resource.
- 5 Check **Create or Update simplified REST services for this REST resource** if you want to generate additional services with a simplified REST API. This allows you to use a simple HTTP-GET method without a payload.

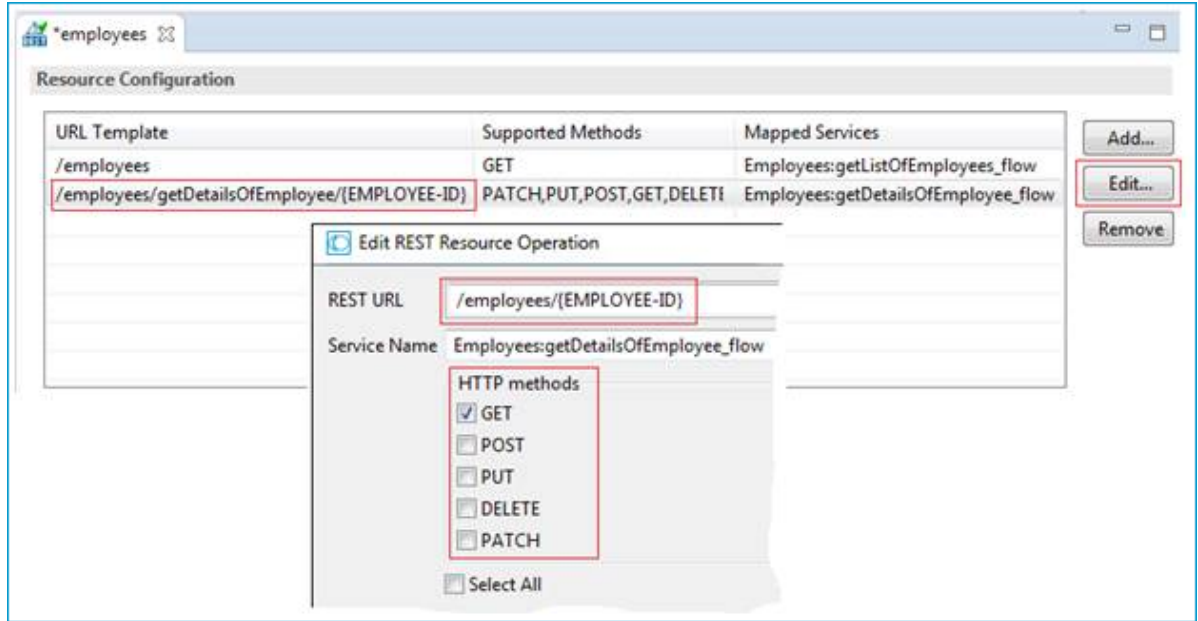


Note: This applies only to APIs where input signature parameters do not contain arrays.

Step 6: Edit the REST Resource

➤ To edit the REST resource

- 1 Switch to the **Service Development** perspective of Software AG Designer.



- 2 Refresh the package where you generated the REST Resource in the previous step.
- 3 Open the REST Resource.
- 4 Edit the REST URLs and HTTP Methods to match the functionality of your application.

For more information see:

- *REST Developer's Guide* in the webMethods Integration Server documentation for information on using the generated REST resources
- *REST Developer's Guide* in the webMethods Integration Server documentation

Task 3: Execute the Call from the REST Client to COBOL

Use an appropriate REST client, specify URL, Method and Headers and invoke your REST resource:

URL http://localhost:5555/restv2/employees

Method GET

Headers Accept application/json

200 OK

```
{
  "errorCode": "00000000",
  "errorFlag": "false",
  "response": {
    "EMPLOYEES-LIST": {
      "EMPLOYEES": [
        {
          "IDENT": "11100102",
          "FIRSTNAME": "Edgar",
          "SURNAME": "Schindler",
          "DATE-BIRTH": "Dec 4, 1962"
        },
        {
          "IDENT": "11100105",
          "FIRSTNAME": "Christian",
          "SURNAME": "Schirm",
          "DATE-BIRTH": "Mar 15, 1961"
        },
        {
          "IDENT": "11100106",
          "FIRSTNAME": "Rainer",
          "SURNAME": "Schmitt",
          "DATE-BIRTH": "Feb 13, 1955"
        },
        {
          "IDENT": "11100107",
          "FIRSTNAME": "Helga",
          "SURNAME": "Schmidt",
          "DATE-BIRTH": "May 26, 1961"
        }
      ]
    }
  }
}
```

URL http://localhost:5555/restv2/employees/111000105

Method GET

Headers Accept application/json

200 OK

```
{
  "errorCode": "00000000",
  "errorFlag": "false",
  "response": {
    "EMPLOYEE-DETAILS": {
      "FIRSTNAME": "Christian",
      "SURNAME": "Schirm",
      "DATE-BIRTH": "Mar 15, 1961",
      "DETAIL": {
        "ANNUAL-SALARY": "68000",
        "VACATION": "21",
        "LANGUAGE": "GER"
      }
    }
  }
}
```


7

Calling COBOL DFHCOMMAREA (Zero Footprint using CICS ECI) on z/OS CICS from REST

■ Introduction	82
■ What do I need to Install for this Scenario?	84
■ Task 1: Extract the Interface of a COBOL Server	85
■ Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server	97
■ Task 3: Execute the Call from the REST Client to COBOL	103

This scenario uses the tools IDL Extractor for COBOL and Integration Server Wrapper of the Designer.

Introduction



- 1 Extract the interface of the COBOL server program. See *Using the IDL Extractor for COBOL - Overview*.
- 2 Generate REST resources, connection and adapter services in Integration Server.
- 3 Execute the call from the REST client to the COBOL server program.

This scenario makes the following important assumptions:

For Task 1:

- You have a working COBOL DFHCOMMAREA server. For illustration and examples on such a server, see *CICS with DFHCOMMAREA Calling Convention* under *Introduction to the IDL Extractor for COBOL*.
- You have access to the related COBOL sources and copybooks as files on your local machine. The minimum requirement is the `DATA DIVISION` of the interface.



For Tasks 2 and 3:

- You have a REST client.
- To call the COBOL server program at runtime using the *EntireX CICS ECI* connection method, you need to configure the CICS ECI TCP/IP service within your CICS region. See *Preparing IBM CICS for ECI* and *Connection Parameters for CICS ECI Connections*.



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.

Task 1: Extract the Interface of a COBOL Server

- [Introduction](#)
- [Sample COBOL Server used in the Examples](#)
- [Extracting a COBOL Server - Fast-track Method](#)
- [Extracting a COBOL Server - Modern Method with User-defined Mapping](#)
- [Comparison Chart: Fast-track versus User-defined Mapping](#)
- [Testing the Extraction Results](#)

Introduction

Follow the instructions for extracting COBOL, see *Using the IDL Extractor for COBOL - Overview* and choose *Scenario I: Create New IDL and Server Mapping Files* if this is your first extraction. Learn how to invoke the IDL Extractor for COBOL, create a COBOL Extractor Environment and about the possibilities to select the COBOL source.

The extraction process creates the following EntireX metafiles:

- IDL file. A Software AG IDL file contains definitions of the interface between client and server. See *Software AG IDL File* in the IDL Editor documentation in the IDL Editor documentation.
- Server mapping file (optional). The mapping file is a Designer file with extension .cvm that contains COBOL-specific mapping information. See *Server Mapping Files for COBOL* in the Designer documentation.

Sample COBOL Server used in the Examples

The following COBOL server is used to illustrate the features of the IDL Extractor for COBOL. Imagine an existing COBOL server called `EMPLOYEE`. It implements the access logic for a `LIST` and `DETAILS` function to a database view `EMPLOYEE`:

```

IDENTIFICATION DIVISION.
    PROGRAM-ID.      EMPLOYEE.
*****
*   Program Name    = EMPLOYEE
*   String Literal  = QUOTE
*****
01 DFHCOMMAREA.
    02 OPERATION                                PIC X(1).
        88 OPERATION-LIST                      VALUE 'L'.
        88 OPERATION-DETAILS                   VALUE 'D'.
    02 EMPLOYEE-ID                              PIC X(10).
    02 EMPLOYEE-DETAILS.
        03 FIRSTNAME                           PIC X(20).
        03 SURNAME                             PIC X(20).
        03 DATE-BIRTH                          PIC X(12).
        03 DETAILS                             PIC X(100).
        03 FILLER REDEFINES DETAILS.
            04 ANNUAL-SALARY                    PACKED-DECIMAL PIC S9(9).
            04 VACATION                         PIC S9(2).
            04 LANGUAGE                         PIC X(3).
            04 FILLER                           PIC X(90).
    02 EMPLOYEES-COUNT                          PIC 9(8) BINARY.
    02 EMPLOYEES OCCURS 1 TO 99 DEPENDING ON EMPLOYEES-COUNT.
        03 IDENT                              PIC X(10).
        03 FIRSTNAME                          PIC X(20).
        03 SURNAME                            PIC X(20).
        03 DATE-BIRTH                         PIC X(12).

```

Two approaches for extracting the COBOL server are described below:

- **Fast-track**

Learn how EntireX helps you to connect the COBOL Server with quick results.

- **User-defined Mapping**

The following features are among those available with a user-defined mapping:

- **Shape the interface to your COBOL server**

You can minimize the interface by suppressing COBOL server fields or by providing constant values. Renaming interfaces allows you to specify a readable long name. These two features together - constants and renaming of interfaces - are most powerful when multiple interfaces are implemented in a single COBOL server. Each function of the COBOL server triggered by an operation code or function code can be modelled as a separate REST resource method/operation.

■ Select alternative mappings

You can select alternative mappings for COBOL data items (REDEFINES, tables etc.) to enable scenarios with COBOL servers where data exchange is not fully described by the LINKAGE SECTION COBOL data items.

These approaches are described under [Fast-track Method](#) and [Modern Method](#) below.

Extracting a COBOL Server - Fast-track Method

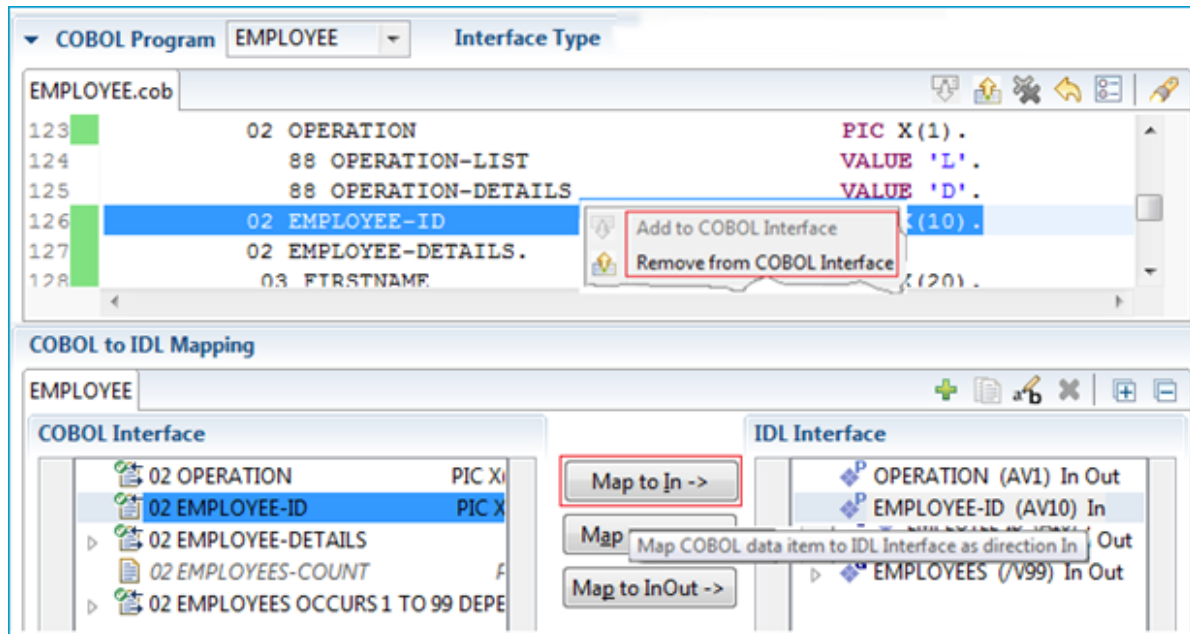
➤ To extract the COBOL Server

- 1 Switch to the **EntireX** perspective.
- 2 Call the IDL Extractor for COBOL.
- 3 Set the correct interface type as described under *Step 4: Define the Extraction Settings and Start Extraction* under *Scenario I: Create New IDL and Server Mapping Files* in the IDL Extractor for COBOL documentation.

- 4 Set the correct value for checkbox **Input Message same as Output Message**.
 - Check the checkbox if the COBOL data structure of the CICS input message is the same as the structure of the CICS output message, that is, there is one COBOL layout for both directions. See the following COBOL server examples in the IDL Extractor for COBOL documentation:
 - *Example 1: Redefines*
 - *Example 2: Buffer Technique*
 - *Example 3: COBOL SET ADDRESS Statements*
 - Clear the checkbox if the COBOL data structure of the CICS input message is different to the structure of the CICS output message, that is, the output overlays the input, which means there is a second COBOL layout for the reply. See the following COBOL server examples in the IDL Extractor for COBOL documentation:
 - *Example 1: Redefines*
 - *Example 2: Buffer Technique*
 - *Example 3: COBOL SET ADDRESS Statements*

For our example, check the check box, because the COBOL layout for input is the same as the COBOL layout for output.

- 5 Press **Next** to enter the COBOL Mapping Editor.



The purpose of the COBOL Mapping Editor is to map COBOL data items to IDL (Interface Definition Language) parameters. It consists of three main areas:

- The upper window pane is the COBOL program source.
- The window pane on the left-hand side is the COBOL interface, consisting of COBOL data items representing the parameters in your COBOL server program.
- The window pane on the right-hand side is the IDL interface consisting of IDL parameters.

What you select depends on check-box **Input Message same as Output Message**:

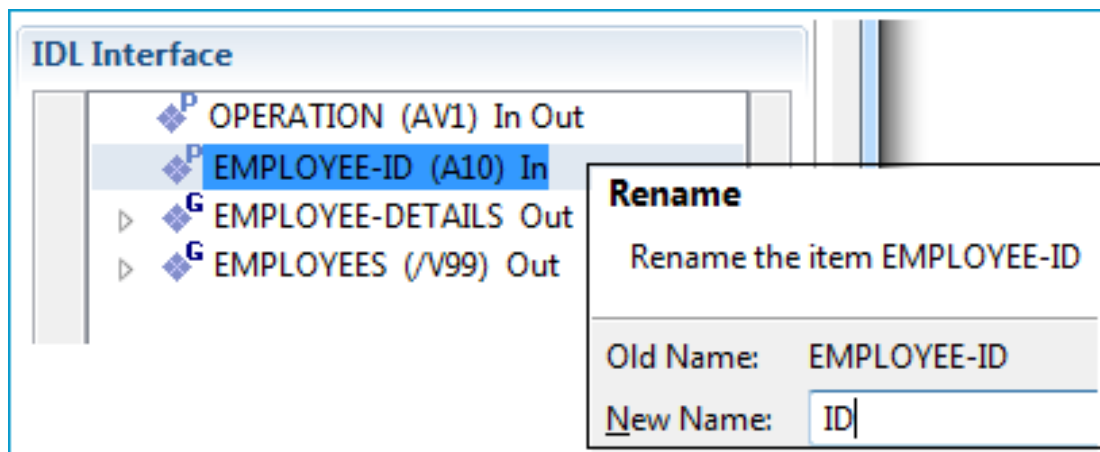
- If there is a single COBOL data structure of the CICS message for input and output - check box **Input Message same as Output Message** is checked - you select a single COBOL data structure in the COBOL Mapping Editor. This structure is used for COBOL input as well as for COBOL output. For more information in this case see *Mapping Editor User Interface* in the IDL Extractor for COBOL documentation.
- If there is a second COBOL data structure for the CICS output message - check box **Input Message same as Output Message** is cleared - you will select two COBOL data structures in the COBOL Mapping Editor. One is used for COBOL input and the other is for COBOL output. For more information in this case see *Mapping Editor User Interface* in the IDL Extractor for COBOL documentation.

COBOL extraction is a two-step process:

- First, select the COBOL parameters using the **Add to** and **Remove from COBOL Interface** from the context menu on the COBOL data items in the upper window pane (COBOL source); in many situations the COBOL interface is detected by the Mapping Editor directly and you can omit this step.
- Then map the COBOL parameters to IDL directions using the **Map to In/Out/InOut** buttons. COBOL parameters do not have directions, they are always INOUT. This step can be left out for a quick test in non-production scenarios; for real production it is recommended to set IDL directions to reduce data transfer. You also have the option to assign a meaningful name to an IDL parameter if the original COBOL data item is not self-explanatory.

For our example:

- Set **Map to In** for COBOL data item `EMPLOYEE-ID` as IDL direction (see screenshot in previous step). For `EMPLOYEES-DETAILS` and `EMPLOYEES` perform the similar **Map to Out**.
- Use the function **Rename** from the context menu on the IDL parameters in the **IDL Interface** pane to rename COBOL parameter `EMPLOYEES-ID` to IDL parameter `ID`. Do the same for `EMPLOYEE-DETAILS`: Rename to `EMPLOYEE`.



- 6 Press **Finish**. The default extraction settings ensure you get useful results. The outcome is a Software AG IDL file (interface definition language), a Designer file with extension `.idl`:

```
library 'EMPLAPPL' is
program 'EMPLOYEE' is
define data parameter
  1 OPERATION (AV1) In Out
  1 ID (AV10) In /* Original Name:EMPLOYEE-ID
  1 EMPLOYEE Out /* Original Name:EMPLOYEE-DETAILS
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
    2 DETAILS (AV100)
  1 EMPLOYEES (/V99) Out
    2 IDENT (AV10)
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
end-define
```

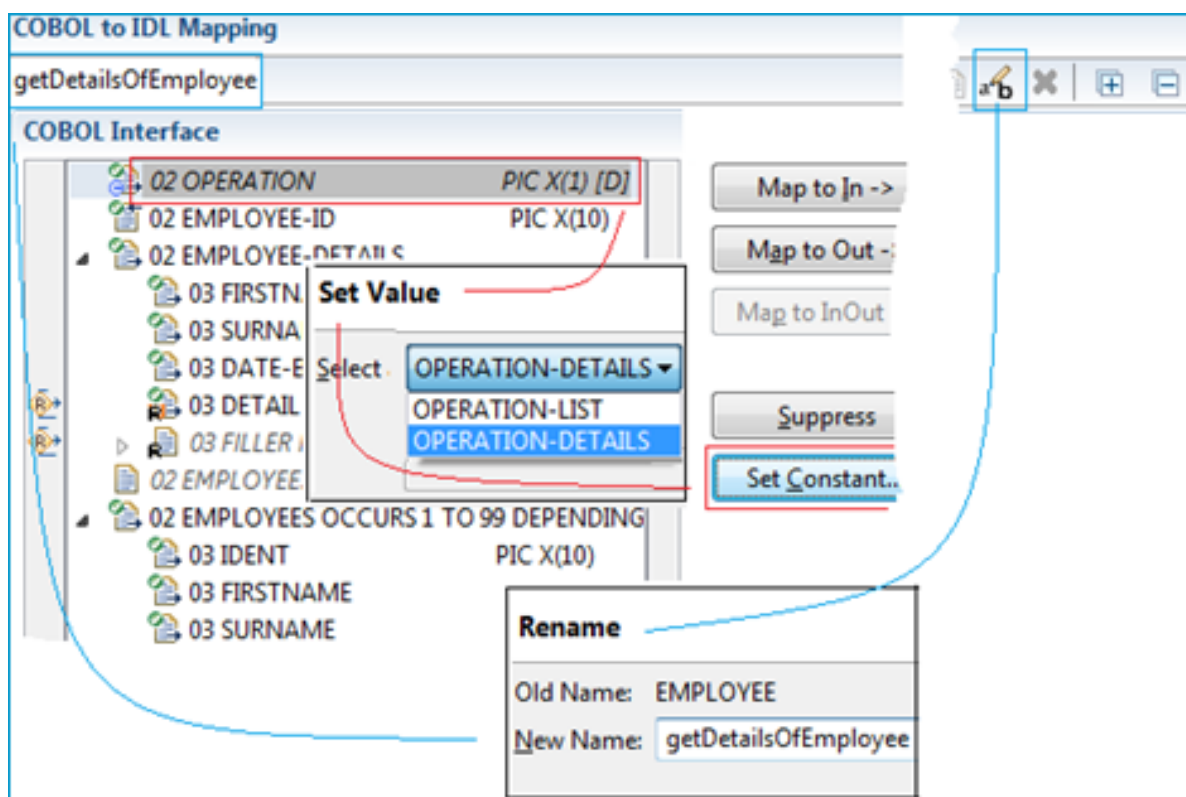
If required, a server mapping file with the same name as the IDL file but the extension `.cvm` may be generated too, for technical reasons. This file must always be stored together with the concomitant IDL file. For more information, see *Server Mapping Files for COBOL* in the Designer documentation. By default, the interface name `EMPLOYEE` is taken from the COBOL server program name.


Extracting a COBOL Server - Modern Method with User-defined Mapping

EntireX allows you to extract all implemented interfaces of the COBOL server separately. Instead of a large `EMPLOYEE` interface, separate interfaces `getListOfEmployees` and `getDetailsOfEmployee` are extracted. Each interface contains required parameters; obsolete parameters for an interface are suppressed, improving its usability.

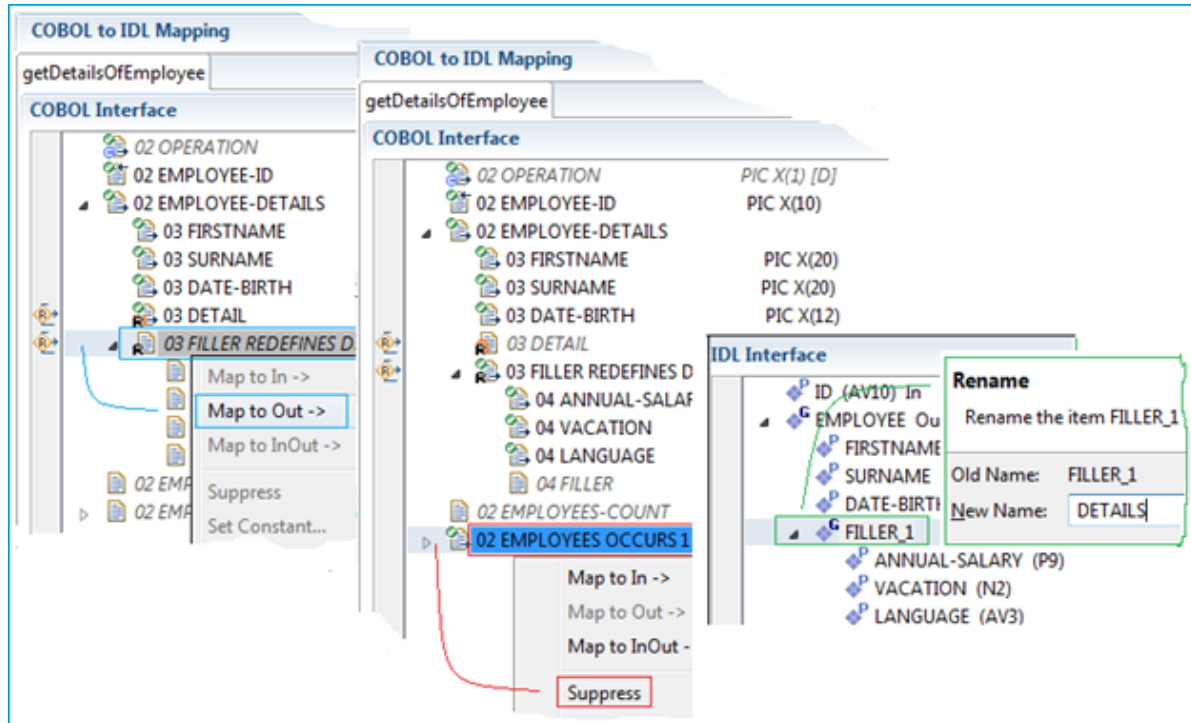
➤ To extract a COBOL server with a user-defined mapping

- 1 Shape `EMPLOYEE` to `getDetailsOfEmployee`:



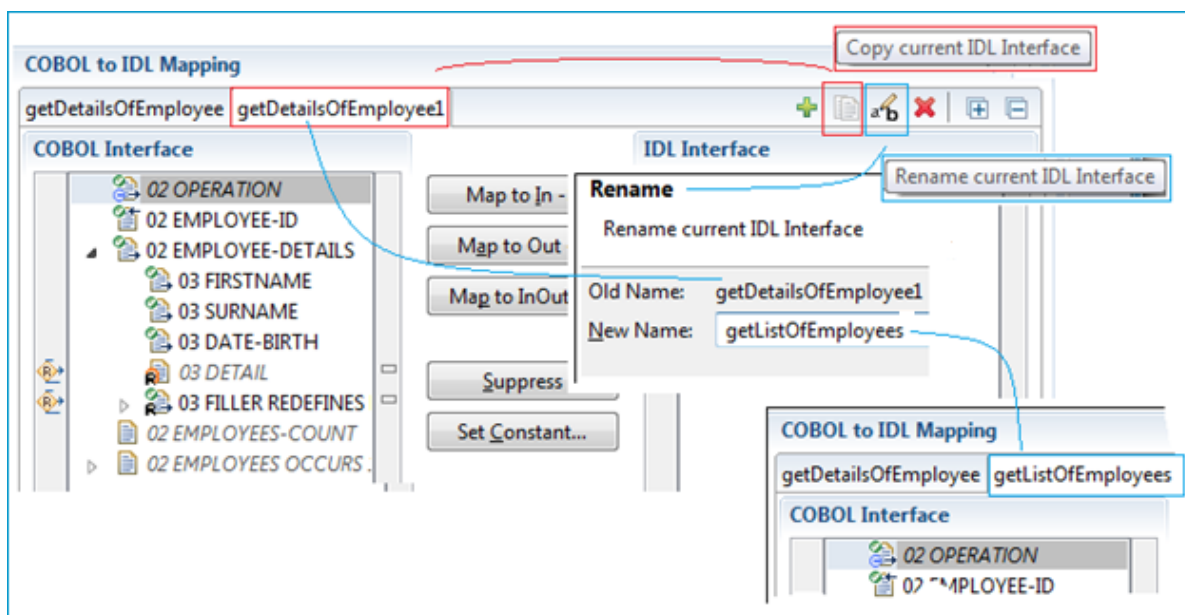
- Mark the `OPERATION` field in the **COBOL Interface** pane and press the **Set Constant** button. In the pop-up window, select `OPERATION-DETAILS` (red markers). The `OPERATION-DETAILS` value originates from COBOL level-88 data item (blue markers). If there is no COBOL level-88 enumeration type in the COBOL program defined, you can enter the function code manually in the pop-up window.
- By default the IDL interface gets the same name as your COBOL server: `EMPLOYEE`. Use the **Rename** button  from the toolbar of the COBOL Mapping Editor and change the IDL interface name to `getDetailsOfEmployee` (blue markers). This name is used as the IS service name later.

2 Continue shaping, making `getDetailsOfEmployee` easier to use:




- Use the context menu of the redefinition of the `DETAIL` field in the **COBOL Interface** pane, `FILLER REDEFINES DETAIL` and choose **Map to Out** (blue markers). A redefinition is a secondary layout (field types) of the same memory area. The redefinition contains more specific information (`ANNUAL-SALARY`, `VACATION`, etc.) than character buffer `DETAIL`. This is the reason why it is more useful in your interface.
- Use the context menu of `FILLER_1` in the IDL interface pane and rename `FILLER_1` to `DETAILS` (green markers) for a readable IS service field name later.
- The `DETAILS` function in the COBOL server does not make use of the `EMPLOYEES` field. Therefore you can leave it out of the interface: use the context menu of the `EMPLOYEES` field in the **COBOL Interface** pane and choose **Suppress** (red markers).


3 Shape `EMPLOYEE` to `getListOfEmployees` Function:



The COBOL Mapping Editor allows you to define multiple IDL interfaces to the same COBOL server:

- You create the additional IDL interface using the toolbar button **Copy current IDL interface**

 (red markers). A new tab `getDetailsOfEmployee1` is created. The previously extracted `getDetailsOfEmployee` interface still exists in the first tab. Once you reactivate the first tab, you will see the interface of `getDetailsOfEmployee` again.

- Use the **Rename** button  from the toolbar of the COBOL Mapping Editor and change the IDL interface name to `getListOfEmployees` (blue markers). This name is used as the IS service name later.

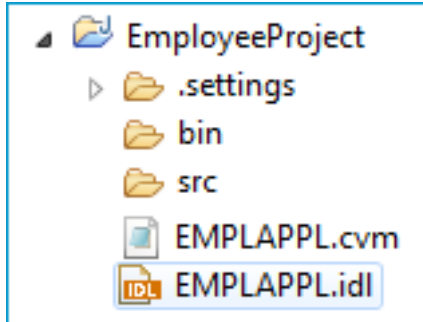
- For the `LIST` function you need the `OPERATION-LIST` value in the `OPERATION` field: Mark the `OPERATION` field in the **COBOL Interface** pane and press the **Set Constant** button. In the pop-up window, select `OPERATION-LIST` (similar to Step 1. above *Shape EMPLOYEE to getDetailsOfEmployee*; red markers).

- Use the context menu of the `EMPLOYEES` field in the **COBOL Interface** pane, `EMPLOYEES OCCURS 1 TO 99` and choose **Map to Out** (Similar to Step 2. above *Continue shaping, making getDetailsOfEmployee easier to use*; blue markers).

- Because the `EMPLOYEE` and `EMPLOYEE-DETAIL` fields are not used in the COBOL server `LIST` function, you leave them out in the IDL interface: use the context menu of `EMPLOYEE-DETAIL` field in the **COBOL Interface** pane and choose **Suppress** (similar to Step 2. above *Continue shaping, making getDetailsOfEmployee easier to use*; red markers). Do the same for the `EMPLOYEE-ID` field.

- 4 Press **Finish** to retrieve the extraction result in the form of a Software AG IDL file. At the same time, a server mapping file for COBOL (Designer file with extension `.cvm`) is created.

See *Server Mapping Files for COBOL* in the Designer documentation. The *Software AG IDL File* in the IDL Editor documentation describes the interfaces from the client point of view, while the server mapping file contains the mapping to the real COBOL server. Both of these files must be kept together and in sync, otherwise a call to the COBOL server may fail.



To summarize: We created two IDL interfaces. In the IDL file, these resulted in two IDL programs: `getListOfEmployees` and `getDetailsOfEmployee`. Both IDL programs were given readable names (Steps 1 and 3). Meaningful fields were kept, while superfluous fields were suppressed (Steps 1 and 3). The program `getDetailsOfEmployee` contains the redefined fields of parameter `DETAILS` (see below) mapped during extraction (Step 1).



Note: At runtime, the RPC client generated with the extracted interface will send data for the redesigned interfaces, while your COBOL server still expects `EMPLOYEE` data. The EntireX runtime transforms the incoming data stream from the RPC client, using the server mapping file.

```

program 'getDetailsOfEmployee' is
  define data parameter
    1 ID (AV10) In /* Original Name:EMPLOYEE-ID
    1 EMPLOYEE Out /* Original Name:EMPLOYEE-DETAILS
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
    2 DETAILS /* Original Name:FILLER_1
    3 ANNUAL-SALARY (P9)
    3 VACATION (N2)
    3 LANGUAGE (AV3)
  end-define
  program 'getListOfEmployees' is
    1 EMPLOYEES (/V99) Out
    2 IDENT (AV10)
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
  end-define
    
```

Comparison Chart: Fast-track versus User-defined Mapping

The following table highlights the differences when shaping an interface with a *user-defined mapping* compared with a *fast-path extraction*.

	Interface Shaping with User-defined Mapping	Fast-path Extraction
General	User-defined interface with dedicated mapping. For our example, the interfaces are small and tidy and more self-explanatory with long, readable interface names. They are easier to use with the hidden <code>OPERATION</code> field and the suppressed COBOL fields, which are not needed.	Automatic, quick result extraction with COBOL-like interfaces on IS. For our example, the IS interface <code>EMPLOYEE</code> matches exactly 1:1, field by field.
IS service(s)	Multiple small and handy IS services; each <code>OPERATION</code> code is mapped to a separate IS service.	One big IS service.
IS service name	Readable long name.	Short Subprogram name; up to 8 characters.
IS fields	Usage of Suppress and Set Constant reduces the message length. This keeps focus on relevant data items and keeps the client's interface clean. It may also improve performance.	The IS fields and COBOL server parameters match 1:1. As COBOL layout descriptions are sometimes used for many different purposes, irrelevant data items appear and clutter up the IS interface.
<code>OPERATION</code> parameter	Suppressed: the <code>OPERATION</code> parameter does not exist in the IS service as an IS field.	The <code>OPERATION</code> parameter exists in the IS service as an IS field and needs to be filled-in by the client endpoint.
<code>OPERATION</code> code	The <code>OPERATION</code> code is provided internally by EntireX runtime in the <code>OPERATION</code> field.	The <code>OPERATION</code> code needs to be specified in the IS service in the <code>OPERATION</code> field by the client endpoint.
<code>REDEFINE</code> parameters	Either the parameter that is redefined or one of its redefinitions is available as an IS field.	Only the parameter that is redefined is available as an IS field. Redefinitions of the parameter are not available as IS fields.

A user-defined mapping enables you also to define alternative mappings for COBOL data items (`REDEFINES`, tables etc.). These enable scenarios with COBOL servers where data exchange is not fully described by the `LINKAGE SECTION` COBOL data items. For more information refer to the IDL Extractor for COBOL documentation:

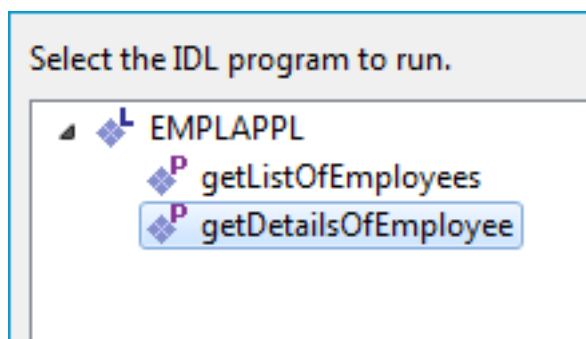
- *Mapping Editor IDL Interface Mapping Functions* if the check-box **Input Message same as Output Message** was checked.
- *Mapping Editor IDL Interface Mapping Functions* if the check-box **Input Message same as Output Message** was cleared.
- *User-defined Mapping*

Testing the Extraction Results

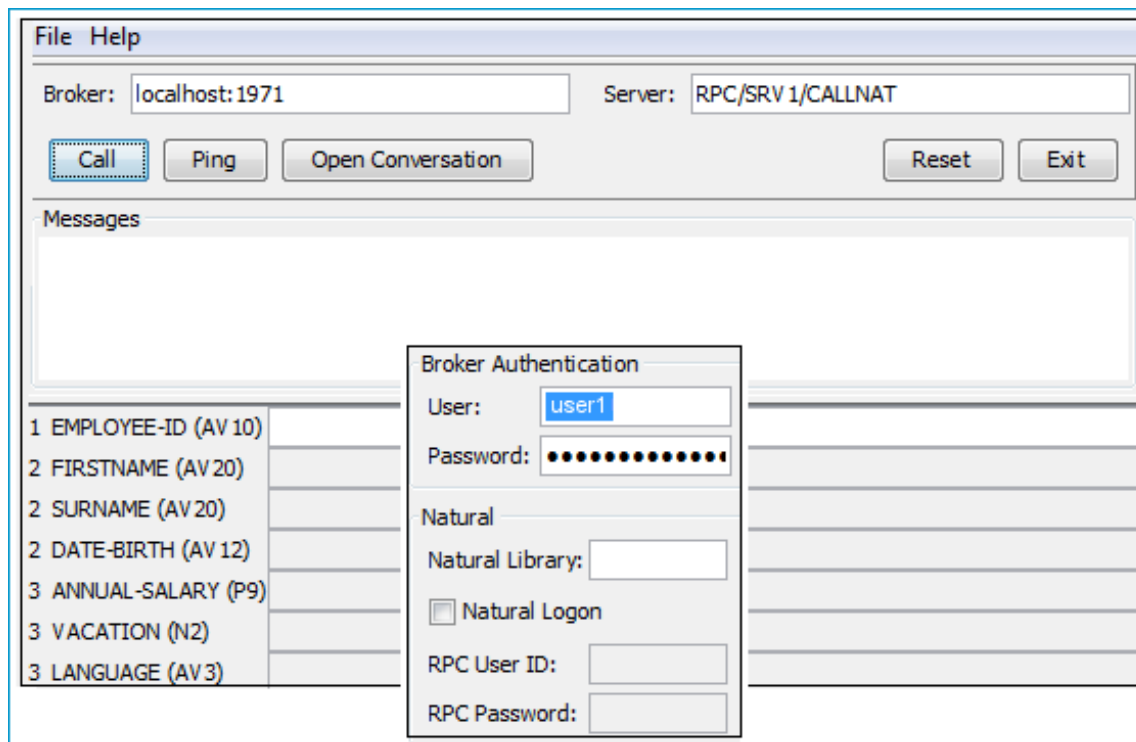
The following pictures use the extraction results described under *Extracting a COBOL Server - Modern Method with User-defined Mapping*.

> To test the extraction results (optional)

- 1 You can test the results of the extraction operation and the REST server back end, using the EntireX IDL Tester. From the context menu of the IDL file in the Designer, choose **Software AG IDL Tester**.



- 2 Select `getDetailsOfEmployee`.



Note that the broker and server parameters contain the explicit route to call the server program, and you can optionally ping the connection from this client. With the **File > Options** dialog, you can set:

- **User** and **Password** for *broker* authentication
- **RPC User ID** and **RPC Password** for *server* authentication

See *EntireX IDL Tester* in the Designer documentation.

- 3 Check the Integration Server log, the EntireX Adapter log or the RPC logs. Applies to all connection methods.

Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server

This section describes your first steps to create a new Integration Server connection. This is described in more detail under *Using the Integration Server Wrapper*, for example working with existing Integration Server connections. This section covers the following topics:

- [Step 1: Start the Integration Server Wrapper Wizard](#)
- [Step 2: Create a New Integration Server Connection](#)
- [Step 3: Select the Connection Type](#)
- [Step 4: Define Adapter Services for a CICS ECI Connection](#)
- [Step 5: Create or Update a REST Resource](#)
- [Step 6: Edit the REST Resource](#)

Step 1: Start the Integration Server Wrapper Wizard

➤ To start the Integration Server Wrapper wizard

- 1 In the context menu of a Software AG IDL file, choose **Integration Server > Generate web-Methods IS Connection**.

This starts the wizard with a list of existing Integration Server Wrapper connections.



Note: If the selected IDL file is not valid because of a syntax error, an error dialog comes up and the wizard does not start.

- 2 Continue with *Step 2: Create a New Integration Server Connection*.

Step 2: Create a New Integration Server Connection

Define New Integration Server Connection

Add host:port, user and password for a new Integration Server

Server:

User:

Password:

☐
Use secure connection

➤ To create a new Integration Server connection

- 1 Define the new Integration Server connection on the wizard page.



Notes:

1. The only required field is **Server**. Enter the hostname of the Integration Server including an optional port number. If no port number is specified, port number defaults to "5555". The **Integration Server Authentication** can be passed with the **User** and **Password** fields.
 2. Optional settings are for secure connections. The **Truststore for HTTPS** contains all signed certificates and must be a valid truststore.
 3. The check box **Verify host name** checks that the hostname is entered in the stored certificate.
 4. When the Integration Server has **Client Authentication** enabled, you can specify your **Keystore** file and keystore **Password**.
 5. For managing Integration Server connections, see [Preferences](#).
- 2 Choose **Next** and continue with *Step 3: Select the Connection Type*.

Step 3: Select the Connection Type

☒ Create a new EntireX Adapter connection of type **CICS ECI Connection**

☐ Update Adapter Services or Listeners to an existing Integration Server connection:

List of Connections

Connection Name	Package Name	Connection Type	Enabled
Employees:EmployeesConnection	Employees	EntireX RPC Connection	Yes

Total: 1

☒ Map Software AG IDL data types to String

☒ Create or Update REST resource

> To create a new connection

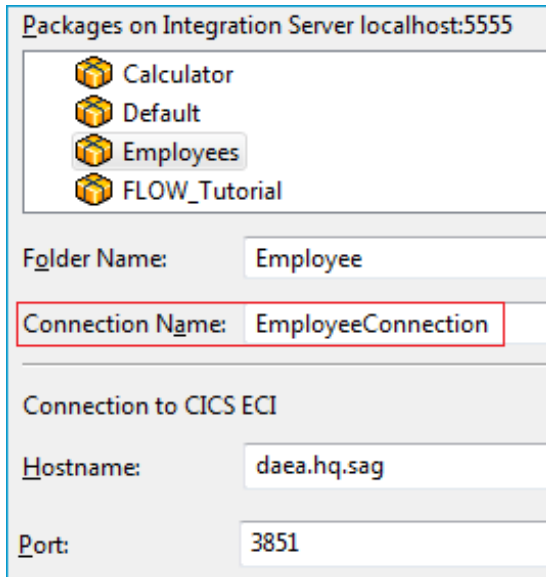
- 1 Select a connection type from the drop down list. Connection types are described under *EntireX Adapter Connections* in the EntireX Adapter documentation and *Introduction to the Integration Server Wrapper*.



Note: The list of connection types is filtered: connection types that require a license are only shown if a corresponding license file is available. Reliable RPC connections are only shown if all IDL programs contain only IN parameters. Also, if a server mapping file is available, only those connection types that support the interface type specified in the server mapping file are shown.

- 2 Check the box **Create or Update REST resource**.
- 3 Click **Next** and continue with *Step 4: Define Adapter Services for a CICS ECI Connection*.

Step 4: Define Adapter Services for a CICS ECI Connection



Packages on Integration Server localhost:5555

- Calculator
- Default
- Employees
- FLOW_Tutorial

Folder Name: Employee

Connection Name: EmployeeConnection

Connection to CICS ECI

Hostname: daea.hq.sag

Port: 3851

➤ To create a connection and related adapter services

- 1 Select a package for the created objects.
- 2 Define a folder name. If the folder does not exist, it will be created.
- 3 Define a connection name.
- 4 Define the parameters of the connection type. For details, see *Connection Parameters for CICS ECI Connections* in the EntireX Adapter documentation.

As a result, the folder will contain the connection and the adapter services (one for each IDL program). The name of a service is the same as the respective IDL program.

The default settings for the adapter services are:

- the **Default** package; if not available, the first package
- the IDL library name for the **Folder Name**
- the IDL library name with the suffix "Connection" for the **Connection Name**



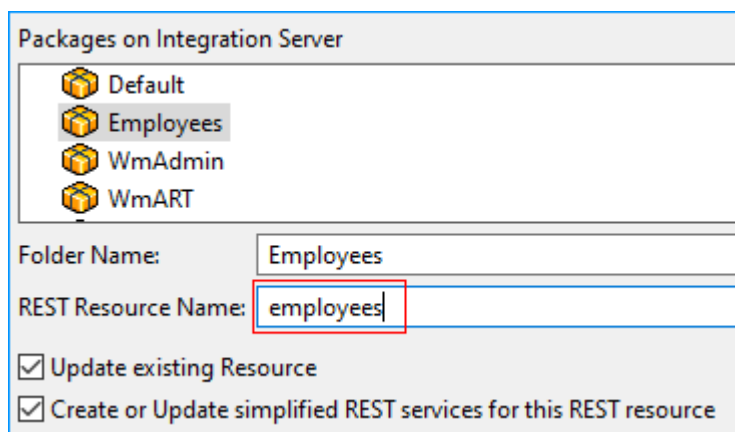
Note: When creating a connection, a package dependency is added such that the selected package depends on webMethods EntireX (the package `WmEntireX`) with the version currently used.

Step 5: Create or Update a REST Resource

➤ To create or update a REST resource

Make sure checkbox **Create or Update REST resource** is checked.

- 1 Select a package.



Packages on Integration Server

- Default
- Employees
- WmAdmin
- WmART

Folder Name: Employees

REST Resource Name: employees

☒ Update existing Resource

☒ Create or Update simplified REST services for this REST resource

- 2 Define a folder name. If the folder does not exist, it will be created.
- 3 Define the REST Resource name.
- 4 Check **Update existing Resource** if you want to add additional services to an existing resource.
- 5 Check **Create or Update simplified REST services for this REST resource** if you want to generate additional services with a simplified REST API. This allows you to use a simple HTTP-GET method without a payload.

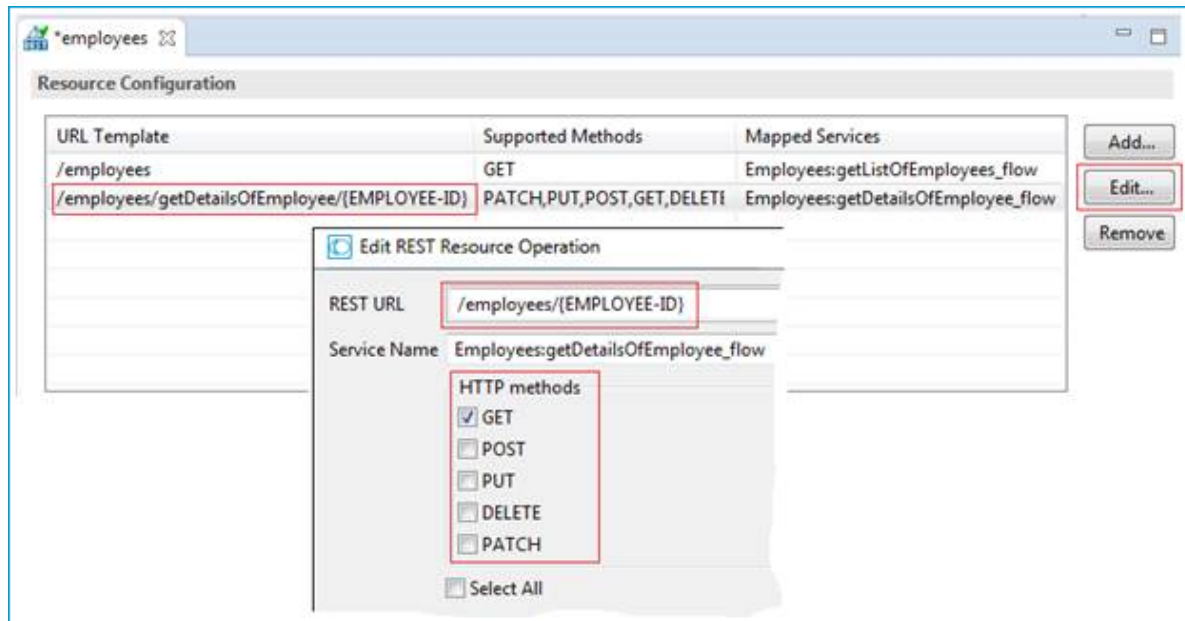


Note: This applies only to APIs where input signature parameters do not contain arrays.

Step 6: Edit the REST Resource

➤ To edit the REST resource

- 1 Switch to the **Service Development** perspective of Software AG Designer.



- 2 Refresh the package where you generated the REST Resource in the previous step.
- 3 Open the REST Resource.
- 4 Edit the REST URLs and HTTP Methods to match the functionality of your application.

For more information see:

- *REST Developer's Guide* in the webMethods Integration Server documentation for information on using the generated REST resources
- *REST Developer's Guide* in the webMethods Integration Server documentation

Task 3: Execute the Call from the REST Client to COBOL

Use an appropriate REST client, specify URL, Method and Headers and invoke your REST resource:

URL http://localhost:5555/restv2/employees

Method GET

Headers Accept application/json

200 OK

```
{
  "errorCode": "00000000",
  "errorFlag": "false",
  "response": {
    "EMPLOYEES-LIST": {
      "EMPLOYEES": [
        {
          "IDENT": "11100102",
          "FIRSTNAME": "Edgar",
          "SURNAME": "Schindler",
          "DATE-BIRTH": "Dec 4, 1962"
        },
        {
          "IDENT": "11100105",
          "FIRSTNAME": "Christian",
          "SURNAME": "Schirm",
          "DATE-BIRTH": "Mar 15, 1961"
        },
        {
          "IDENT": "11100106",
          "FIRSTNAME": "Rainer",
          "SURNAME": "Schmitt",
          "DATE-BIRTH": "Feb 13, 1955"
        },
        {
          "IDENT": "11100107",
          "FIRSTNAME": "Helga",
          "SURNAME": "Schmidt",
          "DATE-BIRTH": "May 26, 1961"
        }
      ]
    }
  }
}
```

URL http://localhost:5555/restv2/employees/111000105

Method GET

Headers Accept application/json

200 OK

```
{
  "errorCode": "00000000",
  "errorFlag": "false",
  "response": {
    "EMPLOYEE-DETAILS": {
      "FIRSTNAME": "Christian",
      "SURNAME": "Schirm",
      "DATE-BIRTH": "Mar 15, 1961",
      "DETAIL": {
        "ANNUAL-SALARY": "68000",
        "VACATION": "21",
        "LANGUAGE": "GER"
      }
    }
  }
}
```


8

Calling COBOL DFHCOMMAREA (Zero Footprint using CICS IPIC) on z/OS CICS from REST

■ Introduction	106
■ What do I need to Install for this Scenario?	107
■ Task 1: Extract the Interface of a COBOL Server	108
■ Task 2: Generate the Connection and Adapter Services in Integration Server	121
■ Task 3: Execute the Call from the REST Client to COBOL	127

This scenario uses the tools IDL Extractor for COBOL and Integration Server Wrapper of the Designer.

Introduction



This scenario makes the following important assumptions:

For Task 1:

- You have a working COBOL DFHCOMMAREA server. For illustration and examples on such a server, see *CICS with DFHCOMMAREA Calling Convention* under *Introduction to the IDL Extractor for COBOL*.
- You have access to the related COBOL sources and copybooks as files on your local machine. The minimum requirement is the DATA DIVISION of the interface.



For Tasks 2 and 3:

- To call the COBOL server program at runtime using the *EntireX CICS IPIC* connection method, you need to configure the CICS IPIC TCP/IP service within your CICS region. See *Preparing IBM CICS for IPIC* and *Connection Parameters for CICS IPIC Connections*.



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.

Task 1: Extract the Interface of a COBOL Server

- [Introduction](#)
- [Sample COBOL Server used in the Examples](#)
- [Extracting a COBOL Server - Fast-track Method](#)
- [Extracting a COBOL Server - Modern Method with User-defined Mapping](#)
- [Comparison Chart: Fast-track versus User-defined Mapping](#)
- [Testing the Extraction Results](#)

Introduction

Follow the instructions for extracting COBOL, see *Using the IDL Extractor for COBOL - Overview* and choose *Scenario I: Create New IDL and Server Mapping Files* if this is your first extraction. Learn how to invoke the IDL Extractor for COBOL, create a COBOL Extractor Environment and about the possibilities to select the COBOL source.

The extraction process creates the following EntireX metafiles:

- IDL file. A Software AG IDL file contains definitions of the interface between client and server. See *Software AG IDL File* in the IDL Editor documentation in the IDL Editor documentation.
- Server mapping file (optional). The mapping file is a Designer file with extension .cvm that contains COBOL-specific mapping information. See *Server Mapping Files for COBOL* in the Designer documentation.

Sample COBOL Server used in the Examples

The following COBOL server is used to illustrate the features of the IDL Extractor for COBOL. Imagine an existing COBOL server called `EMPLOYEE`. It implements the access logic for a `LIST` and `DETAILS` function to a database view `EMPLOYEE`:

```

IDENTIFICATION DIVISION.
    PROGRAM-ID.      EMPLOYEE.
*****
*   Program Name    = EMPLOYEE
*   String Literal  = QUOTE
*****
01 DFHCOMMAREA.
    02 OPERATION                                PIC X(1).
        88 OPERATION-LIST                      VALUE 'L'.
        88 OPERATION-DETAILS                  VALUE 'D'.
    02 EMPLOYEE-ID                              PIC X(10).
    02 EMPLOYEE-DETAILS.
        03 FIRSTNAME                          PIC X(20).
        03 SURNAME                          PIC X(20).
        03 DATE-BIRTH                        PIC X(12).
        03 DETAILS                          PIC X(100).
        03 FILLER REDEFINES DETAILS.
            04 ANNUAL-SALARY                PACKED-DECIMAL PIC S9(9).
            04 VACATION                      PIC S9(2).
            04 LANGUAGE                      PIC X(3).
            04 FILLER                      PIC X(90).
    02 EMPLOYEES-COUNT                          PIC 9(8) BINARY.
    02 EMPLOYEES OCCURS 1 TO 99 DEPENDING ON EMPLOYEES-COUNT.
        03 IDENT                            PIC X(10).
        03 FIRSTNAME                        PIC X(20).
        03 SURNAME                        PIC X(20).
        03 DATE-BIRTH                      PIC X(12).

```

Two approaches for extracting the COBOL server are described below:

- **Fast-track**

Learn how EntireX helps you to connect the COBOL Server with quick results.

- **User-defined Mapping**

The following features are among those available with a user-defined mapping:

- **Shape the interface to your COBOL server**

You can minimize the interface by suppressing COBOL server fields or by providing constant values. Renaming interfaces allows you to specify a readable long name. These two features together - constants and renaming of interfaces - are most powerful when multiple interfaces are implemented in a single COBOL server. Each function of the COBOL server triggered by an operation code or function code can be modelled as a separate REST resource method/operation.

■ Select alternative mappings

You can select alternative mappings for COBOL data items (REDEFINES, tables etc.) to enable scenarios with COBOL servers where data exchange is not fully described by the LINKAGE SECTION COBOL data items.

These approaches are described under [Fast-track Method](#) and [Modern Method](#) below.

Extracting a COBOL Server - Fast-track Method

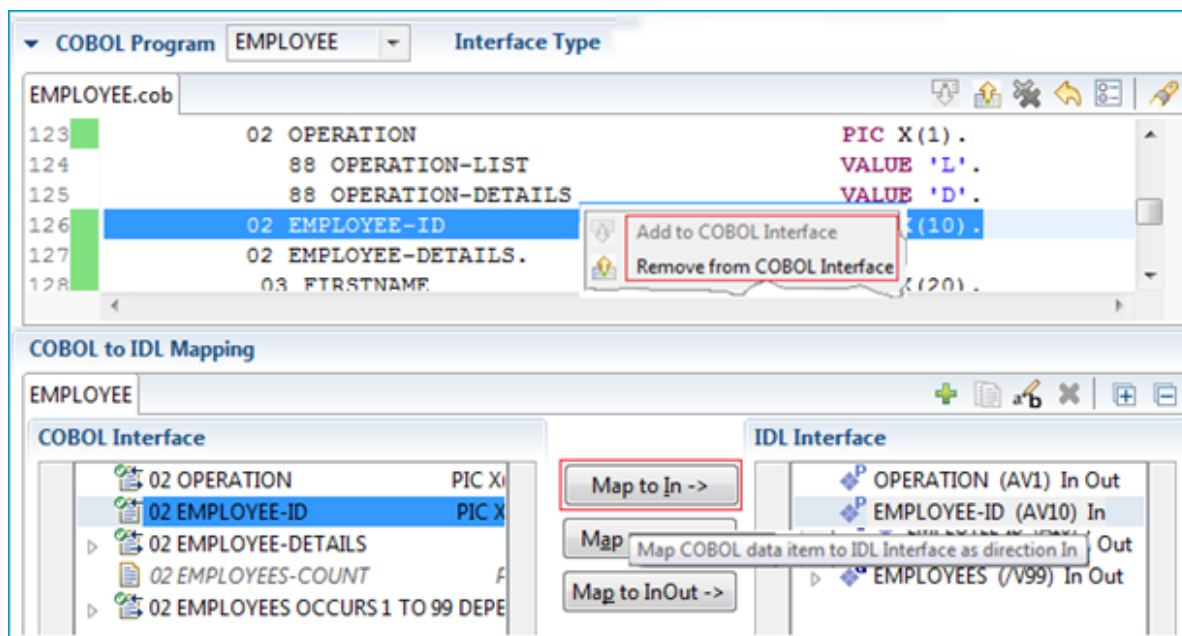
➤ To extract the COBOL Server

- 1 Switch to the **EntireX** perspective.
- 2 Call the IDL Extractor for COBOL.
- 3 Set the correct interface type as described under *Step 4: Define the Extraction Settings and Start Extraction* under *Scenario I: Create New IDL and Server Mapping Files* in the IDL Extractor for COBOL documentation.

- 4 Set the correct value for checkbox **Input Message same as Output Message**.
 - Check the checkbox if the COBOL data structure of the CICS input message is the same as the structure of the CICS output message, that is, there is one COBOL layout for both directions. See the following COBOL server examples in the IDL Extractor for COBOL documentation:
 - *Example 1: Redefines*
 - *Example 2: Buffer Technique*
 - *Example 3: COBOL SET ADDRESS Statements*
 - Clear the checkbox if the COBOL data structure of the CICS input message is different to the structure of the CICS output message, that is, the output overlays the input, which means there is a second COBOL layout for the reply. See the following COBOL server examples in the IDL Extractor for COBOL documentation:
 - *Example 1: Redefines*
 - *Example 2: Buffer Technique*
 - *Example 3: COBOL SET ADDRESS Statements*

For our example, check the check box, because the COBOL layout for input is the same as the COBOL layout for output.

- 5 Press **Next** to enter the COBOL Mapping Editor.



The purpose of the COBOL Mapping Editor is to map COBOL data items to IDL (Interface Definition Language) parameters. It consists of three main areas:

- The upper window pane is the COBOL program source.
- The window pane on the left-hand side is the COBOL interface, consisting of COBOL data items representing the parameters in your COBOL server program.
- The window pane on the right-hand side is the IDL interface consisting of IDL parameters.

What you select depends on check-box **Input Message same as Output Message**:

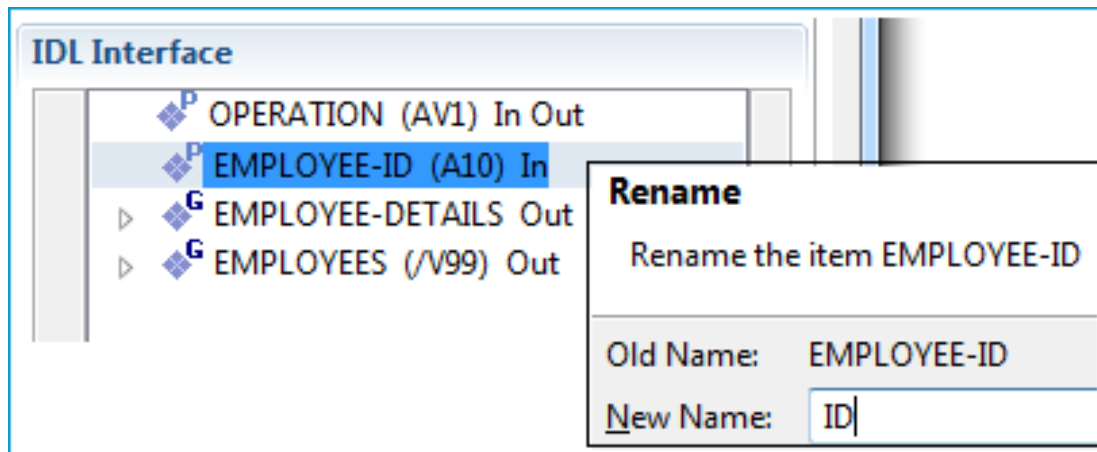
- If there is a single COBOL data structure of the CICS message for input and output - check box **Input Message same as Output Message** is checked - you select a single COBOL data structure in the COBOL Mapping Editor. This structure is used for COBOL input as well as for COBOL output. For more information in this case see *Mapping Editor User Interface* in the IDL Extractor for COBOL documentation.
- If there is a second COBOL data structure for the CICS output message - check box **Input Message same as Output Message** is cleared - you will select two COBOL data structures in the COBOL Mapping Editor. One is used for COBOL input and the other is for COBOL output. For more information in this case see *Mapping Editor User Interface* in the IDL Extractor for COBOL documentation.

COBOL extraction is a two-step process:

- First, select the COBOL parameters using the **Add to** and **Remove from COBOL Interface** from the context menu on the COBOL data items in the upper window pane (COBOL source); in many situations the COBOL interface is detected by the Mapping Editor directly and you can omit this step.
- Then map the COBOL parameters to IDL directions using the **Map to In/Out/InOut** buttons. COBOL parameters do not have directions, they are always INOUT. This step can be left out for a quick test in non-production scenarios; for real production it is recommended to set IDL directions to reduce data transfer. You also have the option to assign a meaningful name to an IDL parameter if the original COBOL data item is not self-explanatory.

For our example:

- Set **Map to In** for COBOL data item `EMPLOYEE-ID` as IDL direction (see screenshot in previous step). For `EMPLOYEES-DETAILS` and `EMPLOYEES` perform the similar **Map to Out**.
- Use the function **Rename** from the context menu on the IDL parameters in the **IDL Interface** pane to rename COBOL parameter `EMPLOYEES-ID` to IDL parameter `ID`. Do the same for `EMPLOYEE-DETAILS`: Rename to `EMPLOYEE`.



- 6 Press **Finish**. The default extraction settings ensure you get useful results. The outcome is a Software AG IDL file (interface definition language), a Designer file with extension `.idl`:


```

library 'EMPLAPPL' is
program 'EMPLOYEE' is
  define data parameter
    1 OPERATION (AV1) In Out
    1 ID (AV10) In /* Original Name:EMPLOYEE-ID
    1 EMPLOYEE Out /* Original Name:EMPLOYEE-DETAILS
      2 FIRSTNAME (AV20)
      2 SURNAME (AV20)
      2 DATE-BIRTH (AV12)
      2 DETAILS (AV100)
    1 EMPLOYEES (/V99) Out
      2 IDENT (AV10)
      2 FIRSTNAME (AV20)
      2 SURNAME (AV20)
      2 DATE-BIRTH (AV12)
  end-define

```

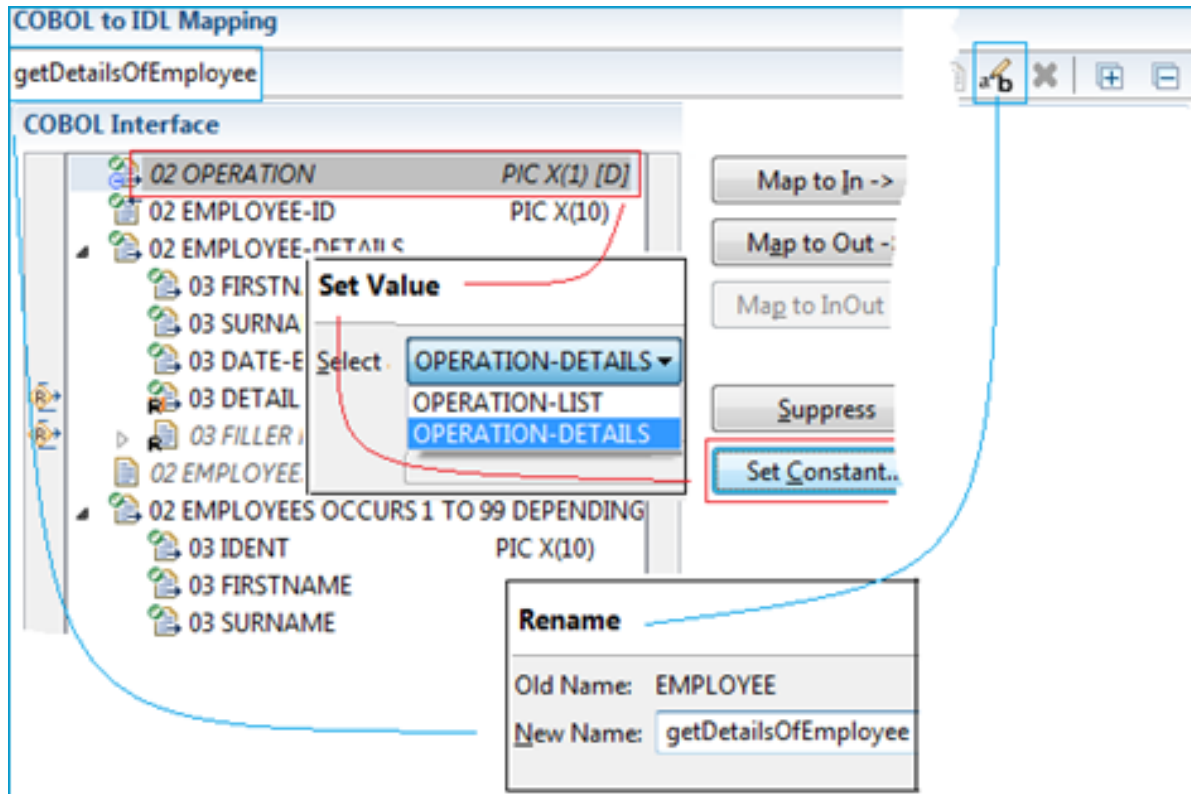
If required, a server mapping file with the same name as the IDL file but the extension .cvm may be generated too, for technical reasons. This file must always be stored together with the concomitant IDL file. For more information, see *Server Mapping Files for COBOL* in the Designer documentation. By default, the interface name `EMPLOYEE` is taken from the COBOL server program name.


Extracting a COBOL Server - Modern Method with User-defined Mapping

EntireX allows you to extract all implemented interfaces of the COBOL server separately. Instead of a large `EMPLOYEE` interface, separate interfaces `getListOfEmployees` and `getDetailsOfEmployee` are extracted. Each interface contains required parameters; obsolete parameters for an interface are suppressed, improving its usability.

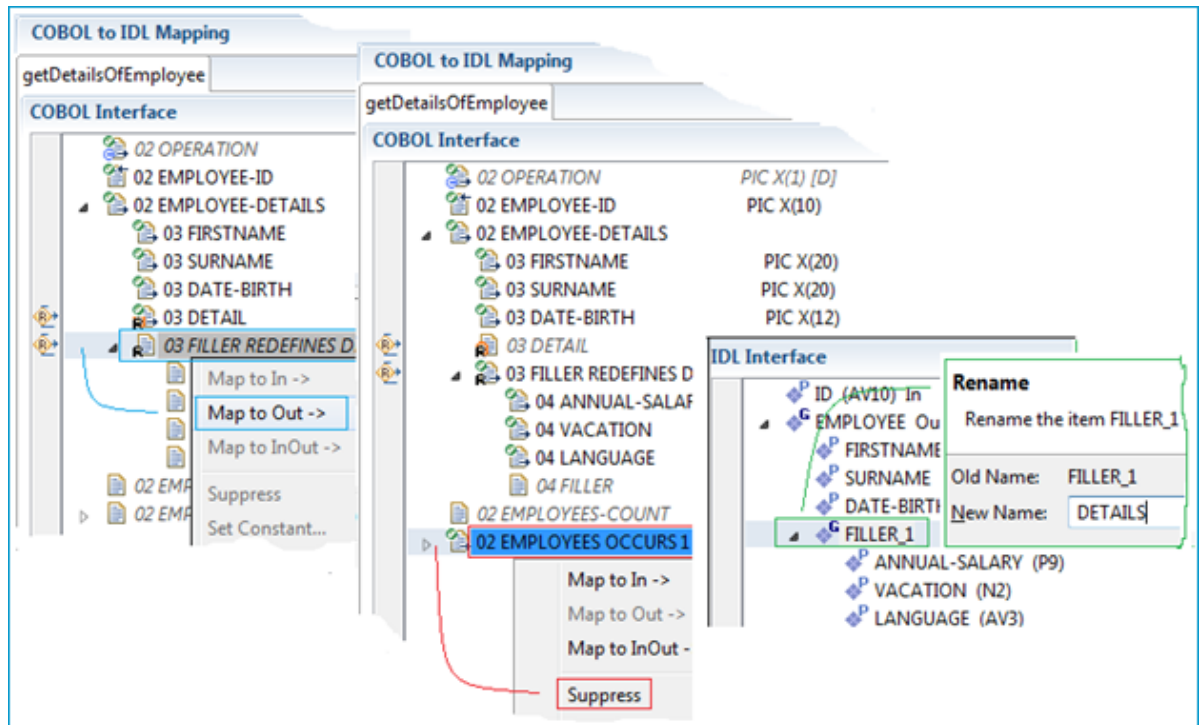
➤ To extract a COBOL server with a user-defined mapping

- 1 Shape `EMPLOYEE` to `getDetailsOfEmployee`:



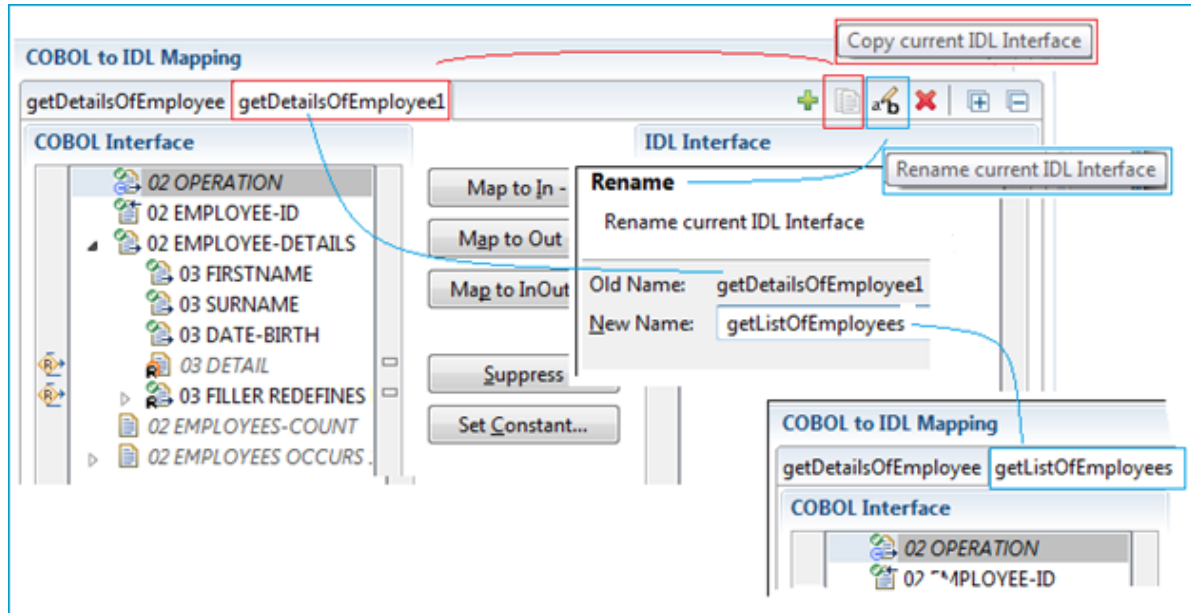
- Mark the `OPERATION` field in the **COBOL Interface** pane and press the **Set Constant** button. In the pop-up window, select `OPERATION-DETAILS` (red markers). The `OPERATION-DETAILS` value originates from COBOL level-88 data item (blue markers). If there is no COBOL level-88 enumeration type in the COBOL program defined, you can enter the function code manually in the pop-up window.
- By default the IDL interface gets the same name as your COBOL server: `EMPLOYEE`. Use the **Rename** button  from the toolbar of the COBOL Mapping Editor and change the IDL interface name to `getDetailsOfEmployee` (blue markers). This name is used as the IS service name later.

2 Continue shaping, making `getDetailsOfEmployee` easier to use:




- Use the context menu of the redefinition of the `DETAIL` field in the **COBOL Interface** pane, `FILLER REDEFINES DETAIL` and choose **Map to Out** (blue markers). A redefinition is a secondary layout (field types) of the same memory area. The redefinition contains more specific information (`ANNUAL-SALARY`, `VACATION`, etc.) than character buffer `DETAIL`. This is the reason why it is more useful in your interface.
- Use the context menu of `FILLER_1` in the IDL interface pane and rename `FILLER_1` to `DETAILS` (green markers) for a readable IS service field name later.
- The `DETAILS` function in the COBOL server does not make use of the `EMPLOYEES` field. Therefore you can leave it out of the interface: use the context menu of the `EMPLOYEES` field in the **COBOL Interface** pane and choose **Suppress** (red markers).


3 Shape `EMPLOYEE` to `getListOfEmployees` Function:



The COBOL Mapping Editor allows you to define multiple IDL interfaces to the same COBOL server:

- You create the additional IDL interface using the toolbar button **Copy current IDL interface**

 (red markers). A new tab `getDetailsOfEmployee1` is created. The previously extracted `getDetailsOfEmployee` interface still exists in the first tab. Once you reactivate the first tab, you will see the interface of `getDetailsOfEmployee` again.

- Use the **Rename** button  from the toolbar of the COBOL Mapping Editor and change the IDL interface name to `getListOfEmployees` (blue markers). This name is used as the IS service name later.

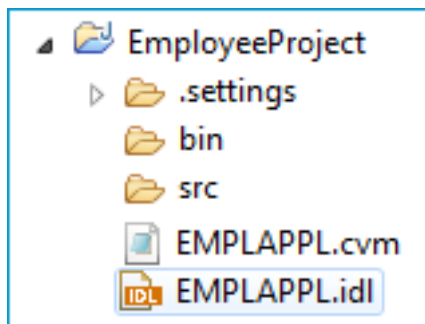
- For the `LIST` function you need the `OPERATION-LIST` value in the `OPERATION` field: Mark the `OPERATION` field in the **COBOL Interface** pane and press the **Set Constant** button. In the pop-up window, select `OPERATION-LIST` (similar to Step 1. above *Shape `EMPLOYEE` to `getDetailsOfEmployee`*; red markers).

- Use the context menu of the `EMPLOYEES` field in the **COBOL Interface** pane, `EMPLOYEES OCCURS 1 TO 99` and choose **Map to Out** (Similar to Step 2. above *Continue shaping, making `getDetailsOfEmployee` easier to use*; blue markers).

- Because the `EMPLOYEE` and `EMPLOYEE-DETAIL` fields are not used in the COBOL server `LIST` function, you leave them out in the IDL interface: use the context menu of `EMPLOYEE-DETAIL` field in the **COBOL Interface** pane and choose **Suppress** (similar to Step 2. above *Continue shaping, making `getDetailsOfEmployee` easier to use*; red markers). Do the same for the `EMPLOYEE-ID` field.

- 4 Press **Finish** to retrieve the extraction result in the form of a Software AG IDL file. At the same time, a server mapping file for COBOL (Designer file with extension `.cvm`) is created.

See *Server Mapping Files for COBOL* in the Designer documentation. The *Software AG IDL File* in the IDL Editor documentation describes the interfaces from the client point of view, while the server mapping file contains the mapping to the real COBOL server. Both of these files must be kept together and in sync, otherwise a call to the COBOL server may fail.



To summarize: We created two IDL interfaces. In the IDL file, these resulted in two IDL programs: `getListOfEmployees` and `getDetailsOfEmployee`. Both IDL programs were given readable names (Steps 1 and 3). Meaningful fields were kept, while superfluous fields were suppressed (Steps 1 and 3). The program `getDetailsOfEmployee` contains the redefined fields of parameter `DETAILS` (see below) mapped during extraction (Step 1).



Note: At runtime, the RPC client generated with the extracted interface will send data for the redesigned interfaces, while your COBOL server still expects `EMPLOYEE` data. The EntireX runtime transforms the incoming data stream from the RPC client, using the server mapping file.

```

program 'getDetailsOfEmployee' is
  define data parameter
    1 ID (AV10) In /* Original Name:EMPLOYEE-ID
    1 EMPLOYEE Out /* Original Name:EMPLOYEE-DETAILS
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
    2 DETAILS /* Original Name:FILLER_1
    3 ANNUAL-SALARY (P9)
    3 VACATION (N2)
    3 LANGUAGE (AV3)
  end-define
  program 'getListOfEmployees' is
    1 EMPLOYEES (/V99) Out
    2 IDENT (AV10)
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
  end-define

```

Comparison Chart: Fast-track versus User-defined Mapping

The following table highlights the differences when shaping an interface with a *user-defined mapping* compared with a *fast-path extraction*.

	Interface Shaping with User-defined Mapping	Fast-path Extraction
General	User-defined interface with dedicated mapping. For our example, the interfaces are small and tidy and more self-explanatory with long, readable interface names. They are easier to use with the hidden <code>OPERATION</code> field and the suppressed COBOL fields, which are not needed.	Automatic, quick result extraction with COBOL-like interfaces on IS. For our example, the IS interface <code>EMPLOYEE</code> matches exactly 1:1, field by field.
IS service(s)	Multiple small and handy IS services; each <code>OPERATION</code> code is mapped to a separate IS service.	One big IS service.
IS service name	Readable long name.	Short Subprogram name; up to 8 characters.
IS fields	Usage of Suppress and Set Constant reduces the message length. This keeps focus on relevant data items and keeps the client's interface clean. It may also improve performance.	The IS fields and COBOL server parameters match 1:1. As COBOL layout descriptions are sometimes used for many different purposes, irrelevant data items appear and clutter up the IS interface.
<code>OPERATION</code> parameter	Suppressed: the <code>OPERATION</code> parameter does not exist in the IS service as an IS field.	The <code>OPERATION</code> parameter exists in the IS service as an IS field and needs to be filled-in by the client endpoint.
<code>OPERATION</code> code	The <code>OPERATION</code> code is provided internally by EntireX runtime in the <code>OPERATION</code> field.	The <code>OPERATION</code> code needs to be specified in the IS service in the <code>OPERATION</code> field by the client endpoint.
<code>REDEFINE</code> parameters	Either the parameter that is redefined or one of its redefinitions is available as an IS field.	Only the parameter that is redefined is available as an IS field. Redefinitions of the parameter are not available as IS fields.

A user-defined mapping enables you also to define alternative mappings for COBOL data items (`REDEFINES`, tables etc.). These enable scenarios with COBOL servers where data exchange is not fully described by the `LINKAGE SECTION` COBOL data items. For more information refer to the IDL Extractor for COBOL documentation:

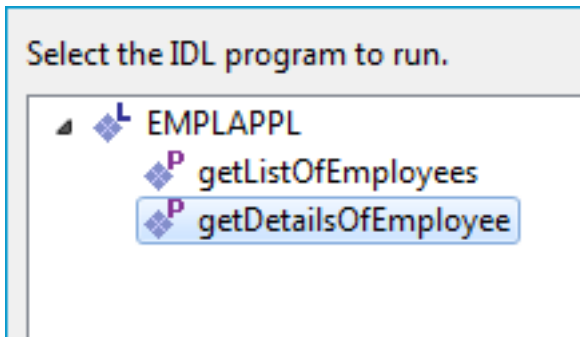
- *Mapping Editor IDL Interface Mapping Functions* if the check-box **Input Message same as Output Message** was checked.
- *Mapping Editor IDL Interface Mapping Functions* if the check-box **Input Message same as Output Message** was cleared.
- *User-defined Mapping*

Testing the Extraction Results

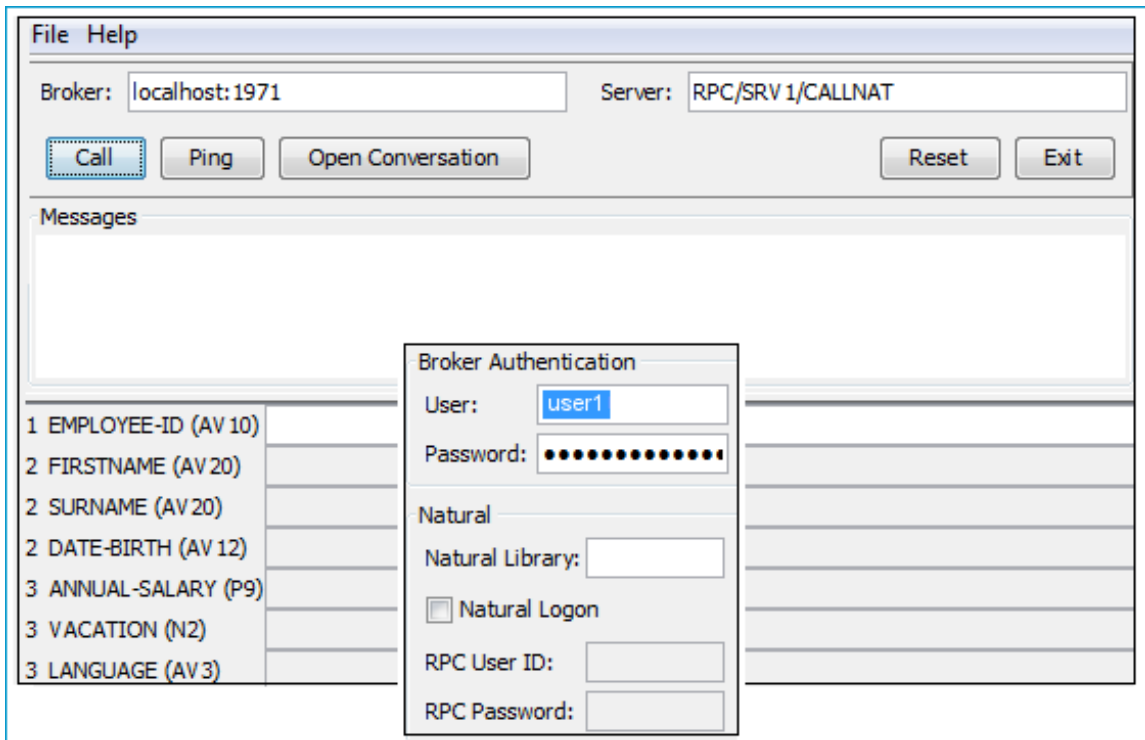
The following pictures use the extraction results described under [Extracting a COBOL Server - Modern Method with User-defined Mapping](#).

> To test the extraction results (optional)

- 1 You can test the results of the extraction operation and the REST server back end, using the EntireX IDL Tester. From the context menu of the IDL file in the Designer, choose **Software AG IDL Tester**.



- 2 Select `getDetailsOfEmployee`.



Note that the broker and server parameters contain the explicit route to call the server program, and you can optionally ping the connection from this client. With the **File > Options** dialog, you can set:

- **User** and **Password** for *broker* authentication
- **RPC User ID** and **RPC Password** for *server* authentication

See *EntireX IDL Tester* in the Designer documentation.

- 3 Check the Integration Server log, the EntireX Adapter log or the RPC logs. Applies to all connection methods.

Task 2: Generate the Connection and Adapter Services in Integration Server

This section describes your first steps to create a new Integration Server connection. This is described in more detail under *Using the Integration Server Wrapper*, for example working with existing Integration Server connections. This section covers the following topics:

- [Step 1: Start the Integration Server Wrapper Wizard](#)
- [Step 2: Create a New Integration Server Connection](#)
- [Step 3: Select the Connection Type](#)
- [Step 4: Define Adapter Services for a CICS IPIC Connection](#)
- [Step 5: Create or Update a REST Resource](#)
- [Step 6: Edit the REST Resource](#)

Step 1: Start the Integration Server Wrapper Wizard

➤ To start the Integration Server Wrapper wizard

- 1 In the context menu of a Software AG IDL file, choose **Integration Server > Generate web-Methods IS Connection**.

This starts the wizard with a list of existing Integration Server Wrapper connections.



Note: If the selected IDL file is not valid because of a syntax error, an error dialog comes up and the wizard does not start.

- 2 Continue with *Step 2: Create a New Integration Server Connection*.

Step 2: Create a New Integration Server Connection

Define New Integration Server Connection

Add host:port, user and password for a new Integration Server

Server:

User:

Password:

☐
Use secure connection

➤ To create a new Integration Server connection

- 1 Define the new Integration Server connection on the wizard page.



Notes:

1. The only required field is **Server**. Enter the hostname of the Integration Server including an optional port number. If no port number is specified, port number defaults to "5555". The **Integration Server Authentication** can be passed with the **User** and **Password** fields.
 2. Optional settings are for secure connections. The **Truststore for HTTPS** contains all signed certificates and must be a valid truststore.
 3. The check box **Verify host name** checks that the hostname is entered in the stored certificate.
 4. When the Integration Server has **Client Authentication** enabled, you can specify your **Keystore** file and keystore **Password**.
 5. For managing Integration Server connections, see [Preferences](#).
- 2 Choose **Next** and continue with *Step 3: Select the Connection Type*.

Step 3: Select the Connection Type

☒ Create a new EntireX Adapter connection of type **CICS IPIC Connection**

☐ Update Adapter Services or Listeners to an existing Integration Server connection:

List of Connections

Connection Name	Package Name	Connection Type	Enabled
Employees:EmployeesConnection	Employees	EntireX RPC Connection	Yes

Total: 1

☒ Map Software AG IDL data types to String

☒ Create or Update REST resource

> To create a new connection

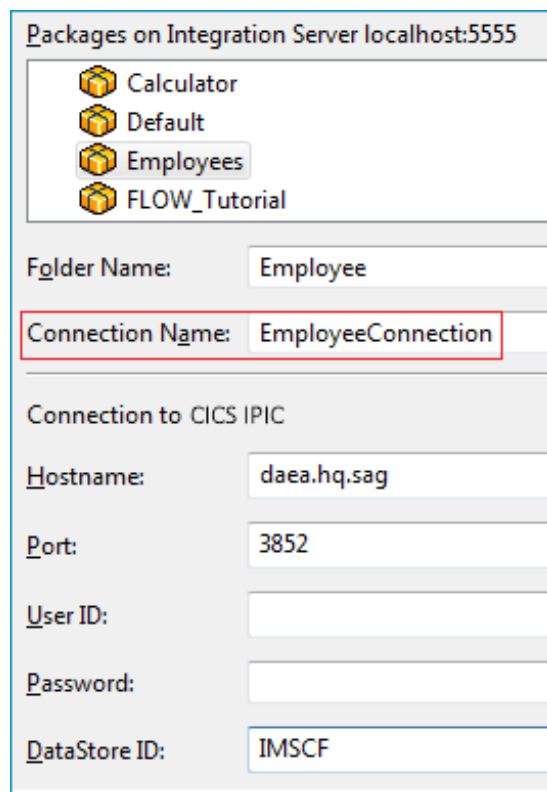
- 1 Select a connection type from the drop down list. Connection types are described under *EntireX Adapter Connections* in the EntireX Adapter documentation and *Introduction to the Integration Server Wrapper*.



Note: The list of connection types is filtered: connection types that require a license are only shown if a corresponding license file is available. Reliable RPC connections are only shown if all IDL programs contain only IN parameters. Also, if a server mapping file is available, only those connection types that support the interface type specified in the server mapping file are shown.

- 2 Check the box **Create or Update REST resource**.
- 3 Click **Next** and continue with *Step 4: Define Adapter Services for a CICS IPIC Connection*.

Step 4: Define Adapter Services for a CICS IPIC Connection



Packages on Integration Server localhost:5555

- Calculator
- Default
- Employees
- FLOW_Tutorial

Folder Name: Employee

Connection Name: EmployeeConnection

Connection to CICS IPIC

Hostname: daea.hq.sag

Port: 3852

User ID:

Password:

DataStore ID: IMSCF

➤ To create a connection and related adapter services

- 1 Select a package for the created objects.
- 2 Define a folder name. If the folder does not exist, it will be created.
- 3 Define a connection name.
- 4 Define the parameters of the connection type. For details, see *Connection Parameters for CICS IPIC Connections* in the EntireX Adapter documentation.

As a result, the folder will contain the connection and the adapter services (one for each IDL program). The name of a service is the same as the respective IDL program.

The default settings for the adapter services are:

- the **Default** package; if not available, the first package
- the IDL library name for the **Folder Name**
- the IDL library name with the suffix "Connection" for the **Connection Name**



Note: 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.

Step 5: Create or Update a REST Resource

> To create or update a REST resource

Make sure checkbox **Create or Update REST resource** is checked.

- 1 Select a package.

The screenshot shows a dialog box titled "Packages on Integration Server". It contains a list of packages: Default, Employees (selected), WmAdmin, and WmART. Below the list, there are two text input fields: "Folder Name:" with the value "Employees" and "REST Resource Name:" with the value "employees". At the bottom, there are two checked checkboxes: "Update existing Resource" and "Create or Update simplified REST services for this REST resource".

- 2 Define a folder name. If the folder does not exist, it will be created.
- 3 Define the REST Resource name.
- 4 Check **Update existing Resource** if you want to add additional services to an existing resource.
- 5 Check **Create or Update simplified REST services for this REST resource** if you want to generate additional services with a simplified REST API. This allows you to use a simple HTTP-GET method without a payload.

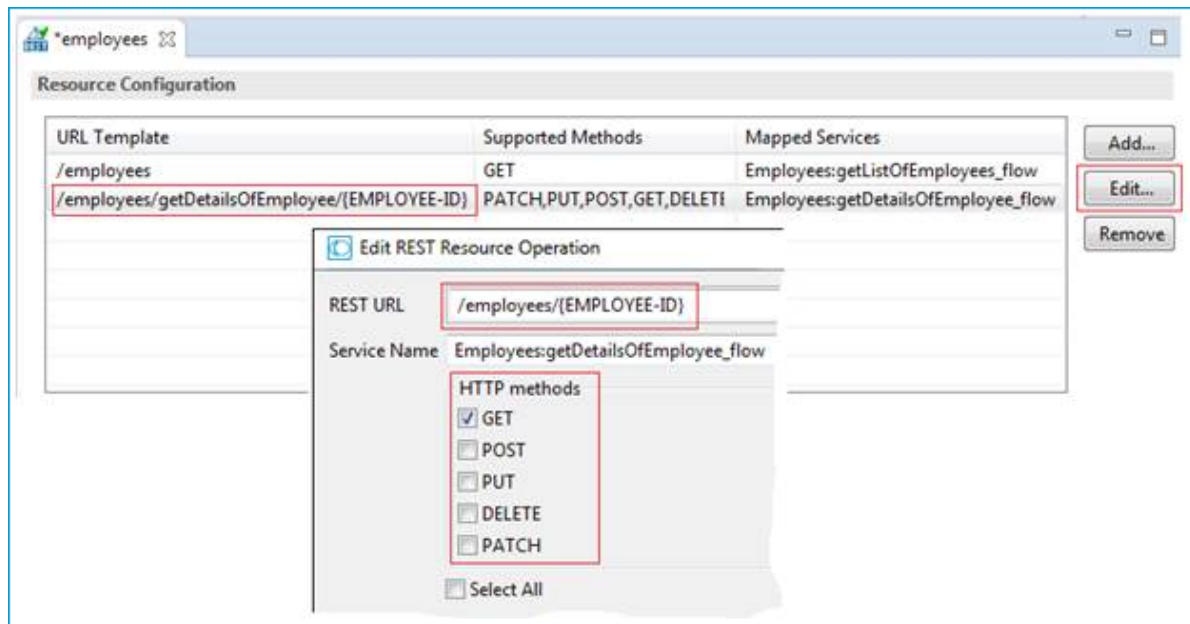


Note: This applies only to APIs where input signature parameters do not contain arrays.

Step 6: Edit the REST Resource

> To edit the REST resource

- 1 Switch to the **Service Development** perspective of Software AG Designer.



- 2 Refresh the package where you generated the REST Resource in the previous step.
- 3 Open the REST Resource.
- 4 Edit the REST URLs and HTTP Methods to match the functionality of your application.

For more information see:

- *REST Developer's Guide* in the webMethods Integration Server documentation for information on using the generated REST resources
- *REST Developer's Guide* in the webMethods Integration Server documentation

Task 3: Execute the Call from the REST Client to COBOL

From the **Service Development** perspective, refresh the package where the connection service was written, select the adapter service and use the service test to **Run Service**. This invokes the adapter service through the connector service.

The screenshot shows the Service Development console with the 'Employees' package expanded. The 'getListOfEmployees' service is selected, and the 'Run As' dropdown is set to '2 Run Service'. Below the service list, the 'outRec' table displays the following data:

Name	Value
outRec	
EMPLOYEES	
EMPLOYEES[0]	
IDENT	11100102
FIRSTNAME	EDGAR
SURNAME	SCHINDLER
DATE-BIRTH	Dec 4, 1962
EMPLOYEES[1]	
EMPLOYEES[2]	
EMPLOYEES[3]	
EMPLOYEES[4]	
errorCode	00000000
errorFlag	false

In case of error or unexpected results:

- Check the Integration Server log or the EntireX Adapter log.

9

Calling COBOL DFHCOMMAREA (Minimal Footprint Using CICS Socket Listener) from REST

■ Introduction	130
■ What do I need to Install for this Scenario?	132
■ Task 1: Extract the Interface of a COBOL Server	133
■ Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server	146
■ Task 3: Execute the Call from the REST Client to COBOL	152

This scenario uses the tools IDL Extractor for COBOL and Integration Server Wrapper of the Designer.

Introduction



- ❶ Extract the interface of the COBOL server program. See *Using the IDL Extractor for COBOL - Overview*.
- ❷ Generate REST resources, connection and adapter services in Integration Server.
- ❸ Execute the call from the REST client to the COBOL server program.

This scenario makes the following important assumptions:

For Task 1:

- You have a working COBOL DFHCOMMAREA server. For illustration and examples on such a server, see *CICS with DFHCOMMAREA Calling Convention* under *Introduction to the IDL Extractor for COBOL*.
- You have access to the related COBOL sources and copybooks as files on your local machine. The minimum requirement is the `DATA DIVISION` of the interface.



For Tasks 2 and 3:

- You have a REST client.
- To call the COBOL server program at runtime using the *EntireX CICS Socket Listener* connection method, you need to install the CICS Socket Listener within your CICS region: see *Preparing for CICS Socket Listener*. For configuration, see *Connection Parameters for CICS Socket Listener Connections*.



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.
- The CICS Socket Listener, which is installed
 - together with the RPC Server for CICS, see *Installing the RPC Server for CICS*, or
 - separately, see *EntireX CICS® Socket Listener* in the z/OS Installation documentation

Task 1: Extract the Interface of a COBOL Server

- [Introduction](#)
- [Sample COBOL Server used in the Examples](#)
- [Extracting a COBOL Server - Fast-track Method](#)
- [Extracting a COBOL Server - Modern Method with User-defined Mapping](#)
- [Comparison Chart: Fast-track versus User-defined Mapping](#)
- [Testing the Extraction Results](#)

Introduction

Follow the instructions for extracting COBOL, see *Using the IDL Extractor for COBOL - Overview* and choose *Scenario I: Create New IDL and Server Mapping Files* if this is your first extraction. Learn how to invoke the IDL Extractor for COBOL, create a COBOL Extractor Environment and about the possibilities to select the COBOL source.

The extraction process creates the following EntireX metafiles:

- IDL file. A Software AG IDL file contains definitions of the interface between client and server. See *Software AG IDL File* in the IDL Editor documentation in the IDL Editor documentation.
- Server mapping file (optional). The mapping file is a Designer file with extension .cvm that contains COBOL-specific mapping information. See *Server Mapping Files for COBOL* in the Designer documentation.

Sample COBOL Server used in the Examples

The following COBOL server is used to illustrate the features of the IDL Extractor for COBOL. Imagine an existing COBOL server called `EMPLOYEE`. It implements the access logic for a `LIST` and `DETAILS` function to a database view `EMPLOYEE`:

```

IDENTIFICATION DIVISION.
    PROGRAM-ID.      EMPLOYEE.
*****
*   Program Name    = EMPLOYEE
*   String Literal  = QUOTE
*****
01 DFHCOMMAREA.
    02 OPERATION                                PIC X(1).
        88 OPERATION-LIST                      VALUE 'L'.
        88 OPERATION-DETAILS                   VALUE 'D'.
    02 EMPLOYEE-ID                             PIC X(10).
    02 EMPLOYEE-DETAILS.
        03 FIRSTNAME                          PIC X(20).
        03 SURNAME                            PIC X(20).
        03 DATE-BIRTH                         PIC X(12).
        03 DETAILS                            PIC X(100).
        03 FILLER REDEFINES DETAILS.
            04 ANNUAL-SALARY                    PACKED-DECIMAL PIC S9(9).
            04 VACATION                         PIC S9(2).
            04 LANGUAGE                         PIC X(3).
            04 FILLER                           PIC X(90).
    02 EMPLOYEES-COUNT                         PIC 9(8) BINARY.
    02 EMPLOYEES OCCURS 1 TO 99 DEPENDING ON EMPLOYEES-COUNT.
        03 IDENT                              PIC X(10).
        03 FIRSTNAME                          PIC X(20).
        03 SURNAME                            PIC X(20).
        03 DATE-BIRTH                         PIC X(12).

```

Two approaches for extracting the COBOL server are described below:

- **Fast-track**

Learn how EntireX helps you to connect the COBOL Server with quick results.

- **User-defined Mapping**

The following features are among those available with a user-defined mapping:

- **Shape the interface to your COBOL server**

You can minimize the interface by suppressing COBOL server fields or by providing constant values. Renaming interfaces allows you to specify a readable long name. These two features together - constants and renaming of interfaces - are most powerful when multiple interfaces are implemented in a single COBOL server. Each function of the COBOL server triggered by an operation code or function code can be modelled as a separate REST resource method/operation.

■ Select alternative mappings

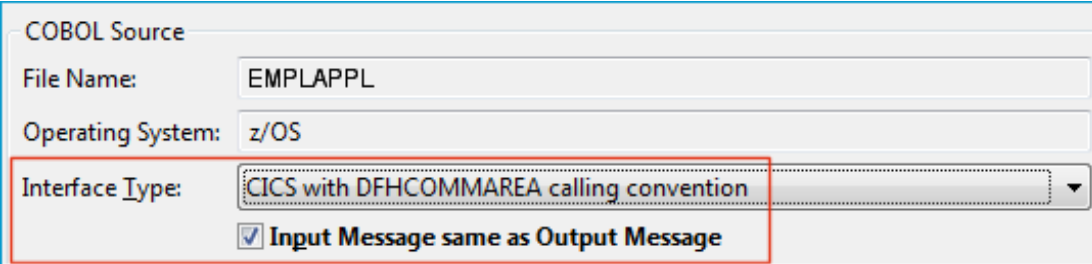
You can select alternative mappings for COBOL data items (REDEFINES, tables etc.) to enable scenarios with COBOL servers where data exchange is not fully described by the LINKAGE SECTION COBOL data items.

These approaches are described under [Fast-track Method](#) and [Modern Method](#) below.

Extracting a COBOL Server - Fast-track Method

➤ To extract the COBOL Server

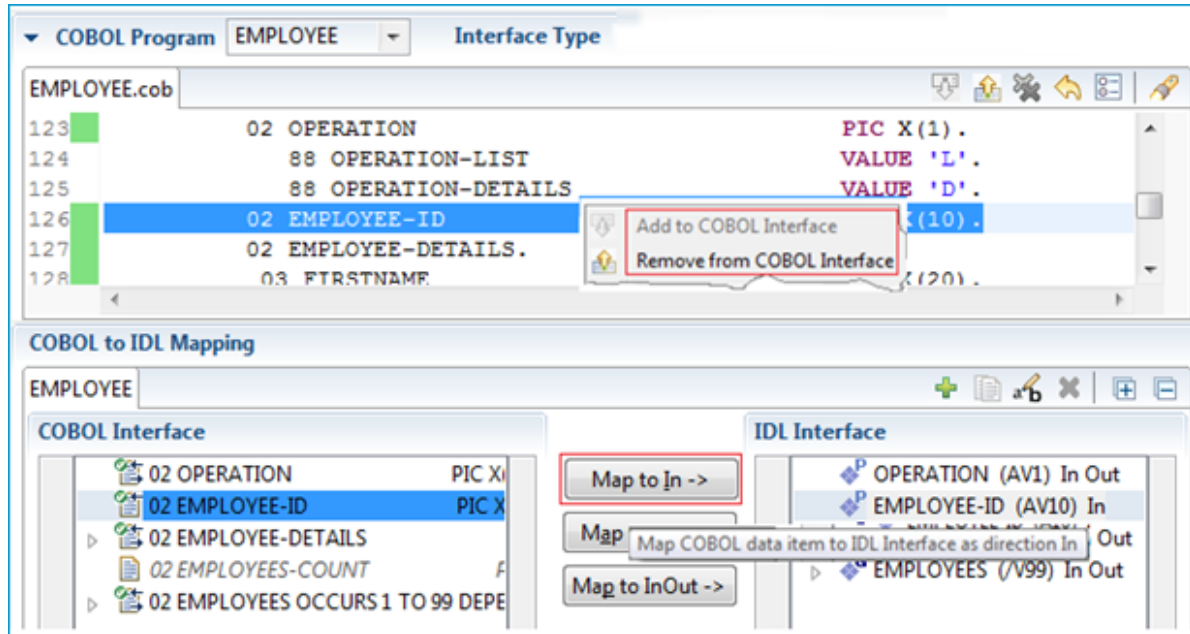
- 1 Switch to the **EntireX** perspective.
- 2 Call the IDL Extractor for COBOL.
- 3 Set the correct interface type as described under *Step 4: Define the Extraction Settings and Start Extraction* under *Scenario I: Create New IDL and Server Mapping Files* in the IDL Extractor for COBOL documentation.



- 4 Set the correct value for checkbox **Input Message same as Output Message**.
 - Check the checkbox if the COBOL data structure of the CICS input message is the same as the structure of the CICS output message, that is, there is one COBOL layout for both directions. See the following COBOL server examples in the IDL Extractor for COBOL documentation:
 - *Example 1: Redefines*
 - *Example 2: Buffer Technique*
 - *Example 3: COBOL SET ADDRESS Statements*
 - Clear the checkbox if the COBOL data structure of the CICS input message is different to the structure of the CICS output message, that is, the output overlays the input, which means there is a second COBOL layout for the reply. See the following COBOL server examples in the IDL Extractor for COBOL documentation:
 - *Example 1: Redefines*
 - *Example 2: Buffer Technique*
 - *Example 3: COBOL SET ADDRESS Statements*

For our example, check the check box, because the COBOL layout for input is the same as the COBOL layout for output.

- 5 Press **Next** to enter the COBOL Mapping Editor.



The purpose of the COBOL Mapping Editor is to map COBOL data items to IDL (Interface Definition Language) parameters. It consists of three main areas:

- The upper window pane is the COBOL program source.
- The window pane on the left-hand side is the COBOL interface, consisting of COBOL data items representing the parameters in your COBOL server program.
- The window pane on the right-hand side is the IDL interface consisting of IDL parameters.

What you select depends on check-box **Input Message same as Output Message**:

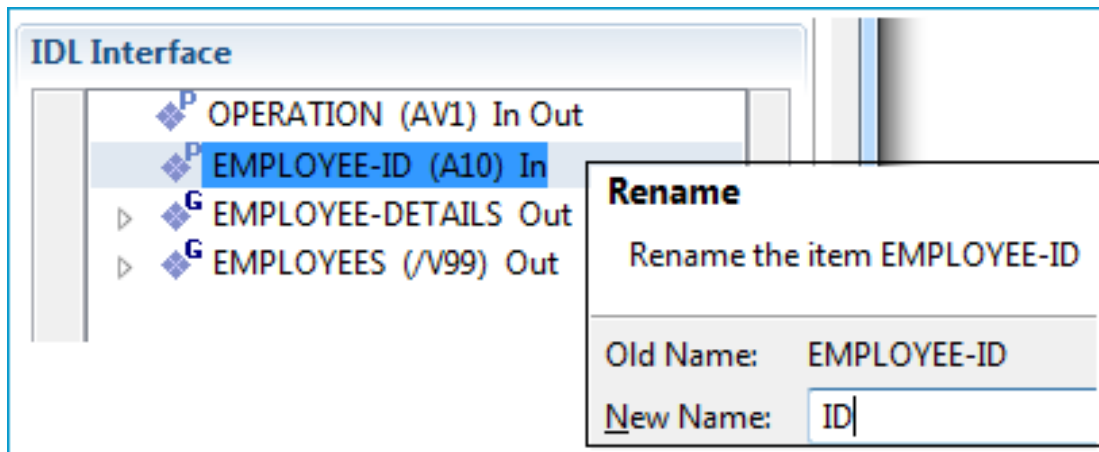
- If there is a single COBOL data structure of the CICS message for input and output - check box **Input Message same as Output Message** is checked - you select a single COBOL data structure in the COBOL Mapping Editor. This structure is used for COBOL input as well as for COBOL output. For more information in this case see *Mapping Editor User Interface* in the IDL Extractor for COBOL documentation.
- If there is a second COBOL data structure for the CICS output message - check box **Input Message same as Output Message** is cleared - you will select two COBOL data structures in the COBOL Mapping Editor. One is used for COBOL input and the other is for COBOL output. For more information in this case see *Mapping Editor User Interface* in the IDL Extractor for COBOL documentation.

COBOL extraction is a two-step process:

- First, select the COBOL parameters using the **Add to** and **Remove from COBOL Interface** from the context menu on the COBOL data items in the upper window pane (COBOL source); in many situations the COBOL interface is detected by the Mapping Editor directly and you can omit this step.
- Then map the COBOL parameters to IDL directions using the **Map to In/Out/InOut** buttons. COBOL parameters do not have directions, they are always INOUT. This step can be left out for a quick test in non-production scenarios; for real production it is recommended to set IDL directions to reduce data transfer. You also have the option to assign a meaningful name to an IDL parameter if the original COBOL data item is not self-explanatory.

For our example:

- Set **Map to In** for COBOL data item `EMPLOYEE-ID` as IDL direction (see screenshot in previous step). For `EMPLOYEES-DETAILS` and `EMPLOYEES` perform the similar **Map to Out**.
- Use the function **Rename** from the context menu on the IDL parameters in the **IDL Interface** pane to rename COBOL parameter `EMPLOYEES-ID` to IDL parameter `ID`. Do the same for `EMPLOYEE-DETAILS`: Rename to `EMPLOYEE`.



- 6 Press **Finish**. The default extraction settings ensure you get useful results. The outcome is a Software AG IDL file (interface definition language), a Designer file with extension `.idl`:

```
library 'EMPLAPPL' is
program 'EMPLOYEE' is
define data parameter
  1 OPERATION (AV1) In Out
  1 ID (AV10) In /* Original Name:EMPLOYEE-ID
  1 EMPLOYEE Out /* Original Name:EMPLOYEE-DETAILS
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
    2 DETAILS (AV100)
  1 EMPLOYEES (/V99) Out
    2 IDENT (AV10)
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
end-define
```

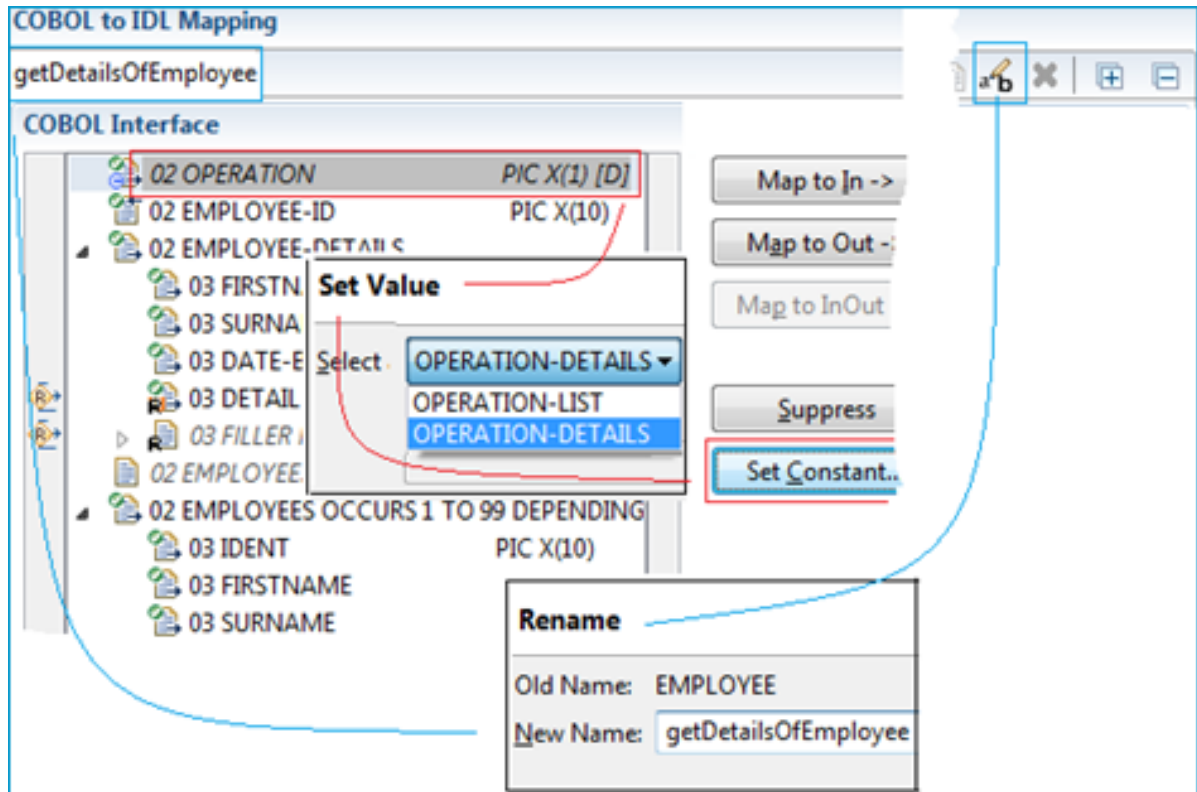
If required, a server mapping file with the same name as the IDL file but the extension .cvm may be generated too, for technical reasons. This file must always be stored together with the concomitant IDL file. For more information, see *Server Mapping Files for COBOL* in the Designer documentation. By default, the interface name `EMPLOYEE` is taken from the COBOL server program name.


Extracting a COBOL Server - Modern Method with User-defined Mapping

EntireX allows you to extract all implemented interfaces of the COBOL server separately. Instead of a large `EMPLOYEE` interface, separate interfaces `getListOfEmployees` and `getDetailsOfEmployee` are extracted. Each interface contains required parameters; obsolete parameters for an interface are suppressed, improving its usability.

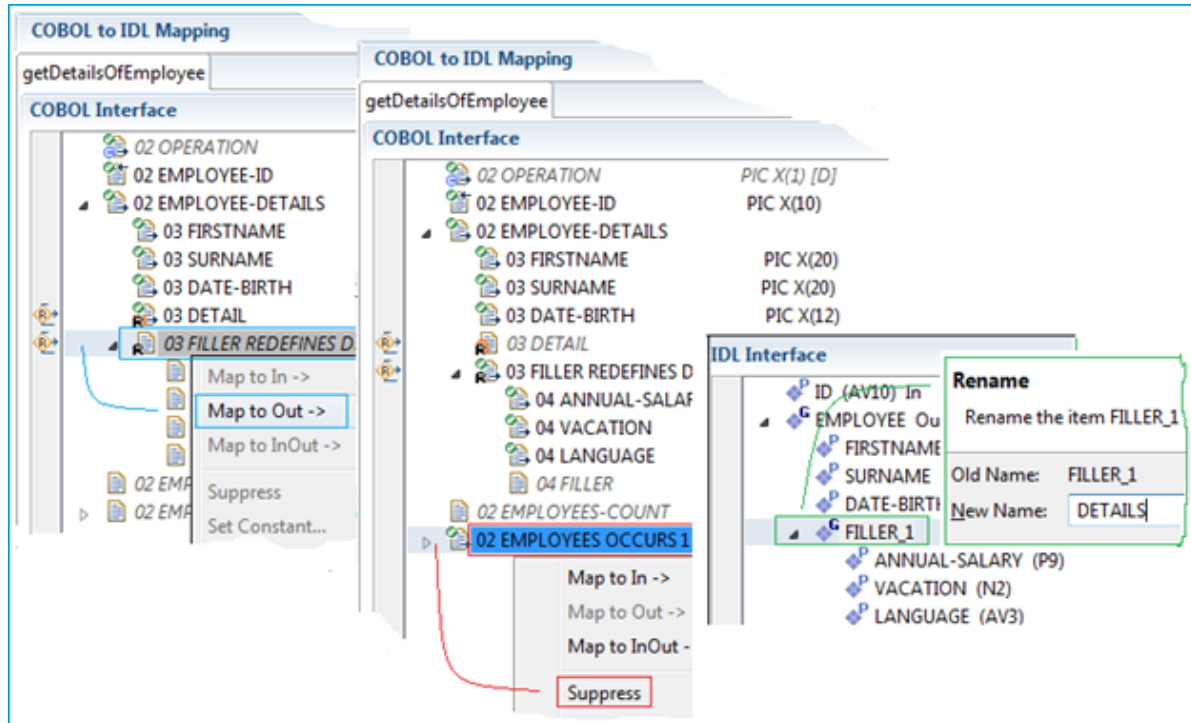
➤ To extract a COBOL server with a user-defined mapping

- 1 Shape `EMPLOYEE` to `getDetailsOfEmployee`:



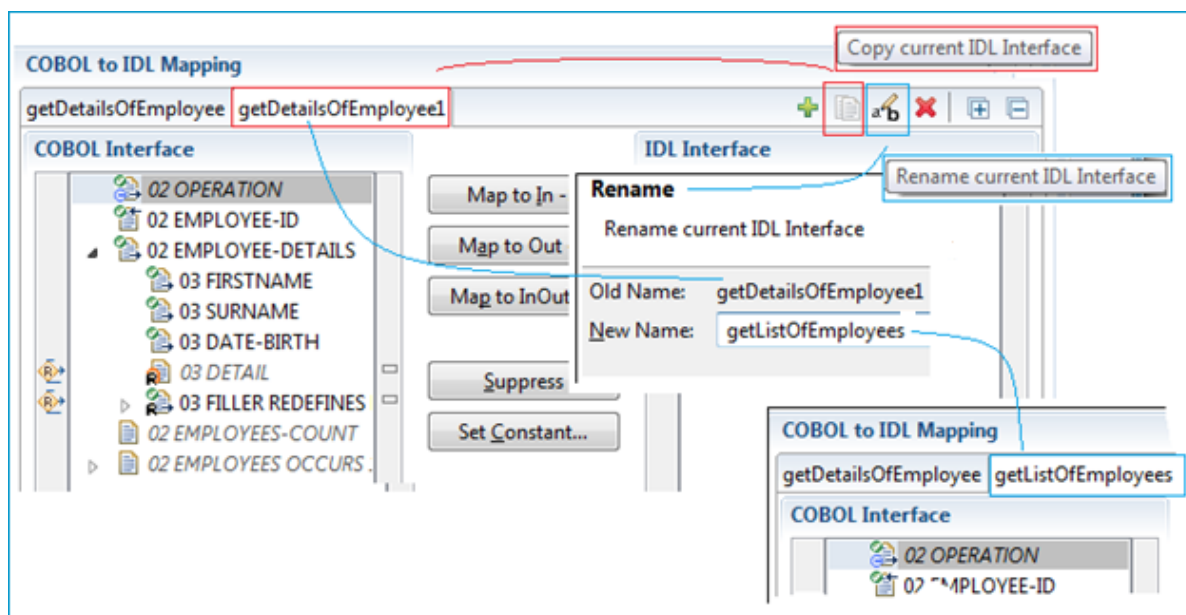
- Mark the `OPERATION` field in the **COBOL Interface** pane and press the **Set Constant** button. In the pop-up window, select `OPERATION-DETAILS` (red markers). The `OPERATION-DETAILS` value originates from COBOL level-88 data item (blue markers). If there is no COBOL level-88 enumeration type in the COBOL program defined, you can enter the function code manually in the pop-up window.
- By default the IDL interface gets the same name as your COBOL server: `EMPLOYEE`. Use the **Rename** button  from the toolbar of the COBOL Mapping Editor and change the IDL interface name to `getDetailsOfEmployee` (blue markers). This name is used as the IS service name later.

2 Continue shaping, making `getDetailsOfEmployee` easier to use:





- Use the context menu of the redefinition of the `DETAIL` field in the **COBOL Interface** pane, `FILLER REDEFINES DETAIL` and choose **Map to Out** (blue markers). A redefinition is a secondary layout (field types) of the same memory area. The redefinition contains more specific information (`ANNUAL-SALARY`, `VACATION`, etc.) than character buffer `DETAIL`. This is the reason why it is more useful in your interface.
- Use the context menu of `FILLER_1` in the IDL interface pane and rename `FILLER_1` to `DETAILS` (green markers) for a readable IS service field name later.
- The `DETAILS` function in the COBOL server does not make use of the `EMPLOYEES` field. Therefore you can leave it out of the interface: use the context menu of the `EMPLOYEES` field in the **COBOL Interface** pane and choose **Suppress** (red markers).

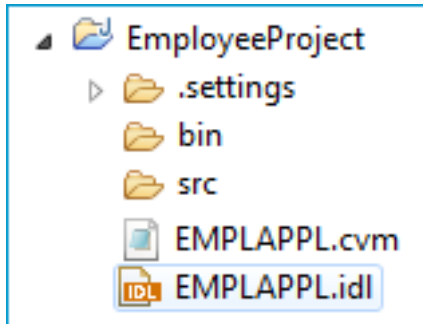
3 Shape `EMPLOYEE` to `getListOfEmployees` Function:



The COBOL Mapping Editor allows you to define multiple IDL interfaces to the same COBOL server:

-  . (red markers). A new tab `getDetailsOfEmployee1` is created. The previously extracted `getDetailsOfEmployee` interface still exists in the first tab. Once you reactivate the first tab, you will see the interface of `getDetailsOfEmployee` again.
- Use the **Rename** button  from the toolbar of the COBOL Mapping Editor and change the IDL interface name to **`getListOfEmployees`** (blue markers). This name is used as the IS service name later.
- For the `LIST` function you need the `OPERATION-LIST` value in the `OPERATION` field: Mark the `OPERATION` field in the **COBOL Interface** pane and press the **Set Constant** button. In the pop-up window, select `OPERATION-LIST` (similar to Step 1. above *Shape `EMPLOYEE` to `getDetailsOfEmployee`*; red markers).
- Use the context menu of the `EMPLOYEES` field in the **COBOL Interface** pane, `EMPLOYEES OCCURS 1 TO 99` and choose **Map to Out** (Similar to Step 2. above *Continue shaping, making `getDetailsOfEmployee` easier to use*; blue markers).
- Because the `EMPLOYEE` and `EMPLOYEE-DETAIL` fields are not used in the COBOL server `LIST` function, you leave them out in the IDL interface: use the context menu of `EMPLOYEE-DETAIL` field in the **COBOL Interface** pane and choose **Suppress** (similar to Step 2. above *Continue shaping, making `getDetailsOfEmployee` easier to use*; red markers). Do the same for the `EMPLOYEE-ID` field.

See *Server Mapping Files for COBOL* in the Designer documentation. The *Software AG IDL File* in the IDL Editor documentation describes the interfaces from the client point of view, while the server mapping file contains the mapping to the real COBOL server. Both of these files must be kept together and in sync, otherwise a call to the COBOL server may fail.



To summarize: We created two IDL interfaces. In the IDL file, these resulted in two IDL programs: `getListOfEmployees` and `getDetailsOfEmployee`. Both IDL programs were given readable names (Steps 1 and 3). Meaningful fields were kept, while superfluous fields were suppressed (Steps 1 and 3). The program `getDetailsOfEmployee` contains the redefined fields of parameter `DETAILS` (see below) mapped during extraction (Step 1).



Note: At runtime, the RPC client generated with the extracted interface will send data for the redesigned interfaces, while your COBOL server still expects `EMPLOYEE` data. The EntireX runtime transforms the incoming data stream from the RPC client, using the server mapping file.

```

program 'getDetailsOfEmployee' is
  define data parameter
    1 ID (AV10) In /* Original Name:EMPLOYEE-ID
    1 EMPLOYEE Out /* Original Name:EMPLOYEE-DETAILS
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
    2 DETAILS /* Original Name:FILLER_1
    3 ANNUAL-SALARY (P9)
    3 VACATION (N2)
    3 LANGUAGE (AV3)
  end-define
  program 'getListOfEmployees' is
    1 EMPLOYEES (/V99) Out
    2 IDENT (AV10)
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
  end-define

```

Comparison Chart: Fast-track versus User-defined Mapping

The following table highlights the differences when shaping an interface with a *user-defined mapping* compared with a *fast-path extraction*.

	Interface Shaping with User-defined Mapping	Fast-path Extraction
General	User-defined interface with dedicated mapping. For our example, the interfaces are small and tidy and more self-explanatory with long, readable interface names. They are easier to use with the hidden <code>OPERATION</code> field and the suppressed COBOL fields, which are not needed.	Automatic, quick result extraction with COBOL-like interfaces on IS. For our example, the IS interface <code>EMPLOYEE</code> matches exactly 1:1, field by field.
IS service(s)	Multiple small and handy IS services; each <code>OPERATION</code> code is mapped to a separate IS service.	One big IS service.
IS service name	Readable long name.	Short Subprogram name; up to 8 characters.
IS fields	Usage of Suppress and Set Constant reduces the message length. This keeps focus on relevant data items and keeps the client's interface clean. It may also improve performance.	The IS fields and COBOL server parameters match 1:1. As COBOL layout descriptions are sometimes used for many different purposes, irrelevant data items appear and clutter up the IS interface.
<code>OPERATION</code> parameter	Suppressed: the <code>OPERATION</code> parameter does not exist in the IS service as an IS field.	The <code>OPERATION</code> parameter exists in the IS service as an IS field and needs to be filled-in by the client endpoint.
<code>OPERATION</code> code	The <code>OPERATION</code> code is provided internally by EntireX runtime in the <code>OPERATION</code> field.	The <code>OPERATION</code> code needs to be specified in the IS service in the <code>OPERATION</code> field by the client endpoint.
<code>REDEFINE</code> parameters	Either the parameter that is redefined or one of its redefinitions is available as an IS field.	Only the parameter that is redefined is available as an IS field. Redefinitions of the parameter are not available as IS fields.

A user-defined mapping enables you also to define alternative mappings for COBOL data items (`REDEFINES`, tables etc.). These enable scenarios with COBOL servers where data exchange is not fully described by the `LINKAGE SECTION` COBOL data items. For more information refer to the IDL Extractor for COBOL documentation:

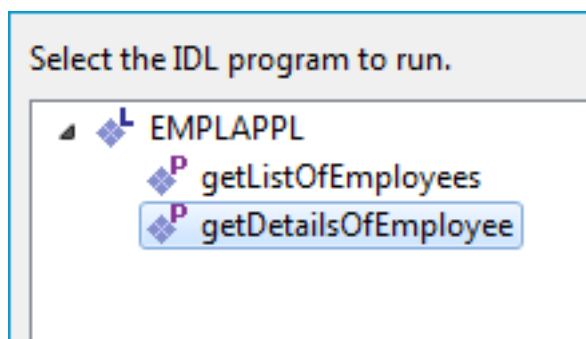
- *Mapping Editor IDL Interface Mapping Functions* if the check-box **Input Message same as Output Message** was checked.
- *Mapping Editor IDL Interface Mapping Functions* if the check-box **Input Message same as Output Message** was cleared.
- *User-defined Mapping*

Testing the Extraction Results

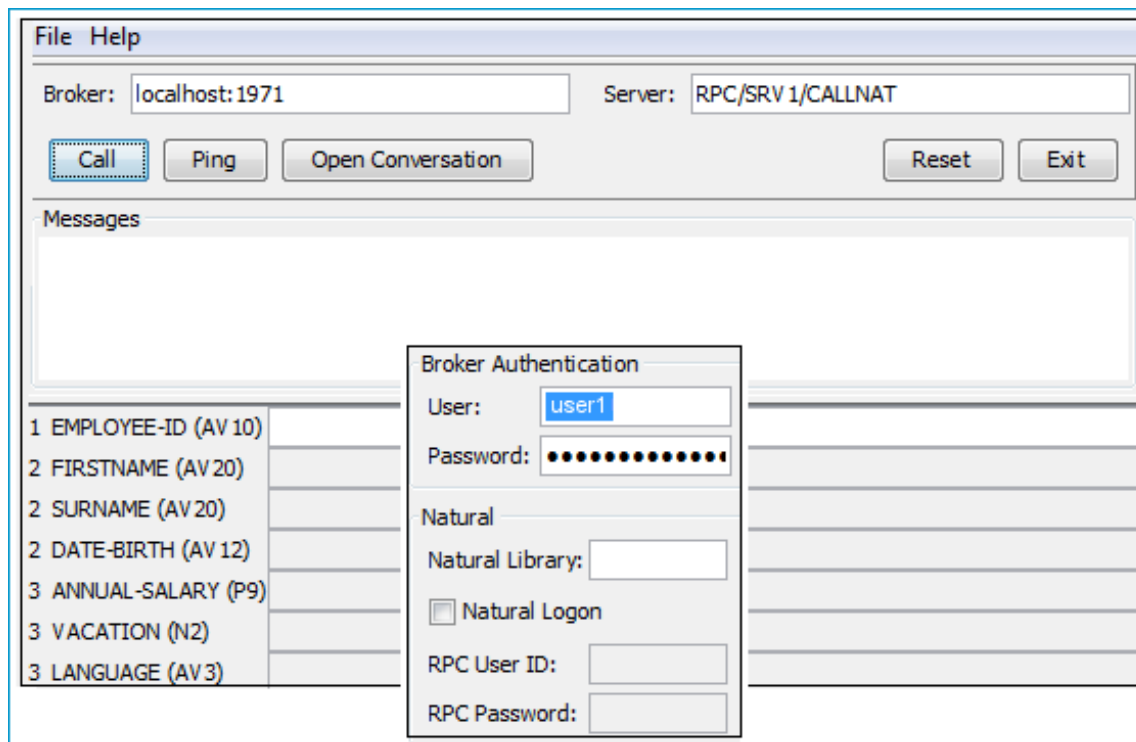
The following pictures use the extraction results described under [Extracting a COBOL Server - Modern Method with User-defined Mapping](#).

> To test the extraction results (optional)

- 1 You can test the results of the extraction operation and the REST server back end, using the EntireX IDL Tester. From the context menu of the IDL file in the Designer, choose **Software AG IDL Tester**.



- 2 Select `getDetailsOfEmployee`.



Note that the broker and server parameters contain the explicit route to call the server program, and you can optionally ping the connection from this client. With the **File > Options** dialog, you can set:

- **User** and **Password** for *broker* authentication
- **RPC User ID** and **RPC Password** for *server* authentication

See *EntireX IDL Tester* in the Designer documentation.

- 3 Check the Integration Server log, the EntireX Adapter log or the RPC logs. Applies to all connection methods.

Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server

This section describes your first steps to create a new Integration Server connection. This is described in more detail under *Using the Integration Server Wrapper*, for example working with existing Integration Server connections. This section covers the following topics:

- [Step 1: Start the Integration Server Wrapper Wizard](#)
- [Step 2: Create a New Integration Server Connection](#)
- [Step 3: Select the Connection Type](#)
- [Step 4: Define Adapter Services for a CICS Socket Listener Connection](#)
- [Step 5: Create or Update a REST Resource](#)
- [Step 6: Edit the REST Resource](#)

Step 1: Start the Integration Server Wrapper Wizard

➤ To start the Integration Server Wrapper wizard

- 1 In the context menu of a Software AG IDL file, choose **Integration Server > Generate web-Methods IS Connection**.

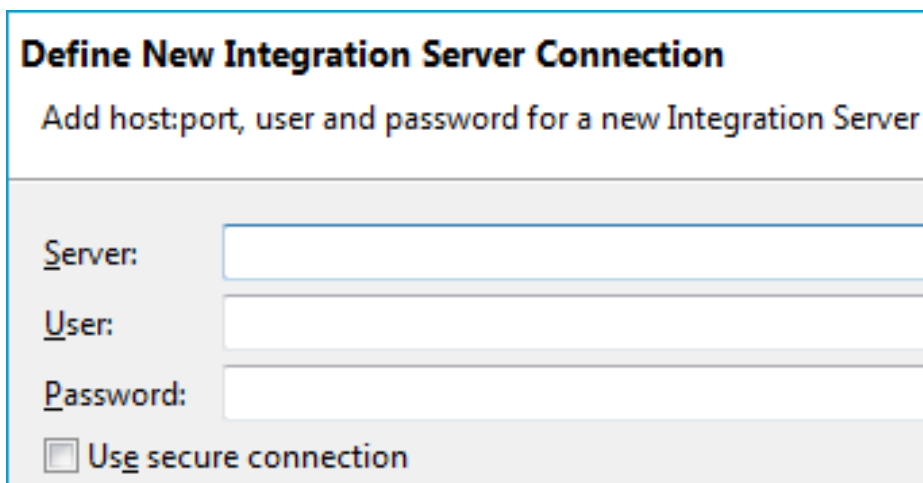
This starts the wizard with a list of existing Integration Server Wrapper connections.



Note: If the selected IDL file is not valid because of a syntax error, an error dialog comes up and the wizard does not start.

- 2 Continue with *Step 2: Create a New Integration Server Connection*.

Step 2: Create a New Integration Server Connection



The screenshot shows a web form titled "Define New Integration Server Connection". Below the title is a subtitle: "Add host:port, user and password for a new Integration Server". The form contains three input fields: "Server:", "User:", and "Password:". Below these fields is a checkbox labeled "Use secure connection".

➤ To create a new Integration Server connection

- 1 Define the new Integration Server connection on the wizard page.



Notes:

1. The only required field is **Server**. Enter the hostname of the Integration Server including an optional port number. If no port number is specified, port number defaults to "5555". The **Integration Server Authentication** can be passed with the **User** and **Password** fields.
2. Optional settings are for secure connections. The **Truststore for HTTPS** contains all signed certificates and must be a valid truststore.
3. The check box **Verify host name** checks that the hostname is entered in the stored certificate.
4. When the Integration Server has **Client Authentication** enabled, you can specify your **Keystore** file and keystore **Password**.
5. For managing Integration Server connections, see [Preferences](#).

- 2 Choose **Next** and continue with *Step 3: Select the Connection Type*.

Step 3: Select the Connection Type

☒ Create a new EntireX Adapter connection of type CICS Socket Listener Connection

☐ Update Adapter Services or Listeners to an existing Integration Server connection:

List of Connections

Connection Name	Package Name	Connection Type	Enabled
Employees:EmployeesConnection	Employees	EntireX RPC Connection	Yes

Total: 1

☒ Map Software AG IDL data types to String
☒ Create or Update REST resource

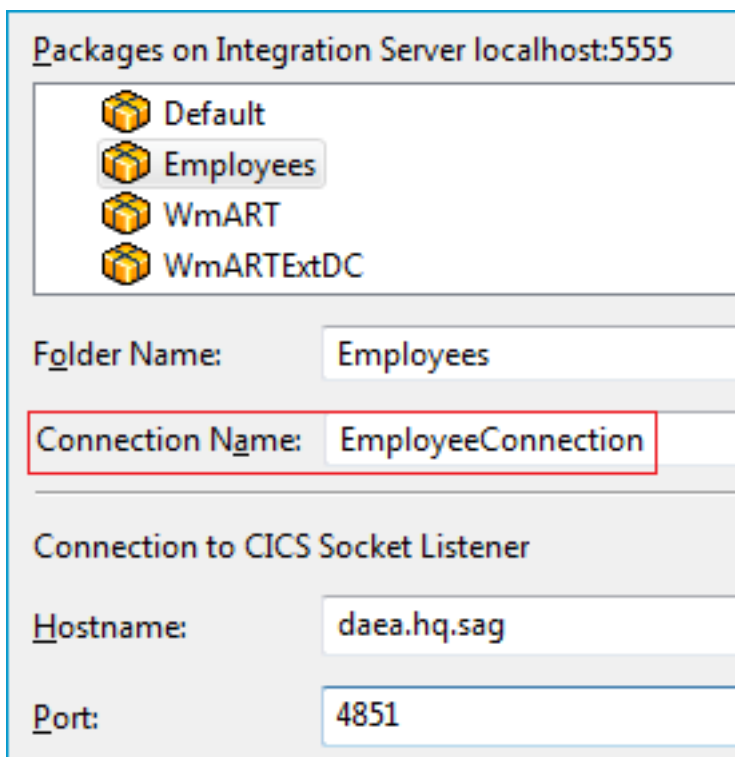
> To create a new connection

- 1 Select a connection type from the drop down list. Connection types are described under *EntireX Adapter Connections* in the EntireX Adapter documentation and *Introduction to the Integration Server Wrapper*.



Note: The list of connection types is filtered: connection types that require a license are only shown if a corresponding license file is available. Reliable RPC connections are only shown if all IDL programs contain only IN parameters. Also, if a server mapping file is available, only those connection types that support the interface type specified in the server mapping file are shown.

- 2 Check the box **Create or Update REST resource**.
- 3 Click **Next** and continue with *Step 4: Define Adapter Services for a CICS Socket Listener Connection*.

Step 4: Define Adapter Services for a CICS Socket Listener Connection


Packages on Integration Server localhost:5555

- Default
- Employees
- WmART
- WmARTEExtDC

Folder Name: Employees

Connection Name: EmployeeConnection

Connection to CICS Socket Listener

Hostname: daea.hq.sag

Port: 4851

➤ **To create a connection and related adapter services**

- 1 Select a package for the created objects.
- 2 Define a folder name. If the folder does not exist, it will be created.
- 3 Define a connection name.
- 4 Define the parameters of the connection type. For details, see *Connection Parameters for CICS Socket Listener Connections* in the EntireX Adapter documentation.

As a result, the folder will contain the connection and the adapter services (one for each IDL program). The name of a service is the same as the respective IDL program.

The default settings for the adapter services are:

- the **Default** package; if not available, the first package
- the IDL library name for the **Folder Name**
- the IDL library name with the suffix "Connection" for the **Connection Name**



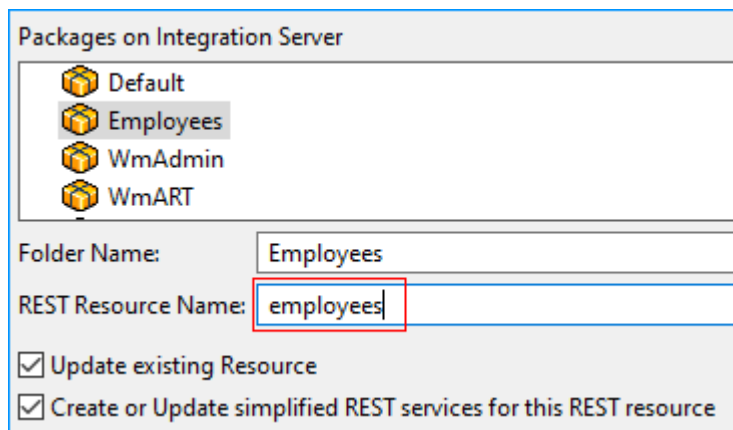
Note: When creating a connection, a package dependency is added such that the selected package depends on webMethods EntireX (the package `WmEntireX`) with the version currently used.

Step 5: Create or Update a REST Resource

➤ To create or update a REST resource

Make sure checkbox **Create or Update REST resource** is checked.

- 1 Select a package.



Packages on Integration Server

- Default
- Employees**
- WmAdmin
- WmART

Folder Name: Employees

REST Resource Name: employees

☒ Update existing Resource

☒ Create or Update simplified REST services for this REST resource

- 2 Define a folder name. If the folder does not exist, it will be created.
- 3 Define the REST Resource name.
- 4 Check **Update existing Resource** if you want to add additional services to an existing resource.
- 5 Check **Create or Update simplified REST services for this REST resource** if you want to generate additional services with a simplified REST API. This allows you to use a simple HTTP-GET method without a payload.



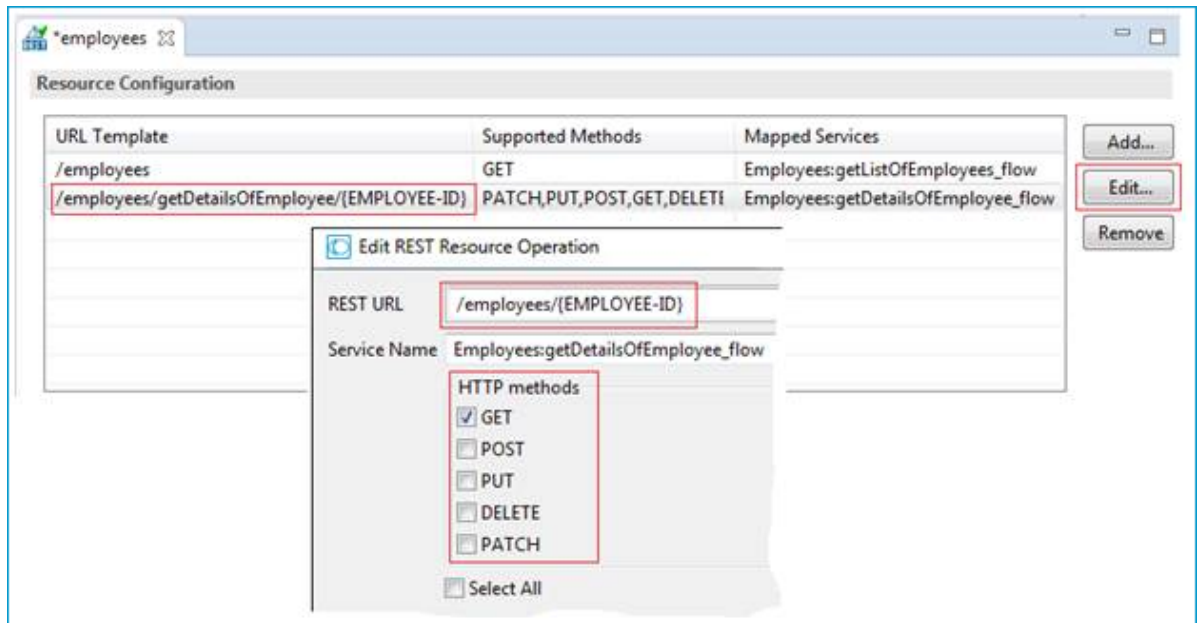
Note: This applies only to APIs where input signature parameters do not contain arrays.

;

Step 6: Edit the REST Resource

➤ To edit the REST resource

- 1 Switch to the **Service Development** perspective of Software AG Designer.



- 2 Refresh the package where you generated the REST Resource in the previous step.
- 3 Open the REST Resource.
- 4 Edit the REST URLs and HTTP Methods to match the functionality of your application.

For more information see:

- *REST Developer's Guide* in the webMethods Integration Server documentation for information on using the generated REST resources
- *REST Developer's Guide* in the webMethods Integration Server documentation

Task 3: Execute the Call from the REST Client to COBOL

Use an appropriate REST client, specify URL, Method and Headers and invoke your REST resource:

URL http://localhost:5555/restv2/employees

Method GET

Headers Accept application/json

200 OK

```
{
  "errorCode": "00000000",
  "errorFlag": "false",
  "response": {
    "EMPLOYEES-LIST": {
      "EMPLOYEES": [
        {
          "IDENT": "11100102",
          "FIRSTNAME": "Edgar",
          "SURNAME": "Schindler",
          "DATE-BIRTH": "Dec 4, 1962"
        },
        {
          "IDENT": "11100105",
          "FIRSTNAME": "Christian",
          "SURNAME": "Schirm",
          "DATE-BIRTH": "Mar 15, 1961"
        },
        {
          "IDENT": "11100106",
          "FIRSTNAME": "Rainer",
          "SURNAME": "Schmitt",
          "DATE-BIRTH": "Feb 13, 1955"
        },
        {
          "IDENT": "11100107",
          "FIRSTNAME": "Helga",
          "SURNAME": "Schmidt",
          "DATE-BIRTH": "May 26, 1961"
        }
      ]
    }
  }
}
```

URL http://localhost:5555/restv2/employees/111000105

Method GET

Headers Accept application/json

200 OK

```
{
  "errorCode": "00000000",
  "errorFlag": "false",
  "response": {
    "EMPLOYEE-DETAILS": {
      "FIRSTNAME": "Christian",
      "SURNAME": "Schirm",
      "DATE-BIRTH": "Mar 15, 1961",
      "DETAIL": {
        "ANNUAL-SALARY": "68000",
        "VACATION": "21",
        "LANGUAGE": "GER"
      }
    }
  }
}
```


10

Calling COBOL DFHCOMMAREA on z/OS CICS from REST

■ Introduction	154
■ What do I need to Install for this Scenario?	155
■ Task 1: Extract the Interface of a COBOL Server	157
■ Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server	170
■ Task 3: Execute the Call from the REST Client to COBOL	176

This scenario uses the tools IDL Extractor for COBOL and Integration Server Wrapper of the Designer.

Introduction



- 1 Extract the interface of the COBOL server program. See *Using the IDL Extractor for COBOL - Overview*.
- 2 Generate REST resources, connection and adapter services in Integration Server.
- 3 Execute the call from the REST client to the COBOL server program.

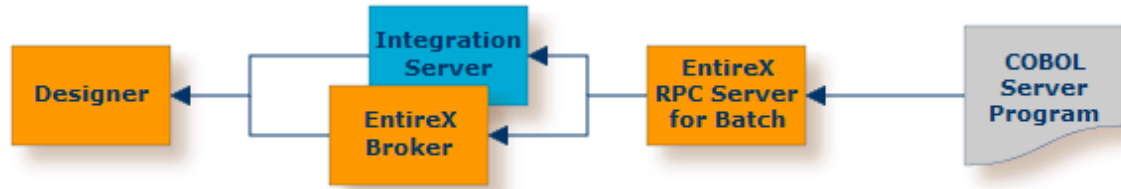
This scenario makes the following important assumptions:

For Task 1:

- You have a working COBOL DFHCOMMAREA server. For illustration and examples on such a server, see *CICS with DFHCOMMAREA Calling Convention* under *Introduction to the IDL Extractor for COBOL*.
- You have access to the related COBOL sources and copybooks. The minimum requirement is the `DATA DIVISION` of the interface. The sources and copybooks must be stored either
 - *locally*, that is, on the same machine where the Designer is running



- or *remotely* in a PDS or CA Librarian data set and accessed via the *RPC Server for Batch*, and either EntireX Broker or Integration Server



For Tasks 2 and 3:

- You have a REST client.
- You can call the COBOL server program 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
 - the *RPC Server for CICS*



- For the *EntireX Direct RPC* connection method you need:
 - the *RPC Server for CICS*



See also *Direct RPC* in the EntireX Adapter documentation.

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.

- You have an RPC Server for CICS installed. See *Installing the RPC Server for CICS* in the z/OS Installation documentation.
- Optionally 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.
- Optionally, for remote extraction only (Task 1), you have an RPC Server for Batch installed. See *Installing the RPC Server for Batch* in the z/OS Installation documentation.

Task 1: Extract the Interface of a COBOL Server

- [Introduction](#)
- [Sample COBOL Server used in the Examples](#)
- [Extracting a COBOL Server - Fast-track Method](#)
- [Extracting a COBOL Server - Modern Method with User-defined Mapping](#)
- [Comparison Chart: Fast-track versus User-defined Mapping](#)
- [Testing the Extraction Results](#)

Introduction

Follow the instructions for extracting COBOL, see *Using the IDL Extractor for COBOL - Overview* and choose *Scenario I: Create New IDL and Server Mapping Files* if this is your first extraction. Learn how to invoke the IDL Extractor for COBOL, create a COBOL Extractor Environment and about the possibilities to select the COBOL source.

The extraction process creates the following EntireX metafiles:

- IDL file. A Software AG IDL file contains definitions of the interface between client and server. See *Software AG IDL File* in the IDL Editor documentation in the IDL Editor documentation.
- Server mapping file (optional). The mapping file is a Designer file with extension .cvm that contains COBOL-specific mapping information. See *Server Mapping Files for COBOL* in the Designer documentation.

Sample COBOL Server used in the Examples

The following COBOL server is used to illustrate the features of the IDL Extractor for COBOL. Imagine an existing COBOL server called `EMPLOYEE`. It implements the access logic for a `LIST` and `DETAILS` function to a database view `EMPLOYEE`:

```

IDENTIFICATION DIVISION.
    PROGRAM-ID.      EMPLOYEE.
*****
*   Program Name    = EMPLOYEE
*   String Literal  = QUOTE
*****
01 DFHCOMMAREA.
    02 OPERATION                                PIC X(1).
        88 OPERATION-LIST                      VALUE 'L'.
        88 OPERATION-DETAILS                  VALUE 'D'.
    02 EMPLOYEE-ID                                PIC X(10).
    02 EMPLOYEE-DETAILS.
        03 FIRSTNAME                            PIC X(20).
        03 SURNAME                              PIC X(20).
        03 DATE-BIRTH                          PIC X(12).
        03 DETAILS                            PIC X(100).
        03 FILLER REDEFINES DETAILS.
            04 ANNUAL-SALARY          PACKED-DECIMAL PIC S9(9).
            04 VACATION                PIC S9(2).
            04 LANGUAGE                PIC X(3).
            04 FILLER                  PIC X(90).
    02 EMPLOYEES-COUNT                          PIC 9(8) BINARY.
    02 EMPLOYEES OCCURS 1 TO 99 DEPENDING ON EMPLOYEES-COUNT.
        03 IDENT                          PIC X(10).
        03 FIRSTNAME                      PIC X(20).
        03 SURNAME                        PIC X(20).
        03 DATE-BIRTH                    PIC X(12).

```

Two approaches for extracting the COBOL server are described below:

- **Fast-track**

Learn how EntireX helps you to connect the COBOL Server with quick results.

- **User-defined Mapping**

The following features are among those available with a user-defined mapping:

- **Shape the interface to your COBOL server**

You can minimize the interface by suppressing COBOL server fields or by providing constant values. Renaming interfaces allows you to specify a readable long name. These two features together - constants and renaming of interfaces - are most powerful when multiple interfaces are implemented in a single COBOL server. Each function of the COBOL server triggered by an operation code or function code can be modelled as a separate REST resource method/operation.

■ Select alternative mappings

You can select alternative mappings for COBOL data items (REDEFINES, tables etc.) to enable scenarios with COBOL servers where data exchange is not fully described by the LINKAGE SECTION COBOL data items.

These approaches are described under [Fast-track Method](#) and [Modern Method](#) below.

Extracting a COBOL Server - Fast-track Method

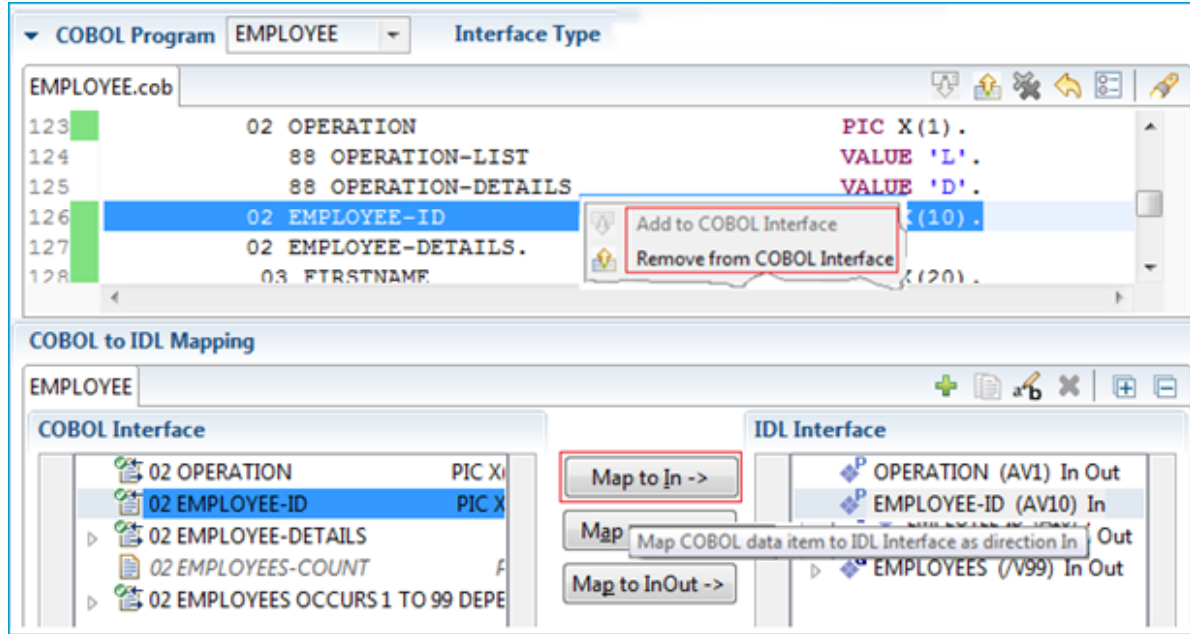
➤ To extract the COBOL Server

- 1 Switch to the **EntireX** perspective.
- 2 Call the IDL Extractor for COBOL.
- 3 Set the correct interface type as described under *Step 4: Define the Extraction Settings and Start Extraction* under *Scenario I: Create New IDL and Server Mapping Files* in the IDL Extractor for COBOL documentation.

- 4 Set the correct value for checkbox **Input Message same as Output Message**.
 - Check the checkbox if the COBOL data structure of the CICS input message is the same as the structure of the CICS output message, that is, there is one COBOL layout for both directions. See the following COBOL server examples in the IDL Extractor for COBOL documentation:
 - *Example 1: Redefines*
 - *Example 2: Buffer Technique*
 - *Example 3: COBOL SET ADDRESS Statements*
 - Clear the checkbox if the COBOL data structure of the CICS input message is different to the structure of the CICS output message, that is, the output overlays the input, which means there is a second COBOL layout for the reply. See the following COBOL server examples in the IDL Extractor for COBOL documentation:
 - *Example 1: Redefines*
 - *Example 2: Buffer Technique*
 - *Example 3: COBOL SET ADDRESS Statements*

For our example, check the check box, because the COBOL layout for input is the same as the COBOL layout for output.

- 5 Press **Next** to enter the COBOL Mapping Editor.



The purpose of the COBOL Mapping Editor is to map COBOL data items to IDL (Interface Definition Language) parameters. It consists of three main areas:

- The upper window pane is the COBOL program source.
- The window pane on the left-hand side is the COBOL interface, consisting of COBOL data items representing the parameters in your COBOL server program.
- The window pane on the right-hand side is the IDL interface consisting of IDL parameters.

What you select depends on check-box **Input Message same as Output Message**:

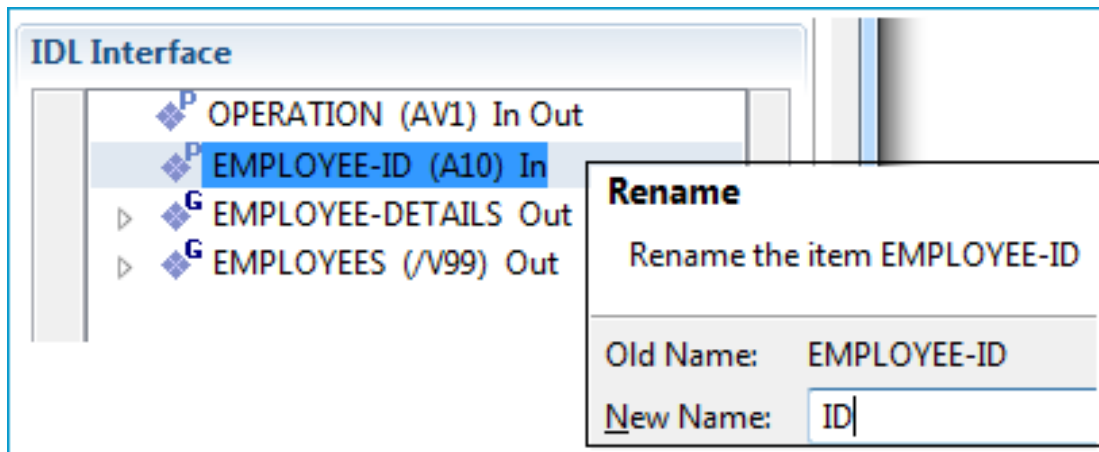
- If there is a single COBOL data structure of the CICS message for input and output - check box **Input Message same as Output Message** is checked - you select a single COBOL data structure in the COBOL Mapping Editor. This structure is used for COBOL input as well as for COBOL output. For more information in this case see *Mapping Editor User Interface* in the IDL Extractor for COBOL documentation.
- If there is a second COBOL data structure for the CICS output message - check box **Input Message same as Output Message** is cleared - you will select two COBOL data structures in the COBOL Mapping Editor. One is used for COBOL input and the other is for COBOL output. For more information in this case see *Mapping Editor User Interface* in the IDL Extractor for COBOL documentation.

COBOL extraction is a two-step process:

- First, select the COBOL parameters using the **Add to** and **Remove from COBOL Interface** from the context menu on the COBOL data items in the upper window pane (COBOL source); in many situations the COBOL interface is detected by the Mapping Editor directly and you can omit this step.
- Then map the COBOL parameters to IDL directions using the **Map to In/Out/InOut** buttons. COBOL parameters do not have directions, they are always INOUT. This step can be left out for a quick test in non-production scenarios; for real production it is recommended to set IDL directions to reduce data transfer. You also have the option to assign a meaningful name to an IDL parameter if the original COBOL data item is not self-explanatory.

For our example:

- Set **Map to In** for COBOL data item `EMPLOYEE-ID` as IDL direction (see screenshot in previous step). For `EMPLOYEES-DETAILS` and `EMPLOYEES` perform the similar **Map to Out**.
- Use the function **Rename** from the context menu on the IDL parameters in the **IDL Interface** pane to rename COBOL parameter `EMPLOYEES-ID` to IDL parameter `ID`. Do the same for `EMPLOYEE-DETAILS`: Rename to `EMPLOYEE`.



- 6 Press **Finish**. The default extraction settings ensure you get useful results. The outcome is a Software AG IDL file (interface definition language), a Designer file with extension `.idl`:

```
library 'EMPLAPPL' is
program 'EMPLOYEE' is
define data parameter
  1 OPERATION (AV1) In Out
  1 ID (AV10) In /* Original Name:EMPLOYEE-ID
  1 EMPLOYEE Out /* Original Name:EMPLOYEE-DETAILS
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
    2 DETAILS (AV100)
  1 EMPLOYEES (/V99) Out
    2 IDENT (AV10)
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
end-define
```

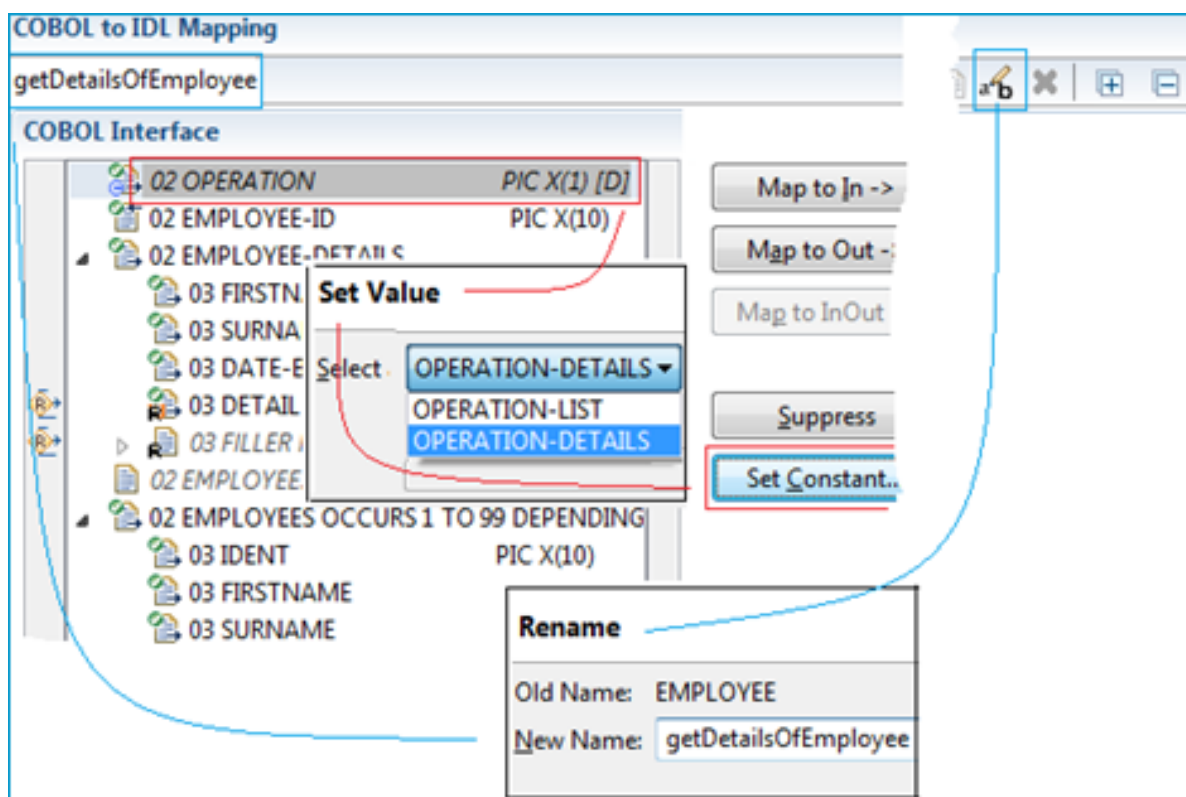
If required, a server mapping file with the same name as the IDL file but the extension `.cvm` may be generated too, for technical reasons. This file must always be stored together with the concomitant IDL file. For more information, see *Server Mapping Files for COBOL* in the Designer documentation. By default, the interface name `EMPLOYEE` is taken from the COBOL server program name.


Extracting a COBOL Server - Modern Method with User-defined Mapping

EntireX allows you to extract all implemented interfaces of the COBOL server separately. Instead of a large `EMPLOYEE` interface, separate interfaces `getListOfEmployees` and `getDetailsOfEmployee` are extracted. Each interface contains required parameters; obsolete parameters for an interface are suppressed, improving its usability.

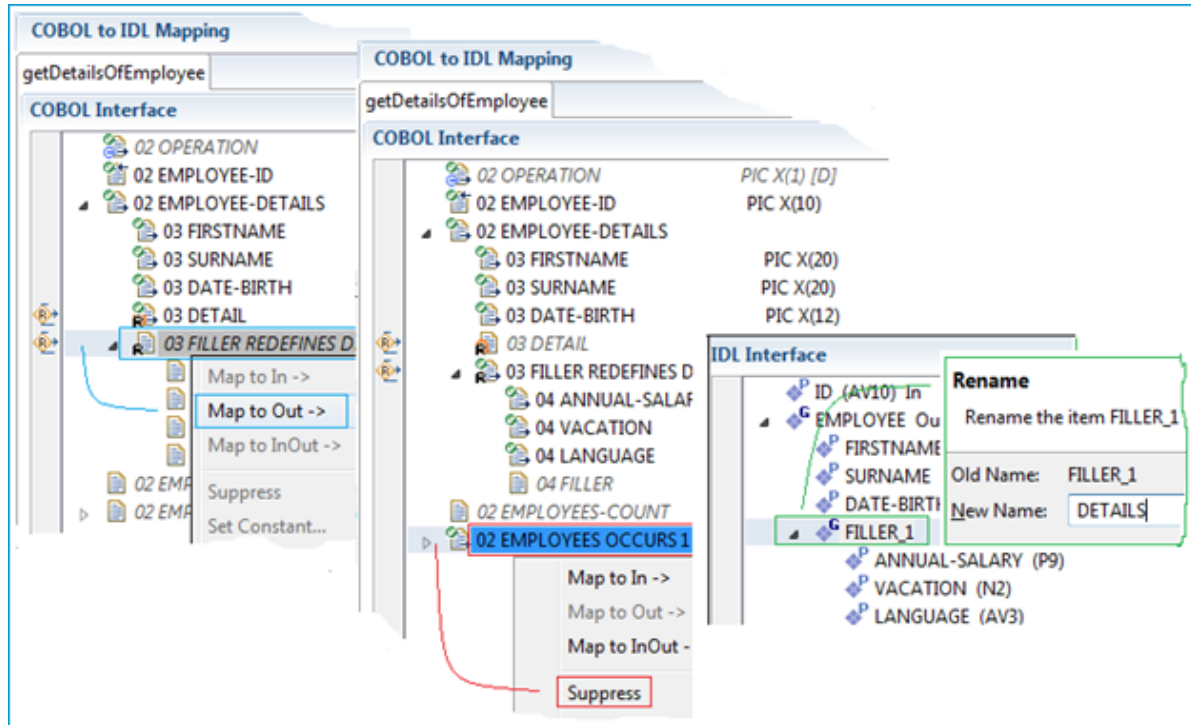
➤ To extract a COBOL server with a user-defined mapping

- 1 Shape `EMPLOYEE` to `getDetailsOfEmployee`:



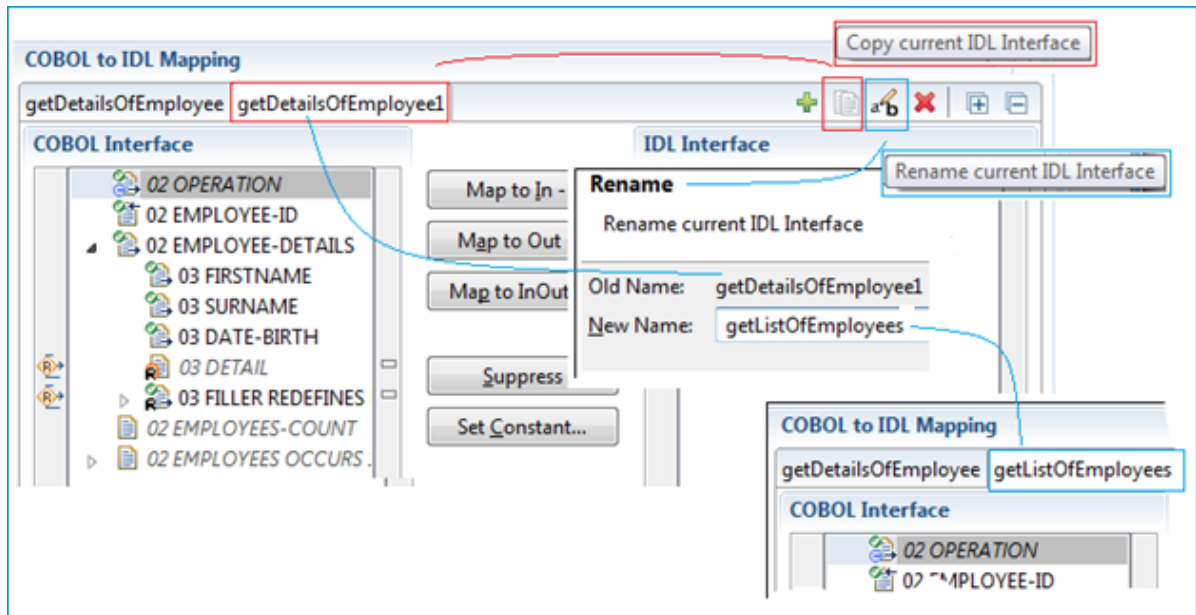
- Mark the `OPERATION` field in the **COBOL Interface** pane and press the **Set Constant** button. In the pop-up window, select `OPERATION-DETAILS` (red markers). The `OPERATION-DETAILS` value originates from COBOL level-88 data item (blue markers). If there is no COBOL level-88 enumeration type in the COBOL program defined, you can enter the function code manually in the pop-up window.
- By default the IDL interface gets the same name as your COBOL server: `EMPLOYEE`. Use the **Rename** button  from the toolbar of the COBOL Mapping Editor and change the IDL interface name to `getDetailsOfEmployee` (blue markers). This name is used as the IS service name later.

2 Continue shaping, making `getDetailsOfEmployee` easier to use:



- Use the context menu of the redefinition of the `DETAIL` field in the **COBOL Interface** pane, `FILLER REDEFINES DETAIL` and choose **Map to Out** (blue markers). A redefinition is a secondary layout (field types) of the same memory area. The redefinition contains more specific information (`ANNUAL-SALARY`, `VACATION`, etc.) than character buffer `DETAIL`. This is the reason why it is more useful in your interface.
- Use the context menu of `FILLER_1` in the IDL interface pane and rename `FILLER_1` to `DETAILS` (green markers) for a readable IS service field name later.
- The `DETAILS` function in the COBOL server does not make use of the `EMPLOYEES` field. Therefore you can leave it out of the interface: use the context menu of the `EMPLOYEES` field in the **COBOL Interface** pane and choose **Suppress** (red markers).

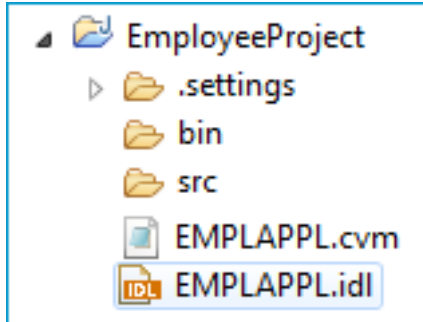
3 Shape `EMPLOYEE` to `getListOfEmployees` Function:



The COBOL Mapping Editor allows you to define multiple IDL interfaces to the same COBOL server:

- You create the additional IDL interface using the toolbar button **Copy current IDL interface** (red markers). A new tab `getDetailsOfEmployee1` is created. The previously extracted `getDetailsOfEmployee` interface still exists in the first tab. Once you reactivate the first tab, you will see the interface of `getDetailsOfEmployee` again.
 - Use the **Rename** button (blue markers) from the toolbar of the COBOL Mapping Editor and change the IDL interface name to `getListOfEmployees` (blue markers). This name is used as the IS service name later.
 - For the `LIST` function you need the `OPERATION-LIST` value in the `OPERATION` field: Mark the `OPERATION` field in the **COBOL Interface** pane and press the **Set Constant** button. In the pop-up window, select `OPERATION-LIST` (similar to Step 1. above *Shape `EMPLOYEE` to `getDetailsOfEmployee`*; red markers).
 - Use the context menu of the `EMPLOYEES` field in the **COBOL Interface** pane, `EMPLOYEES OCCURS 1 TO 99` and choose **Map to Out** (Similar to Step 2. above *Continue shaping, making `getDetailsOfEmployee` easier to use*; blue markers).
 - Because the `EMPLOYEE` and `EMPLOYEE-DETAIL` fields are not used in the COBOL server `LIST` function, you leave them out in the IDL interface: use the context menu of `EMPLOYEE-DETAIL` field in the **COBOL Interface** pane and choose **Suppress** (similar to Step 2. above *Continue shaping, making `getDetailsOfEmployee` easier to use*; red markers). Do the same for the `EMPLOYEE-ID` field.
- 4 Press **Finish** to retrieve the extraction result in the form of a Software AG IDL file. At the same time, a server mapping file for COBOL (Designer file with extension `.cvm`) is created.

See *Server Mapping Files for COBOL* in the Designer documentation. The *Software AG IDL File* in the IDL Editor documentation describes the interfaces from the client point of view, while the server mapping file contains the mapping to the real COBOL server. Both of these files must be kept together and in sync, otherwise a call to the COBOL server may fail.



To summarize: We created two IDL interfaces. In the IDL file, these resulted in two IDL programs: `getListOfEmployees` and `getDetailsOfEmployee`. Both IDL programs were given readable names (Steps 1 and 3). Meaningful fields were kept, while superfluous fields were suppressed (Steps 1 and 3). The program `getDetailsOfEmployee` contains the redefined fields of parameter `DETAILS` (see below) mapped during extraction (Step 1).



Note: At runtime, the RPC client generated with the extracted interface will send data for the redesigned interfaces, while your COBOL server still expects `EMPLOYEE` data. The EntireX runtime transforms the incoming data stream from the RPC client, using the server mapping file.

```

program 'getDetailsOfEmployee' is
  define data parameter
    1 ID (AV10) In /* Original Name:EMPLOYEE-ID
    1 EMPLOYEE Out /* Original Name:EMPLOYEE-DETAILS
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
    2 DETAILS /* Original Name:FILLER_1
    3 ANNUAL-SALARY (P9)
    3 VACATION (N2)
    3 LANGUAGE (AV3)
  end-define
  program 'getListOfEmployees' is
    1 EMPLOYEES (/V99) Out
    2 IDENT (AV10)
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
  end-define
    
```

Comparison Chart: Fast-track versus User-defined Mapping

The following table highlights the differences when shaping an interface with a *user-defined mapping* compared with a *fast-path extraction*.

	Interface Shaping with User-defined Mapping	Fast-path Extraction
General	User-defined interface with dedicated mapping. For our example, the interfaces are small and tidy and more self-explanatory with long, readable interface names. They are easier to use with the hidden <code>OPERATION</code> field and the suppressed COBOL fields, which are not needed.	Automatic, quick result extraction with COBOL-like interfaces on IS. For our example, the IS interface <code>EMPLOYEE</code> matches exactly 1:1, field by field.
IS service(s)	Multiple small and handy IS services; each <code>OPERATION</code> code is mapped to a separate IS service.	One big IS service.
IS service name	Readable long name.	Short Subprogram name; up to 8 characters.
IS fields	Usage of Suppress and Set Constant reduces the message length. This keeps focus on relevant data items and keeps the client's interface clean. It may also improve performance.	The IS fields and COBOL server parameters match 1:1. As COBOL layout descriptions are sometimes used for many different purposes, irrelevant data items appear and clutter up the IS interface.
<code>OPERATION</code> parameter	Suppressed: the <code>OPERATION</code> parameter does not exist in the IS service as an IS field.	The <code>OPERATION</code> parameter exists in the IS service as an IS field and needs to be filled-in by the client endpoint.
<code>OPERATION</code> code	The <code>OPERATION</code> code is provided internally by EntireX runtime in the <code>OPERATION</code> field.	The <code>OPERATION</code> code needs to be specified in the IS service in the <code>OPERATION</code> field by the client endpoint.
<code>REDEFINE</code> parameters	Either the parameter that is redefined or one of its redefinitions is available as an IS field.	Only the parameter that is redefined is available as an IS field. Redefinitions of the parameter are not available as IS fields.

A user-defined mapping enables you also to define alternative mappings for COBOL data items (`REDEFINES`, tables etc.). These enable scenarios with COBOL servers where data exchange is not fully described by the `LINKAGE SECTION` COBOL data items. For more information refer to the IDL Extractor for COBOL documentation:

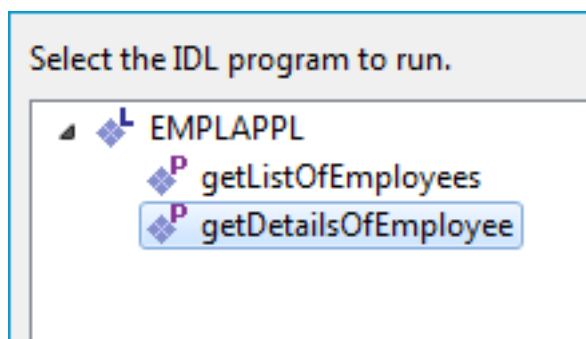
- *Mapping Editor IDL Interface Mapping Functions* if the check-box **Input Message same as Output Message** was checked.
- *Mapping Editor IDL Interface Mapping Functions* if the check-box **Input Message same as Output Message** was cleared.
- *User-defined Mapping*

Testing the Extraction Results

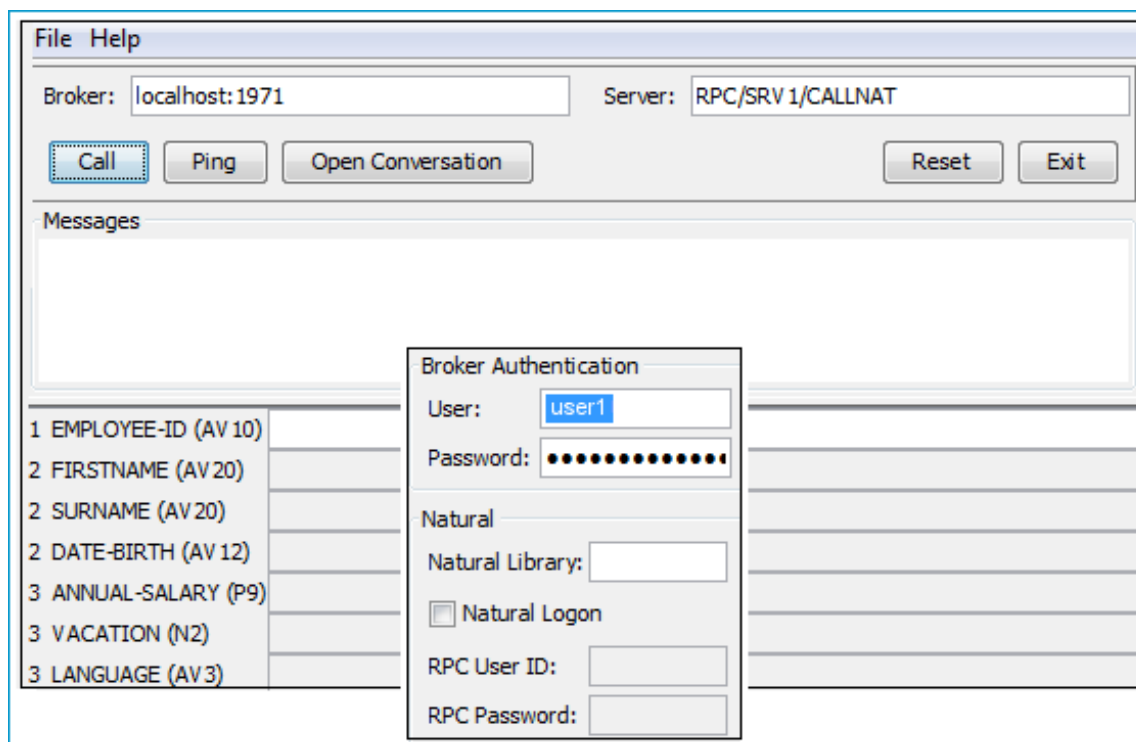
The following pictures use the extraction results described under *Extracting a COBOL Server - Modern Method with User-defined Mapping*.

> To test the extraction results (optional)

- 1 You can test the results of the extraction operation and the REST server back end, using the EntireX IDL Tester. From the context menu of the IDL file in the Designer, choose **Software AG IDL Tester**.



- 2 Select `getDetailsOfEmployee`.



Note that the broker and server parameters contain the explicit route to call the server program, and you can optionally ping the connection from this client. With the **File > Options** dialog, you can set:

- **User** and **Password** for *broker* authentication
- **RPC User ID** and **RPC Password** for *server* authentication

See *EntireX IDL Tester* in the Designer documentation.

- 3 Check the Integration Server log, the EntireX Adapter log or the RPC logs. Applies to all connection methods.

Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server

This section describes your first steps to create a new Integration Server connection. This is described in more detail under *Using the Integration Server Wrapper*, for example working with existing Integration Server connections. This section covers the following topics:

- [Step 1: Start the Integration Server Wrapper Wizard](#)
- [Step 2: Create a New Integration Server Connection](#)
- [Step 3: Select the Connection Type](#)
- [Step 4: Define Adapter Services for an RPC Connection](#)
- [Step 5: Create or Update a REST Resource](#)
- [Step 6: Edit the REST Resource](#)

Step 1: Start the Integration Server Wrapper Wizard

➤ To start the Integration Server Wrapper wizard

- 1 In the context menu of a Software AG IDL file, choose **Integration Server > Generate web-Methods IS Connection**.

This starts the wizard with a list of existing Integration Server Wrapper connections.



Note: If the selected IDL file is not valid because of a syntax error, an error dialog comes up and the wizard does not start.

- 2 Continue with *Step 2: Create a New Integration Server Connection*.

Step 2: Create a New Integration Server Connection

Define New Integration Server Connection

Add host:port, user and password for a new Integration Server

Server:

User:

Password:

☐ Use secure connection

➤ **To create a new Integration Server connection**

- 1 Define the new Integration Server connection on the wizard page.

**Notes:**

1. The only required field is **Server**. Enter the hostname of the Integration Server including an optional port number. If no port number is specified, port number defaults to "5555". The **Integration Server Authentication** can be passed with the **User** and **Password** fields.
2. Optional settings are for secure connections. The **Truststore for HTTPS** contains all signed certificates and must be a valid truststore.
3. The check box **Verify host name** checks that the hostname is entered in the stored certificate.
4. When the Integration Server has **Client Authentication** enabled, you can specify your **Keystore** file and keystore **Password**.
5. For managing Integration Server connections, see [Preferences](#).

- 2 Choose **Next** and continue with *Step 3: Select the Connection Type*.

Step 3: Select the Connection Type

☒ Create a new EntireX Adapter connection of type: **EntireX RPC Connection**

☐ Update Adapter Services or Listeners to an existing Integration Server connection:

List of Connections

Connection Name	Package Name	Connection Type	Enabled
Employees:EmployeesConnection	Employees	EntireX RPC Connection	Yes

Total: 1

☒ Map Software AG IDL data types to String

☒ Create or Update REST resource

> To create a new connection

- 1 Select a connection type from the drop down list. Connection types are described under *EntireX Adapter Connections* in the EntireX Adapter documentation and *Introduction to the Integration Server Wrapper*.



Note: The list of connection types is filtered: connection types that require a license are only shown if a corresponding license file is available. Reliable RPC connections are only shown if all IDL programs contain only IN parameters. Also, if a server mapping file is available, only those connection types that support the interface type specified in the server mapping file are shown.

- 2 Check the box **Create or Update REST resource**.
- 3 Click **Next** and continue with *Step 4: Define Adapter Services for an RPC Connection*.

Step 4: Define Adapter Services for an RPC Connection

Packages on Integration Server localhost:5555

- Default
- Employees**
- WmART
- WmARTEstDC

Folder Name:

Connection Name:

RPC Connection to EntireX

Broker ID:

Server Address:

> To create a connection and related adapter services

- 1 Select an existing package or create a new package for the created objects.
- 2 Define a folder name. If the folder does not exist, it will be created.
- 3 Define a connection name.
- 4 Define the parameters of the connection type. For details, see *Connection Parameters for RPC Connections* in the EntireX Adapter documentation.

As a result, the folder will contain the connection and the adapter services (one for each IDL program). The name of a service is the same as the respective IDL program.

The default settings for the adapter services are:

- the **Default** package; if not available, the first package
- the IDL library name for the **Folder Name**
- the IDL library name with the suffix "Connection" for the **Connection Name**



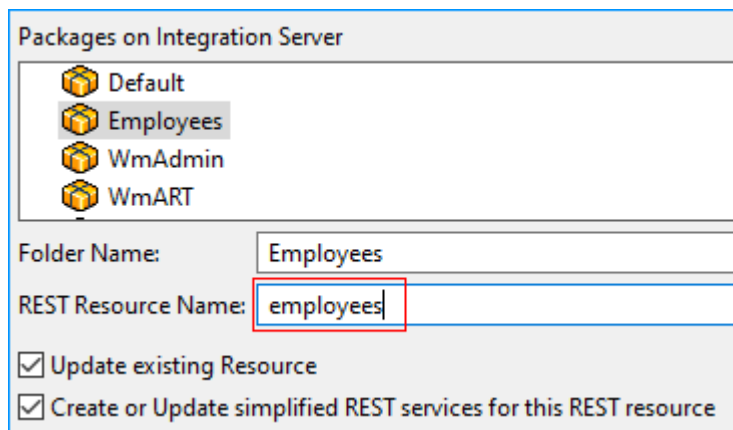
Note: When creating a connection, a package dependency is added such that the selected package depends on webMethods EntireX (the package `WmEntireX`) with the version currently used.

Step 5: Create or Update a REST Resource

➤ To create or update a REST resource

Make sure checkbox **Create or Update REST resource** is checked.

- 1 Select a package.



Packages on Integration Server

- Default
- Employees**
- WmAdmin
- WmART

Folder Name: Employees

REST Resource Name: employees

☒ Update existing Resource

☒ Create or Update simplified REST services for this REST resource

- 2 Define a folder name. If the folder does not exist, it will be created.
- 3 Define the REST Resource name.
- 4 Check **Update existing Resource** if you want to add additional services to an existing resource.
- 5 Check **Create or Update simplified REST services for this REST resource** if you want to generate additional services with a simplified REST API. This allows you to use a simple HTTP-GET method without a payload.

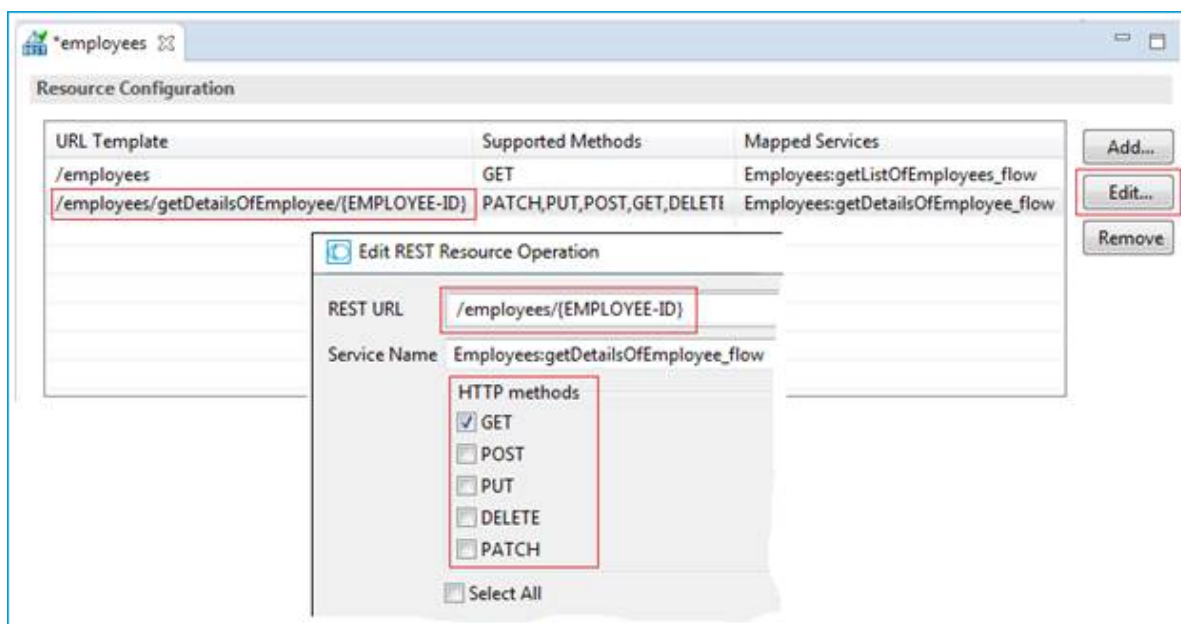


Note: This applies only to APIs where input signature parameters do not contain arrays.

Step 6: Edit the REST Resource

➤ To edit the REST resource

- 1 Switch to the **Service Development** perspective of Software AG Designer.



- 2 Refresh the package where you generated the REST Resource in the previous step.
- 3 Open the REST Resource.
- 4 Edit the REST URLs and HTTP Methods to match the functionality of your application.

For more information see:

- *REST Developer's Guide* in the webMethods Integration Server documentation for information on using the generated REST resources
- *REST Developer's Guide* in the webMethods Integration Server documentation

Task 3: Execute the Call from the REST Client to COBOL

Use an appropriate REST client, specify URL, Method and Headers and invoke your REST resource:

URL http://localhost:5555/restv2/employees

Method GET

Headers Accept application/json

200 OK

```
{
  "errorCode": "00000000",
  "errorFlag": "false",
  "response": {
    "EMPLOYEES-LIST": {
      "EMPLOYEES": [
        {
          "IDENT": "11100102",
          "FIRSTNAME": "Edgar",
          "SURNAME": "Schindler",
          "DATE-BIRTH": "Dec 4, 1962"
        },
        {
          "IDENT": "11100105",
          "FIRSTNAME": "Christian",
          "SURNAME": "Schirm",
          "DATE-BIRTH": "Mar 15, 1961"
        },
        {
          "IDENT": "11100106",
          "FIRSTNAME": "Rainer",
          "SURNAME": "Schmitt",
          "DATE-BIRTH": "Feb 13, 1955"
        },
        {
          "IDENT": "11100107",
          "FIRSTNAME": "Helga",
          "SURNAME": "Schmidt",
          "DATE-BIRTH": "May 26, 1961"
        }
      ]
    }
  }
}
```

URL http://localhost:5555/restv2/employees/111000105

Method GET

Headers Accept application/json

200 OK

```
{
  "errorCode": "00000000",
  "errorFlag": "false",
  "response": {
    "EMPLOYEE-DETAILS": {
      "FIRSTNAME": "Christian",
      "SURNAME": "Schirm",
      "DATE-BIRTH": "Mar 15, 1961",
      "DETAIL": {
        "ANNUAL-SALARY": "68000",
        "VACATION": "21",
        "LANGUAGE": "GER"
      }
    }
  }
}
```


11

Calling COBOL Large Buffer on z/OS CICS from REST

■ Introduction	179
■ What do I need to Install for this Scenario?	181
■ Task 1: Extract the Interface of a COBOL Server	182
■ Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server	195
■ Task 3: Execute the Call from the REST Client to COBOL	201

This scenario uses the tools IDL Extractor for COBOL and Integration Server Wrapper of the Designer.

Introduction



- ① Extract the interface of the COBOL server program. See *Using the IDL Extractor for COBOL - Overview*.
- ② Generate REST resources, connection and adapter services in Integration Server.
- ③ Execute the call from the REST client to the COBOL server program.

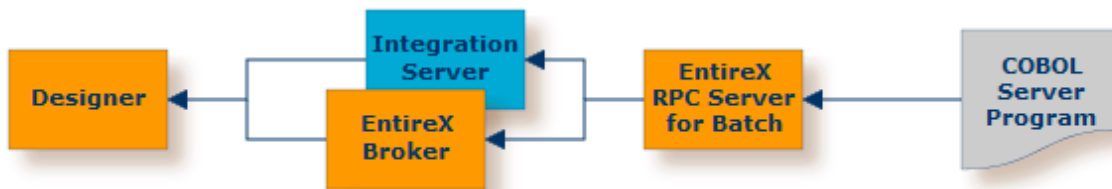
This scenario makes the following important assumptions:

For Task 1:

- You have a working COBOL Large Buffer server. For illustration and examples on such a server, see *CICS with DFHCOMMAREA Large Buffer Interface* under *Introduction to the IDL Extractor for COBOL*.
- You have access to the related COBOL sources and copybooks. The minimum requirement is the `DATA DIVISION` of the interface. The sources and copybooks must be stored either
 - *locally*, that is, on the same machine where the Designer is running



- or *remotely* in a PDS or CA Librarian data set and accessed via the *RPC Server for Batch*, and either EntireX Broker or Integration Server



For Tasks 2 and 3:

- You have a REST client.
- You can call the COBOL server program 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
 - the *RPC Server for CICS*



- For the *EntireX Direct RPC* connection method you need:
 - the *RPC Server for CICS*



See also *Direct RPC* in the EntireX Adapter documentation.

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.
- You have an RPC Server for CICS installed. See *Installing the RPC Server for CICS* in the z/OS Installation documentation.
- Optionally 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.
- Optionally, for remote extraction only (Task 1), you have an RPC Server for Batch installed. See *Installing the RPC Server for Batch* in the z/OS Installation documentation.

Task 1: Extract the Interface of a COBOL Server

- [Introduction](#)
- [Sample COBOL Server used in the Examples](#)
- [Extracting a COBOL Server - Fast-track Method](#)
- [Extracting a COBOL Server - Modern Method with User-defined Mapping](#)
- [Comparison Chart: Fast-track versus User-defined Mapping](#)
- [Testing the Extraction Results](#)

Introduction

Follow the instructions for extracting COBOL, see *Using the IDL Extractor for COBOL - Overview* and choose *Scenario I: Create New IDL and Server Mapping Files* if this is your first extraction. Learn how to invoke the IDL Extractor for COBOL, create a COBOL Extractor Environment and about the possibilities to select the COBOL source.

The extraction process creates the following EntireX metafiles:

- IDL file. A Software AG IDL file contains definitions of the interface between client and server. See *Software AG IDL File* in the IDL Editor documentation in the IDL Editor documentation.
- Server mapping file (optional). The mapping file is a Designer file with extension .cvm that contains COBOL-specific mapping information. See *Server Mapping Files for COBOL* in the Designer documentation.

Sample COBOL Server used in the Examples

The following COBOL server is used to illustrate the features of the IDL Extractor for COBOL. Imagine an existing COBOL server called `EMPLOYEE`. It implements the access logic for a `LIST` and `DETAILS` function to a database view `EMPLOYEE`:

```

IDENTIFICATION DIVISION.
    PROGRAM-ID.      EMPLOYEE.
*****
*   Program Name    = EMPLOYEE
*   String Literal  = QUOTE
*****
01 DFHCOMMAREA.
    02 OPERATION                                PIC X(1).
        88 OPERATION-LIST                      VALUE 'L'.
        88 OPERATION-DETAILS                   VALUE 'D'.
    02 EMPLOYEE-ID                             PIC X(10).
    02 EMPLOYEE-DETAILS.
        03 FIRSTNAME                          PIC X(20).
        03 SURNAME                            PIC X(20).
        03 DATE-BIRTH                         PIC X(12).
        03 DETAILS                            PIC X(100).
        03 FILLER REDEFINES DETAILS.
            04 ANNUAL-SALARY                    PACKED-DECIMAL PIC S9(9).
            04 VACATION                          PIC S9(2).
            04 LANGUAGE                          PIC X(3).
            04 FILLER                            PIC X(90).
    02 EMPLOYEES-COUNT                         PIC 9(8) BINARY.
    02 EMPLOYEES OCCURS 1 TO 99 DEPENDING ON EMPLOYEES-COUNT.
        03 IDENT                              PIC X(10).
        03 FIRSTNAME                          PIC X(20).
        03 SURNAME                            PIC X(20).
        03 DATE-BIRTH                         PIC X(12).

```

Two approaches for extracting the COBOL server are described below:

- **Fast-track**

Learn how EntireX helps you to connect the COBOL Server with quick results.

- **User-defined Mapping**

The following features are among those available with a user-defined mapping:

- **Shape the interface to your COBOL server**

You can minimize the interface by suppressing COBOL server fields or by providing constant values. Renaming interfaces allows you to specify a readable long name. These two features together - constants and renaming of interfaces - are most powerful when multiple interfaces are implemented in a single COBOL server. Each function of the COBOL server triggered by an operation code or function code can be modelled as a separate REST resource method/operation.

■ Select alternative mappings

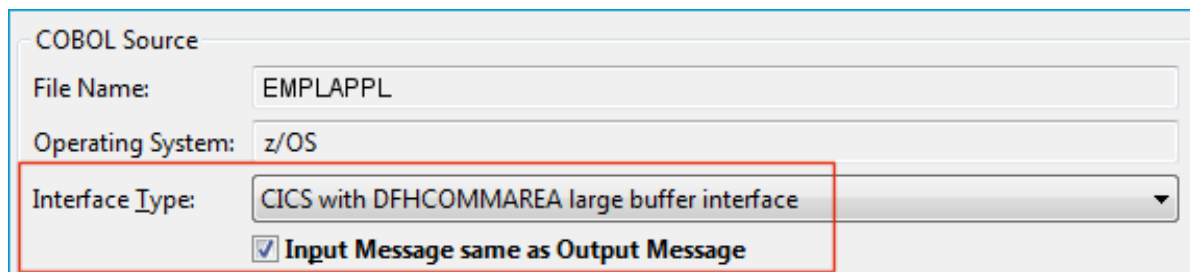
You can select alternative mappings for COBOL data items (REDEFINES, tables etc.) to enable scenarios with COBOL servers where data exchange is not fully described by the LINKAGE SECTION COBOL data items.

These approaches are described under [Fast-track Method](#) and [Modern Method](#) below.

Extracting a COBOL Server - Fast-track Method

> To extract the COBOL Server

- 1 Switch to the **EntireX** perspective.
- 2 Call the IDL Extractor for COBOL.
- 3 Set the correct interface type as described under *Step 4: Define the Extraction Settings and Start Extraction* under *Scenario I: Create New IDL and Server Mapping Files* in the IDL Extractor for COBOL documentation.



COBOL Source

File Name: EMPLAPPL

Operating System: z/OS

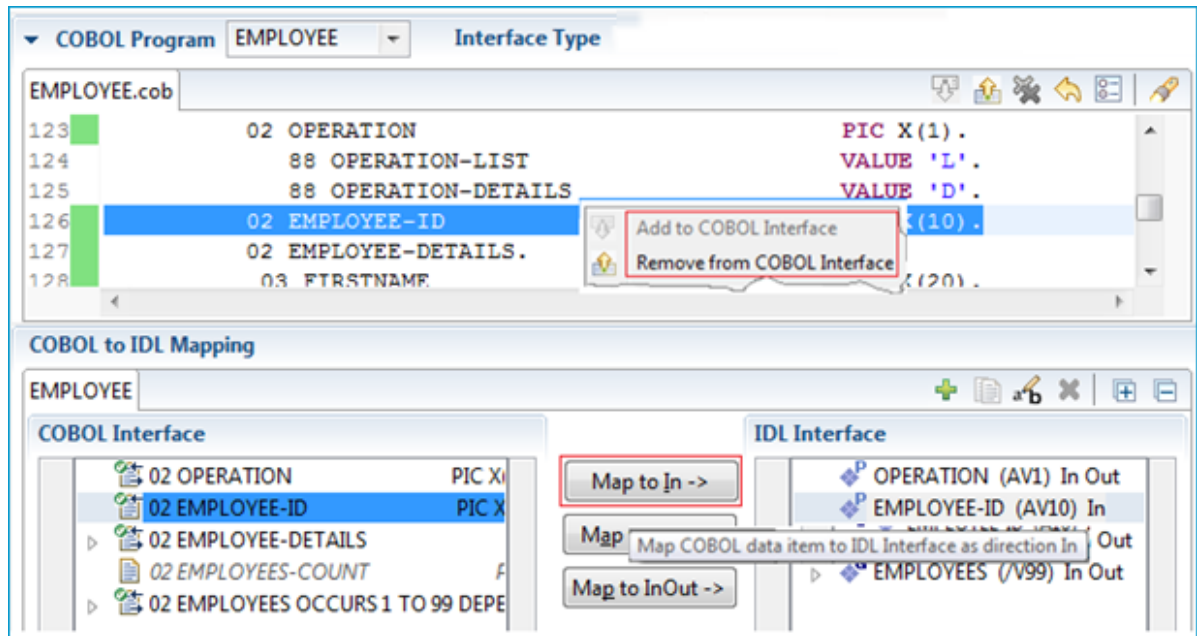
Interface Type: CICS with DFHCOMMAREA large buffer interface

☒ Input Message same as Output Message

- 4 Set the correct value for checkbox **Input Message same as Output Message**.
 - *Check* the checkbox if the COBOL data structure of the CICS input message is the same as the structure of the CICS output message, that is, there is one COBOL layout for both directions.
 - *Clear* the checkbox if the COBOL data structure of the CICS input message is different to the structure of the CICS output message, that is, the output overlays the input, which means there is a second COBOL layout for the reply.

For our example, check the check box, because the COBOL layout for input is the same as the COBOL layout for output.

- 5 Press **Next** to enter the COBOL Mapping Editor.



The purpose of the COBOL Mapping Editor is to map COBOL data items to IDL (Interface Definition Language) parameters. It consists of three main areas:

- The upper window pane is the COBOL program source.
- The window pane on the left-hand side is the COBOL interface, consisting of COBOL data items representing the parameters in your COBOL server program.
- The window pane on the right-hand side is the IDL interface consisting of IDL parameters.

What you select depends on check-box **Input Message same as Output Message**:

- If there is a single COBOL data structure of the CICS message for input and output - check box **Input Message same as Output Message** is checked - you select a single COBOL data structure in the COBOL Mapping Editor. This structure is used for COBOL input as well as for COBOL output. For more information in this case see *Mapping Editor User Interface* in the IDL Extractor for COBOL documentation.
- If there is a second COBOL data structure for the CICS output message - check box **Input Message same as Output Message** is cleared - you will select two COBOL data structures in the COBOL Mapping Editor. One is used for COBOL input and the other is for COBOL output. For more information in this case see *Mapping Editor User Interface* in the IDL Extractor for COBOL documentation.

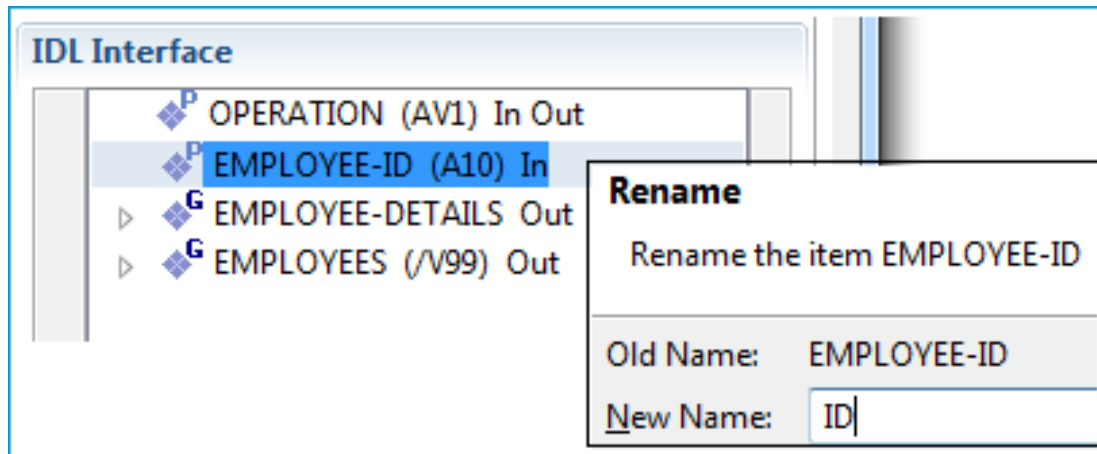
COBOL extraction is a two-step process:

- First, select the COBOL parameters using the **Add to** and **Remove from COBOL Interface** from the context menu on the COBOL data items in the upper window pane (COBOL source); in many situations the COBOL interface is detected by the Mapping Editor directly and you can omit this step.

- Then map the COBOL parameters to IDL directions using the **Map to In/Out/InOut** buttons. COBOL parameters do not have directions, they are always INOUT. This step can be left out for a quick test in non-production scenarios; for real production it is recommended to set IDL directions to reduce data transfer. You also have the option to assign a meaningful name to an IDL parameter if the original COBOL data item is not self-explanatory.

For our example:

- Set **Map to In** for COBOL data item `EMPLOYEE-ID` as IDL direction (see screenshot in previous step). For `EMPLOYEES-DETAILS` and `EMPLOYEES` perform the similar **Map to Out**.
- Use the function **Rename** from the context menu on the IDL parameters in the **IDL Interface** pane to rename COBOL parameter `EMPLOYEES-ID` to IDL parameter `ID`. Do the same for `EMPLOYEE-DETAILS`: Rename to `EMPLOYEE`.



- 6 Press **Finish**. The default extraction settings ensure you get useful results. The outcome is a Software AG IDL file (interface definition language), a Designer file with extension `.idl`:

```

library 'EMPLAPPL' is
program 'EMPLOYEE' is
  define data parameter
    1 OPERATION (AV1) In Out
    1 ID (AV10) In /* Original Name:EMPLOYEE-ID
    1 EMPLOYEE Out /* Original Name:EMPLOYEE-DETAILS
      2 FIRSTNAME (AV20)
      2 SURNAME (AV20)
      2 DATE-BIRTH (AV12)
      2 DETAILS (AV100)
    1 EMPLOYEES (/V99) Out
      2 IDENT (AV10)
      2 FIRSTNAME (AV20)
      2 SURNAME (AV20)
      2 DATE-BIRTH (AV12)
  end-define

```

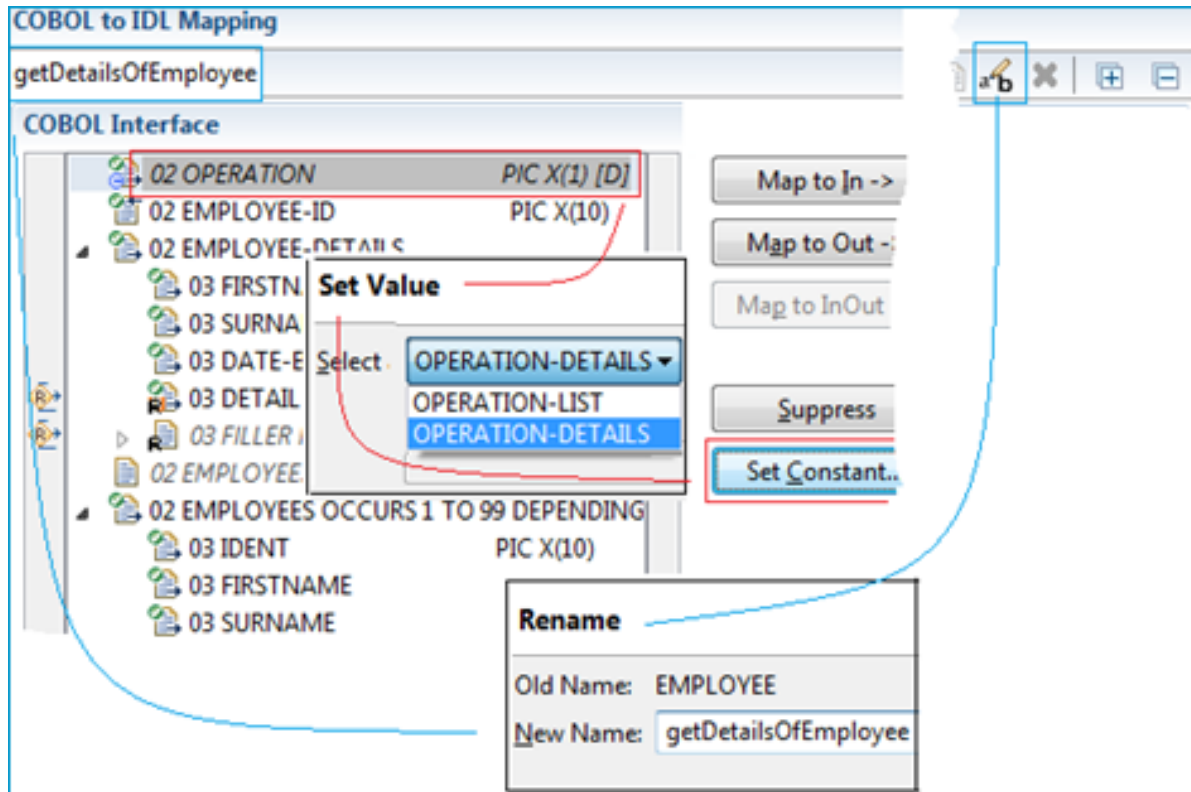
If required, a server mapping file with the same name as the IDL file but the extension .cvm may be generated too, for technical reasons. This file must always be stored together with the concomitant IDL file. For more information, see *Server Mapping Files for COBOL* in the Designer documentation. By default, the interface name `EMPLOYEE` is taken from the COBOL server program name.


Extracting a COBOL Server - Modern Method with User-defined Mapping

EntireX allows you to extract all implemented interfaces of the COBOL server separately. Instead of a large `EMPLOYEE` interface, separate interfaces `getListOfEmployees` and `getDetailsOfEmployee` are extracted. Each interface contains required parameters; obsolete parameters for an interface are suppressed, improving its usability.

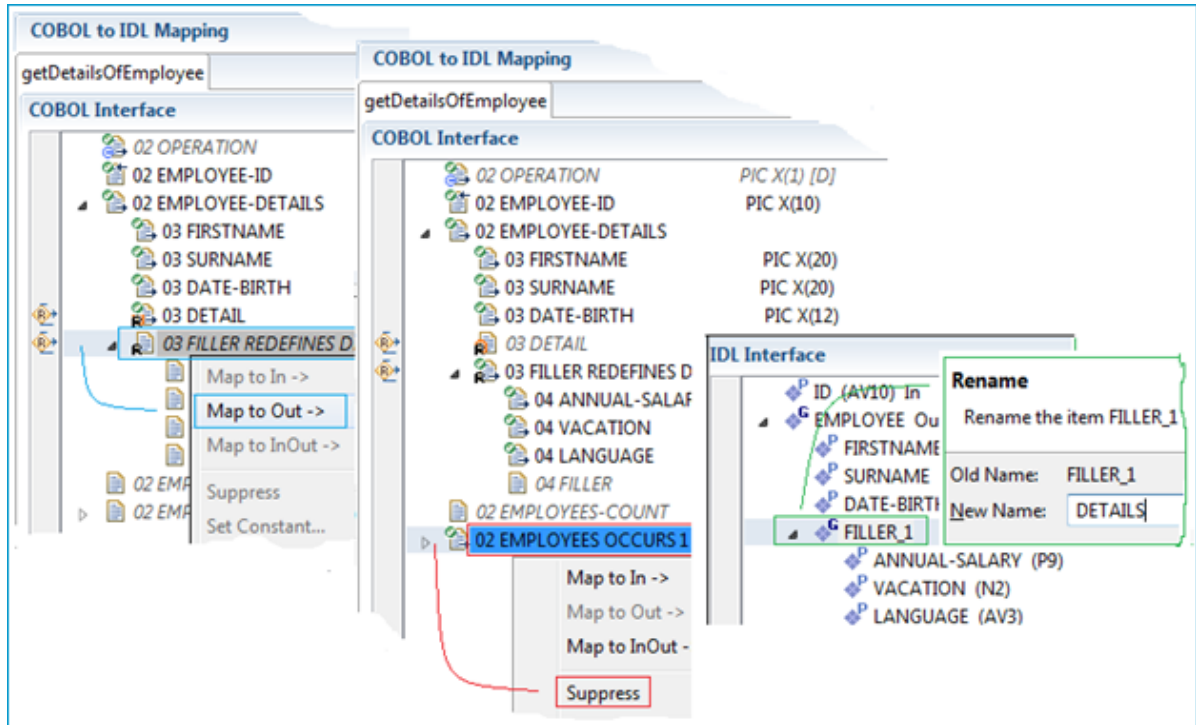
➤ To extract a COBOL server with a user-defined mapping

- 1 Shape `EMPLOYEE` to `getDetailsOfEmployee`:



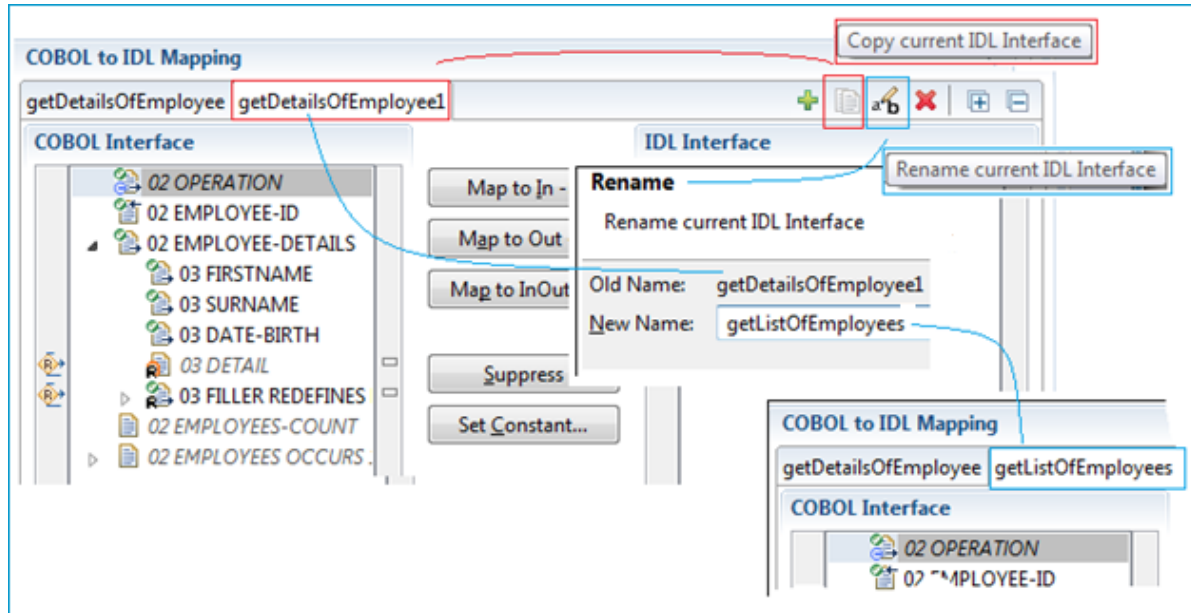
- Mark the `OPERATION` field in the **COBOL Interface** pane and press the **Set Constant** button. In the pop-up window, select `OPERATION-DETAILS` (red markers). The `OPERATION-DETAILS` value originates from COBOL level-88 data item (blue markers). If there is no COBOL level-88 enumeration type in the COBOL program defined, you can enter the function code manually in the pop-up window.
- By default the IDL interface gets the same name as your COBOL server: `EMPLOYEE`. Use the **Rename** button  from the toolbar of the COBOL Mapping Editor and change the IDL interface name to `getDetailsOfEmployee` (blue markers). This name is used as the IS service name later.

2 Continue shaping, making `getDetailsOfEmployee` easier to use:




- Use the context menu of the redefinition of the `DETAIL` field in the **COBOL Interface** pane, `FILLER REDEFINES DETAIL` and choose **Map to Out** (blue markers). A redefinition is a secondary layout (field types) of the same memory area. The redefinition contains more specific information (`ANNUAL-SALARY`, `VACATION`, etc.) than character buffer `DETAIL`. This is the reason why it is more useful in your interface.
- Use the context menu of `FILLER_1` in the IDL interface pane and rename `FILLER_1` to `DETAILS` (green markers) for a readable IS service field name later.
- The `DETAILS` function in the COBOL server does not make use of the `EMPLOYEES` field. Therefore you can leave it out of the interface: use the context menu of the `EMPLOYEES` field in the **COBOL Interface** pane and choose **Suppress** (red markers).


3 Shape `EMPLOYEE` to `getListOfEmployees` Function:



The COBOL Mapping Editor allows you to define multiple IDL interfaces to the same COBOL server:

- You create the additional IDL interface using the toolbar button **Copy current IDL interface**

 (red markers). A new tab `getDetailsOfEmployee1` is created. The previously extracted `getDetailsOfEmployee` interface still exists in the first tab. Once you reactivate the first tab, you will see the interface of `getDetailsOfEmployee` again.

- Use the **Rename** button  from the toolbar of the COBOL Mapping Editor and change the IDL interface name to `getListOfEmployees` (blue markers). This name is used as the IS service name later.

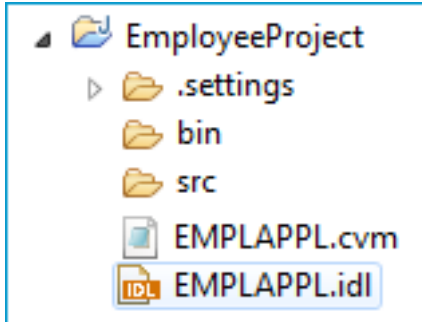
- For the `LIST` function you need the `OPERATION-LIST` value in the `OPERATION` field: Mark the `OPERATION` field in the **COBOL Interface** pane and press the **Set Constant** button. In the pop-up window, select `OPERATION-LIST` (similar to Step 1. above *Shape `EMPLOYEE` to `getDetailsOfEmployee`*; red markers).

- Use the context menu of the `EMPLOYEES` field in the **COBOL Interface** pane, `EMPLOYEES OCCURS 1 TO 99` and choose **Map to Out** (Similar to Step 2. above *Continue shaping, making `getDetailsOfEmployee` easier to use*; blue markers).

- Because the `EMPLOYEE` and `EMPLOYEE-DETAIL` fields are not used in the COBOL server `LIST` function, you leave them out in the IDL interface: use the context menu of `EMPLOYEE-DETAIL` field in the **COBOL Interface** pane and choose **Suppress** (similar to Step 2. above *Continue shaping, making `getDetailsOfEmployee` easier to use*; red markers). Do the same for the `EMPLOYEE-ID` field.

- 4 Press **Finish** to retrieve the extraction result in the form of a Software AG IDL file. At the same time, a server mapping file for COBOL (Designer file with extension `.cvm`) is created.

See *Server Mapping Files for COBOL* in the Designer documentation. The *Software AG IDL File* in the IDL Editor documentation describes the interfaces from the client point of view, while the server mapping file contains the mapping to the real COBOL server. Both of these files must be kept together and in sync, otherwise a call to the COBOL server may fail.



To summarize: We created two IDL interfaces. In the IDL file, these resulted in two IDL programs: `getListOfEmployees` and `getDetailsOfEmployee`. Both IDL programs were given readable names (Steps 1 and 3). Meaningful fields were kept, while superfluous fields were suppressed (Steps 1 and 3). The program `getDetailsOfEmployee` contains the redefined fields of parameter `DETAILS` (see below) mapped during extraction (Step 1).



Note: At runtime, the RPC client generated with the extracted interface will send data for the redesigned interfaces, while your COBOL server still expects `EMPLOYEE` data. The EntireX runtime transforms the incoming data stream from the RPC client, using the server mapping file.

```

program 'getDetailsOfEmployee' is
  define data parameter
    1 ID (AV10) In /* Original Name:EMPLOYEE-ID
    1 EMPLOYEE Out /* Original Name:EMPLOYEE-DETAILS
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
    2 DETAILS /* Original Name:FILLER_1
    3 ANNUAL-SALARY (P9)
    3 VACATION (N2)
    3 LANGUAGE (AV3)
  end-define
  program 'getListOfEmployees' is
    1 EMPLOYEES (/V99) Out
    2 IDENT (AV10)
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
  end-define

```


Comparison Chart: Fast-track versus User-defined Mapping

The following table highlights the differences when shaping an interface with a *user-defined mapping* compared with a *fast-path extraction*.

	Interface Shaping with User-defined Mapping	Fast-path Extraction
General	User-defined interface with dedicated mapping. For our example, the interfaces are small and tidy and more self-explanatory with long, readable interface names. They are easier to use with the hidden <code>OPERATION</code> field and the suppressed COBOL fields, which are not needed.	Automatic, quick result extraction with COBOL-like interfaces on IS. For our example, the IS interface <code>EMPLOYEE</code> matches exactly 1:1, field by field.
IS service(s)	Multiple small and handy IS services; each <code>OPERATION</code> code is mapped to a separate IS service.	One big IS service.
IS service name	Readable long name.	Short Subprogram name; up to 8 characters.
IS fields	Usage of Suppress and Set Constant reduces the message length. This keeps focus on relevant data items and keeps the client's interface clean. It may also improve performance.	The IS fields and COBOL server parameters match 1:1. As COBOL layout descriptions are sometimes used for many different purposes, irrelevant data items appear and clutter up the IS interface.
<code>OPERATION</code> parameter	Suppressed: the <code>OPERATION</code> parameter does not exist in the IS service as an IS field.	The <code>OPERATION</code> parameter exists in the IS service as an IS field and needs to be filled-in by the client endpoint.
<code>OPERATION</code> code	The <code>OPERATION</code> code is provided internally by EntireX runtime in the <code>OPERATION</code> field.	The <code>OPERATION</code> code needs to be specified in the IS service in the <code>OPERATION</code> field by the client endpoint.
<code>REDEFINE</code> parameters	Either the parameter that is redefined or one of its redefinitions is available as an IS field.	Only the parameter that is redefined is available as an IS field. Redefinitions of the parameter are not available as IS fields.

A user-defined mapping enables you also to define alternative mappings for COBOL data items (`REDEFINES`, tables etc.). These enable scenarios with COBOL servers where data exchange is not fully described by the `LINKAGE SECTION COBOL` data items. For more information refer to the IDL Extractor for COBOL documentation:

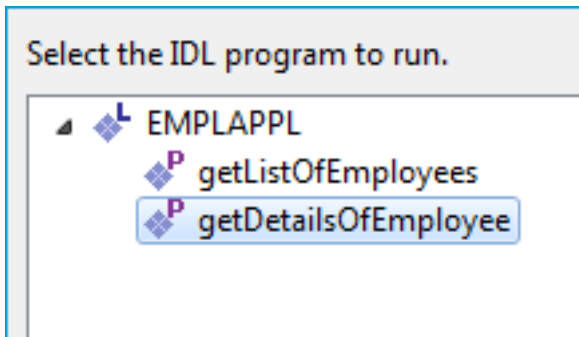
- *Mapping Editor IDL Interface Mapping Functions* if check-box **Input Message same as Output Message** was checked.
- *Mapping Editor IDL Interface Mapping Functions* if check-box **Input Message same as Output Message** was cleared.
- *User-defined Mapping*

Testing the Extraction Results

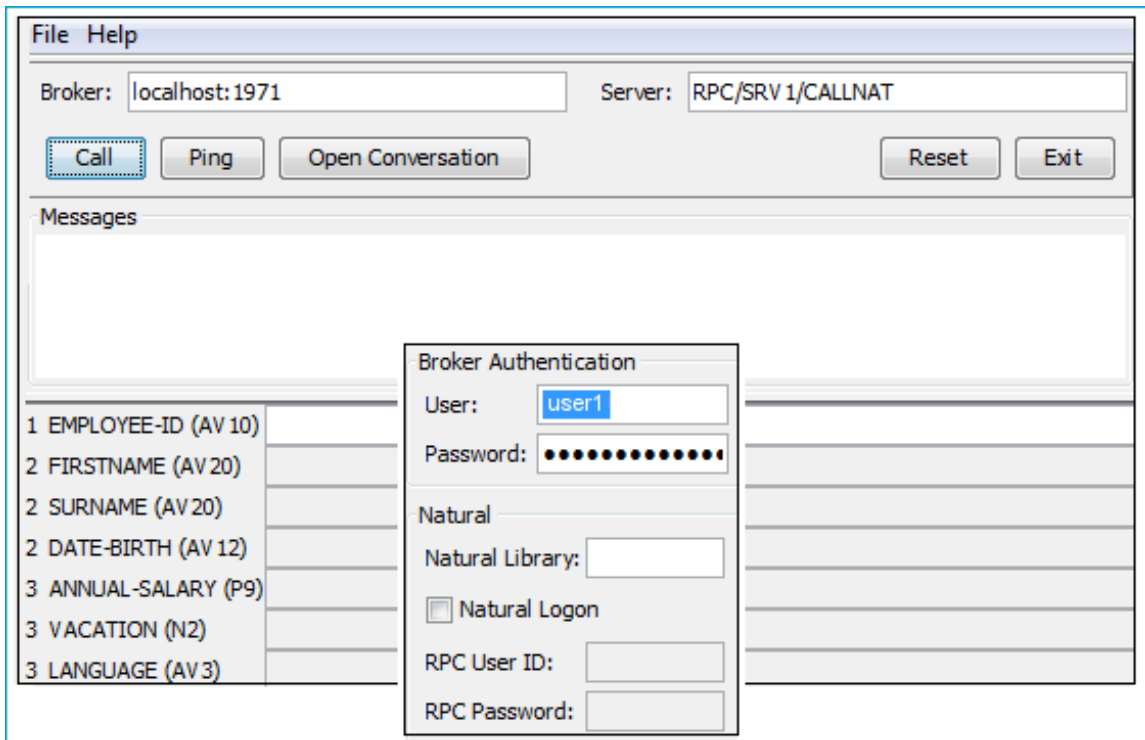
The following pictures use the extraction results described under *Extracting a COBOL Server - Modern Method with User-defined Mapping*.

> To test the extraction results (optional)

- 1 You can test the results of the extraction operation and the REST server back end, using the EntireX IDL Tester. From the context menu of the IDL file in the Designer, choose **Software AG IDL Tester**.



- 2 Select `getDetailsOfEmployee`.



Note that the broker and server parameters contain the explicit route to call the server program, and you can optionally ping the connection from this client. With the **File > Options** dialog, you can set:

- **User** and **Password** for *broker* authentication
- **RPC User ID** and **RPC Password** for *server* authentication

See *EntireX IDL Tester* in the Designer documentation.

- 3 Check the Integration Server log, the EntireX Adapter log or the RPC logs. Applies to all connection methods.

Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server

This section describes your first steps to create a new Integration Server connection. This is described in more detail under *Using the Integration Server Wrapper*, for example working with existing Integration Server connections. This section covers the following topics:

- [Step 1: Start the Integration Server Wrapper Wizard](#)
- [Step 2: Create a New Integration Server Connection](#)
- [Step 3: Select the Connection Type](#)
- [Step 4: Define Adapter Services for an RPC Connection](#)
- [Step 5: Create or Update a REST Resource](#)
- [Step 6: Edit the REST Resource](#)

Step 1: Start the Integration Server Wrapper Wizard

➤ To start the Integration Server Wrapper wizard

- 1 In the context menu of a Software AG IDL file, choose **Integration Server > Generate web-Methods IS Connection**.

This starts the wizard with a list of existing Integration Server Wrapper connections.



Note: If the selected IDL file is not valid because of a syntax error, an error dialog comes up and the wizard does not start.

- 2 Continue with *Step 2: Create a New Integration Server Connection*.

Step 2: Create a New Integration Server Connection

Define New Integration Server Connection

Add host:port, user and password for a new Integration Server

Server:

User:

Password:

☐
Use secure connection

➤ To create a new Integration Server connection

- 1 Define the new Integration Server connection on the wizard page.



Notes:

1. The only required field is **Server**. Enter the hostname of the Integration Server including an optional port number. If no port number is specified, port number defaults to "5555". The **Integration Server Authentication** can be passed with the **User** and **Password** fields.
 2. Optional settings are for secure connections. The **Truststore for HTTPS** contains all signed certificates and must be a valid truststore.
 3. The check box **Verify host name** checks that the hostname is entered in the stored certificate.
 4. When the Integration Server has **Client Authentication** enabled, you can specify your **Keystore** file and keystore **Password**.
 5. For managing Integration Server connections, see [Preferences](#).
- 2 Choose **Next** and continue with *Step 3: Select the Connection Type*.

Step 3: Select the Connection Type

☒ Create a new EntireX Adapter connection of type: **EntireX RPC Connection**

☐ Update Adapter Services or Listeners to an existing Integration Server connection:

List of Connections

Connection Name	Package Name	Connection Type	Enabled
Employees:EmployeesConnection	Employees	EntireX RPC Connection	Yes

Total: 1

☒ Map Software AG IDL data types to String

☒ Create or Update REST resource

> To create a new connection

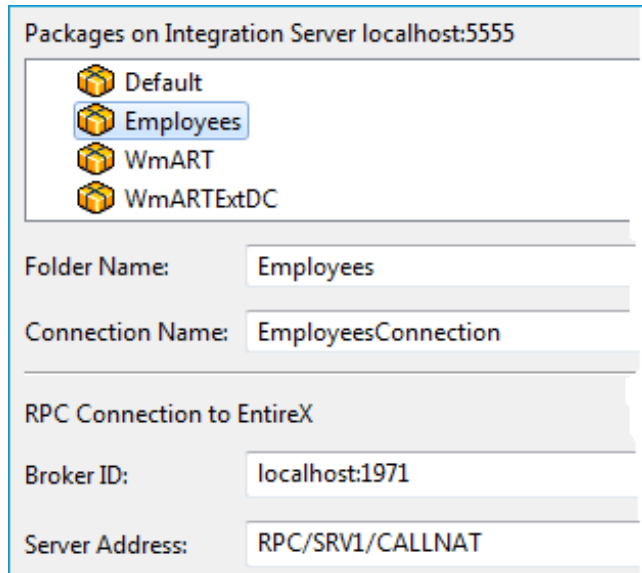
- 1 Select a connection type from the drop down list. Connection types are described under *EntireX Adapter Connections* in the EntireX Adapter documentation and *Introduction to the Integration Server Wrapper*.



Note: The list of connection types is filtered: connection types that require a license are only shown if a corresponding license file is available. Reliable RPC connections are only shown if all IDL programs contain only IN parameters. Also, if a server mapping file is available, only those connection types that support the interface type specified in the server mapping file are shown.

- 2 Check the box **Create or Update REST resource**.
- 3 Click **Next** and continue with *Step 4: Define Adapter Services for an RPC Connection*.

Step 4: Define Adapter Services for an RPC Connection



Packages on Integration Server localhost:5555

- Default
- Employees**
- WmART
- WmARTEExtDC

Folder Name: Employees

Connection Name: EmployeesConnection

RPC Connection to EntireX

Broker ID: localhost:1971

Server Address: RPC/SRV1/CALLNAT

> To create a connection and related adapter services

- 1 Select an existing package or create a new package for the created objects.
- 2 Define a folder name. If the folder does not exist, it will be created.
- 3 Define a connection name.
- 4 Define the parameters of the connection type. For details, see *Connection Parameters for RPC Connections* in the EntireX Adapter documentation.

As a result, the folder will contain the connection and the adapter services (one for each IDL program). The name of a service is the same as the respective IDL program.

The default settings for the adapter services are:

- the **Default** package; if not available, the first package
- the IDL library name for the **Folder Name**
- the IDL library name with the suffix "Connection" for the **Connection Name**



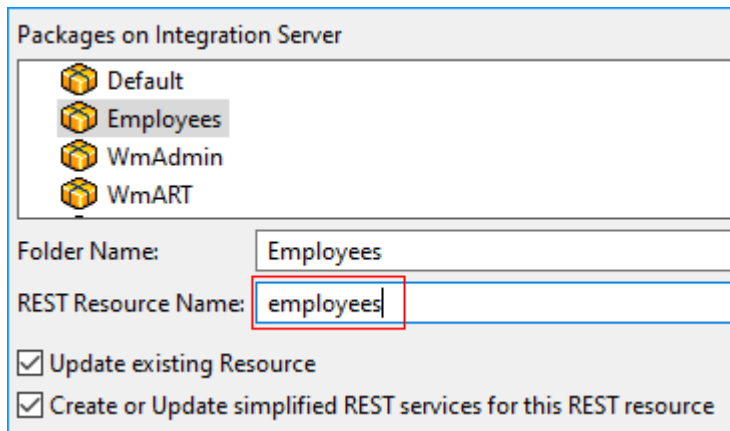
Note: When creating a connection, a package dependency is added such that the selected package depends on webMethods EntireX (the package `WmEntireX`) with the version currently used.

Step 5: Create or Update a REST Resource

➤ To create or update a REST resource

Make sure checkbox **Create or Update REST resource** is checked.

- 1 Select a package.



Packages on Integration Server

- Default
- Employees
- WmAdmin
- WmART

Folder Name: Employees

REST Resource Name: employees

☒ Update existing Resource

☒ Create or Update simplified REST services for this REST resource

- 2 Define a folder name. If the folder does not exist, it will be created.
- 3 Define the REST Resource name.
- 4 Check **Update existing Resource** if you want to add additional services to an existing resource.
- 5 Check **Create or Update simplified REST services for this REST resource** if you want to generate additional services with a simplified REST API. This allows you to use a simple HTTP-GET method without a payload.

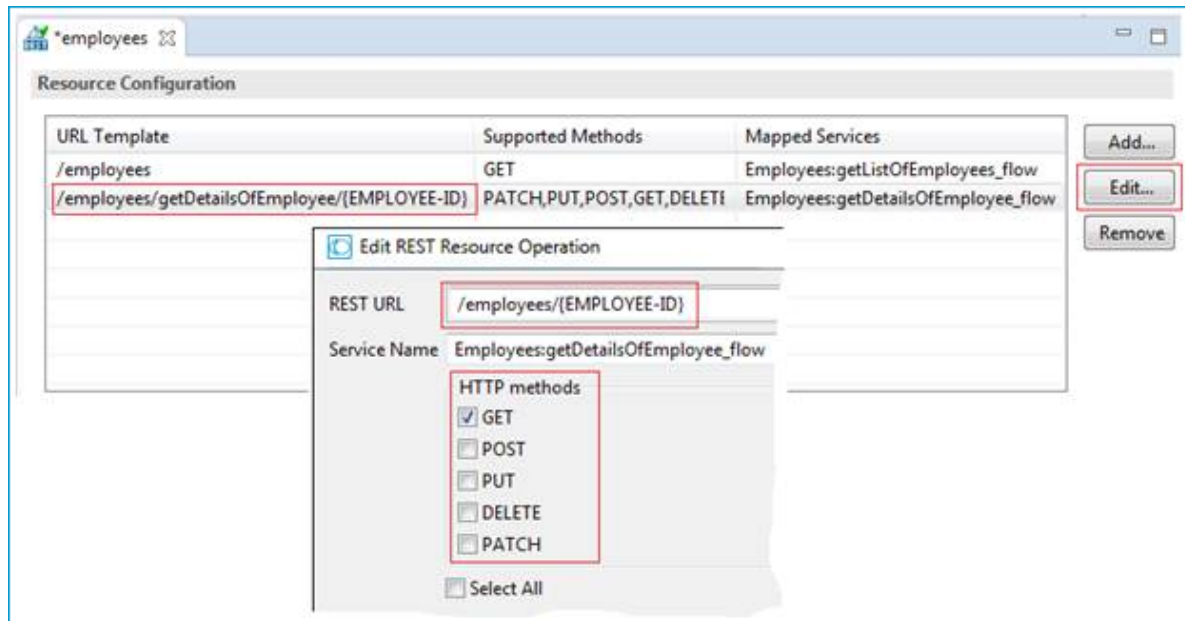


Note: This applies only to APIs where input signature parameters do not contain arrays.

Step 6: Edit the REST Resource

➤ To edit the REST resource

- 1 Switch to the **Service Development** perspective of Software AG Designer.



- 2 Refresh the package where you generated the REST Resource in the previous step.
- 3 Open the REST Resource.
- 4 Edit the REST URLs and HTTP Methods to match the functionality of your application.

For more information see:

- *REST Developer's Guide* in the webMethods Integration Server documentation for information on using the generated REST resources
- *REST Developer's Guide* in the webMethods Integration Server documentation

Task 3: Execute the Call from the REST Client to COBOL

Use an appropriate REST client, specify URL, Method and Headers and invoke your REST resource:

URL http://localhost:5555/restv2/employees

Method GET

Headers Accept application/json

200 OK

```
{
  "errorCode": "00000000",
  "errorFlag": "false",
  "response": {
    "EMPLOYEES-LIST": {
      "EMPLOYEES": [
        {
          "IDENT": "11100102",
          "FIRSTNAME": "Edgar",
          "SURNAME": "Schindler",
          "DATE-BIRTH": "Dec 4, 1962"
        },
        {
          "IDENT": "11100105",
          "FIRSTNAME": "Christian",
          "SURNAME": "Schirm",
          "DATE-BIRTH": "Mar 15, 1961"
        },
        {
          "IDENT": "11100106",
          "FIRSTNAME": "Rainer",
          "SURNAME": "Schmitt",
          "DATE-BIRTH": "Feb 13, 1955"
        },
        {
          "IDENT": "11100107",
          "FIRSTNAME": "Helga",
          "SURNAME": "Schmidt",
          "DATE-BIRTH": "May 26, 1961"
        }
      ]
    }
  }
}
```

URL http://localhost:5555/restv2/employees/111000105

Method GET

Headers Accept application/json

200 OK

```
{
  "errorCode": "00000000",
  "errorFlag": "false",
  "response": {
    "EMPLOYEE-DETAILS": {
      "FIRSTNAME": "Christian",
      "SURNAME": "Schirm",
      "DATE-BIRTH": "Mar 15, 1961",
      "DETAIL": {
        "ANNUAL-SALARY": "68000",
        "VACATION": "21",
        "LANGUAGE": "GER"
      }
    }
  }
}
```

12

Calling COBOL Large Buffer (Minimal Footprint using CICS Socket Listener) on z/OS CICS from REST

■ Introduction	205
■ What do I need to Install for this Scenario?	207
■ Task 1: Extract the Interface of a COBOL Server	208
■ Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server	221
■ Task 3: Execute the Call from the REST Client to COBOL	227

This scenario uses the tools IDL Extractor for COBOL and Integration Server Wrapper of the Designer.

Introduction



- ❶ Extract the interface of the COBOL server program. See *Using the IDL Extractor for COBOL - Overview*.
- ❷ Generate REST resources, connection and adapter services in Integration Server.
- ❸ Execute the call from the REST client to the COBOL server program.

This scenario makes the following important assumptions:

For Task 1:

- You have a working COBOL Large Buffer server. For illustration and examples on such a server, see *CICS with DFHCOMMAREA Large Buffer Interface* under *Introduction to the IDL Extractor for COBOL*.
- You have access to the related COBOL sources and copybooks as files on your local machine. The minimum requirement is the `DATA DIVISION` of the interface.



For Tasks 2 and 3:

- You have a REST client.
- To call the COBOL server program at runtime using the *EntireX CICS Socket Listener* connection method, you need to install the CICS Socket Listener within your CICS region: see *Preparing for CICS Socket Listener*. For configuration, see *Connection Parameters for CICS Socket Listener Connections*.



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.
- The CICS Socket Listener, which is installed
 - together with the RPC Server for CICS, see *Installing the RPC Server for CICS*, or
 - separately, see *EntireX CICS® Socket Listener* in the z/OS Installation documentation

Task 1: Extract the Interface of a COBOL Server

- [Introduction](#)
- [Sample COBOL Server used in the Examples](#)
- [Extracting a COBOL Server - Fast-track Method](#)
- [Extracting a COBOL Server - Modern Method with User-defined Mapping](#)
- [Comparison Chart: Fast-track versus User-defined Mapping](#)
- [Testing the Extraction Results](#)

Introduction

Follow the instructions for extracting COBOL, see *Using the IDL Extractor for COBOL - Overview* and choose *Scenario I: Create New IDL and Server Mapping Files* if this is your first extraction. Learn how to invoke the IDL Extractor for COBOL, create a COBOL Extractor Environment and about the possibilities to select the COBOL source.

The extraction process creates the following EntireX metafiles:

- IDL file. A Software AG IDL file contains definitions of the interface between client and server. See *Software AG IDL File* in the IDL Editor documentation in the IDL Editor documentation.
- Server mapping file (optional). The mapping file is a Designer file with extension `.cvm` that contains COBOL-specific mapping information. See *Server Mapping Files for COBOL* in the Designer documentation.

Sample COBOL Server used in the Examples

The following COBOL server is used to illustrate the features of the IDL Extractor for COBOL. Imagine an existing COBOL server called `EMPLOYEE`. It implements the access logic for a `LIST` and `DETAILS` function to a database view `EMPLOYEE`:


```

IDENTIFICATION DIVISION.
    PROGRAM-ID.      EMPLOYEE.
*****
*   Program Name    = EMPLOYEE
*   String Literal  = QUOTE
*****
01 DFHCOMMAREA.
    02 OPERATION                                PIC X(1).
        88 OPERATION-LIST                      VALUE 'L'.
        88 OPERATION-DETAILS                   VALUE 'D'.
    02 EMPLOYEE-ID                                PIC X(10).
    02 EMPLOYEE-DETAILS.
        03 FIRSTNAME                            PIC X(20).
        03 SURNAME                             PIC X(20).
        03 DATE-BIRTH                          PIC X(12).
        03 DETAILS                            PIC X(100).
        03 FILLER REDEFINES DETAILS.
            04 ANNUAL-SALARY                    PACKED-DECIMAL PIC S9(9).
            04 VACATION                        PIC S9(2).
            04 LANGUAGE                        PIC X(3).
            04 FILLER                          PIC X(90).
    02 EMPLOYEES-COUNT                          PIC 9(8) BINARY.
    02 EMPLOYEES OCCURS 1 TO 99 DEPENDING ON EMPLOYEES-COUNT.
        03 IDENT                              PIC X(10).
        03 FIRSTNAME                          PIC X(20).
        03 SURNAME                            PIC X(20).
        03 DATE-BIRTH                        PIC X(12).

```

Two approaches for extracting the COBOL server are described below:

- **Fast-track**

Learn how EntireX helps you to connect the COBOL Server with quick results.

- **User-defined Mapping**

The following features are among those available with a user-defined mapping:

- **Shape the interface to your COBOL server**

You can minimize the interface by suppressing COBOL server fields or by providing constant values. Renaming interfaces allows you to specify a readable long name. These two features together - constants and renaming of interfaces - are most powerful when multiple interfaces are implemented in a single COBOL server. Each function of the COBOL server triggered by an operation code or function code can be modelled as a separate REST resource method/operation.

■ **Select alternative mappings**

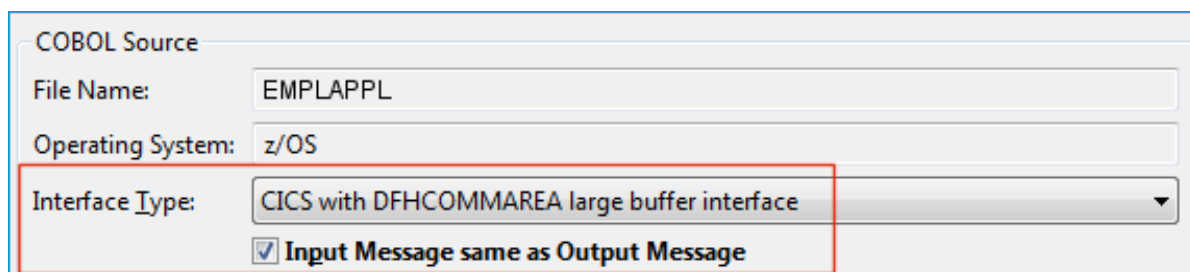
You can select alternative mappings for COBOL data items (REDEFINES, tables etc.) to enable scenarios with COBOL servers where data exchange is not fully described by the LINKAGE SECTION COBOL data items.

These approaches are described under [Fast-track Method](#) and [Modern Method](#) below.

Extracting a COBOL Server - Fast-track Method

➤ **To extract the COBOL Server**

- 1 Switch to the **EntireX** perspective.
- 2 Call the IDL Extractor for COBOL.
- 3 Set the correct interface type as described under *Step 4: Define the Extraction Settings and Start Extraction* under *Scenario I: Create New IDL and Server Mapping Files* in the IDL Extractor for COBOL documentation.



COBOL Source

File Name: EMPLAPPL

Operating System: z/OS

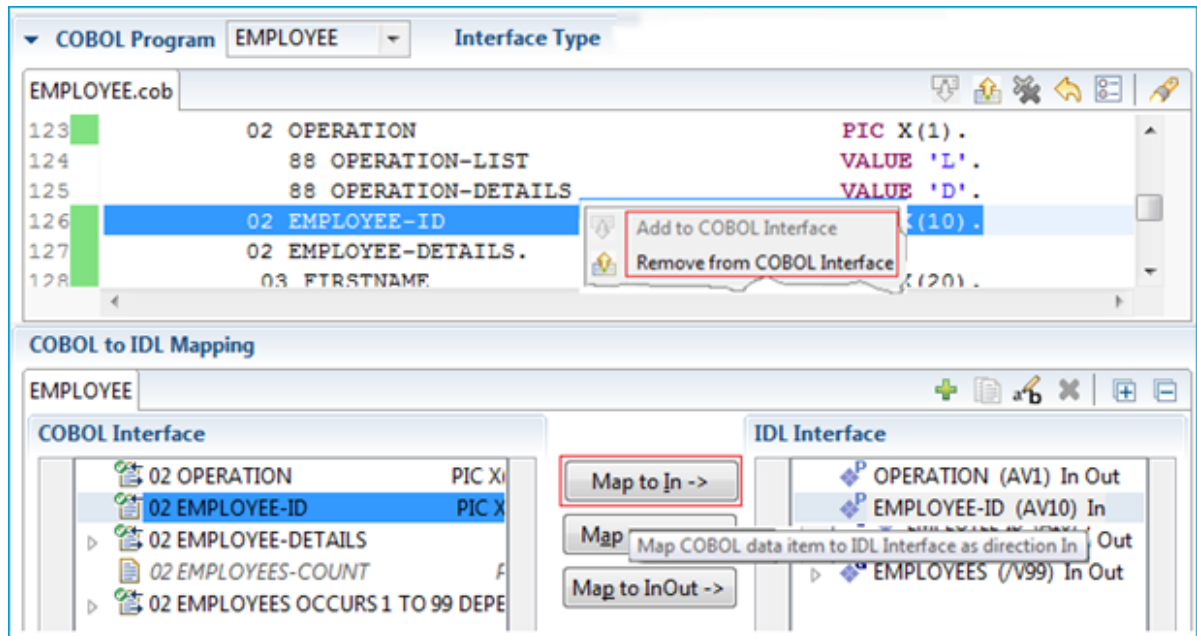
Interface Type: CICS with DFHCOMMAREA large buffer interface

☒ Input Message same as Output Message

- 4 Set the correct value for checkbox **Input Message same as Output Message**.
 - *Check* the checkbox if the COBOL data structure of the CICS input message is the same as the structure of the CICS output message, that is, there is one COBOL layout for both directions.
 - *Clear* the checkbox if the COBOL data structure of the CICS input message is different to the structure of the CICS output message, that is, the output overlays the input, which means there is a second COBOL layout for the reply.

For our example, check the check box, because the COBOL layout for input is the same as the COBOL layout for output.

- 5 Press **Next** to enter the COBOL Mapping Editor.



The purpose of the COBOL Mapping Editor is to map COBOL data items to IDL (Interface Definition Language) parameters. It consists of three main areas:

- The upper window pane is the COBOL program source.
- The window pane on the left-hand side is the COBOL interface, consisting of COBOL data items representing the parameters in your COBOL server program.
- The window pane on the right-hand side is the IDL interface consisting of IDL parameters.

What you select depends on check-box **Input Message same as Output Message**:

- If there is a single COBOL data structure of the CICS message for input and output - check box **Input Message same as Output Message** is checked - you select a single COBOL data structure in the COBOL Mapping Editor. This structure is used for COBOL input as well as for COBOL output. For more information in this case see *Mapping Editor User Interface* in the IDL Extractor for COBOL documentation.
- If there is a second COBOL data structure for the CICS output message - check box **Input Message same as Output Message** is cleared - you will select two COBOL data structures in the COBOL Mapping Editor. One is used for COBOL input and the other is for COBOL output. For more information in this case see *Mapping Editor User Interface* in the IDL Extractor for COBOL documentation.

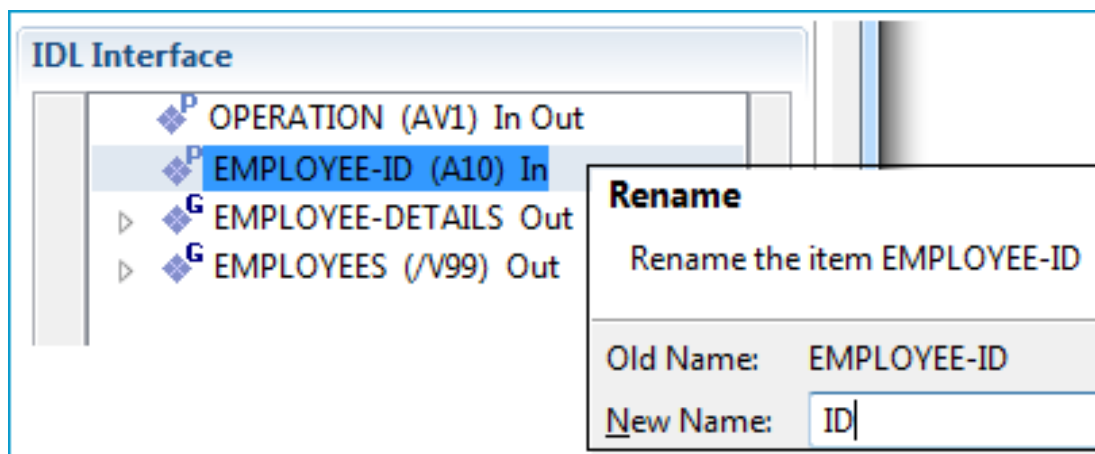
COBOL extraction is a two-step process:

- First, select the COBOL parameters using the **Add to** and **Remove from COBOL Interface** from the context menu on the COBOL data items in the upper window pane (COBOL source); in many situations the COBOL interface is detected by the Mapping Editor directly and you can omit this step.

- Then map the COBOL parameters to IDL directions using the **Map to In/Out/InOut** buttons. COBOL parameters do not have directions, they are always INOUT. This step can be left out for a quick test in non-production scenarios; for real production it is recommended to set IDL directions to reduce data transfer. You also have the option to assign a meaningful name to an IDL parameter if the original COBOL data item is not self-explanatory.

For our example:

- Set **Map to In** for COBOL data item `EMPLOYEE-ID` as IDL direction (see screenshot in previous step). For `EMPLOYEES-DETAILS` and `EMPLOYEES` perform the similar **Map to Out**.
- Use the function **Rename** from the context menu on the IDL parameters in the **IDL Interface** pane to rename COBOL parameter `EMPLOYEES-ID` to IDL parameter `ID`. Do the same for `EMPLOYEE-DETAILS`: Rename to `EMPLOYEE`.



- 6 Press **Finish**. The default extraction settings ensure you get useful results. The outcome is a Software AG IDL file (interface definition language), a Designer file with extension `.idl`:

```
library 'EMPLAPPL' is
program 'EMPLOYEE' is
  define data parameter
    1 OPERATION (AV1) In Out
    1 ID (AV10) In /* Original Name:EMPLOYEE-ID
    1 EMPLOYEE Out /* Original Name:EMPLOYEE-DETAILS
      2 FIRSTNAME (AV20)
      2 SURNAME (AV20)
      2 DATE-BIRTH (AV12)
      2 DETAILS (AV100)
    1 EMPLOYEES (/V99) Out
      2 IDENT (AV10)
      2 FIRSTNAME (AV20)
      2 SURNAME (AV20)
      2 DATE-BIRTH (AV12)
  end-define
```

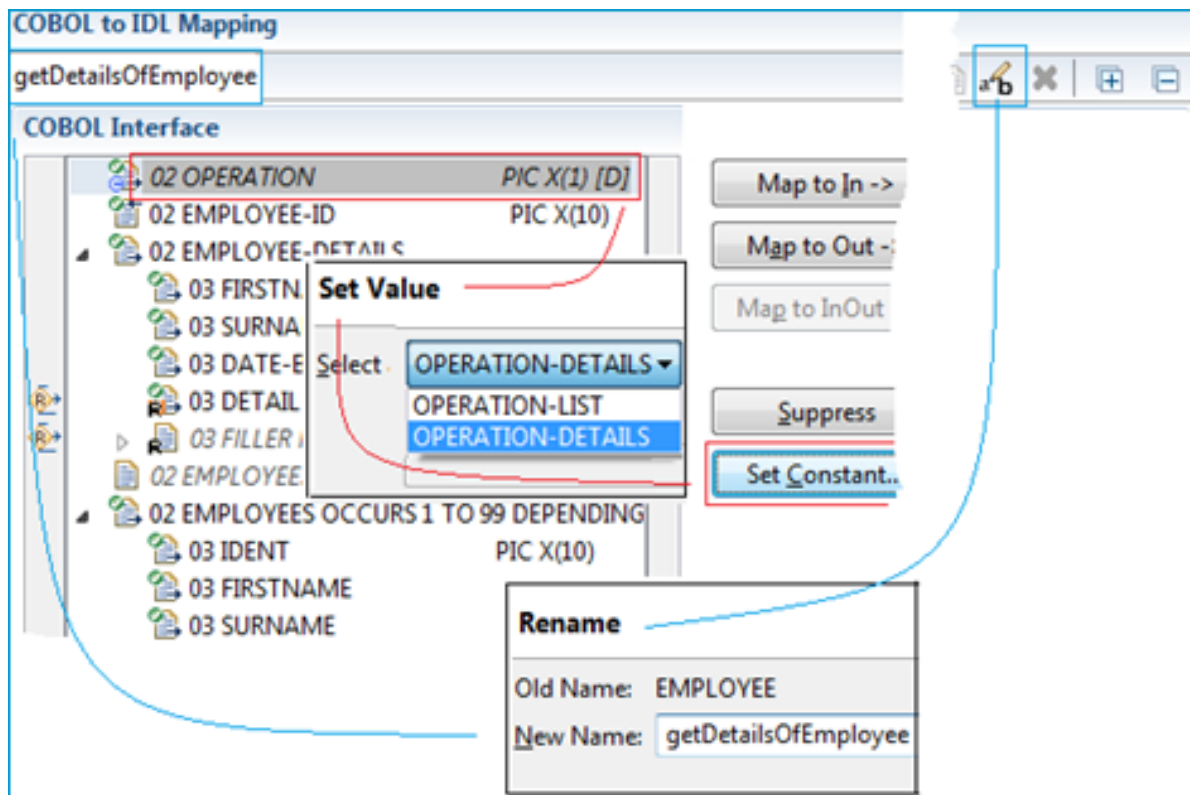
If required, a server mapping file with the same name as the IDL file but the extension .cvm may be generated too, for technical reasons. This file must always be stored together with the concomitant IDL file. For more information, see *Server Mapping Files for COBOL* in the Designer documentation. By default, the interface name `EMPLOYEE` is taken from the COBOL server program name.


Extracting a COBOL Server - Modern Method with User-defined Mapping

EntireX allows you to extract all implemented interfaces of the COBOL server separately. Instead of a large `EMPLOYEE` interface, separate interfaces `getListOfEmployees` and `getDetailsOfEmployee` are extracted. Each interface contains required parameters; obsolete parameters for an interface are suppressed, improving its usability.

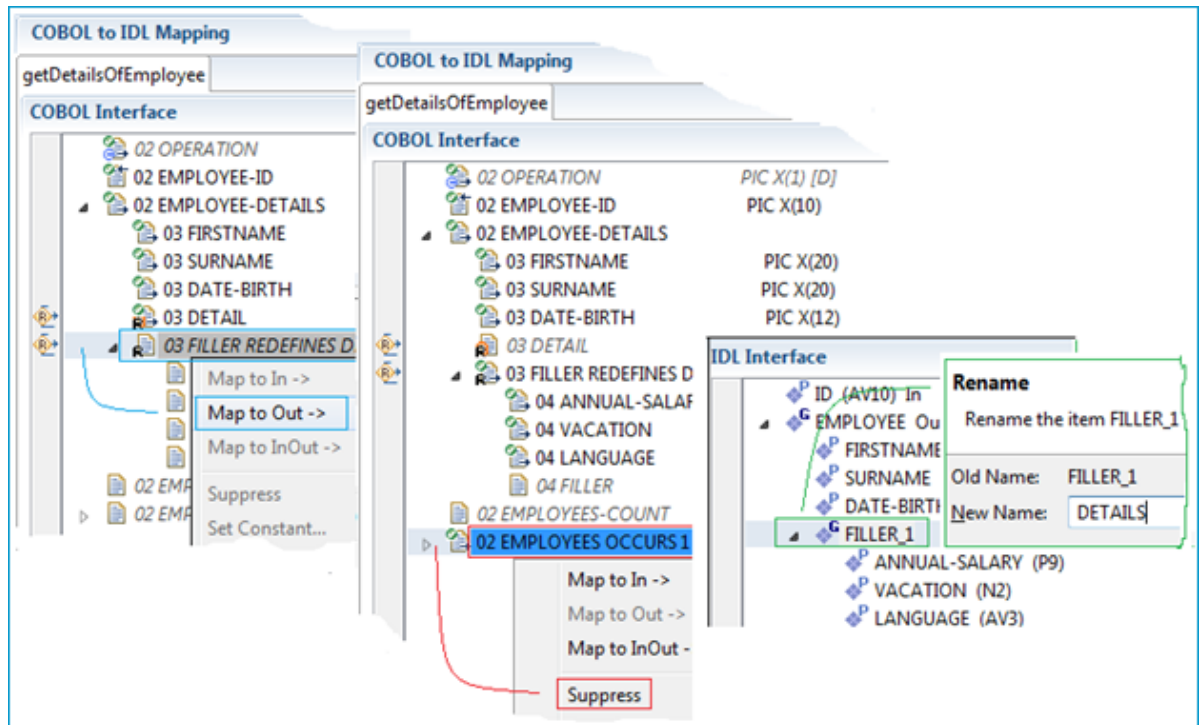
» To extract a COBOL server with a user-defined mapping

- 1 Shape `EMPLOYEE` to `getDetailsOfEmployee`:



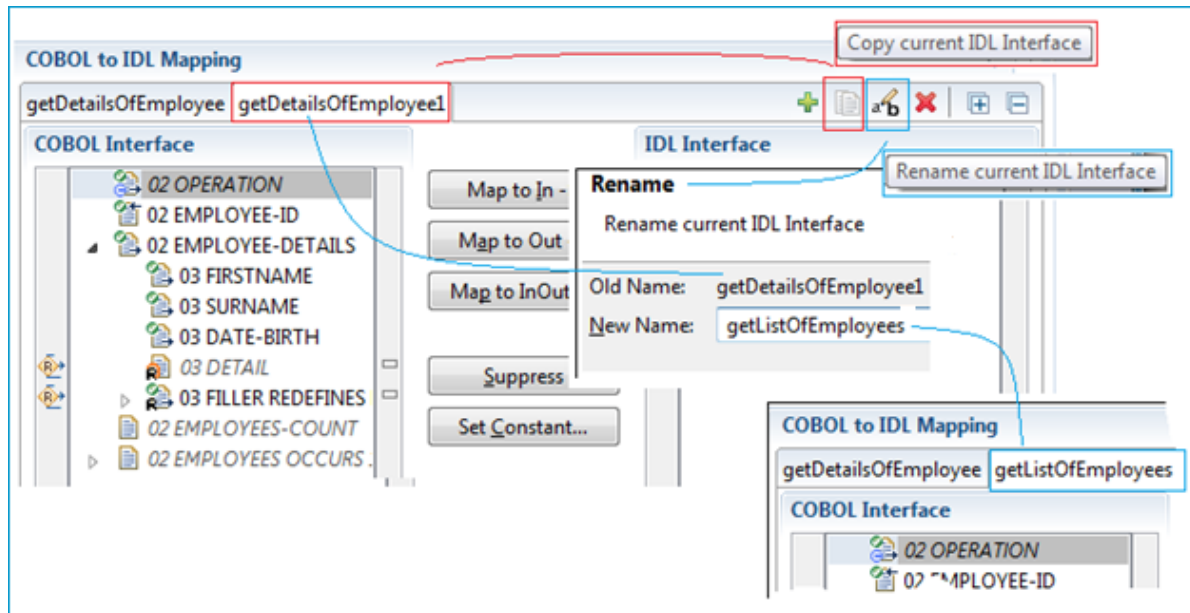
- Mark the `OPERATION` field in the **COBOL Interface** pane and press the **Set Constant** button. In the pop-up window, select `OPERATION-DETAILS` (red markers). The `OPERATION-DETAILS` value originates from COBOL level-88 data item (blue markers). If there is no COBOL level-88 enumeration type in the COBOL program defined, you can enter the function code manually in the pop-up window.
- By default the IDL interface gets the same name as your COBOL server: `EMPLOYEE`. Use the **Rename** button  from the toolbar of the COBOL Mapping Editor and change the IDL interface name to `getDetailsOfEmployee` (blue markers). This name is used as the IS service name later.

2 Continue shaping, making `getDetailsOfEmployee` easier to use:




- Use the context menu of the redefinition of the `DETAIL` field in the **COBOL Interface** pane, `FILLER REDEFINES DETAIL` and choose **Map to Out** (blue markers). A redefinition is a secondary layout (field types) of the same memory area. The redefinition contains more specific information (`ANNUAL-SALARY`, `VACATION`, etc.) than character buffer `DETAIL`. This is the reason why it is more useful in your interface.
- Use the context menu of `FILLER_1` in the IDL interface pane and rename `FILLER_1` to `DETAILS` (green markers) for a readable IS service field name later.
- The `DETAILS` function in the COBOL server does not make use of the `EMPLOYEES` field. Therefore you can leave it out of the interface: use the context menu of the `EMPLOYEES` field in the **COBOL Interface** pane and choose **Suppress** (red markers).


3 Shape `EMPLOYEE` to `getListOfEmployees` Function:



The COBOL Mapping Editor allows you to define multiple IDL interfaces to the same COBOL server:

- You create the additional IDL interface using the toolbar button **Copy current IDL interface**

 (red markers). A new tab `getDetailsOfEmployee1` is created. The previously extracted `getDetailsOfEmployee` interface still exists in the first tab. Once you reactivate the first tab, you will see the interface of `getDetailsOfEmployee` again.

- Use the **Rename** button  from the toolbar of the COBOL Mapping Editor and change the IDL interface name to `getListOfEmployees` (blue markers). This name is used as the IS service name later.

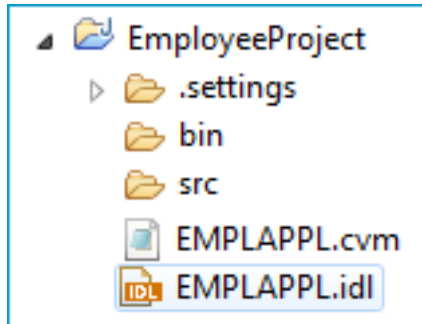
- For the `LIST` function you need the `OPERATION-LIST` value in the `OPERATION` field: Mark the `OPERATION` field in the **COBOL Interface** pane and press the **Set Constant** button. In the pop-up window, select `OPERATION-LIST` (similar to Step 1. above *Shape EMPLOYEE to getDetailsOfEmployee*; red markers).

- Use the context menu of the `EMPLOYEES` field in the **COBOL Interface** pane, `EMPLOYEES OCCURS 1 TO 99` and choose **Map to Out** (Similar to Step 2. above *Continue shaping, making getDetailsOfEmployee easier to use*; blue markers).

- Because the `EMPLOYEE` and `EMPLOYEE-DETAIL` fields are not used in the COBOL server `LIST` function, you leave them out in the IDL interface: use the context menu of `EMPLOYEE-DETAIL` field in the **COBOL Interface** pane and choose **Suppress** (similar to Step 2. above *Continue shaping, making getDetailsOfEmployee easier to use*; red markers). Do the same for the `EMPLOYEE-ID` field.

- 4 Press **Finish** to retrieve the extraction result in the form of a Software AG IDL file. At the same time, a server mapping file for COBOL (Designer file with extension `.cvm`) is created.

See *Server Mapping Files for COBOL* in the Designer documentation. The *Software AG IDL File* in the IDL Editor documentation describes the interfaces from the client point of view, while the server mapping file contains the mapping to the real COBOL server. Both of these files must be kept together and in sync, otherwise a call to the COBOL server may fail.



To summarize: We created two IDL interfaces. In the IDL file, these resulted in two IDL programs: `getListOfEmployees` and `getDetailsOfEmployee`. Both IDL programs were given readable names (Steps 1 and 3). Meaningful fields were kept, while superfluous fields were suppressed (Steps 1 and 3). The program `getDetailsOfEmployee` contains the redefined fields of parameter `DETAILS` (see below) mapped during extraction (Step 1).



Note: At runtime, the RPC client generated with the extracted interface will send data for the redesigned interfaces, while your COBOL server still expects `EMPLOYEE` data. The EntireX runtime transforms the incoming data stream from the RPC client, using the server mapping file.

```

program 'getDetailsOfEmployee' is
  define data parameter
    1 ID (AV10) In /* Original Name:EMPLOYEE-ID
    1 EMPLOYEE Out /* Original Name:EMPLOYEE-DETAILS
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
    2 DETAILS /* Original Name:FILLER_1
    3 ANNUAL-SALARY (P9)
    3 VACATION (N2)
    3 LANGUAGE (AV3)
  end-define
  program 'getListOfEmployees' is
    1 EMPLOYEES (/V99) Out
    2 IDENT (AV10)
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
  end-define
  
```

Comparison Chart: Fast-track versus User-defined Mapping

The following table highlights the differences when shaping an interface with a *user-defined mapping* compared with a *fast-path extraction*.

	Interface Shaping with User-defined Mapping	Fast-path Extraction
General	User-defined interface with dedicated mapping. For our example, the interfaces are small and tidy and more self-explanatory with long, readable interface names. They are easier to use with the hidden <code>OPERATION</code> field and the suppressed COBOL fields, which are not needed.	Automatic, quick result extraction with COBOL-like interfaces on IS. For our example, the IS interface <code>EMPLOYEE</code> matches exactly 1:1, field by field.
IS service(s)	Multiple small and handy IS services; each <code>OPERATION</code> code is mapped to a separate IS service.	One big IS service.
IS service name	Readable long name.	Short Subprogram name; up to 8 characters.
IS fields	Usage of Suppress and Set Constant reduces the message length. This keeps focus on relevant data items and keeps the client's interface clean. It may also improve performance.	The IS fields and COBOL server parameters match 1:1. As COBOL layout descriptions are sometimes used for many different purposes, irrelevant data items appear and clutter up the IS interface.
<code>OPERATION</code> parameter	Suppressed: the <code>OPERATION</code> parameter does not exist in the IS service as an IS field.	The <code>OPERATION</code> parameter exists in the IS service as an IS field and needs to be filled-in by the client endpoint.
<code>OPERATION</code> code	The <code>OPERATION</code> code is provided internally by EntireX runtime in the <code>OPERATION</code> field.	The <code>OPERATION</code> code needs to be specified in the IS service in the <code>OPERATION</code> field by the client endpoint.
<code>REDEFINE</code> parameters	Either the parameter that is redefined or one of its redefinitions is available as an IS field.	Only the parameter that is redefined is available as an IS field. Redefinitions of the parameter are not available as IS fields.

A user-defined mapping enables you also to define alternative mappings for COBOL data items (`REDEFINES`, tables etc.). These enable scenarios with COBOL servers where data exchange is not fully described by the `LINKAGE SECTION` COBOL data items. For more information refer to the IDL Extractor for COBOL documentation:

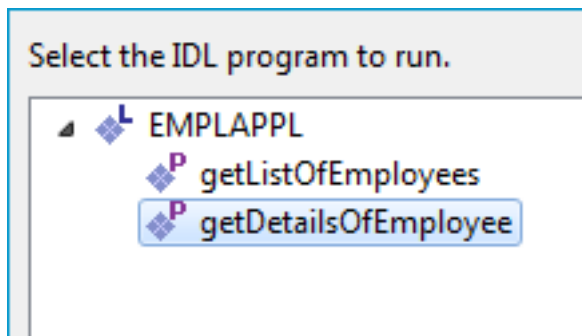
- *Mapping Editor IDL Interface Mapping Functions* if check-box **Input Message same as Output Message** was checked.
- *Mapping Editor IDL Interface Mapping Functions* if check-box **Input Message same as Output Message** was cleared.
- *User-defined Mapping*

Testing the Extraction Results

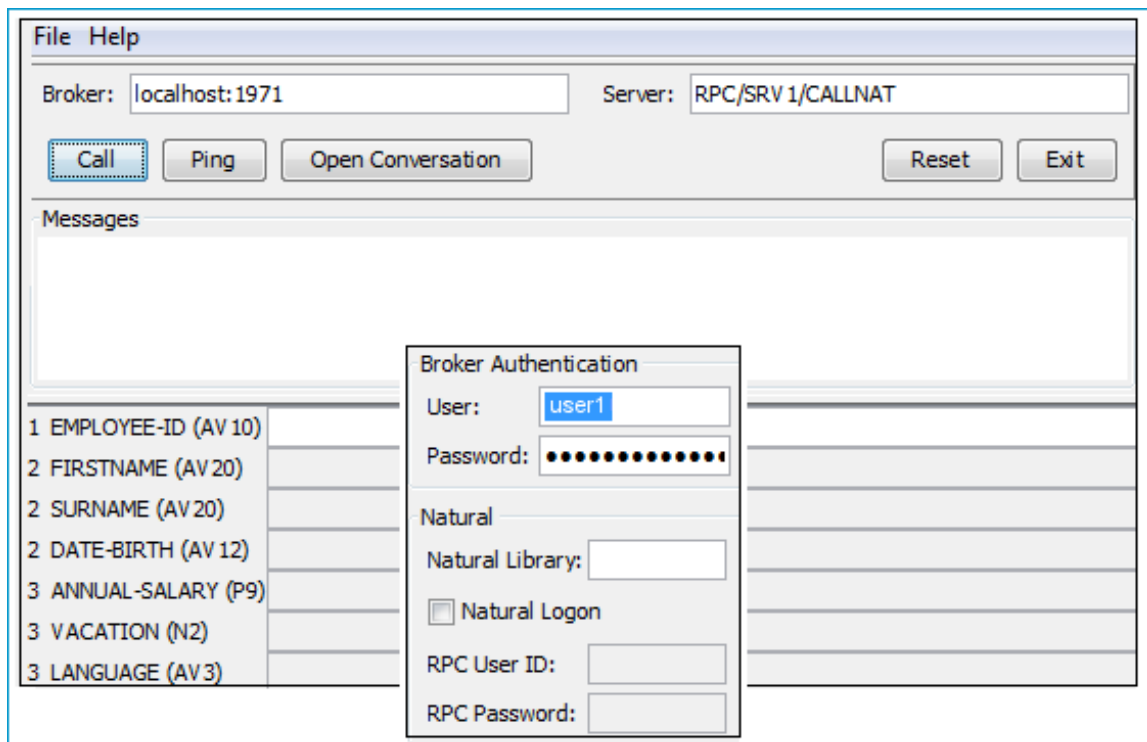
The following pictures use the extraction results described under *Extracting a COBOL Server - Modern Method with User-defined Mapping*.

> To test the extraction results (optional)

- 1 You can test the results of the extraction operation and the REST server back end, using the EntireX IDL Tester. From the context menu of the IDL file in the Designer, choose **Software AG IDL Tester**.



- 2 Select `getDetailsOfEmployee`.



Note that the broker and server parameters contain the explicit route to call the server program, and you can optionally ping the connection from this client. With the **File > Options** dialog, you can set:

- **User** and **Password** for *broker* authentication
- **RPC User ID** and **RPC Password** for *server* authentication

See *EntireX IDL Tester* in the Designer documentation.

- 3 Check the Integration Server log, the EntireX Adapter log or the RPC logs. Applies to all connection methods.

Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server

This section describes your first steps to create a new Integration Server connection. This is described in more detail under *Using the Integration Server Wrapper*, for example working with existing Integration Server connections. This section covers the following topics:

- [Step 1: Start the Integration Server Wrapper Wizard](#)
- [Step 2: Create a New Integration Server Connection](#)
- [Step 3: Select the Connection Type](#)
- [Step 4: Define Adapter Services for a CICS Socket Listener Connection](#)
- [Step 5: Create or Update a REST Resource](#)
- [Step 6: Edit the REST Resource](#)

Step 1: Start the Integration Server Wrapper Wizard

➤ To start the Integration Server Wrapper wizard

- 1 In the context menu of a Software AG IDL file, choose **Integration Server > Generate web-Methods IS Connection**.

This starts the wizard with a list of existing Integration Server Wrapper connections.



Note: If the selected IDL file is not valid because of a syntax error, an error dialog comes up and the wizard does not start.

- 2 Continue with *Step 2: Create a New Integration Server Connection*.

Step 2: Create a New Integration Server Connection

Define New Integration Server Connection
Add host:port, user and password for a new Integration Server

Server:

User:

Password:

☐ Use secure connection

➤ To create a new Integration Server connection

- 1 Define the new Integration Server connection on the wizard page.



Notes:

1. The only required field is **Server**. Enter the hostname of the Integration Server including an optional port number. If no port number is specified, port number defaults to "5555". The **Integration Server Authentication** can be passed with the **User** and **Password** fields.
 2. Optional settings are for secure connections. The **Truststore for HTTPS** contains all signed certificates and must be a valid truststore.
 3. The check box **Verify host name** checks that the hostname is entered in the stored certificate.
 4. When the Integration Server has **Client Authentication** enabled, you can specify your **Keystore** file and keystore **Password**.
 5. For managing Integration Server connections, see [Preferences](#).
- 2 Choose **Next** and continue with *Step 3: Select the Connection Type*.

Step 3: Select the Connection Type

☒ Create a new EntireX Adapter connection of type CICS Socket Listener Connection

☐ Update Adapter Services or Listeners to an existing Integration Server connection:

List of Connections

Connection Name	Package Name	Connection Type	Enabled
Employees:EmployeesConnection	Employees	EntireX RPC Connection	Yes

Total: 1

☒ Map Software AG IDL data types to String

☒ Create or Update REST resource

➤ To create a new connection

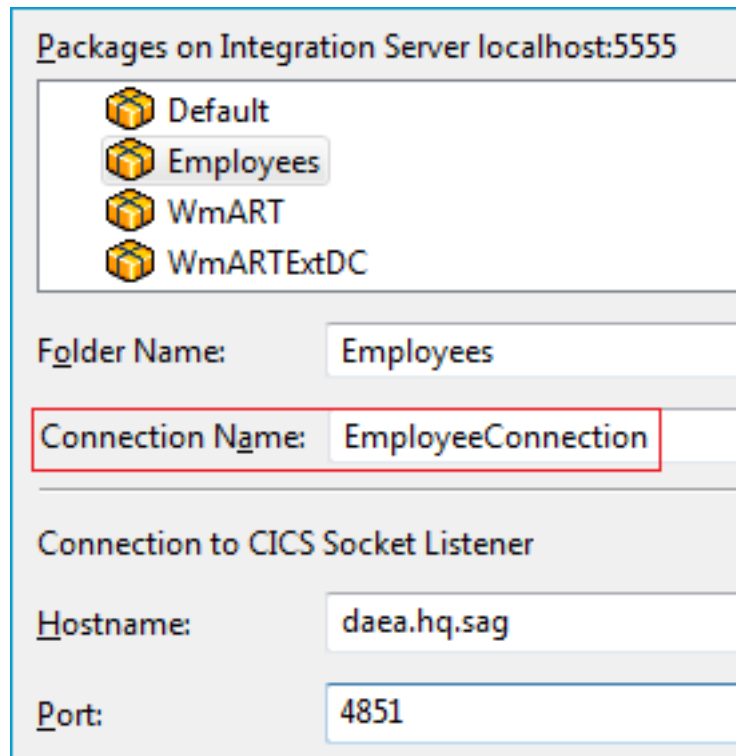
- 1 Select a connection type from the drop down list. Connection types are described under *EntireX Adapter Connections* in the EntireX Adapter documentation and *Introduction to the Integration Server Wrapper*.



Note: The list of connection types is filtered: connection types that require a license are only shown if a corresponding license file is available. Reliable RPC connections are only shown if all IDL programs contain only IN parameters. Also, if a server mapping file is available, only those connection types that support the interface type specified in the server mapping file are shown.

- 2 Check the box **Create or Update REST resource**.
- 3 Click **Next** and continue with *Step 4: Define Adapter Services for a CICS Socket Listener Connection*.

Step 4: Define Adapter Services for a CICS Socket Listener Connection



Packages on Integration Server localhost:5555

- Default
- Employees
- WmART
- WmARTEExtDC

Folder Name: Employees

Connection Name: EmployeeConnection

Connection to CICS Socket Listener

Hostname: daea.hq.sag

Port: 4851

➤ To create a connection and related adapter services

- 1 Select a package for the created objects.
- 2 Define a folder name. If the folder does not exist, it will be created.
- 3 Define a connection name.
- 4 Define the parameters of the connection type. For details, see *Connection Parameters for CICS Socket Listener Connections* in the EntireX Adapter documentation.

As a result, the folder will contain the connection and the adapter services (one for each IDL program). The name of a service is the same as the respective IDL program.

The default settings for the adapter services are:

- the **Default** package; if not available, the first package
- the IDL library name for the **Folder Name**
- the IDL library name with the suffix "Connection" for the **Connection Name**



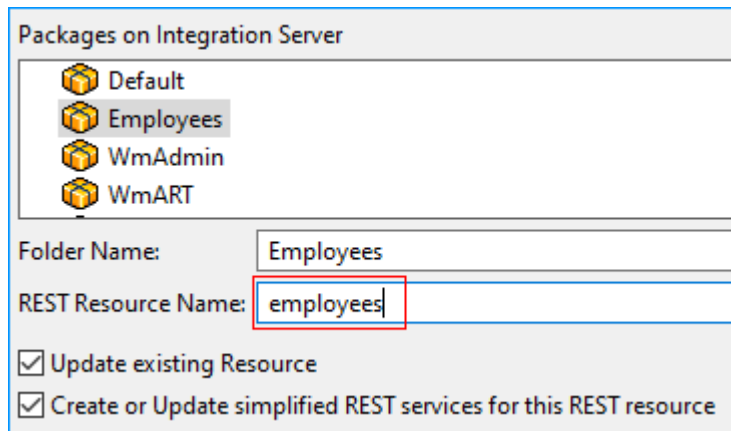
Note: When creating a connection, a package dependency is added such that the selected package depends on webMethods EntireX (the package `WmEntireX`) with the version currently used.

Step 5: Create or Update a REST Resource

➤ To create or update a REST resource

Make sure checkbox **Create or Update REST resource** is checked.

- 1 Select a package.



Packages on Integration Server

- Default
- Employees
- WmAdmin
- WmART

Folder Name: Employees

REST Resource Name: employees

☒ Update existing Resource

☒ Create or Update simplified REST services for this REST resource

- 2 Define a folder name. If the folder does not exist, it will be created.
- 3 Define the REST Resource name.
- 4 Check **Update existing Resource** if you want to add additional services to an existing resource.
- 5 Check **Create or Update simplified REST services for this REST resource** if you want to generate additional services with a simplified REST API. This allows you to use a simple HTTP-GET method without a payload.



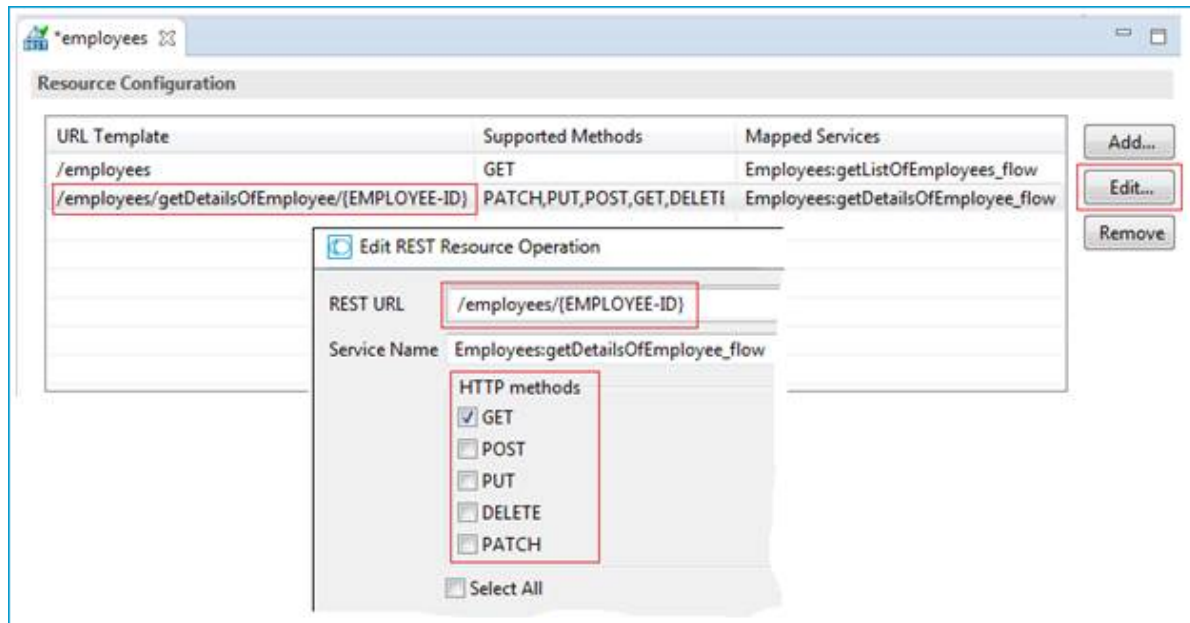
Note: This applies only to APIs where input signature parameters do not contain arrays.

;

Step 6: Edit the REST Resource

➤ To edit the REST resource

- 1 Switch to the **Service Development** perspective of Software AG Designer.



- 2 Refresh the package where you generated the REST Resource in the previous step.
- 3 Open the REST Resource.
- 4 Edit the REST URLs and HTTP Methods to match the functionality of your application.

For more information see:

- *REST Developer's Guide* in the webMethods Integration Server documentation for information on using the generated REST resources
- *REST Developer's Guide* in the webMethods Integration Server documentation

Task 3: Execute the Call from the REST Client to COBOL

Use an appropriate REST client, specify URL, Method and Headers and invoke your REST resource:

URL http://localhost:5555/restv2/employees	URL http://localhost:5555/restv2/employees/111000105
Method GET	Method GET
Headers Accept application/json	Headers Accept application/json
200 OK	200 OK
<pre>{ "errorCode": "00000000", "errorFlag": "false", "response": { "EMPLOYEES-LIST": { "EMPLOYEES": [{ "IDENT": "11100102", "FIRSTNAME": "Edgar", "SURNAME": "Schindler", "DATE-BIRTH": "Dec 4, 1962" }, { "IDENT": "11100105", "FIRSTNAME": "Christian", "SURNAME": "Schirm", "DATE-BIRTH": "Mar 15, 1961" }, { "IDENT": "11100106", "FIRSTNAME": "Rainer", "SURNAME": "Schmitt", "DATE-BIRTH": "Feb 13, 1955" }, { "IDENT": "11100107", "FIRSTNAME": "Helga", "SURNAME": "Schmidt", "DATE-BIRTH": "May 26, 1961" }] } } }</pre>	<pre>{ "errorCode": "00000000", "errorFlag": "false", "response": { "EMPLOYEE-DETAILS": { "FIRSTNAME": "Christian", "SURNAME": "Schirm", "DATE-BIRTH": "Mar 15, 1961", "DETAIL": { "ANNUAL-SALARY": "68000", "VACATION": "21", "LANGUAGE": "GER" } } } }</pre>

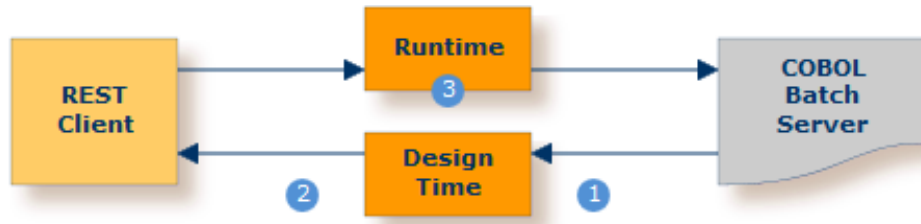
13

Calling COBOL on z/OS Batch from REST

■ Introduction	231
■ What do I need to Install for this Scenario?	233
■ Task 1: Extract the Interface of a COBOL Server	234
■ Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server	246
■ Task 3: Execute the Call from the REST Client to COBOL	252

This scenario uses the tools IDL Extractor for COBOL and Integration Server Wrapper of the Designer.

Introduction



- ❶ Extract the interface of the COBOL server program. See *Using the IDL Extractor for COBOL - Overview*.
- ❷ Generate REST resources, connection and adapter services in Integration Server.
- ❸ Execute the call from the REST client to the COBOL server program.

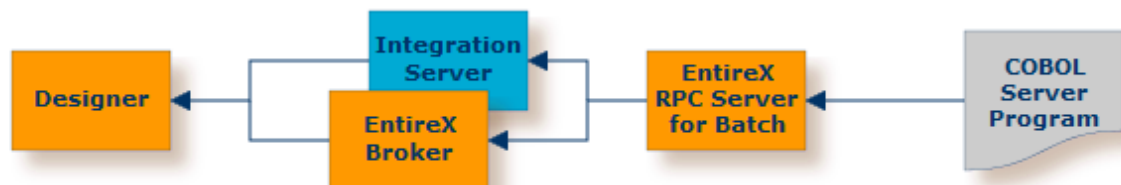
This scenario makes the following important assumptions:

For Task 1:

- You have a working COBOL batch server. For illustration and examples on such a server, see *Batch with Standard Linkage Calling Convention* under *Introduction to the IDL Extractor for COBOL*.
- You have access to the related COBOL sources and copybooks. The minimum requirement is the `DATA DIVISION` of the interface. The sources and copybooks must be stored either
 - *locally*, that is, on the same machine where the Designer is running



- or *remotely* in a PDS or CA Librarian data set and accessed via the *RPC Server for Batch*, and either EntireX Broker or Integration Server



For Tasks 2 and 3:

- You have a REST client.
- You can call the COBOL server program 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
 - The *RPC Server for Batch*



- For the *EntireX Direct RPC* connection method you need:
 - the *RPC Server for Batch*



See also *Direct RPC* in the EntireX Adapter documentation.

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.
- You have an RPC Server for Batch installed. See *Installing the RPC Server for Batch* in the z/OS Installation documentation.
- Optionally, for remote extraction (Task 1) or RPC Connection method (Task 2 and 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.

Task 1: Extract the Interface of a COBOL Server

- [Introduction](#)
- [Sample COBOL Server used in the Examples](#)
- [Extracting a COBOL Server - Fast-track Method](#)
- [Extracting a COBOL Server - Modern Method with User-defined Mapping](#)
- [Comparison Chart: Fast-track versus User-defined Mapping](#)
- [Testing the Extraction Results](#)

Introduction

Follow the instructions for extracting COBOL, see *Using the IDL Extractor for COBOL - Overview* and choose *Scenario I: Create New IDL and Server Mapping Files* if this is your first extraction. Learn how to invoke the IDL Extractor for COBOL, create a COBOL Extractor Environment and about the possibilities to select the COBOL source.

The extraction process creates the following EntireX metafiles:

- IDL file. A Software AG IDL file contains definitions of the interface between client and server. See *Software AG IDL File* in the IDL Editor documentation in the IDL Editor documentation.
- Server mapping file (optional). The mapping file is a Designer file with extension .cvm that contains COBOL-specific mapping information. See *Server Mapping Files for COBOL* in the Designer documentation.

Sample COBOL Server used in the Examples

The following COBOL server is used to illustrate the features of the IDL Extractor for COBOL. Imagine an existing COBOL server called `EMPLOYEE`. It implements the access logic for a `LIST` and `DETAILS` function to a database view `EMPLOYEE`:

```

IDENTIFICATION DIVISION.
    PROGRAM-ID.      EMPLOYEE.
*****
*   Program Name    = EMPLOYEE
*   String Literal  = QUOTE
*****
01 DFHCOMMAREA.
    02 OPERATION                                PIC X(1).
        88 OPERATION-LIST                      VALUE 'L'.
        88 OPERATION-DETAILS                  VALUE 'D'.
    02 EMPLOYEE-ID                             PIC X(10).
    02 EMPLOYEE-DETAILS.
        03 FIRSTNAME                          PIC X(20).
        03 SURNAME                           PIC X(20).
        03 DATE-BIRTH                        PIC X(12).
        03 DETAILS                          PIC X(100).
        03 FILLER REDEFINES DETAILS.
            04 ANNUAL-SALARY                 PACKED-DECIMAL PIC S9(9).
            04 VACATION                      PIC S9(2).
            04 LANGUAGE                      PIC X(3).
            04 FILLER                        PIC X(90).
    02 EMPLOYEES-COUNT                         PIC 9(8) BINARY.
    02 EMPLOYEES OCCURS 1 TO 99 DEPENDING ON EMPLOYEES-COUNT.
        03 IDENT                            PIC X(10).
        03 FIRSTNAME                        PIC X(20).
        03 SURNAME                         PIC X(20).
        03 DATE-BIRTH                      PIC X(12).

```

Two approaches for extracting the COBOL server are described below:

- **Fast-track**

Learn how EntireX helps you to connect the COBOL Server with quick results.

- **User-defined Mapping**

The following features are among those available with a user-defined mapping:

- **Shape the interface to your COBOL server**

You can minimize the interface by suppressing COBOL server fields or by providing constant values. Renaming interfaces allows you to specify a readable long name. These two features together - constants and renaming of interfaces - are most powerful when multiple interfaces are implemented in a single COBOL server. Each function of the COBOL server triggered by an operation code or function code can be modelled as a separate REST resource method/operation.

■ Select alternative mappings

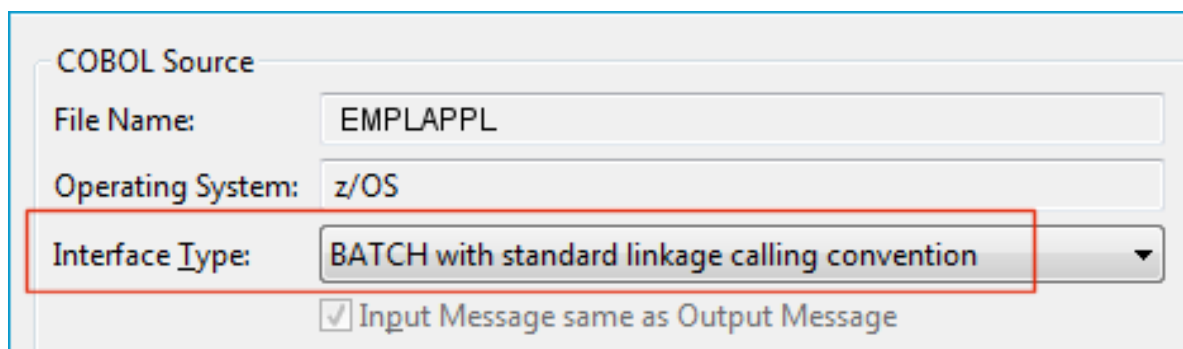
You can select alternative mappings for COBOL data items (REDEFINES, tables etc.) to enable scenarios with COBOL servers where data exchange is not fully described by the LINKAGE SECTION COBOL data items.

These approaches are described under [Fast-track Method](#) and [Modern Method](#) below.

Extracting a COBOL Server - Fast-track Method

> To extract the COBOL Server

- 1 Switch to the **EntireX** perspective.
- 2 Call the IDL Extractor for COBOL.
- 3 Set the correct interface type as described under *Step 4: Define the Extraction Settings and Start Extraction* under *Scenario I: Create New IDL and Server Mapping Files* in the IDL Extractor for COBOL documentation.



COBOL Source

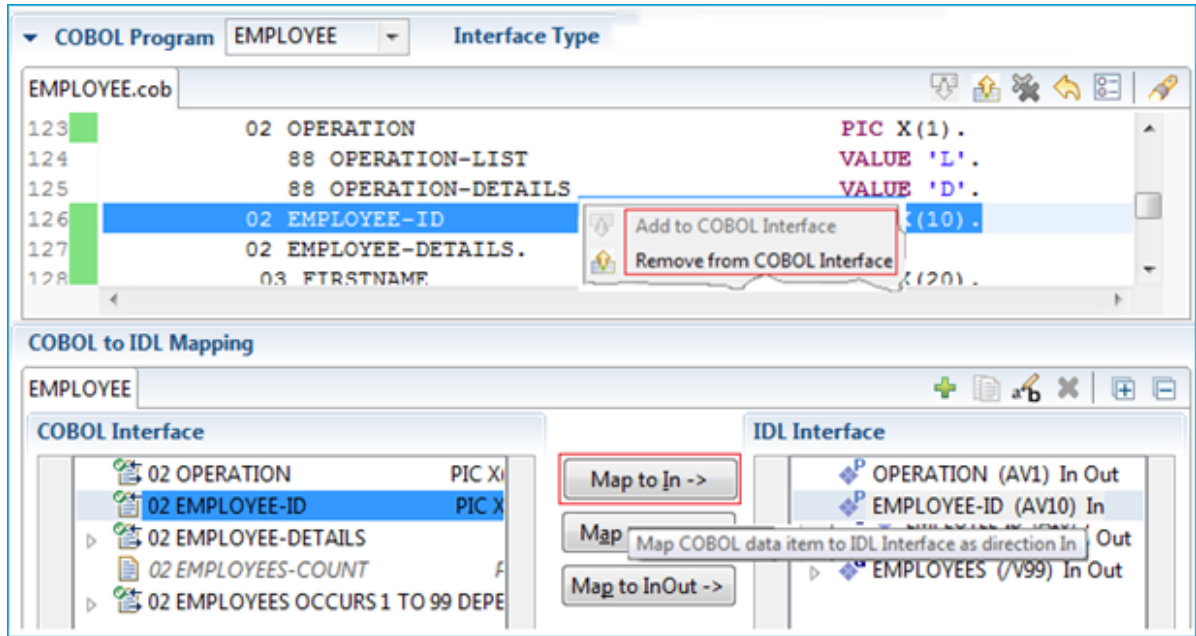
File Name: EMPLAPPL

Operating System: z/OS

Interface Type: BATCH with standard linkage calling convention ▼

☒ Input Message same as Output Message

- 4 Press **Next** to enter the COBOL Mapping Editor.



The purpose of the COBOL Mapping Editor is to map COBOL data items to IDL (Interface Definition Language) parameters. It consists of three main areas:

- The upper window pane is the COBOL program source.
- The window pane on the left-hand side is the COBOL interface, consisting of COBOL data items representing the parameters in your COBOL server program.
- The window pane on the right-hand side is the IDL interface consisting of IDL parameters.

For more information see *Mapping Editor User Interface* in the IDL Extractor for COBOL documentation.

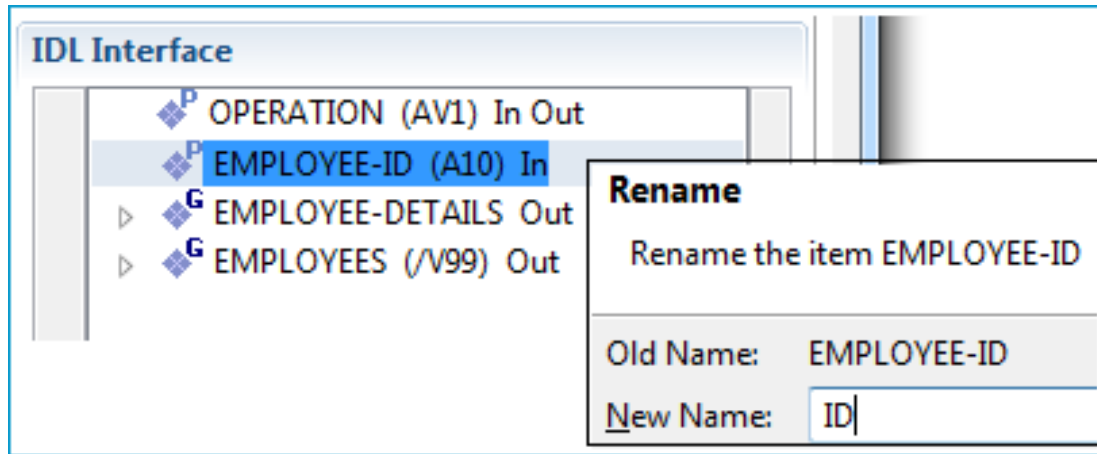
COBOL extraction is a two-step process:

- First, select the COBOL parameters using the **Add to** and **Remove from COBOL Interface** from the context menu on the COBOL data items in the upper window pane (COBOL source); in many situations the COBOL interface is detected by the Mapping Editor directly and you can omit this step.
- Then map the COBOL parameters to IDL directions using the **Map to In/Out/InOut** buttons. COBOL parameters do not have directions, they are always INOUT. This step can be left out for a quick test in non-production scenarios; for real production it is recommended to set IDL directions to reduce data transfer. You also have the option to assign a meaningful name to an IDL parameter if the original COBOL data item is not self-explanatory.

For our example:

- Set **Map to In** for COBOL data item `EMPLOYEE-ID` as IDL direction (see screenshot in previous step). For `EMPLOYEES-DETAILS` and `EMPLOYEES` perform the similar **Map to Out**.

- Use the function **Rename** from the context menu on the IDL parameters in the **IDL Interface** pane to rename COBOL parameter `EMPLOYEES-ID` to IDL parameter `ID`. Do the same for `EMPLOYEE-DETAILS`: Rename to `EMPLOYEE`.



- 5 Press **Finish**. The default extraction settings ensure you get useful results. The outcome is a Software AG IDL file (interface definition language), a Designer file with extension `.idl`:

```
library 'EMPLAPPL' is
program 'EMPLOYEE' is
define data parameter
  1 OPERATION (AV1) In Out
  1 ID (AV10) In /* Original Name:EMPLOYEE-ID
  1 EMPLOYEE Out /* Original Name:EMPLOYEE-DETAILS
  2 FIRSTNAME (AV20)
  2 SURNAME (AV20)
  2 DATE-BIRTH (AV12)
  2 DETAILS (AV100)
  1 EMPLOYEES (/V99) Out
  2 IDENT (AV10)
  2 FIRSTNAME (AV20)
  2 SURNAME (AV20)
  2 DATE-BIRTH (AV12)
end-define
```

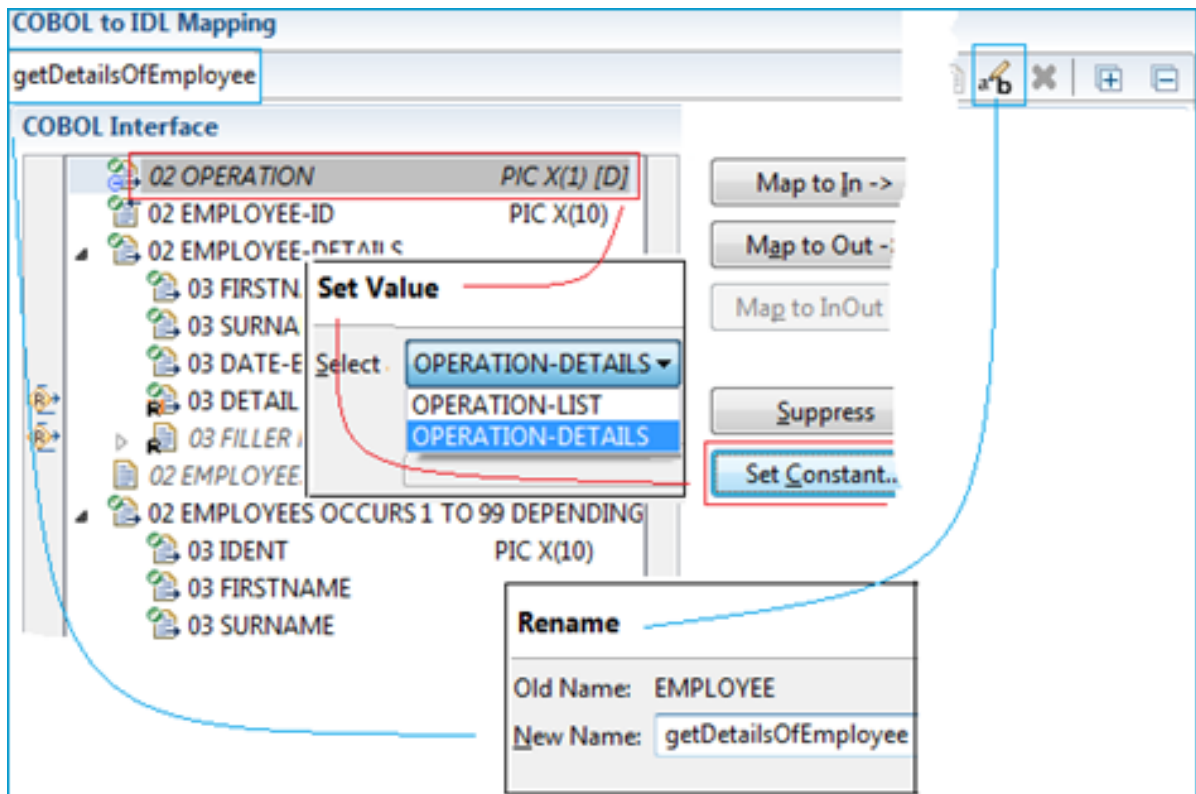
If required, a server mapping file with the same name as the IDL file but the extension `.cvm` may be generated too, for technical reasons. This file must always be stored together with the concomitant IDL file. For more information, see *Server Mapping Files for COBOL* in the Designer documentation. By default, the interface name `EMPLOYEE` is taken from the COBOL server program name.


Extracting a COBOL Server - Modern Method with User-defined Mapping

EntireX allows you to extract all implemented interfaces of the COBOL server separately. Instead of a large `EMPLOYEE` interface, separate interfaces `getListOfEmployees` and `getDetailsOfEmployee` are extracted. Each interface contains required parameters; obsolete parameters for an interface are suppressed, improving its usability.

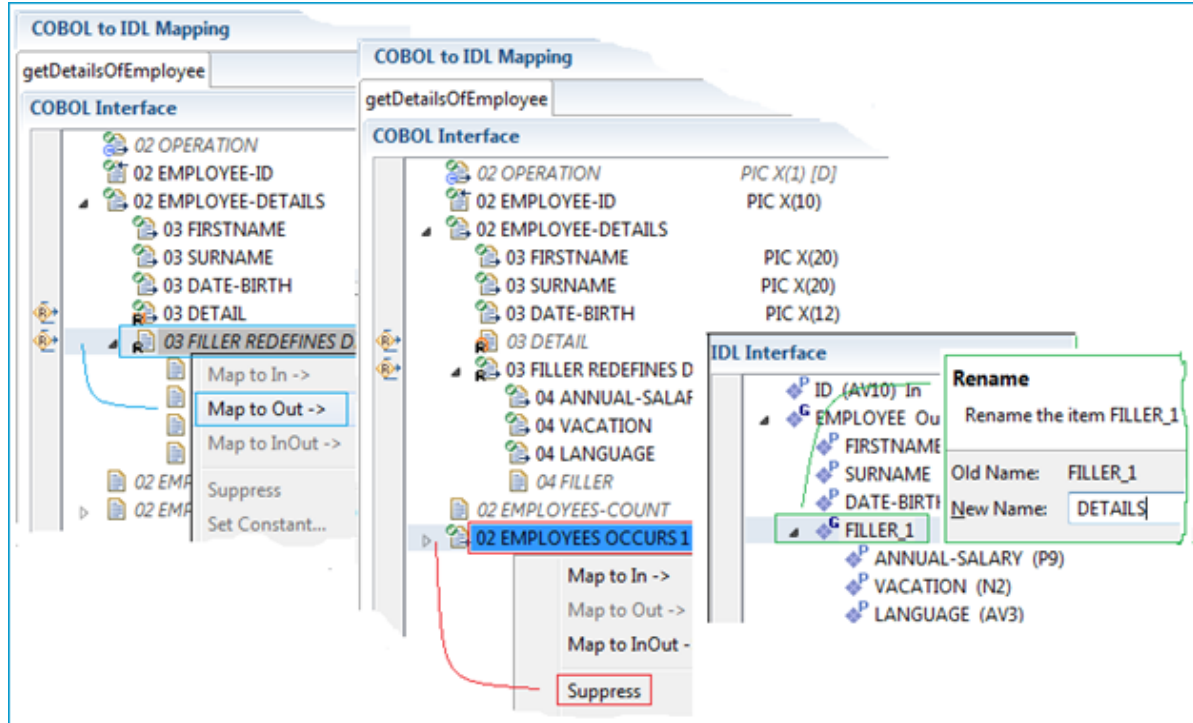
➤ To extract a COBOL server with a user-defined mapping

- 1 Shape `EMPLOYEE` to `getDetailsOfEmployee`:



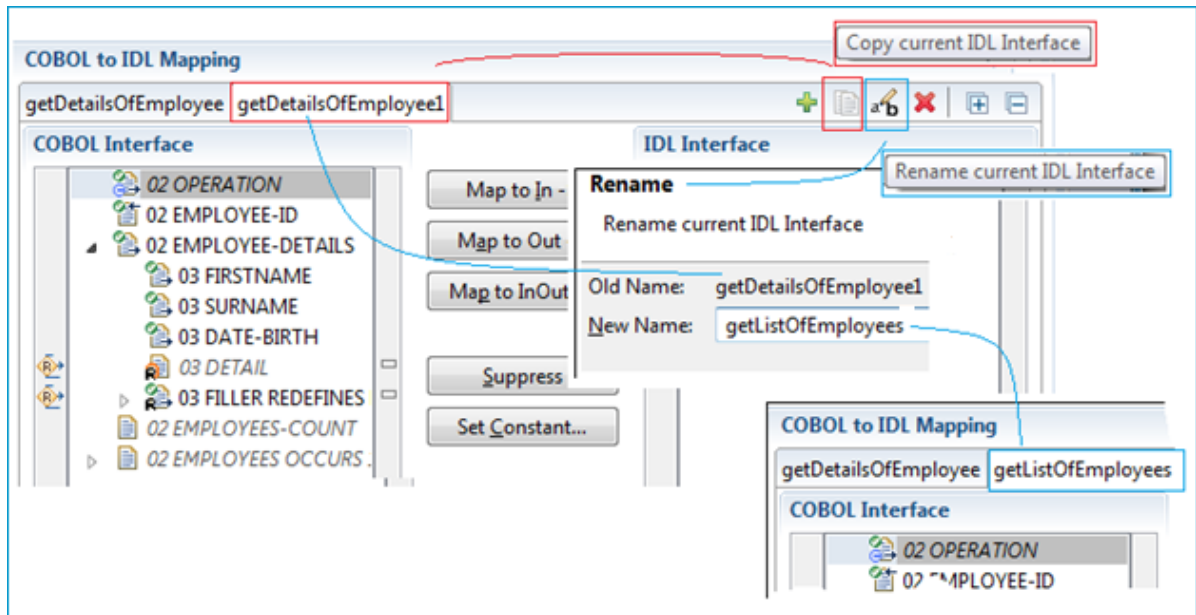
- Mark the `OPERATION` field in the COBOL **Interface** pane and press the **Set Constant** button. In the pop-up window, select `OPERATION-DETAILS` (red markers). The `OPERATION-DETAILS` value originates from COBOL level-88 data item (blue markers). If there is no COBOL level-88 enumeration type in the COBOL program defined, you can enter the function code manually in the pop-up window.
- By default the IDL interface gets the same name as your COBOL server: `EMPLOYEE`. Use the **Rename** button  from the toolbar of the COBOL Mapping Editor and change the IDL interface name to `getDetailsOfEmployee` (blue markers). This name is used as the IS service name later.

- 2 Continue shaping, making `getDetailsOfEmployee` easier to use:



- Use the context menu of the redefinition of the `DETAIL` field in the **COBOL Interface** pane, `FILLER REDEFINES DETAIL` and choose **Map to Out** (blue markers). A redefinition is a secondary layout (field types) of the same memory area. The redefinition contains more specific information (`ANNUAL-SALARY`, `VACATION`, etc.) than character buffer `DETAIL`. This is the reason why it is more useful in your interface.
- Use the context menu of `FILLER_1` in the IDL interface pane and rename `FILLER_1` to `DETAILS` (green markers) for a readable IS service field name later.
- The `DETAILS` function in the COBOL server does not make use of the `EMPLOYEES` field. Therefore you can leave it out of the interface: use the context menu of the `EMPLOYEES` field in the **COBOL Interface** pane and choose **Suppress** (red markers).

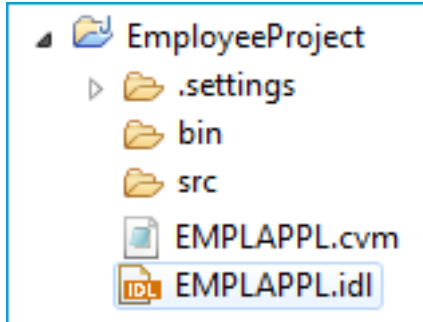
3 Shape `EMPLOYEE` to `getListOfEmployees` Function:



The COBOL Mapping Editor allows you to define multiple IDL interfaces to the same COBOL server:

- You create the additional IDL interface using the toolbar button **Copy current IDL interface** (red markers). A new tab `getDetailsOfEmployee1` is created. The previously extracted `getDetailsOfEmployee` interface still exists in the first tab. Once you reactivate the first tab, you will see the interface of `getDetailsOfEmployee` again.
 - Use the **Rename** button (blue markers) from the toolbar of the COBOL Mapping Editor and change the IDL interface name to `getListOfEmployees` (blue markers). This name is used as the IS service name later.
 - For the `LIST` function you need the `OPERATION-LIST` value in the `OPERATION` field: Mark the `OPERATION` field in the **COBOL Interface** pane and press the **Set Constant** button. In the pop-up window, select `OPERATION-LIST` (similar to Step 1. above *Shape `EMPLOYEE` to `getDetailsOfEmployee`*; red markers).
 - Use the context menu of the `EMPLOYEES` field in the **COBOL Interface** pane, `EMPLOYEES OCCURS 1 TO 99` and choose **Map to Out** (Similar to Step 2. above *Continue shaping, making `getDetailsOfEmployee` easier to use*; blue markers).
 - Because the `EMPLOYEE` and `EMPLOYEE-DETAIL` fields are not used in the COBOL server `LIST` function, you leave them out in the IDL interface: use the context menu of `EMPLOYEE-DETAIL` field in the **COBOL Interface** pane and choose **Suppress** (similar to Step 2. above *Continue shaping, making `getDetailsOfEmployee` easier to use*; red markers). Do the same for the `EMPLOYEE-ID` field.
- 4 Press **Finish** to retrieve the extraction result in the form of a Software AG IDL file. At the same time, a server mapping file for COBOL (Designer file with extension `.cvm`) is created.

See *Server Mapping Files for COBOL* in the Designer documentation. The *Software AG IDL File* in the IDL Editor documentation describes the interfaces from the client point of view, while the server mapping file contains the mapping to the real COBOL server. Both of these files must be kept together and in sync, otherwise a call to the COBOL server may fail.



To summarize: We created two IDL interfaces. In the IDL file, these resulted in two IDL programs: `getListOfEmployees` and `getDetailsOfEmployee`. Both IDL programs were given readable names (Steps 1 and 3). Meaningful fields were kept, while superfluous fields were suppressed (Steps 1 and 3). The program `getDetailsOfEmployee` contains the redefined fields of parameter `DETAILS` (see below) mapped during extraction (Step 1).



Note: At runtime, the RPC client generated with the extracted interface will send data for the redesigned interfaces, while your COBOL server still expects `EMPLOYEE` data. The EntireX runtime transforms the incoming data stream from the RPC client, using the server mapping file.

```

program 'getDetailsOfEmployee' is
  define data parameter
    1 ID (AV10) In /* Original Name:EMPLOYEE-ID
    1 EMPLOYEE Out /* Original Name:EMPLOYEE-DETAILS
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
    2 DETAILS /* Original Name:FILLER_1
    3 ANNUAL-SALARY (P9)
    3 VACATION (N2)
    3 LANGUAGE (AV3)
  end-define
  program 'getListOfEmployees' is
    1 EMPLOYEES (/V99) Out
    2 IDENT (AV10)
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
  end-define

```

Comparison Chart: Fast-track versus User-defined Mapping

The following table highlights the differences when shaping an interface with a *user-defined mapping* compared with a *fast-path extraction*.

	Interface Shaping with User-defined Mapping	Fast-path Extraction
General	User-defined interface with dedicated mapping. For our example, the interfaces are small and tidy and more self-explanatory with long, readable interface names. They are easier to use with the hidden <code>OPERATION</code> field and the suppressed COBOL fields, which are not needed.	Automatic, quick result extraction with COBOL-like interfaces on IS. For our example, the IS interface <code>EMPLOYEE</code> matches exactly 1:1, field by field.
IS service(s)	Multiple small and handy IS services; each <code>OPERATION</code> code is mapped to a separate IS service.	One big IS service.
IS service name	Readable long name.	Short Subprogram name; up to 8 characters.
IS fields	Usage of Suppress and Set Constant reduces the message length. This keeps focus on relevant data items and keeps the client's interface clean. It may also improve performance.	The IS fields and COBOL server parameters match 1:1. As COBOL layout descriptions are sometimes used for many different purposes, irrelevant data items appear and clutter up the IS interface.
<code>OPERATION</code> parameter	Suppressed: the <code>OPERATION</code> parameter does not exist in the IS service as an IS field.	The <code>OPERATION</code> parameter exists in the IS service as an IS field and needs to be filled-in by the client endpoint.
<code>OPERATION</code> code	The <code>OPERATION</code> code is provided internally by EntireX runtime in the <code>OPERATION</code> field.	The <code>OPERATION</code> code needs to be specified in the IS service in the <code>OPERATION</code> field by the client endpoint.
<code>REDEFINE</code> parameters	Either the parameter that is redefined or one of its redefinitions is available as an IS field.	Only the parameter that is redefined is available as an IS field. Redefinitions of the parameter are not available as IS fields.

A user-defined mapping enables you also to define alternative mappings for COBOL data items (`REDEFINES`, tables etc.). These enable scenarios with COBOL servers where data exchange is not fully described by the `LINKAGE SECTION` COBOL data items. For more information refer to the IDL Extractor for COBOL documentation:

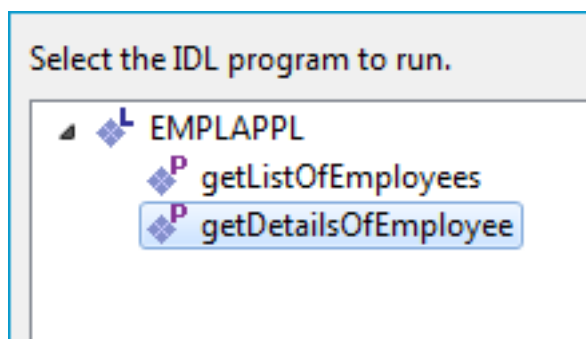
- *Mapping Editor IDL Interface Mapping Functions*
- *User-defined Mapping*

Testing the Extraction Results

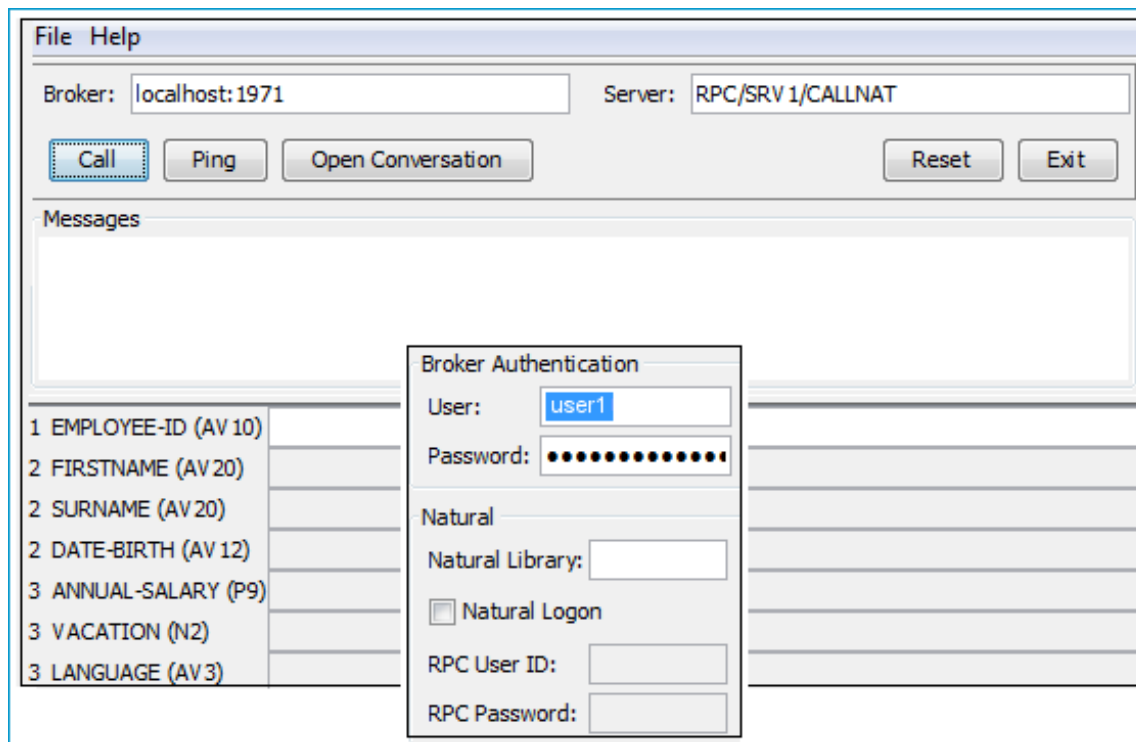
The following pictures use the extraction results described under *Extracting a COBOL Server - Modern Method with User-defined Mapping*.

> To test the extraction results (optional)

- 1 You can test the results of the extraction operation and the REST server back end, using the EntireX IDL Tester. From the context menu of the IDL file in the Designer, choose **Software AG IDL Tester**.



- 2 Select `getDetailsOfEmployee`.



Note that the broker and server parameters contain the explicit route to call the server program, and you can optionally ping the connection from this client. With the **File > Options** dialog, you can set:

- **User** and **Password** for *broker* authentication
- **RPC User ID** and **RPC Password** for *server* authentication

See *EntireX IDL Tester* in the Designer documentation.

- 3 Check the Integration Server log, the EntireX Adapter log or the RPC logs. Applies to all connection methods.

Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server

This section describes your first steps to create a new Integration Server connection. This is described in more detail under *Using the Integration Server Wrapper*, for example working with existing Integration Server connections. This section covers the following topics:

- [Step 1: Start the Integration Server Wrapper Wizard](#)
- [Step 2: Create a New Integration Server Connection](#)
- [Step 3: Select the Connection Type](#)
- [Step 4: Define Adapter Services for an RPC Connection](#)
- [Step 5: Create or Update a REST Resource](#)
- [Step 6: Edit the REST Resource](#)

Step 1: Start the Integration Server Wrapper Wizard

➤ To start the Integration Server Wrapper wizard

- 1 In the context menu of a Software AG IDL file, choose **Integration Server > Generate web-Methods IS Connection**.

This starts the wizard with a list of existing Integration Server Wrapper connections.



Note: If the selected IDL file is not valid because of a syntax error, an error dialog comes up and the wizard does not start.

- 2 Continue with *Step 2: Create a New Integration Server Connection*.

Step 2: Create a New Integration Server Connection

Define New Integration Server Connection

Add host:port, user and password for a new Integration Server

Server:

User:

Password:

☐
Use secure connection

➤ To create a new Integration Server connection

- 1 Define the new Integration Server connection on the wizard page.



Notes:

1. The only required field is **Server**. Enter the hostname of the Integration Server including an optional port number. If no port number is specified, port number defaults to "5555". The **Integration Server Authentication** can be passed with the **User** and **Password** fields.
 2. Optional settings are for secure connections. The **Truststore for HTTPS** contains all signed certificates and must be a valid truststore.
 3. The check box **Verify host name** checks that the hostname is entered in the stored certificate.
 4. When the Integration Server has **Client Authentication** enabled, you can specify your **Keystore** file and keystore **Password**.
 5. For managing Integration Server connections, see [Preferences](#).
- 2 Choose **Next** and continue with *Step 3: Select the Connection Type*.

Step 3: Select the Connection Type

☒ Create a new EntireX Adapter connection of type: **EntireX RPC Connection**

☐ Update Adapter Services or Listeners to an existing Integration Server connection:

List of Connections

Connection Name	Package Name	Connection Type	Enabled
Employees:EmployeesConnection	Employees	EntireX RPC Connection	Yes

Total: 1

☒ Map Software AG IDL data types to String

☒ Create or Update REST resource

> To create a new connection

- 1 Select a connection type from the drop down list. Connection types are described under *EntireX Adapter Connections* in the EntireX Adapter documentation and *Introduction to the Integration Server Wrapper*.



Note: The list of connection types is filtered: connection types that require a license are only shown if a corresponding license file is available. Reliable RPC connections are only shown if all IDL programs contain only IN parameters. Also, if a server mapping file is available, only those connection types that support the interface type specified in the server mapping file are shown.

- 2 Check the box **Create or Update REST resource**.
- 3 Click **Next** and continue with *Step 4: Define Adapter Services for an RPC Connection*.

Step 4: Define Adapter Services for an RPC Connection

Packages on Integration Server localhost:5555

- Default
- Employees**
- WmART
- WmARTEstDC

Folder Name:

Connection Name:

RPC Connection to EntireX

Broker ID:

Server Address:

➤ To create a connection and related adapter services

- 1 Select an existing package or create a new package for the created objects.
- 2 Define a folder name. If the folder does not exist, it will be created.
- 3 Define a connection name.
- 4 Define the parameters of the connection type. For details, see *Connection Parameters for RPC Connections* in the EntireX Adapter documentation.

As a result, the folder will contain the connection and the adapter services (one for each IDL program). The name of a service is the same as the respective IDL program.

The default settings for the adapter services are:

- the **Default** package; if not available, the first package
- the IDL library name for the **Folder Name**
- the IDL library name with the suffix "Connection" for the **Connection Name**



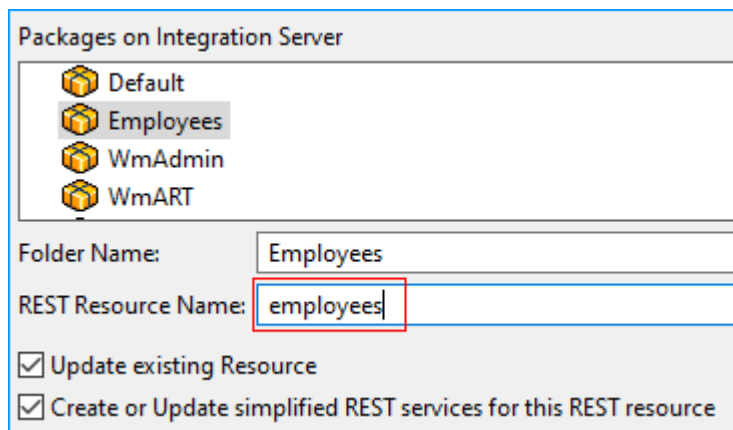
Note: When creating a connection, a package dependency is added such that the selected package depends on webMethods EntireX (the package `WmEntireX`) with the version currently used.

Step 5: Create or Update a REST Resource

➤ To create or update a REST resource

Make sure checkbox **Create or Update REST resource** is checked.

- 1 Select a package.



Packages on Integration Server

- Default
- Employees**
- WmAdmin
- WmART

Folder Name: Employees

REST Resource Name: employees

☒ Update existing Resource

☒ Create or Update simplified REST services for this REST resource

- 2 Define a folder name. If the folder does not exist, it will be created.
- 3 Define the REST Resource name.
- 4 Check **Update existing Resource** if you want to add additional services to an existing resource.
- 5 Check **Create or Update simplified REST services for this REST resource** if you want to generate additional services with a simplified REST API. This allows you to use a simple HTTP-GET method without a payload.

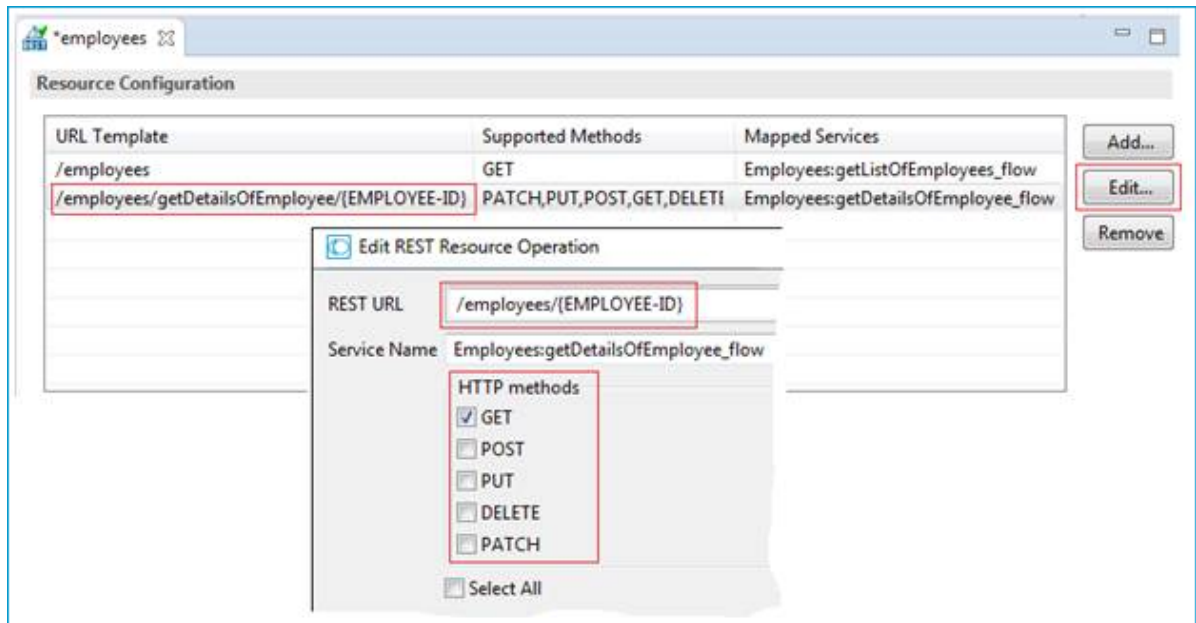


Note: This applies only to APIs where input signature parameters do not contain arrays.

Step 6: Edit the REST Resource

➤ To edit the REST resource

- 1 Switch to the **Service Development** perspective of Software AG Designer.



- 2 Refresh the package where you generated the REST Resource in the previous step.
- 3 Open the REST Resource.
- 4 Edit the REST URLs and HTTP Methods to match the functionality of your application.

For more information see:

- *REST Developer's Guide* in the webMethods Integration Server documentation for information on using the generated REST resources
- *REST Developer's Guide* in the webMethods Integration Server documentation

Task 3: Execute the Call from the REST Client to COBOL

Use an appropriate REST client, specify URL, Method and Headers and invoke your REST resource:

URL http://localhost:5555/restv2/employees

Method GET

Headers Accept application/json

200 OK

```
{
  "errorCode": "00000000",
  "errorFlag": "false",
  "response": {
    "EMPLOYEES-LIST": {
      "EMPLOYEES": [
        {
          "IDENT": "11100102",
          "FIRSTNAME": "Edgar",
          "SURNAME": "Schindler",
          "DATE-BIRTH": "Dec 4, 1962"
        },
        {
          "IDENT": "11100105",
          "FIRSTNAME": "Christian",
          "SURNAME": "Schirm",
          "DATE-BIRTH": "Mar 15, 1961"
        },
        {
          "IDENT": "11100106",
          "FIRSTNAME": "Rainer",
          "SURNAME": "Schmitt",
          "DATE-BIRTH": "Feb 13, 1955"
        },
        {
          "IDENT": "11100107",
          "FIRSTNAME": "Helga",
          "SURNAME": "Schmidt",
          "DATE-BIRTH": "May 26, 1961"
        }
      ]
    }
  }
}
```

URL http://localhost:5555/restv2/employees/111000105

Method GET

Headers Accept application/json

200 OK

```
{
  "errorCode": "00000000",
  "errorFlag": "false",
  "response": {
    "EMPLOYEE-DETAILS": {
      "FIRSTNAME": "Christian",
      "SURNAME": "Schirm",
      "DATE-BIRTH": "Mar 15, 1961",
      "DETAIL": {
        "ANNUAL-SALARY": "68000",
        "VACATION": "21",
        "LANGUAGE": "GER"
      }
    }
  }
}
```

14

Calling COBOL on z/OS IMS from REST

In IMS there are two styles of programs: message processing programs (MPP) and batch message processing programs (BMP). MPP programs can be called via IMS Connect.



- ❶ Extract the interface of the COBOL server program. See *Using the IDL Extractor for COBOL - Overview*.
- ❷ Generate REST resources, connection and adapter services in Integration Server.
- ❸ Execute the call from the REST client to the COBOL server program.

It is important to know whether your COBOL server runs in MPP (online transaction) mode or BMP (batch processing) mode. If you are unsure, consult a COBOL IMS specialist or see description of interface type in the IDL Extractor for COBOL documentation for details and examples: *IMS MPP | IMS BMP*.

When you are sure which programming style you are using, continue with the appropriate scenario:

- **MPP** - online transactions (zero footprint using IMS Connect)
- **BMP** - batch processing (using RPC Server for IMS)

15

Calling COBOL on z/OS IMS MPP (Zero Footprint using IMS Connect) from REST

■ Introduction	257
■ What do I need to Install for this Scenario?	258
■ Task 1: Extract the Interface of a COBOL Server	259
■ Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server	271
■ Task 3: Execute the Call from the REST Client to COBOL	277

This scenario uses the tools IDL Extractor for COBOL and Integration Server Wrapper of the Designer.

Introduction



- ① Extract the interface of the COBOL server program. See *Using the IDL Extractor for COBOL - Overview*.
- ② Generate REST resources, connection and adapter services in Integration Server.
- ③ Execute the call from the REST client to the COBOL server program.

This scenario makes the following important assumptions:

For Task 1:

- You have a working COBOL IMS MPP server. For illustration and examples on such a server, see *IMS MPP Message Interface (IMS Connect)* under *Introduction to the IDL Extractor for COBOL*.
- You have access to the related COBOL sources and copybooks as files on your local machine. The minimum requirement is the `DATA DIVISION` of the interface.



For Tasks 2 and 3:

- You have a REST client.
- To call the COBOL server program at runtime using the EntireX IMS Connect method, you need an IMS Connect address space running. See *Preparing for IMS Connect* and *Connection Parameters for Connections to IMS Connect*.



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.

Task 1: Extract the Interface of a COBOL Server

- [Introduction](#)
- [Sample COBOL Server used in the Examples](#)
- [Extracting a COBOL Server - Fast-track Method](#)
- [Extracting a COBOL Server - Modern Method with User-defined Mapping](#)
- [Comparison Chart: Fast-track versus User-defined Mapping](#)
- [Testing the Extraction Results](#)

Introduction

Follow the instructions for extracting COBOL, see *Using the IDL Extractor for COBOL - Overview* and choose *Scenario I: Create New IDL and Server Mapping Files* if this is your first extraction. Learn how to invoke the IDL Extractor for COBOL, create a COBOL Extractor Environment and about the possibilities to select the COBOL source.

The extraction process creates the following EntireX metafiles:

- IDL file. A Software AG IDL file contains definitions of the interface between client and server. See *Software AG IDL File* in the IDL Editor documentation in the IDL Editor documentation.
- Server mapping file (optional). The mapping file is a Designer file with extension .cvm that contains COBOL-specific mapping information. See *Server Mapping Files for COBOL* in the Designer documentation.

Sample COBOL Server used in the Examples

The following COBOL server is used to illustrate the features of the IDL Extractor for COBOL. Imagine an existing COBOL server called `EMPLOYEE`. It implements the access logic for a `LIST` and `DETAILS` function to a database view `EMPLOYEE`:

```

IDENTIFICATION DIVISION.
    PROGRAM-ID.      EMPLOYEE.
*****
*   Program Name    = EMPLOYEE
*   String Literal  = QUOTE
*****
01 DFHCOMMAREA.
    02 OPERATION                                PIC X(1).
        88 OPERATION-LIST                      VALUE 'L'.
        88 OPERATION-DETAILS                  VALUE 'D'.
    02 EMPLOYEE-ID                                PIC X(10).
    02 EMPLOYEE-DETAILS.
        03 FIRSTNAME                            PIC X(20).
        03 SURNAME                              PIC X(20).
        03 DATE-BIRTH                          PIC X(12).
        03 DETAILS                            PIC X(100).
        03 FILLER REDEFINES DETAILS.
            04 ANNUAL-SALARY          PACKED-DECIMAL PIC S9(9).
            04 VACATION                PIC S9(2).
            04 LANGUAGE                PIC X(3).
            04 FILLER                  PIC X(90).
    02 EMPLOYEES-COUNT                          PIC 9(8) BINARY.
    02 EMPLOYEES OCCURS 1 TO 99 DEPENDING ON EMPLOYEES-COUNT.
        03 IDENT                          PIC X(10).
        03 FIRSTNAME                      PIC X(20).
        03 SURNAME                        PIC X(20).
        03 DATE-BIRTH                    PIC X(12).

```

Two approaches for extracting the COBOL server are described below:

- **Fast-track**

Learn how EntireX helps you to connect the COBOL Server with quick results.

- **User-defined Mapping**

The following features are among those available with a user-defined mapping:

- **Shape the interface to your COBOL server**

You can minimize the interface by suppressing COBOL server fields or by providing constant values. Renaming interfaces allows you to specify a readable long name. These two features together - constants and renaming of interfaces - are most powerful when multiple interfaces are implemented in a single COBOL server. Each function of the COBOL server triggered by an operation code or function code can be modelled as a separate REST resource method/operation.

■ Select alternative mappings

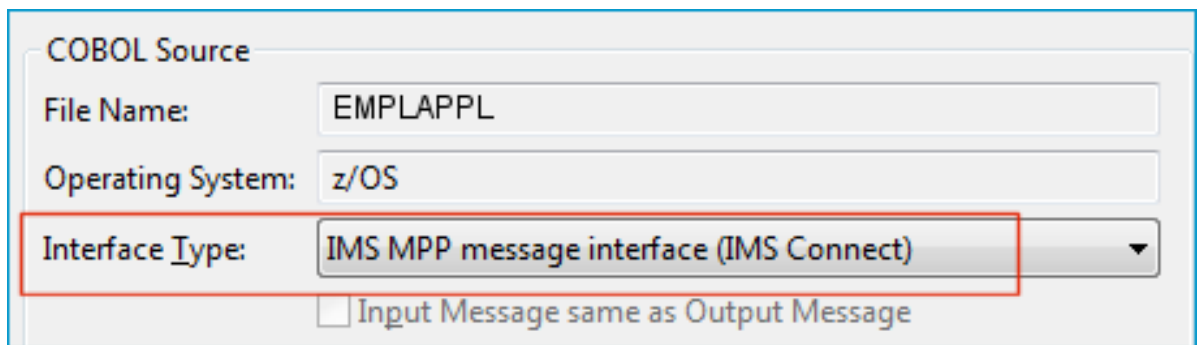
You can select alternative mappings for COBOL data items (REDEFINES, tables etc.) to enable scenarios with COBOL servers where data exchange is not fully described by the LINKAGE SECTION COBOL data items.

These approaches are described under *Fast-track Method* and *Modern Method* below.

Extracting a COBOL Server - Fast-track Method

➤ To extract the COBOL Server

- 1 Switch to the **EntireX** perspective.
- 2 Call the IDL Extractor for COBOL.
- 3 Set the correct interface type as described under *Step 4: Define the Extraction Settings and Start Extraction under Scenario I: Create New IDL and Server Mapping Files in the IDL Extractor for COBOL documentation.*



COBOL Source

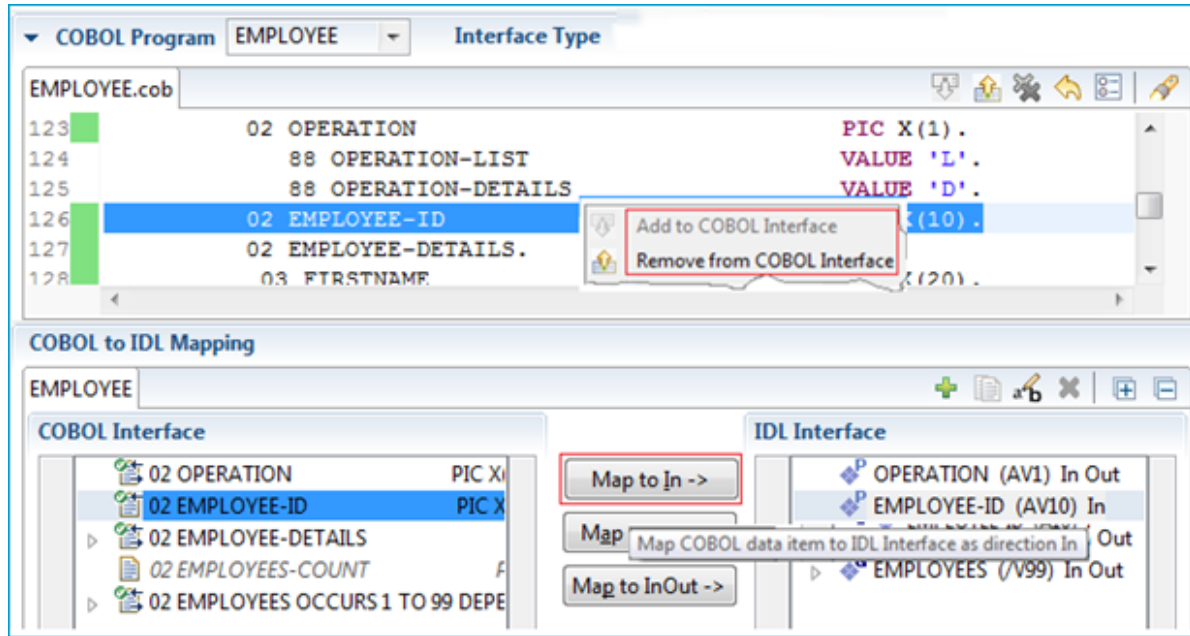
File Name: EMPLAPPL

Operating System: z/OS

Interface Type: IMS MPP message interface (IMS Connect) ▼

☐ Input Message same as Output Message

- 4 Press **Next** to enter the COBOL Mapping Editor.



The purpose of the COBOL Mapping Editor is to map COBOL data items to IDL (Interface Definition Language) parameters. It consists of three main areas:

- The upper window pane is the COBOL program source.
- The window pane on the left-hand side is the COBOL interface, consisting of COBOL data items representing the parameters in your COBOL server program.
- The window pane on the right-hand side is the IDL interface consisting of IDL parameters.

For more information see *Mapping Editor User Interface* in the IDL Extractor for COBOL documentation.

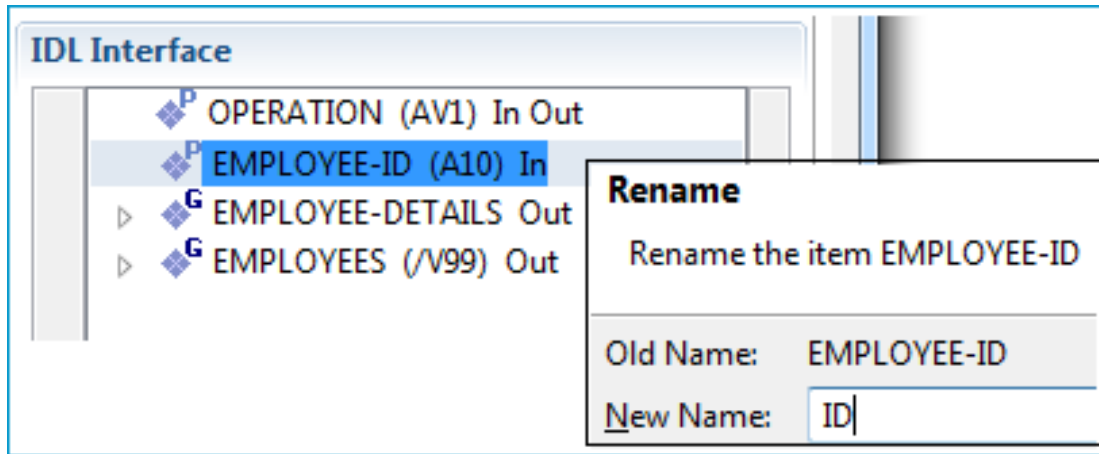
COBOL extraction is a two-step process:

- First, select the COBOL parameters using the **Add to** and **Remove from COBOL Interface** from the context menu on the COBOL data items in the upper window pane (COBOL source); in many situations the COBOL interface is detected by the Mapping Editor directly and you can omit this step.
- Then map the COBOL parameters to IDL directions using the **Map to In/Out/InOut** buttons. COBOL parameters do not have directions, they are always INOUT. This step can be left out for a quick test in non-production scenarios; for real production it is recommended to set IDL directions to reduce data transfer. You also have the option to assign a meaningful name to an IDL parameter if the original COBOL data item is not self-explanatory.

For our example:

- Set **Map to In** for COBOL data item `EMPLOYEE-ID` as IDL direction (see screenshot in previous step). For `EMPLOYEES-DETAILS` and `EMPLOYEES` perform the similar **Map to Out**.

- Use the function **Rename** from the context menu on the IDL parameters in the **IDL Interface** pane to rename COBOL parameter `EMPLOYEE-ID` to IDL parameter `ID`. Do the same for `EMPLOYEE-DETAILS`: Rename to `EMPLOYEE`.



- 5 Press **Finish**. The default extraction settings ensure you get useful results. The outcome is a Software AG IDL file (interface definition language), a Designer file with extension `.idl`:

```
library 'EMPLAPPL' is
program 'EMPLOYEE' is
  define data parameter
    1 OPERATION (AV1) In Out
    1 ID (AV10) In /* Original Name:EMPLOYEE-ID
    1 EMPLOYEE Out /* Original Name:EMPLOYEE-DETAILS
      2 FIRSTNAME (AV20)
      2 SURNAME (AV20)
      2 DATE-BIRTH (AV12)
      2 DETAILS (AV100)
    1 EMPLOYEES (/V99) Out
      2 IDENT (AV10)
      2 FIRSTNAME (AV20)
      2 SURNAME (AV20)
      2 DATE-BIRTH (AV12)
  end-define
```

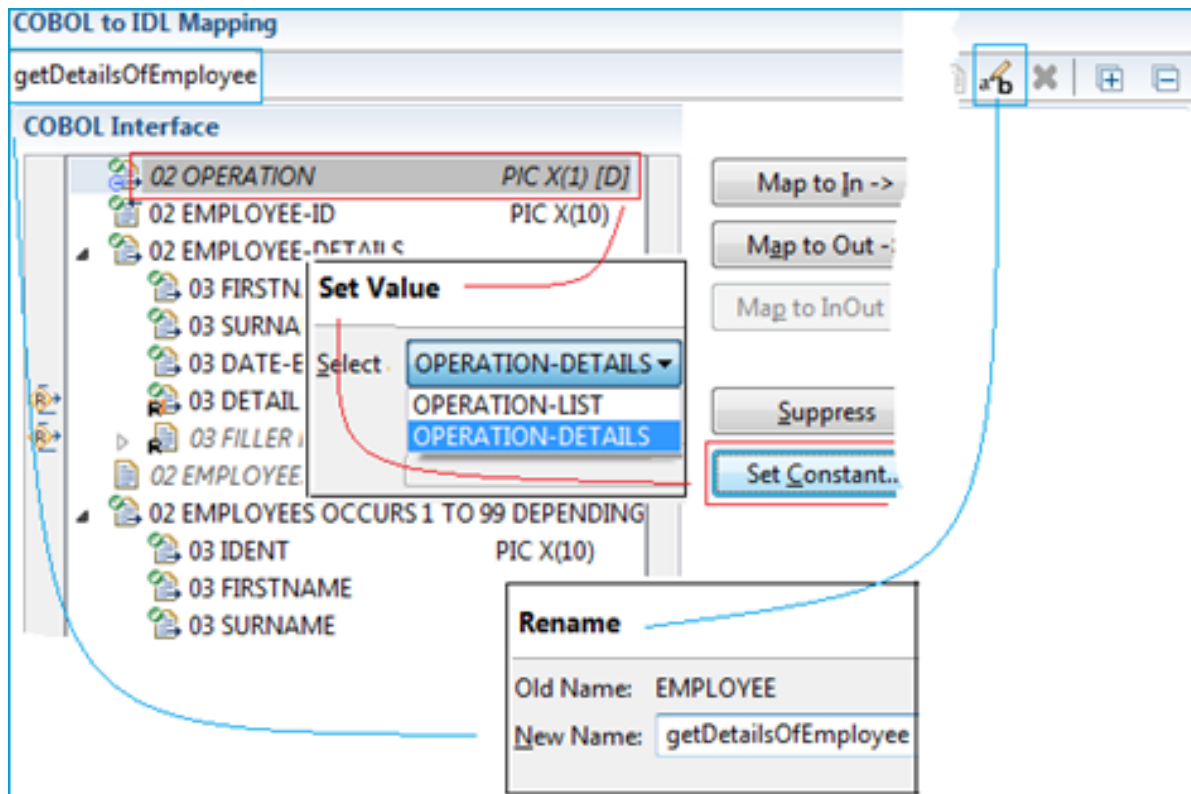
If required, a server mapping file with the same name as the IDL file but the extension `.cvm` may be generated too, for technical reasons. This file must always be stored together with the concomitant IDL file. For more information, see *Server Mapping Files for COBOL* in the Designer documentation. By default, the interface name `EMPLOYEE` is taken from the COBOL server program name.


Extracting a COBOL Server - Modern Method with User-defined Mapping

EntireX allows you to extract all implemented interfaces of the COBOL server separately. Instead of a large `EMPLOYEE` interface, separate interfaces `getListOfEmployees` and `getDetailsOfEmployee` are extracted. Each interface contains required parameters; obsolete parameters for an interface are suppressed, improving its usability.

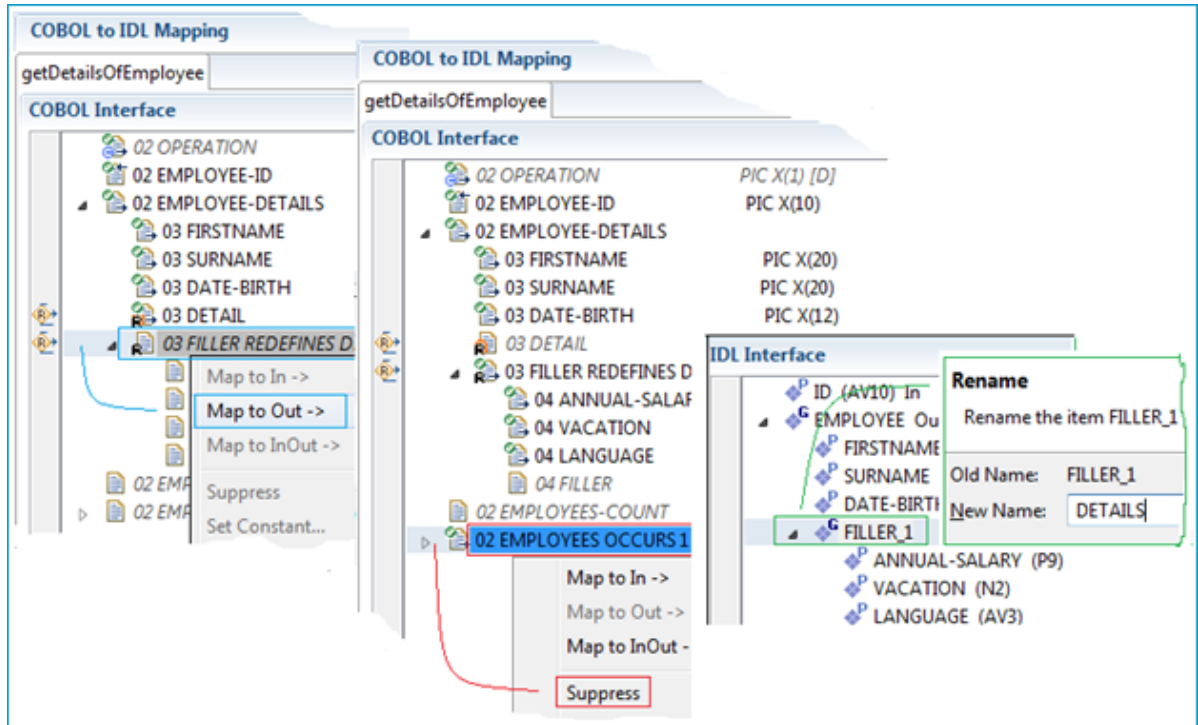
➤ To extract a COBOL server with a user-defined mapping

- 1 Shape `EMPLOYEE` to `getDetailsOfEmployee`:



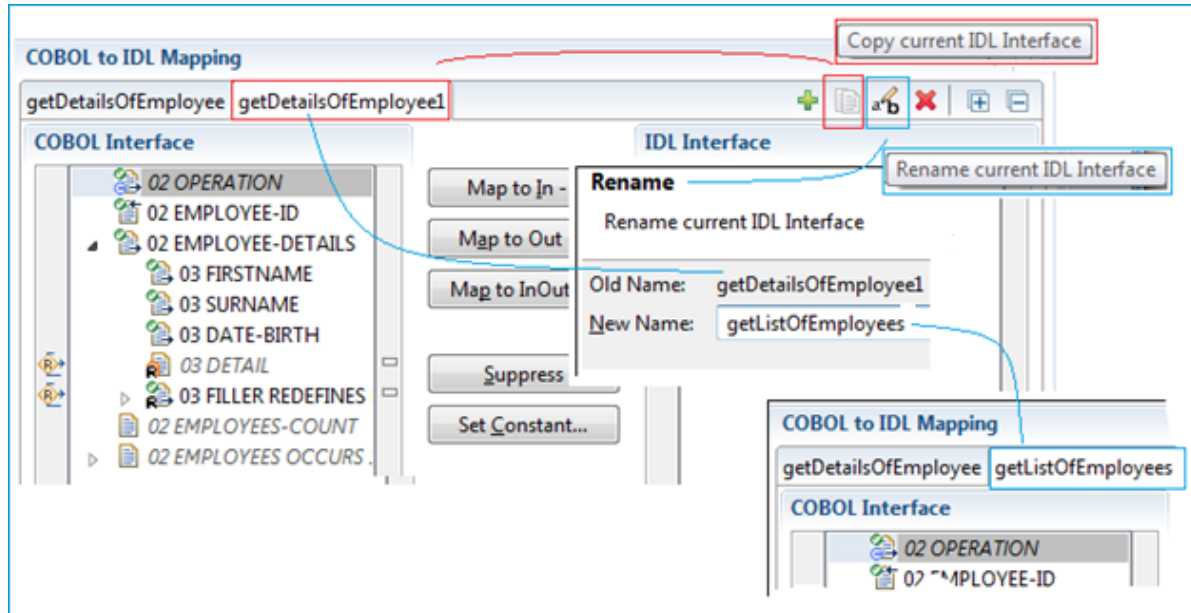
- Mark the `OPERATION` field in the COBOL **Interface** pane and press the **Set Constant** button. In the pop-up window, select `OPERATION-DETAILS` (red markers). The `OPERATION-DETAILS` value originates from COBOL level-88 data item (blue markers). If there is no COBOL level-88 enumeration type in the COBOL program defined, you can enter the function code manually in the pop-up window.
- By default the IDL interface gets the same name as your COBOL server: `EMPLOYEE`. Use the **Rename** button  from the toolbar of the COBOL Mapping Editor and change the IDL interface name to `getDetailsOfEmployee` (blue markers). This name is used as the IS service name later.

- 2 Continue shaping, making `getDetailsOfEmployee` easier to use:




- Use the context menu of the redefinition of the `DETAIL` field in the **COBOL Interface** pane, `FILLER REDEFINES DETAIL` and choose **Map to Out** (blue markers). A redefinition is a secondary layout (field types) of the same memory area. The redefinition contains more specific information (`ANNUAL-SALARY`, `VACATION`, etc.) than character buffer `DETAIL`. This is the reason why it is more useful in your interface.
- Use the context menu of `FILLER_1` in the IDL interface pane and rename `FILLER_1` to `DETAILS` (green markers) for a readable IS service field name later.
- The `DETAILS` function in the COBOL server does not make use of the `EMPLOYEES` field. Therefore you can leave it out of the interface: use the context menu of the `EMPLOYEES` field in the **COBOL Interface** pane and choose **Suppress** (red markers).


3 Shape `EMPLOYEE` to `getListOfEmployees` Function:



The COBOL Mapping Editor allows you to define multiple IDL interfaces to the same COBOL server:

- You create the additional IDL interface using the toolbar button **Copy current IDL interface**

 (red markers). A new tab `getDetailsOfEmployee1` is created. The previously extracted `getDetailsOfEmployee` interface still exists in the first tab. Once you reactivate the first tab, you will see the interface of `getDetailsOfEmployee` again.

- Use the **Rename** button  from the toolbar of the COBOL Mapping Editor and change the IDL interface name to `getListOfEmployees` (blue markers). This name is used as the IS service name later.

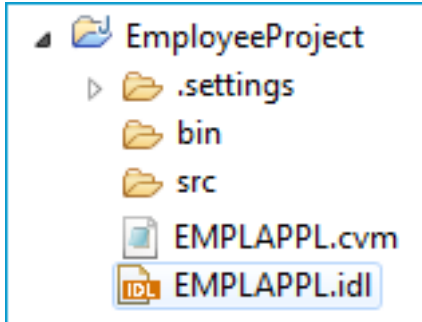
- For the `LIST` function you need the `OPERATION-LIST` value in the `OPERATION` field: Mark the `OPERATION` field in the **COBOL Interface** pane and press the **Set Constant** button. In the pop-up window, select `OPERATION-LIST` (similar to Step 1. above *Shape `EMPLOYEE` to `getDetailsOfEmployee`*; red markers).

- Use the context menu of the `EMPLOYEES` field in the **COBOL Interface** pane, `EMPLOYEES OCCURS 1 TO 99` and choose **Map to Out** (Similar to Step 2. above *Continue shaping, making `getDetailsOfEmployee` easier to use*; blue markers).

- Because the `EMPLOYEE` and `EMPLOYEE-DETAIL` fields are not used in the COBOL server `LIST` function, you leave them out in the IDL interface: use the context menu of `EMPLOYEE-DETAIL` field in the **COBOL Interface** pane and choose **Suppress** (similar to Step 2. above *Continue shaping, making `getDetailsOfEmployee` easier to use*; red markers). Do the same for the `EMPLOYEE-ID` field.

- 4 Press **Finish** to retrieve the extraction result in the form of a Software AG IDL file. At the same time, a server mapping file for COBOL (Designer file with extension `.cvm`) is created.

See *Server Mapping Files for COBOL* in the Designer documentation. The *Software AG IDL File* in the IDL Editor documentation describes the interfaces from the client point of view, while the server mapping file contains the mapping to the real COBOL server. Both of these files must be kept together and in sync, otherwise a call to the COBOL server may fail.



To summarize: We created two IDL interfaces. In the IDL file, these resulted in two IDL programs: `getListOfEmployees` and `getDetailsOfEmployee`. Both IDL programs were given readable names (Steps 1 and 3). Meaningful fields were kept, while superfluous fields were suppressed (Steps 1 and 3). The program `getDetailsOfEmployee` contains the redefined fields of parameter `DETAILS` (see below) mapped during extraction (Step 1).



Note: At runtime, the RPC client generated with the extracted interface will send data for the redesigned interfaces, while your COBOL server still expects `EMPLOYEE` data. The EntireX runtime transforms the incoming data stream from the RPC client, using the server mapping file.

```

program 'getDetailsOfEmployee' is
  define data parameter
    1 ID (AV10) In /* Original Name:EMPLOYEE-ID
    1 EMPLOYEE Out /* Original Name:EMPLOYEE-DETAILS
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
    2 DETAILS /* Original Name:FILLER_1
    3 ANNUAL-SALARY (P9)
    3 VACATION (N2)
    3 LANGUAGE (AV3)
  end-define
  program 'getListOfEmployees' is
    1 EMPLOYEES (/V99) Out
    2 IDENT (AV10)
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
  end-define

```

Comparison Chart: Fast-track versus User-defined Mapping

The following table highlights the differences when shaping an interface with a *user-defined mapping* compared with a *fast-path extraction*.

	Interface Shaping with User-defined Mapping	Fast-path Extraction
General	User-defined interface with dedicated mapping. For our example, the interfaces are small and tidy and more self-explanatory with long, readable interface names. They are easier to use with the hidden <code>OPERATION</code> field and the suppressed COBOL fields, which are not needed.	Automatic, quick result extraction with COBOL-like interfaces on IS. For our example, the IS interface <code>EMPLOYEE</code> matches exactly 1:1, field by field.
IS service(s)	Multiple small and handy IS services; each <code>OPERATION</code> code is mapped to a separate IS service.	One big IS service.
IS service name	Readable long name.	Short Subprogram name; up to 8 characters.
IS fields	Usage of Suppress and Set Constant reduces the message length. This keeps focus on relevant data items and keeps the client's interface clean. It may also improve performance.	The IS fields and COBOL server parameters match 1:1. As COBOL layout descriptions are sometimes used for many different purposes, irrelevant data items appear and clutter up the IS interface.
<code>OPERATION</code> parameter	Suppressed: the <code>OPERATION</code> parameter does not exist in the IS service as an IS field.	The <code>OPERATION</code> parameter exists in the IS service as an IS field and needs to be filled-in by the client endpoint.
<code>OPERATION</code> code	The <code>OPERATION</code> code is provided internally by EntireX runtime in the <code>OPERATION</code> field.	The <code>OPERATION</code> code needs to be specified in the IS service in the <code>OPERATION</code> field by the client endpoint.
<code>REDEFINE</code> parameters	Either the parameter that is redefined or one of its redefinitions is available as an IS field.	Only the parameter that is redefined is available as an IS field. Redefinitions of the parameter are not available as IS fields.

A user-defined mapping enables you also to define alternative mappings for COBOL data items (`REDEFINES`, tables etc.). These enable scenarios with COBOL servers where data exchange is not fully described by the `LINKAGE SECTION` COBOL data items. For more information refer to the IDL Extractor for COBOL documentation:

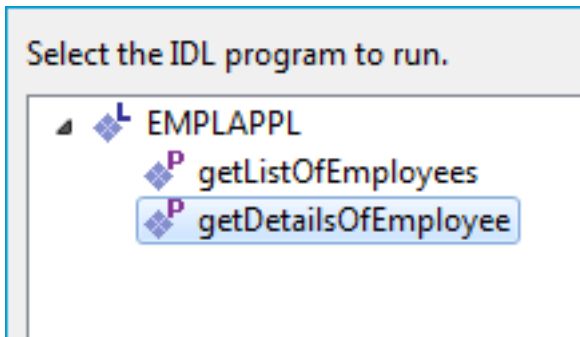
- *Mapping Editor IDL Interface Mapping Functions* if **Input Message same as Output Message** was checked.
- *User-defined Mapping*

Testing the Extraction Results

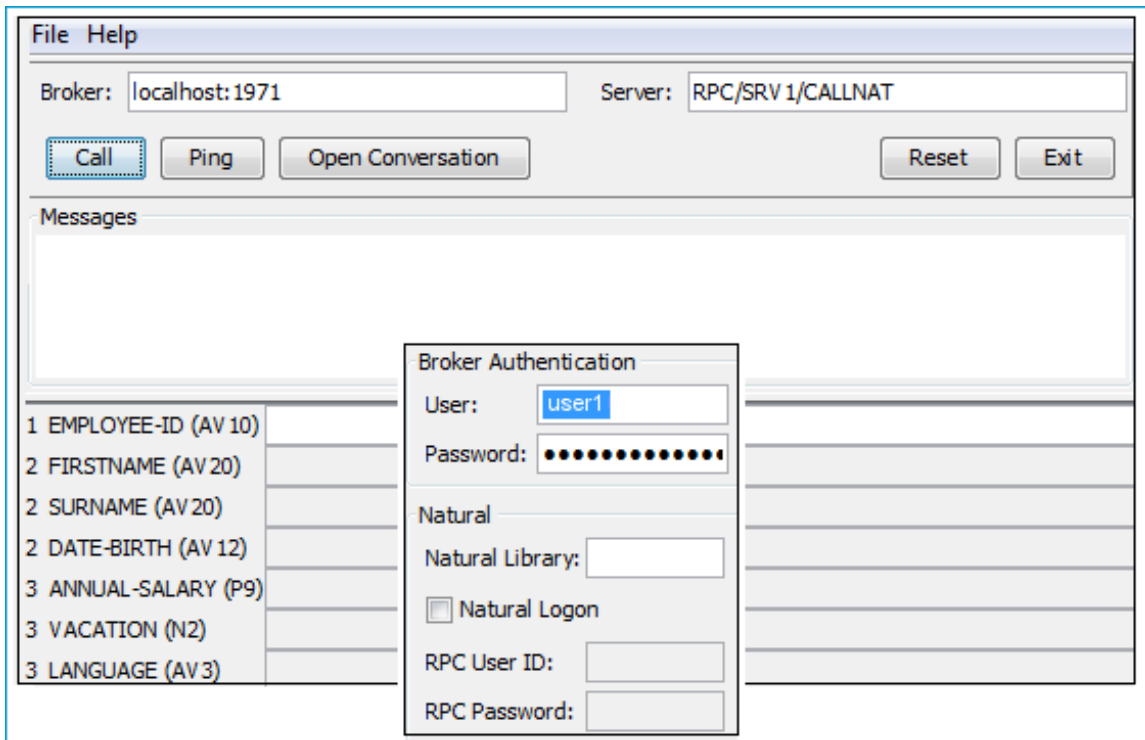
The following pictures use the extraction results described under *Extracting a COBOL Server - Modern Method with User-defined Mapping*.

> To test the extraction results (optional)

- 1 You can test the results of the extraction operation and the REST server back end, using the EntireX IDL Tester. From the context menu of the IDL file in the Designer, choose **Software AG IDL Tester**.



- 2 Select `getDetailsOfEmployee`.



Note that the broker and server parameters contain the explicit route to call the server program, and you can optionally ping the connection from this client. With the **File > Options** dialog, you can set:

- **User** and **Password** for *broker* authentication
- **RPC User ID** and **RPC Password** for *server* authentication

See *EntireX IDL Tester* in the Designer documentation.

- 3 Check the Integration Server log, the EntireX Adapter log or the RPC logs. Applies to all connection methods.

Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server

This section describes your first steps to create a new Integration Server connection. This is described in more detail under *Using the Integration Server Wrapper*, for example working with existing Integration Server connections. This section covers the following topics:

- [Step 1: Start the Integration Server Wrapper Wizard](#)
- [Step 2: Create a New Integration Server Connection](#)
- [Step 3: Select the Connection Type](#)
- [Step 4: Define Adapter Services for an IMS Connect Connection](#)
- [Step 5: Create or Update a REST Resource](#)
- [Step 6: Edit the REST Resource](#)

Step 1: Start the Integration Server Wrapper Wizard

➤ To start the Integration Server Wrapper wizard

- 1 In the context menu of a Software AG IDL file, choose **Integration Server > Generate web-Methods IS Connection**.

This starts the wizard with a list of existing Integration Server Wrapper connections.



Note: If the selected IDL file is not valid because of a syntax error, an error dialog comes up and the wizard does not start.

- 2 Continue with *Step 2: Create a New Integration Server Connection*.

Step 2: Create a New Integration Server Connection

Define New Integration Server Connection

Add host:port, user and password for a new Integration Server

Server:

User:

Password:

☐
Use secure connection

➤ To create a new Integration Server connection

- 1 Define the new Integration Server connection on the wizard page.



Notes:

1. The only required field is **Server**. Enter the hostname of the Integration Server including an optional port number. If no port number is specified, port number defaults to "5555". The **Integration Server Authentication** can be passed with the **User** and **Password** fields.
 2. Optional settings are for secure connections. The **Truststore for HTTPS** contains all signed certificates and must be a valid truststore.
 3. The check box **Verify host name** checks that the hostname is entered in the stored certificate.
 4. When the Integration Server has **Client Authentication** enabled, you can specify your **Keystore** file and keystore **Password**.
 5. For managing Integration Server connections, see [Preferences](#).
- 2 Choose **Next** and continue with *Step 3: Select the Connection Type*.

Step 3: Select the Connection Type

☒ Create a new EntireX Adapter connection of type **IMS Connect Connection**

☐ Update Adapter Services or Listeners to an existing Integration Server connection:

List of Connections

Connection Name	Package Name	Connection Type	Enabled
Employees:EmployeesConnection	Employees	EntireX RPC Connection	Yes


Total: 1

☒ Map Software AG IDL data types to String

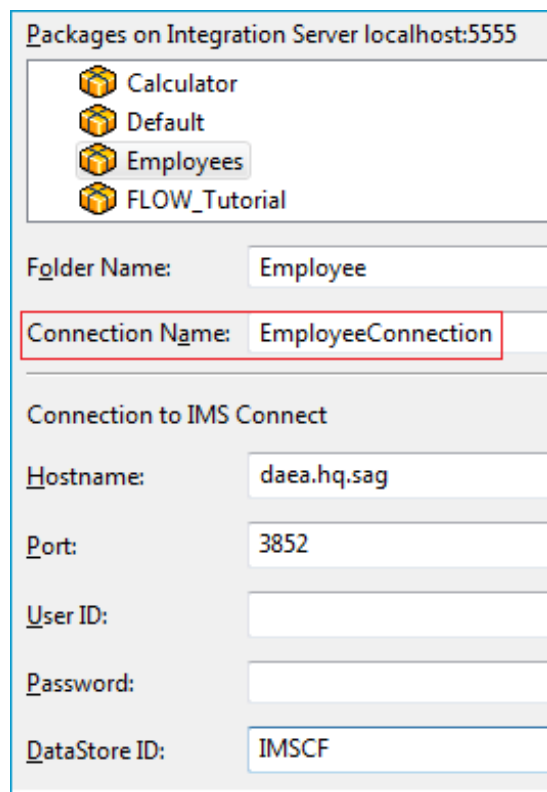
☒ Create or Update REST resource

➤ To create a new connection

- 1 Select a connection type from the drop down list. Connection types are described under *EntireX Adapter Connections* in the EntireX Adapter documentation and *Introduction to the Integration Server Wrapper*.

 **Note:** The list of connection types is filtered: connection types that require a license are only shown if a corresponding license file is available. Reliable RPC connections are only shown if all IDL programs contain only IN parameters. Also, if a server mapping file is available, only those connection types that support the interface type specified in the server mapping file are shown.
- 2 Check the box **Create or Update REST resource**.
- 3 Click **Next** and continue with *Step 4: Define Adapter Services for an IMS Connect Connection*.

Step 4: Define Adapter Services for an IMS Connect Connection



Packages on Integration Server localhost:5555

- Calculator
- Default
- Employees
- FLOW_Tutorial

Folder Name: Employee

Connection Name: EmployeeConnection

Connection to IMS Connect

Hostname: daea.hq.sag

Port: 3852

User ID:

Password:

DataStore ID: IMSCF

➤ To create a connection and related adapter services

- 1 Select a package for the created objects.
- 2 Define a folder name. If the folder does not exist, it will be created.
- 3 Define a connection name.
- 4 Define the parameters of the connection type. For details, see *Connection Parameters for Connections to IMS Connect* in the EntireX Adapter documentation.

As a result, the folder will contain the connection and the adapter services (one for each IDL program). The name of a service is the same as the respective IDL program.

The default settings for the adapter services are:

- the **Default** package; if not available, the first package
- the IDL library name for the **Folder Name**
- the IDL library name with the suffix "Connection" for the **Connection Name**



Note: 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.

Step 5: Create or Update a REST Resource

> To create or update a REST resource

Make sure checkbox **Create or Update REST resource** is checked.

- 1 Select a package.

The screenshot shows a dialog box titled "Packages on Integration Server". It contains a list of packages: Default, Employees (selected), WmAdmin, and WmART. Below the list, there are two text input fields: "Folder Name:" with the value "Employees" and "REST Resource Name:" with the value "employees". At the bottom, there are two checked checkboxes: "Update existing Resource" and "Create or Update simplified REST services for this REST resource".

- 2 Define a folder name. If the folder does not exist, it will be created.
- 3 Define the REST Resource name.
- 4 Check **Update existing Resource** if you want to add additional services to an existing resource.
- 5 Check **Create or Update simplified REST services for this REST resource** if you want to generate additional services with a simplified REST API. This allows you to use a simple HTTP-GET method without a payload.

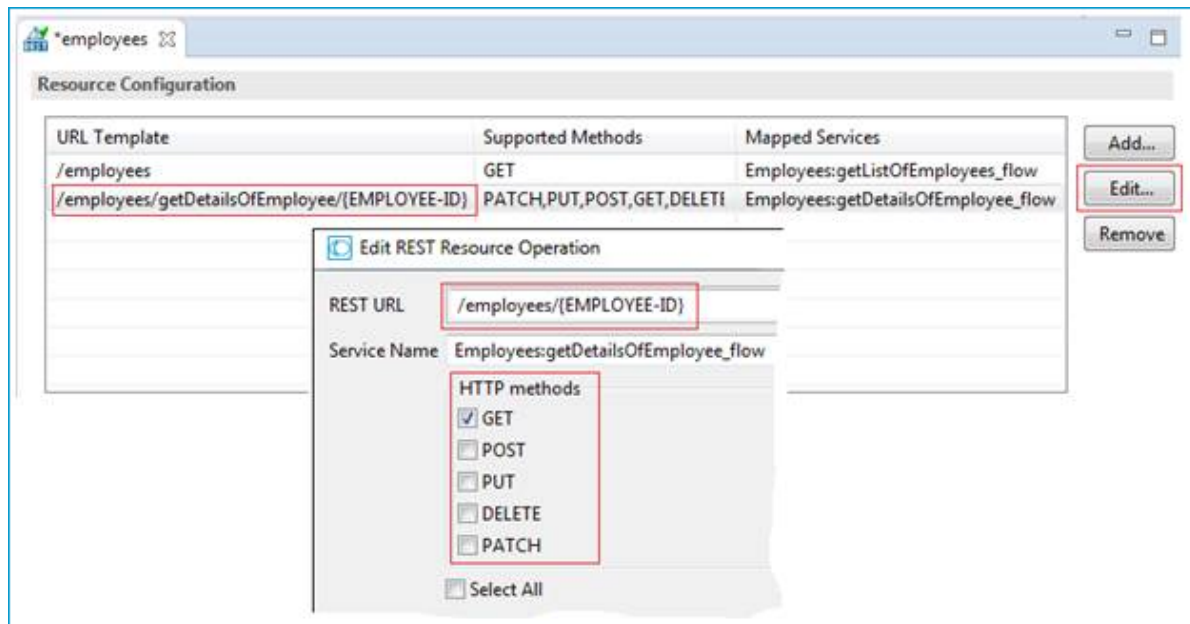


Note: This applies only to APIs where input signature parameters do not contain arrays.

Step 6: Edit the REST Resource

> To edit the REST resource

- 1 Switch to the **Service Development** perspective of Software AG Designer.



- 2 Refresh the package where you generated the REST Resource in the previous step.
- 3 Open the REST Resource.
- 4 Edit the REST URLs and HTTP Methods to match the functionality of your application.

For more information see:

- *REST Developer's Guide* in the webMethods Integration Server documentation for information on using the generated REST resources
- *REST Developer's Guide* in the webMethods Integration Server documentation

Task 3: Execute the Call from the REST Client to COBOL

Use an appropriate REST client, specify URL, Method and Headers and invoke your REST resource:

URL http://localhost:5555/restv2/employees

Method GET

Headers Accept application/json

200 OK

```
{
  "errorCode": "00000000",
  "errorFlag": "false",
  "response": {
    "EMPLOYEES-LIST": {
      "EMPLOYEES": [
        {
          "IDENT": "11100102",
          "FIRSTNAME": "Edgar",
          "SURNAME": "Schindler",
          "DATE-BIRTH": "Dec 4, 1962"
        },
        {
          "IDENT": "11100105",
          "FIRSTNAME": "Christian",
          "SURNAME": "Schirm",
          "DATE-BIRTH": "Mar 15, 1961"
        },
        {
          "IDENT": "11100106",
          "FIRSTNAME": "Rainer",
          "SURNAME": "Schmitt",
          "DATE-BIRTH": "Feb 13, 1955"
        },
        {
          "IDENT": "11100107",
          "FIRSTNAME": "Helga",
          "SURNAME": "Schmidt",
          "DATE-BIRTH": "May 26, 1961"
        }
      ]
    }
  }
}
```

URL http://localhost:5555/restv2/employees/111000105

Method GET

Headers Accept application/json

200 OK

```
{
  "errorCode": "00000000",
  "errorFlag": "false",
  "response": {
    "EMPLOYEE-DETAILS": {
      "FIRSTNAME": "Christian",
      "SURNAME": "Schirm",
      "DATE-BIRTH": "Mar 15, 1961",
      "DETAIL": {
        "ANNUAL-SALARY": "68000",
        "VACATION": "21",
        "LANGUAGE": "GER"
      }
    }
  }
}
```


16

Calling COBOL on z/OS IMS BMP (Batch) from REST

■ Introduction	281
■ What do I need to Install for this Scenario?	283
■ Task 1: Extract the Interface of a COBOL Server	284
■ Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server	296
■ Task 3: Execute the Call from the REST Client to COBOL	302

This scenario uses the tools IDL Extractor for COBOL and Integration Server Wrapper of the Designer.

Introduction



- ❶ Extract the interface of the COBOL server program. See *Using the IDL Extractor for COBOL - Overview*.
- ❷ Generate REST resources, connection and adapter services in Integration Server.
- ❸ Execute the call from the REST client to the COBOL server program.

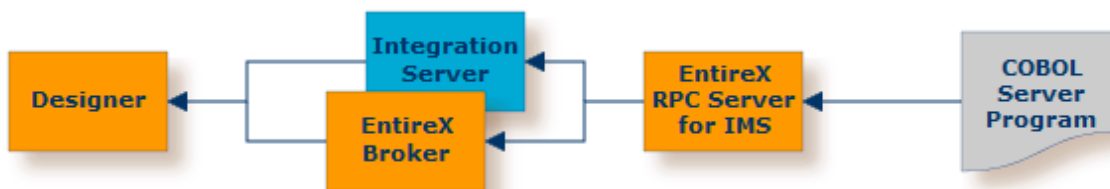
This scenario makes the following important assumptions:

For Task 1:

- You have a working COBOL IMS BMP server. For illustration and examples on such a server, see *IMS BMP with Standard Linkage Calling Convention* under *Introduction to the IDL Extractor for COBOL*.
- You have access to the related COBOL sources and copybooks. The minimum requirement is the `DATA DIVISION` of the interface. The sources and copybooks must be stored either
 - *locally*, that is, on the same machine where the Designer is running



- or *remotely* in a PDS or CA Librarian data set and accessed via the RPC Server for IMS, and either EntireX Broker or Integration Server



For Tasks 2 and 3:

- You have a REST client.
- You can call the COBOL server program 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
 - The *RPC Server for IMS*



- For the *EntireX Direct RPC* connection method you need:
 - the *RPC Server for IMS*



See also *Direct RPC* in the EntireX Adapter documentation.

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.
- You have an RPC Server for IMS installed. See *Installing the RPC Server for IMS* in the z/OS Installation documentation.
- Optionally 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.

Task 1: Extract the Interface of a COBOL Server

- [Introduction](#)
- [Sample COBOL Server used in the Examples](#)
- [Extracting a COBOL Server - Fast-track Method](#)
- [Extracting a COBOL Server - Modern Method with User-defined Mapping](#)
- [Comparison Chart: Fast-track versus User-defined Mapping](#)
- [Testing the Extraction Results](#)

Introduction

Follow the instructions for extracting COBOL, see *Using the IDL Extractor for COBOL - Overview* and choose *Scenario I: Create New IDL and Server Mapping Files* if this is your first extraction. Learn how to invoke the IDL Extractor for COBOL, create a COBOL Extractor Environment and about the possibilities to select the COBOL source.

The extraction process creates the following EntireX metafiles:

- IDL file. A Software AG IDL file contains definitions of the interface between client and server. See *Software AG IDL File* in the IDL Editor documentation in the IDL Editor documentation.
- Server mapping file (optional). The mapping file is a Designer file with extension `.cvm` that contains COBOL-specific mapping information. See *Server Mapping Files for COBOL* in the Designer documentation.

Sample COBOL Server used in the Examples

The following COBOL server is used to illustrate the features of the IDL Extractor for COBOL. Imagine an existing COBOL server called `EMPLOYEE`. It implements the access logic for a `LIST` and `DETAILS` function to a database view `EMPLOYEE`:

```

IDENTIFICATION DIVISION.
    PROGRAM-ID.      EMPLOYEE.
*****
*   Program Name    = EMPLOYEE
*   String Literal  = QUOTE
*****
01 DFHCOMMAREA.
    02 OPERATION                                PIC X(1).
        88 OPERATION-LIST                      VALUE 'L'.
        88 OPERATION-DETAILS                  VALUE 'D'.
    02 EMPLOYEE-ID                             PIC X(10).
    02 EMPLOYEE-DETAILS.
        03 FIRSTNAME                          PIC X(20).
        03 SURNAME                           PIC X(20).
        03 DATE-BIRTH                        PIC X(12).
        03 DETAILS                          PIC X(100).
        03 FILLER REDEFINES DETAILS.
            04 ANNUAL-SALARY                   PACKED-DECIMAL PIC S9(9).
            04 VACATION                       PIC S9(2).
            04 LANGUAGE                       PIC X(3).
            04 FILLER                         PIC X(90).
    02 EMPLOYEES-COUNT                         PIC 9(8) BINARY.
    02 EMPLOYEES OCCURS 1 TO 99 DEPENDING ON EMPLOYEES-COUNT.
        03 IDENT                             PIC X(10).
        03 FIRSTNAME                         PIC X(20).
        03 SURNAME                          PIC X(20).
        03 DATE-BIRTH                       PIC X(12).

```

Two approaches for extracting the COBOL server are described below:

- **Fast-track**

Learn how EntireX helps you to connect the COBOL Server with quick results.

- **User-defined Mapping**

The following features are among those available with a user-defined mapping:

- **Shape the interface to your COBOL server**

You can minimize the interface by suppressing COBOL server fields or by providing constant values. Renaming interfaces allows you to specify a readable long name. These two features together - constants and renaming of interfaces - are most powerful when multiple interfaces are implemented in a single COBOL server. Each function of the COBOL server triggered by an operation code or function code can be modelled as a separate REST resource method/operation.

■ Select alternative mappings

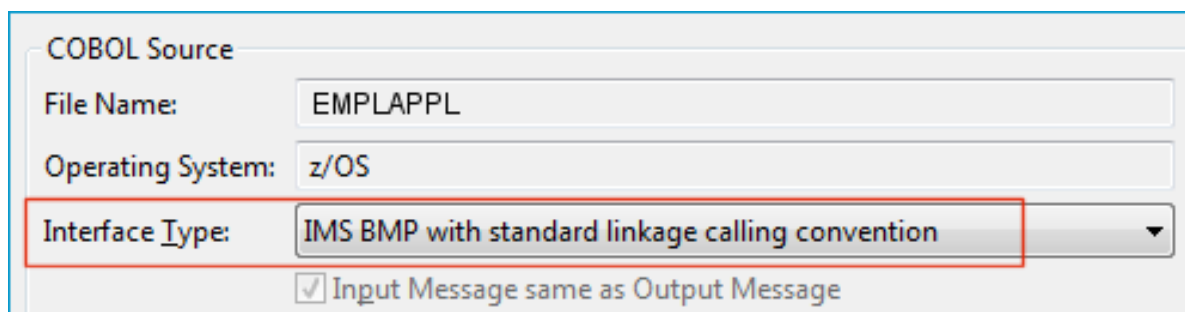
You can select alternative mappings for COBOL data items (REDEFINES, tables etc.) to enable scenarios with COBOL servers where data exchange is not fully described by the LINKAGE SECTION COBOL data items.

These approaches are described under *Fast-track Method* and *Modern Method* below.

Extracting a COBOL Server - Fast-track Method

➤ To extract the COBOL Server

- 1 Switch to the **EntireX** perspective.
- 2 Call the IDL Extractor for COBOL.
- 3 Set the correct interface type as described under *Step 4: Define the Extraction Settings and Start Extraction* under *Scenario I: Create New IDL and Server Mapping Files* in the IDL Extractor for COBOL documentation.



COBOL Source

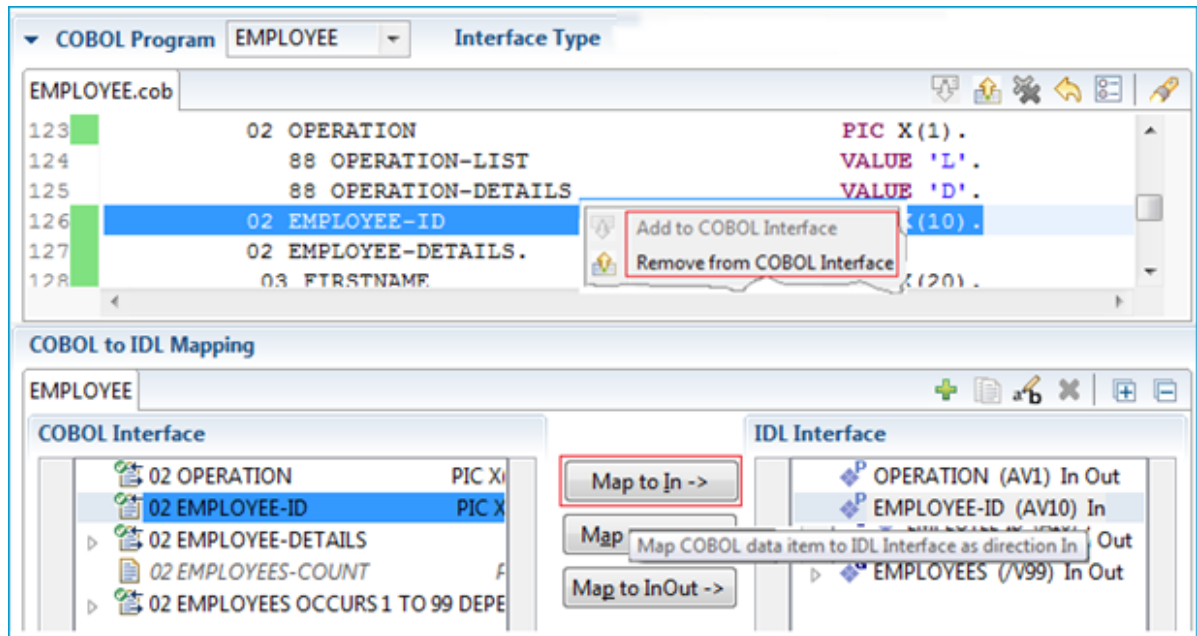
File Name: EMPLAPPL

Operating System: z/OS

Interface Type: IMS BMP with standard linkage calling convention ▼

☒ Input Message same as Output Message

- 4 Press **Next** to enter the COBOL Mapping Editor.



The purpose of the COBOL Mapping Editor is to map COBOL data items to IDL (Interface Definition Language) parameters. It consists of three main areas:

- The upper window pane is the COBOL program source.
- The window pane on the left-hand side is the COBOL interface, consisting of COBOL data items representing the parameters in your COBOL server program.
- The window pane on the right-hand side is the IDL interface consisting of IDL parameters.

For more information see *Mapping Editor User Interface* in the IDL Extractor for COBOL documentation.

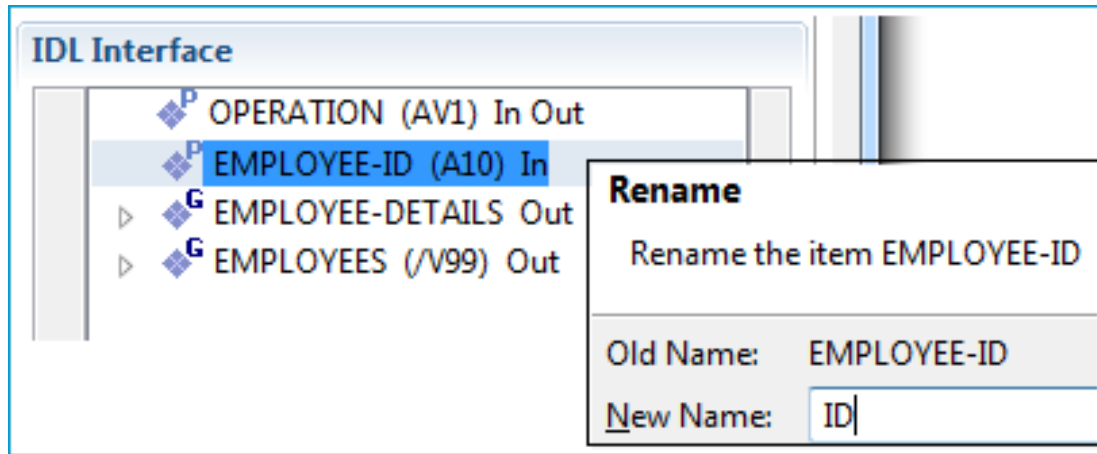
COBOL extraction is a two-step process:

- First, select the COBOL parameters using the **Add to** and **Remove from COBOL Interface** from the context menu on the COBOL data items in the upper window pane (COBOL source); in many situations the COBOL interface is detected by the Mapping Editor directly and you can omit this step.
- Then map the COBOL parameters to IDL directions using the **Map to In/Out/InOut** buttons. COBOL parameters do not have directions, they are always INOUT. This step can be left out for a quick test in non-production scenarios; for real production it is recommended to set IDL directions to reduce data transfer. You also have the option to assign a meaningful name to an IDL parameter if the original COBOL data item is not self-explanatory.

For our example:

- Set **Map to In** for COBOL data item `EMPLOYEE-ID` as IDL direction (see screenshot in previous step). For `EMPLOYEES-DETAILS` and `EMPLOYEES` perform the similar **Map to Out**.

- Use the function **Rename** from the context menu on the IDL parameters in the **IDL Interface** pane to rename COBOL parameter `EMPLOYEES-ID` to IDL parameter `ID`. Do the same for `EMPLOYEE-DETAILS`: Rename to `EMPLOYEE`.



- 5 Press **Finish**. The default extraction settings ensure you get useful results. The outcome is a Software AG IDL file (interface definition language), a Designer file with extension `.idl`:

```
library 'EMPLAPPL' is
program 'EMPLOYEE' is
  define data parameter
    1 OPERATION (AV1) In Out
    1 ID (AV10) In /* Original Name:EMPLOYEE-ID
    1 EMPLOYEE Out /* Original Name:EMPLOYEE-DETAILS
      2 FIRSTNAME (AV20)
      2 SURNAME (AV20)
      2 DATE-BIRTH (AV12)
      2 DETAILS (AV100)
    1 EMPLOYEES (/V99) Out
      2 IDENT (AV10)
      2 FIRSTNAME (AV20)
      2 SURNAME (AV20)
      2 DATE-BIRTH (AV12)
  end-define
```

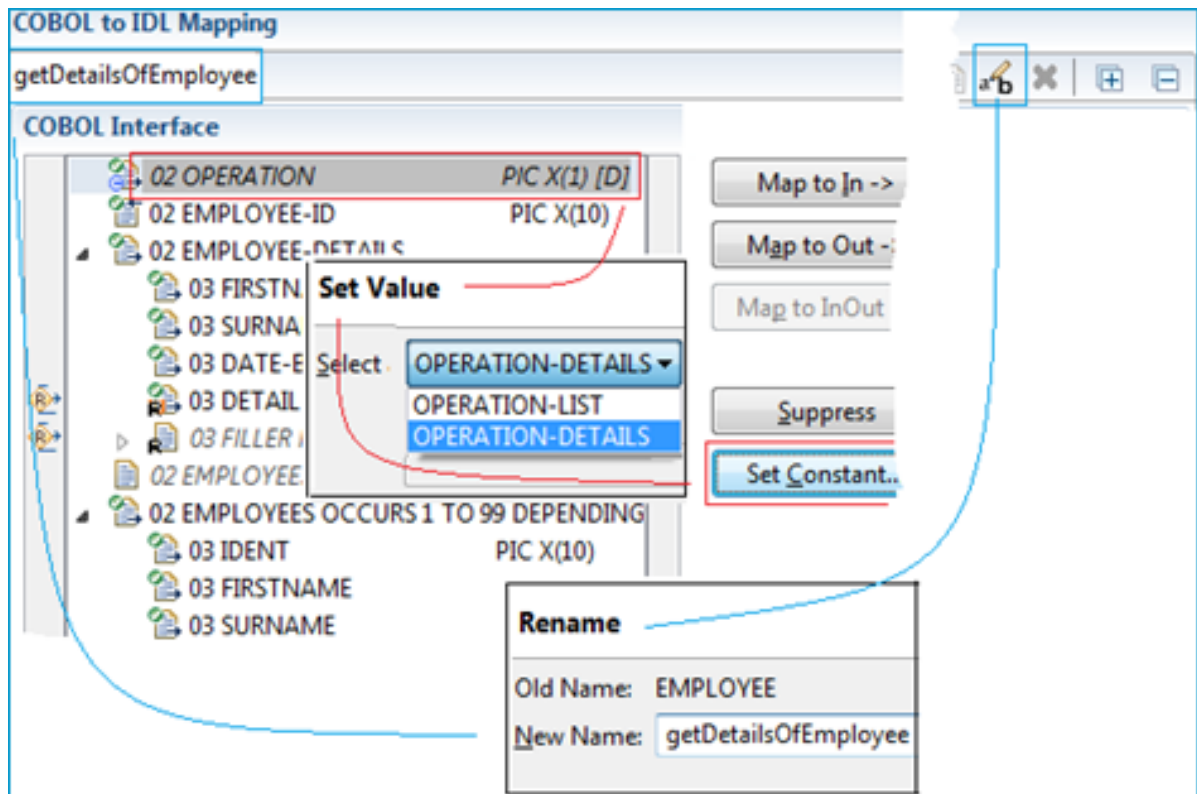
If required, a server mapping file with the same name as the IDL file but the extension `.cvm` may be generated too, for technical reasons. This file must always be stored together with the concomitant IDL file. For more information, see *Server Mapping Files for COBOL* in the Designer documentation. By default, the interface name `EMPLOYEE` is taken from the COBOL server program name.


Extracting a COBOL Server - Modern Method with User-defined Mapping

EntireX allows you to extract all implemented interfaces of the COBOL server separately. Instead of a large `EMPLOYEE` interface, separate interfaces `getListOfEmployees` and `getDetailsOfEmployee` are extracted. Each interface contains required parameters; obsolete parameters for an interface are suppressed, improving its usability.

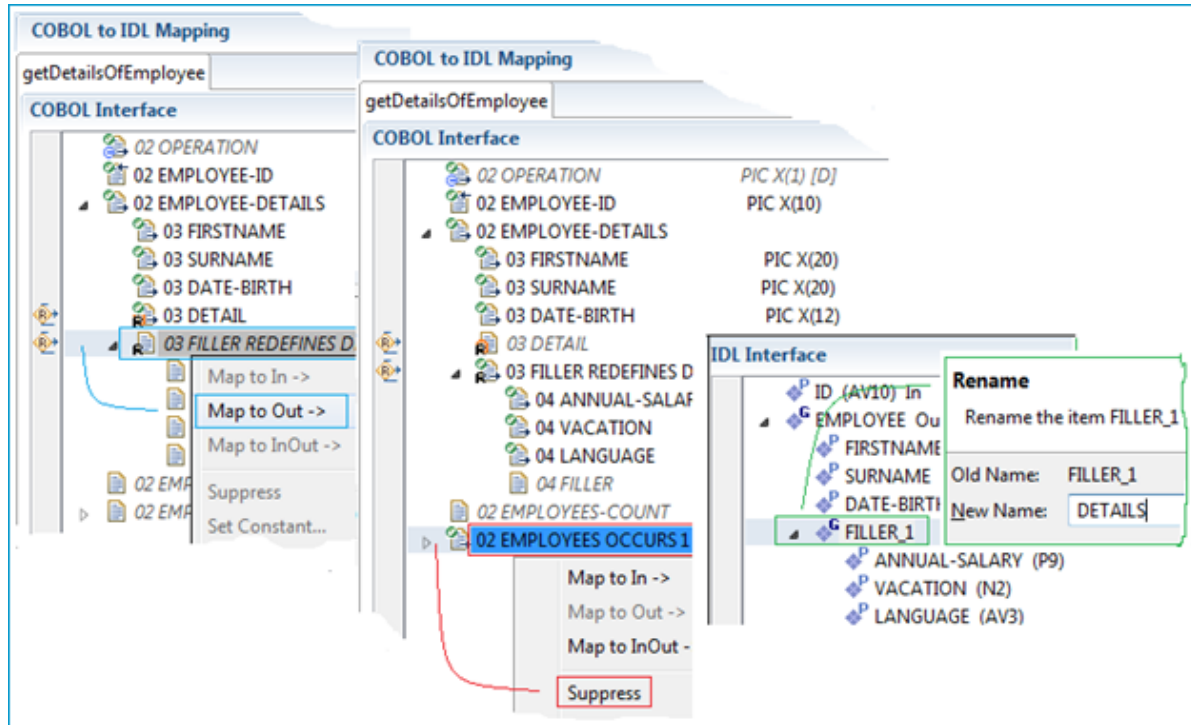
➤ To extract a COBOL server with a user-defined mapping

- 1 Shape `EMPLOYEE` to `getDetailsOfEmployee`:



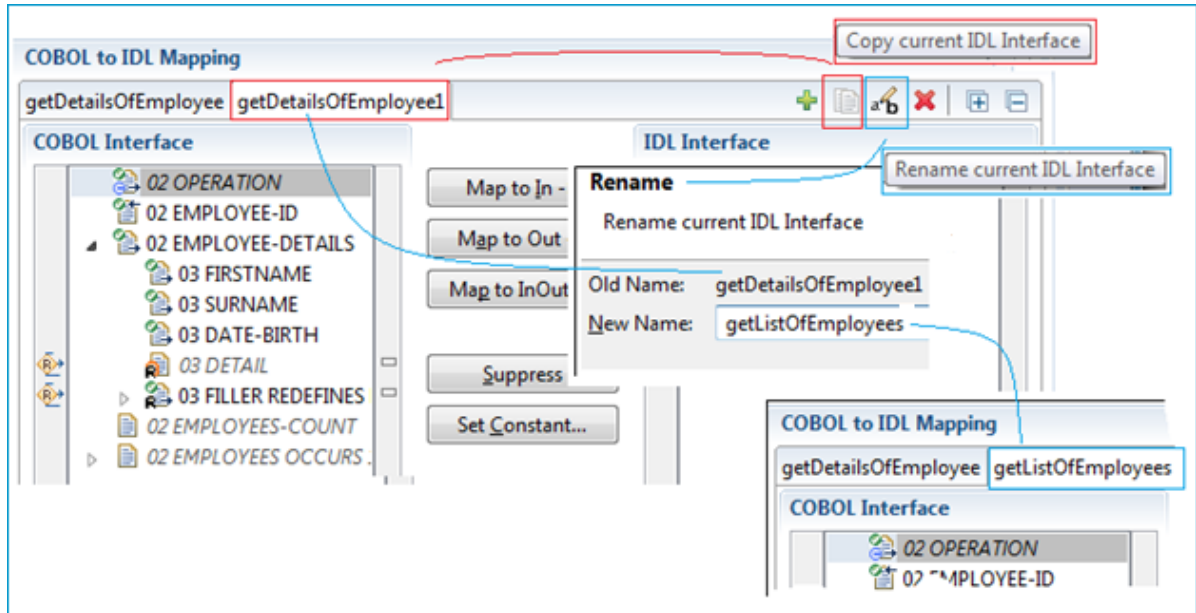
- Mark the `OPERATION` field in the COBOL **Interface** pane and press the **Set Constant** button. In the pop-up window, select `OPERATION-DETAILS` (red markers). The `OPERATION-DETAILS` value originates from COBOL level-88 data item (blue markers). If there is no COBOL level-88 enumeration type in the COBOL program defined, you can enter the function code manually in the pop-up window.
- By default the IDL interface gets the same name as your COBOL server: `EMPLOYEE`. Use the **Rename** button  from the toolbar of the COBOL Mapping Editor and change the IDL interface name to `getDetailsOfEmployee` (blue markers). This name is used as the IS service name later.

- 2 Continue shaping, making `getDetailsOfEmployee` easier to use:





- Use the context menu of the redefinition of the `DETAIL` field in the **COBOL Interface** pane, `FILLER REDEFINES DETAIL` and choose **Map to Out** (blue markers). A redefinition is a secondary layout (field types) of the same memory area. The redefinition contains more specific information (`ANNUAL-SALARY`, `VACATION`, etc.) than character buffer `DETAIL`. This is the reason why it is more useful in your interface.
- Use the context menu of `FILLER_1` in the IDL interface pane and rename `FILLER_1` to `DETAILS` (green markers) for a readable IS service field name later.
- The `DETAILS` function in the COBOL server does not make use of the `EMPLOYEES` field. Therefore you can leave it out of the interface: use the context menu of the `EMPLOYEES` field in the **COBOL Interface** pane and choose **Suppress** (red markers).

3 Shape `EMPLOYEE` to `getListOfEmployees` Function:

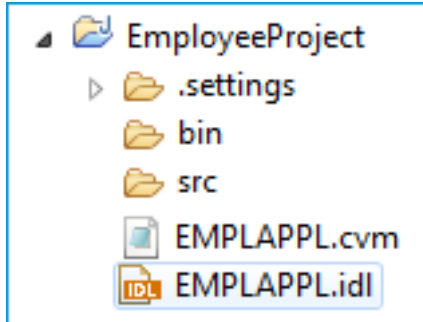


The COBOL Mapping Editor allows you to define multiple IDL interfaces to the same COBOL server:

- You create the additional IDL interface using the toolbar button **Copy current IDL interface**  (red markers). A new tab `getDetailsOfEmployee1` is created. The previously extracted `getDetailsOfEmployee` interface still exists in the first tab. Once you reactivate the first tab, you will see the interface of `getDetailsOfEmployee` again.
- Use the **Rename** button  from the toolbar of the COBOL Mapping Editor and change the IDL interface name to **getListOfEmployees** (blue markers). This name is used as the IS service name later.
- For the **LIST** function you need the **OPERATION-*LIST*** value in the **OPERATION** field: Mark the **OPERATION** field in the **COBOL Interface** pane and press the **Set Constant** button. In the pop-up window, select **OPERATION-*LIST*** (similar to Step 1. above *Shape EMPLOYEE to getDetailsOfEmployee*; red markers).
- Use the context menu of the **EMPLOYEES** field in the **COBOL Interface** pane, **EMPLOYEES OCCURS 1 TO 99** and choose **Map to Out** (Similar to Step 2. above *Continue shaping, making getDetailsOfEmployee easier to use*; blue markers).
- Because the **EMPLOYEE** and **EMPLOYEE-DETAIL** fields are not used in the COBOL server **LIST** function, you leave them out in the IDL interface: use the context menu of **EMPLOYEE-DETAIL** field in the **COBOL Interface** pane and choose **Suppress** (similar to Step 2. above *Continue shaping, making getDetailsOfEmployee easier to use*; red markers). Do the same for the **EMPLOYEE-ID** field.

- 4 Press **Finish** to retrieve the extraction result in the form of a Software AG IDL file. At the same time, a server mapping file for COBOL (Designer file with extension `.cvm`) is created.

See *Server Mapping Files for COBOL* in the Designer documentation. The *Software AG IDL File* in the IDL Editor documentation describes the interfaces from the client point of view, while the server mapping file contains the mapping to the real COBOL server. Both of these files must be kept together and in sync, otherwise a call to the COBOL server may fail.



To summarize: We created two IDL interfaces. In the IDL file, these resulted in two IDL programs: `getListOfEmployees` and `getDetailsOfEmployee`. Both IDL programs were given readable names (Steps 1 and 3). Meaningful fields were kept, while superfluous fields were suppressed (Steps 1 and 3). The program `getDetailsOfEmployee` contains the redefined fields of parameter `DETAILS` (see below) mapped during extraction (Step 1).



Note: At runtime, the RPC client generated with the extracted interface will send data for the redesigned interfaces, while your COBOL server still expects `EMPLOYEE` data. The EntireX runtime transforms the incoming data stream from the RPC client, using the server mapping file.

```

program 'getDetailsOfEmployee' is
  define data parameter
    1 ID (AV10) In /* Original Name:EMPLOYEE-ID
    1 EMPLOYEE Out /* Original Name:EMPLOYEE-DETAILS
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
    2 DETAILS /* Original Name:FILLER_1
    3 ANNUAL-SALARY (P9)
    3 VACATION (N2)
    3 LANGUAGE (AV3)
  end-define
  program 'getListOfEmployees' is
    1 EMPLOYEES (/V99) Out
    2 IDENT (AV10)
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
  end-define

```

Comparison Chart: Fast-track versus User-defined Mapping

The following table highlights the differences when shaping an interface with a *user-defined mapping* compared with a *fast-path extraction*.

	Interface Shaping with User-defined Mapping	Fast-path Extraction
General	User-defined interface with dedicated mapping. For our example, the interfaces are small and tidy and more self-explanatory with long, readable interface names. They are easier to use with the hidden <code>OPERATION</code> field and the suppressed COBOL fields, which are not needed.	Automatic, quick result extraction with COBOL-like interfaces on IS. For our example, the IS interface <code>EMPLOYEE</code> matches exactly 1:1, field by field.
IS service(s)	Multiple small and handy IS services; each <code>OPERATION</code> code is mapped to a separate IS service.	One big IS service.
IS service name	Readable long name.	Short Subprogram name; up to 8 characters.
IS fields	Usage of Suppress and Set Constant reduces the message length. This keeps focus on relevant data items and keeps the client's interface clean. It may also improve performance.	The IS fields and COBOL server parameters match 1:1. As COBOL layout descriptions are sometimes used for many different purposes, irrelevant data items appear and clutter up the IS interface.
<code>OPERATION</code> parameter	Suppressed: the <code>OPERATION</code> parameter does not exist in the IS service as an IS field.	The <code>OPERATION</code> parameter exists in the IS service as an IS field and needs to be filled-in by the client endpoint.
<code>OPERATION</code> code	The <code>OPERATION</code> code is provided internally by EntireX runtime in the <code>OPERATION</code> field.	The <code>OPERATION</code> code needs to be specified in the IS service in the <code>OPERATION</code> field by the client endpoint.
<code>REDEFINE</code> parameters	Either the parameter that is redefined or one of its redefinitions is available as an IS field.	Only the parameter that is redefined is available as an IS field. Redefinitions of the parameter are not available as IS fields.

A user-defined mapping enables you also to define alternative mappings for COBOL data items (`REDEFINES`, tables etc.). These enable scenarios with COBOL servers where data exchange is not fully described by the `LINKAGE SECTION` COBOL data items. For more information refer to the IDL Extractor for COBOL documentation:

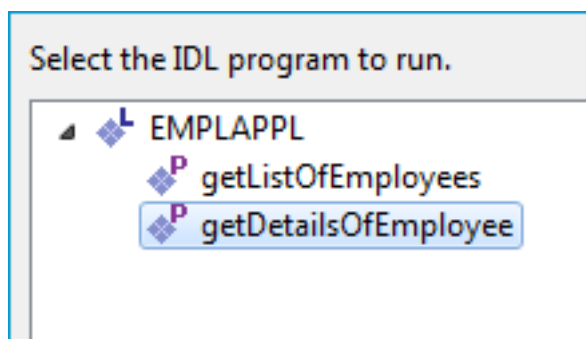
- *Mapping Editor IDL Interface Mapping Functions*
- *User-defined Mapping*

Testing the Extraction Results

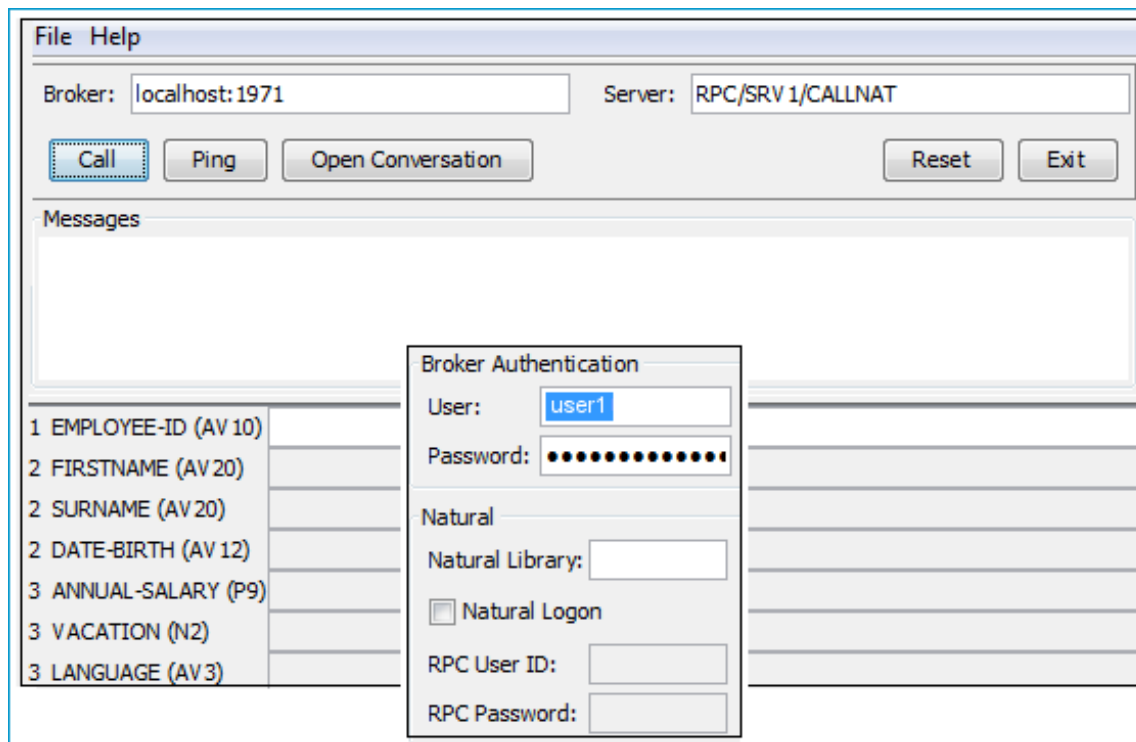
The following pictures use the extraction results described under *Extracting a COBOL Server - Modern Method with User-defined Mapping*.

> To test the extraction results (optional)

- 1 You can test the results of the extraction operation and the REST server back end, using the EntireX IDL Tester. From the context menu of the IDL file in the Designer, choose **Software AG IDL Tester**.



- 2 Select `getDetailsOfEmployee`.



Note that the broker and server parameters contain the explicit route to call the server program, and you can optionally ping the connection from this client. With the **File > Options** dialog, you can set:

- **User** and **Password** for *broker* authentication
- **RPC User ID** and **RPC Password** for *server* authentication

See *EntireX IDL Tester* in the Designer documentation.

- 3 Check the Integration Server log, the EntireX Adapter log or the RPC logs. Applies to all connection methods.

Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server

This section describes your first steps to create a new Integration Server connection. This is described in more detail under *Using the Integration Server Wrapper*, for example working with existing Integration Server connections. This section covers the following topics:

- [Step 1: Start the Integration Server Wrapper Wizard](#)
- [Step 2: Create a New Integration Server Connection](#)
- [Step 3: Select the Connection Type](#)
- [Step 4: Define Adapter Services for an RPC Connection](#)
- [Step 5: Create or Update a REST Resource](#)
- [Step 6: Edit the REST Resource](#)

Step 1: Start the Integration Server Wrapper Wizard

➤ To start the Integration Server Wrapper wizard

- 1 In the context menu of a Software AG IDL file, choose **Integration Server > Generate web-Methods IS Connection**.

This starts the wizard with a list of existing Integration Server Wrapper connections.



Note: If the selected IDL file is not valid because of a syntax error, an error dialog comes up and the wizard does not start.

- 2 Continue with *Step 2: Create a New Integration Server Connection*.

Step 2: Create a New Integration Server Connection

Define New Integration Server Connection

Add host:port, user and password for a new Integration Server

Server:

User:

Password:

☐ Use secure connection

➤ To create a new Integration Server connection

- 1 Define the new Integration Server connection on the wizard page.



Notes:

1. The only required field is **Server**. Enter the hostname of the Integration Server including an optional port number. If no port number is specified, port number defaults to "5555". The **Integration Server Authentication** can be passed with the **User** and **Password** fields.
2. Optional settings are for secure connections. The **Truststore for HTTPS** contains all signed certificates and must be a valid truststore.
3. The check box **Verify host name** checks that the hostname is entered in the stored certificate.
4. When the Integration Server has **Client Authentication** enabled, you can specify your **Keystore** file and keystore **Password**.
5. For managing Integration Server connections, see [Preferences](#).

- 2 Choose **Next** and continue with *Step 3: Select the Connection Type*.

Step 3: Select the Connection Type

☒ Create a new EntireX Adapter connection of type: **EntireX RPC Connection**

☐ Update Adapter Services or Listeners to an existing Integration Server connection:

List of Connections

Connection Name	Package Name	Connection Type	Enabled
Employees:EmployeesConnection	Employees	EntireX RPC Connection	Yes

Total: 1

☒ Map Software AG IDL data types to String

☒ Create or Update REST resource

> To create a new connection

- 1 Select a connection type from the drop down list. Connection types are described under *EntireX Adapter Connections* in the EntireX Adapter documentation and *Introduction to the Integration Server Wrapper*.



Note: The list of connection types is filtered: connection types that require a license are only shown if a corresponding license file is available. Reliable RPC connections are only shown if all IDL programs contain only IN parameters. Also, if a server mapping file is available, only those connection types that support the interface type specified in the server mapping file are shown.

- 2 Check the box **Create or Update REST resource**.
- 3 Click **Next** and continue with *Step 4: Define Adapter Services for an RPC Connection*.

Step 4: Define Adapter Services for an RPC Connection

Packages on Integration Server localhost:5555

- Default
- Employees**
- WmART
- WmARTEstDC

Folder Name:

Connection Name:

RPC Connection to EntireX

Broker ID:

Server Address:

> To create a connection and related adapter services

- 1 Select an existing package or create a new package for the created objects.
- 2 Define a folder name. If the folder does not exist, it will be created.
- 3 Define a connection name.
- 4 Define the parameters of the connection type. For details, see *Connection Parameters for RPC Connections* in the EntireX Adapter documentation.

As a result, the folder will contain the connection and the adapter services (one for each IDL program). The name of a service is the same as the respective IDL program.

The default settings for the adapter services are:

- the **Default** package; if not available, the first package
- the IDL library name for the **Folder Name**
- the IDL library name with the suffix "Connection" for the **Connection Name**



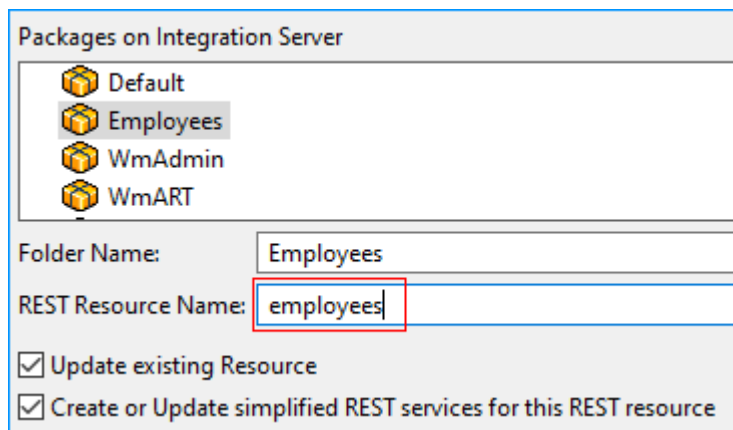
Note: When creating a connection, a package dependency is added such that the selected package depends on webMethods EntireX (the package `WmEntireX`) with the version currently used.

Step 5: Create or Update a REST Resource

➤ To create or update a REST resource

Make sure checkbox **Create or Update REST resource** is checked.

- 1 Select a package.



Packages on Integration Server

- Default
- Employees**
- WmAdmin
- WmART

Folder Name: Employees

REST Resource Name: employees

☒ Update existing Resource

☒ Create or Update simplified REST services for this REST resource

- 2 Define a folder name. If the folder does not exist, it will be created.
- 3 Define the REST Resource name.
- 4 Check **Update existing Resource** if you want to add additional services to an existing resource.
- 5 Check **Create or Update simplified REST services for this REST resource** if you want to generate additional services with a simplified REST API. This allows you to use a simple HTTP-GET method without a payload.

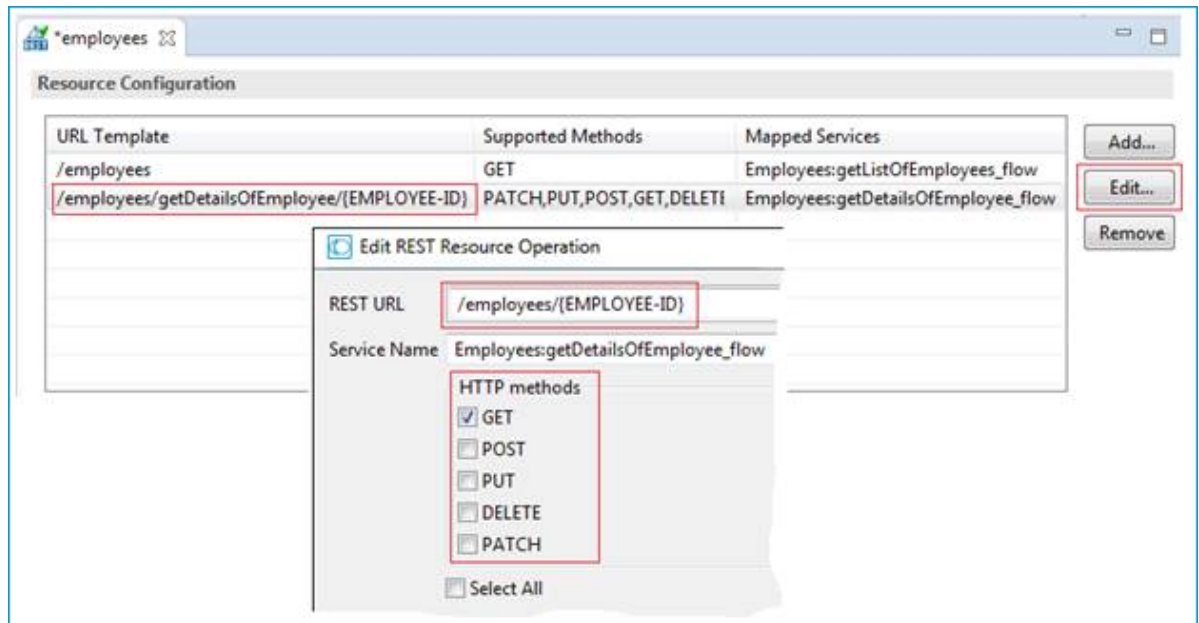


Note: This applies only to APIs where input signature parameters do not contain arrays.

Step 6: Edit the REST Resource

➤ To edit the REST resource

- 1 Switch to the **Service Development** perspective of Software AG Designer.



- 2 Refresh the package where you generated the REST Resource in the previous step.
- 3 Open the REST Resource.
- 4 Edit the REST URLs and HTTP Methods to match the functionality of your application.

For more information see:

- *REST Developer's Guide* in the webMethods Integration Server documentation for information on using the generated REST resources
- *REST Developer's Guide* in the webMethods Integration Server documentation

Task 3: Execute the Call from the REST Client to COBOL

Use an appropriate REST client, specify URL, Method and Headers and invoke your REST resource:

URL http://localhost:5555/restv2/employees

Method GET

Headers Accept application/json

200 OK

```
{
  "errorCode": "00000000",
  "errorFlag": "false",
  "response": {
    "EMPLOYEES-LIST": {
      "EMPLOYEES": [
        {
          "IDENT": "11100102",
          "FIRSTNAME": "Edgar",
          "SURNAME": "Schindler",
          "DATE-BIRTH": "Dec 4, 1962"
        },
        {
          "IDENT": "11100105",
          "FIRSTNAME": "Christian",
          "SURNAME": "Schirm",
          "DATE-BIRTH": "Mar 15, 1961"
        },
        {
          "IDENT": "11100106",
          "FIRSTNAME": "Rainer",
          "SURNAME": "Schmitt",
          "DATE-BIRTH": "Feb 13, 1955"
        },
        {
          "IDENT": "11100107",
          "FIRSTNAME": "Helga",
          "SURNAME": "Schmidt",
          "DATE-BIRTH": "May 26, 1961"
        }
      ]
    }
  }
}
```

URL http://localhost:5555/restv2/employees/111000105

Method GET

Headers Accept application/json

200 OK

```
{
  "errorCode": "00000000",
  "errorFlag": "false",
  "response": {
    "EMPLOYEE-DETAILS": {
      "FIRSTNAME": "Christian",
      "SURNAME": "Schirm",
      "DATE-BIRTH": "Mar 15, 1961",
      "DETAIL": {
        "ANNUAL-SALARY": "68000",
        "VACATION": "21",
        "LANGUAGE": "GER"
      }
    }
  }
}
```

17

Calling COBOL on BS2000 from REST

■ Introduction	305
■ What do I need to Install for this Scenario?	306
■ Task 1: Extract the Interface of a COBOL Server	307
■ Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server	319
■ Task 3: Execute the Call from the REST Client to COBOL	325

This scenario uses the tools IDL Extractor for COBOL and Integration Server Wrapper of the Designer.

Introduction



- ① Extract the interface of the COBOL server program. See *Using the IDL Extractor for COBOL - Overview*.
- ② Generate REST resources, connection and adapter services in Integration Server.
- ③ Execute the call from the REST client to the COBOL server program.

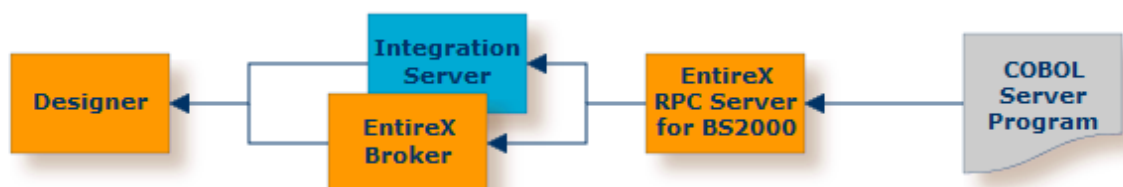
This scenario makes the following important assumptions:

For Task 1:

- You have a working COBOL batch server. For illustration and examples on such a server, see *Batch with Standard Linkage Calling Convention* under *Introduction to the IDL Extractor for COBOL*.
- You have access to the related COBOL sources and copybooks. The minimum requirement is the `DATA DIVISION` of the interface. The sources and copybooks must be stored either
 - *locally*, that is, on the same machine where the Designer is running



- or *remotely* in LMS libraries and accessed via the *RPC Server for BS2000* and either EntireX Broker or Integration Server



For Tasks 2 and 3:

- You have a REST client.
- You can call the COBOL server program 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
 - the *RPC Server for BS2000*



- For the *EntireX Direct RPC* connection method you need:
 - the *RPC Server for BS2000*



See also *Direct RPC* in the EntireX Adapter documentation.

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.
- You have an RPC Server for BS2000 installed. See *Installing the RPC Server for BS2000* in the BS2000 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.

Task 1: Extract the Interface of a COBOL Server

- [Introduction](#)
- [Sample COBOL Server used in the Examples](#)
- [Extracting a COBOL Server - Fast-track Method](#)
- [Extracting a COBOL Server - Modern Method with User-defined Mapping](#)
- [Comparison Chart: Fast-track versus User-defined Mapping](#)
- [Testing the Extraction Results](#)

Introduction

Follow the instructions for extracting COBOL, see *Using the IDL Extractor for COBOL - Overview* and choose *Scenario I: Create New IDL and Server Mapping Files* if this is your first extraction. Learn how to invoke the IDL Extractor for COBOL, create a COBOL Extractor Environment and about the possibilities to select the COBOL source.

The extraction process creates the following EntireX metafiles:

- IDL file. A Software AG IDL file contains definitions of the interface between client and server. See *Software AG IDL File* in the IDL Editor documentation in the IDL Editor documentation.
- Server mapping file (optional). The mapping file is a Designer file with extension .cvm that contains COBOL-specific mapping information. See *Server Mapping Files for COBOL* in the Designer documentation.

Sample COBOL Server used in the Examples

The following COBOL server is used to illustrate the features of the IDL Extractor for COBOL. Imagine an existing COBOL server called `EMPLOYEE`. It implements the access logic for a `LIST` and `DETAILS` function to a database view `EMPLOYEE`:

```

IDENTIFICATION DIVISION.
    PROGRAM-ID.      EMPLOYEE.
*****
*   Program Name    = EMPLOYEE
*   String Literal  = QUOTE
*****
01 DFHCOMMAREA.
    02 OPERATION                                PIC X(1).
        88 OPERATION-LIST                      VALUE 'L'.
        88 OPERATION-DETAILS                  VALUE 'D'.
    02 EMPLOYEE-ID                                PIC X(10).
    02 EMPLOYEE-DETAILS.
        03 FIRSTNAME                            PIC X(20).
        03 SURNAME                              PIC X(20).
        03 DATE-BIRTH                          PIC X(12).
        03 DETAILS                            PIC X(100).
        03 FILLER REDEFINES DETAILS.
            04 ANNUAL-SALARY          PACKED-DECIMAL PIC S9(9).
            04 VACATION                PIC S9(2).
            04 LANGUAGE                PIC X(3).
            04 FILLER                  PIC X(90).
    02 EMPLOYEES-COUNT                          PIC 9(8) BINARY.
    02 EMPLOYEES OCCURS 1 TO 99 DEPENDING ON EMPLOYEES-COUNT.
        03 IDENT                          PIC X(10).
        03 FIRSTNAME                      PIC X(20).
        03 SURNAME                        PIC X(20).
        03 DATE-BIRTH                     PIC X(12).

```

Two approaches for extracting the COBOL server are described below:

- **Fast-track**

Learn how EntireX helps you to connect the COBOL Server with quick results.

- **User-defined Mapping**

The following features are among those available with a user-defined mapping:

- **Shape the interface to your COBOL server**

You can minimize the interface by suppressing COBOL server fields or by providing constant values. Renaming interfaces allows you to specify a readable long name. These two features together - constants and renaming of interfaces - are most powerful when multiple interfaces are implemented in a single COBOL server. Each function of the COBOL server triggered by an operation code or function code can be modelled as a separate REST resource method/operation.

■ Select alternative mappings

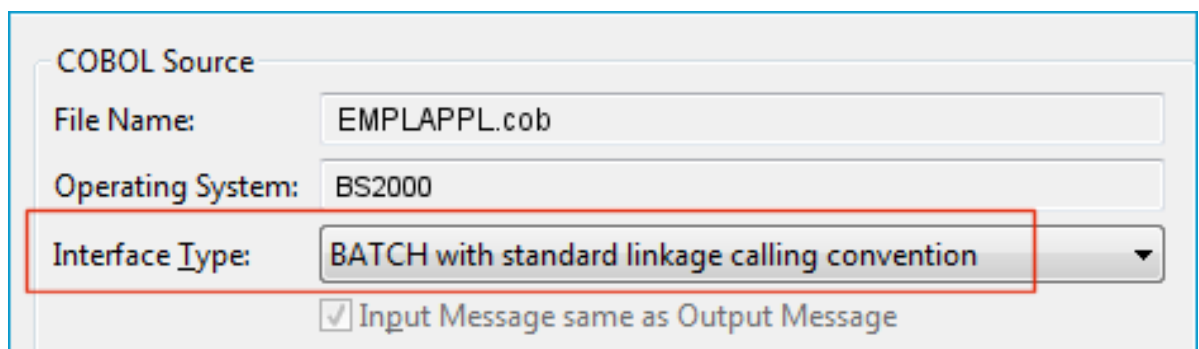
You can select alternative mappings for COBOL data items (REDEFINES, tables etc.) to enable scenarios with COBOL servers where data exchange is not fully described by the LINKAGE SECTION COBOL data items.

These approaches are described under *Fast-track Method* and *Modern Method* below.

Extracting a COBOL Server - Fast-track Method

➤ To extract the COBOL Server

- 1 Switch to the **EntireX** perspective.
- 2 Call the IDL Extractor for COBOL.
- 3 Set the correct interface type as described under *Step 4: Define the Extraction Settings and Start Extraction* under *Scenario I: Create New IDL and Server Mapping Files* in the IDL Extractor for COBOL documentation.



COBOL Source

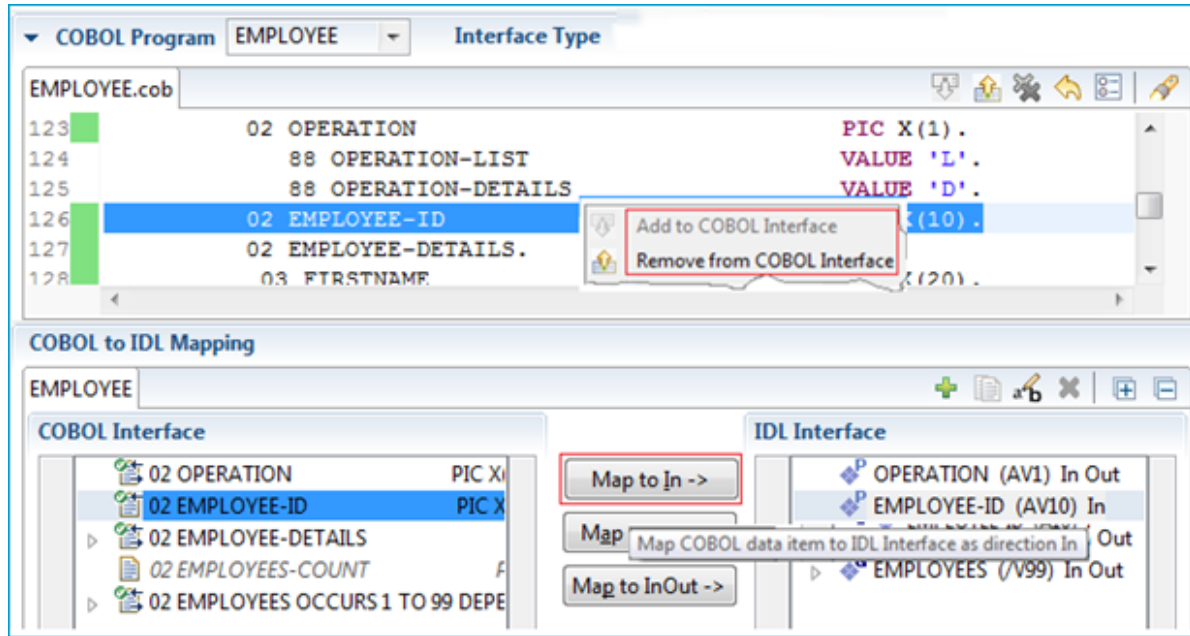
File Name: EMPLAPPL.cob

Operating System: BS2000

Interface Type: BATCH with standard linkage calling convention ▼

☒ Input Message same as Output Message

- 4 Press **Next** to enter the COBOL Mapping Editor.



The purpose of the COBOL Mapping Editor is to map COBOL data items to IDL (Interface Definition Language) parameters. It consists of three main areas:

- The upper window pane is the COBOL program source.
- The window pane on the left-hand side is the COBOL interface, consisting of COBOL data items representing the parameters in your COBOL server program.
- The window pane on the right-hand side is the IDL interface consisting of IDL parameters.

For more information see *Mapping Editor User Interface* in the IDL Extractor for COBOL documentation.

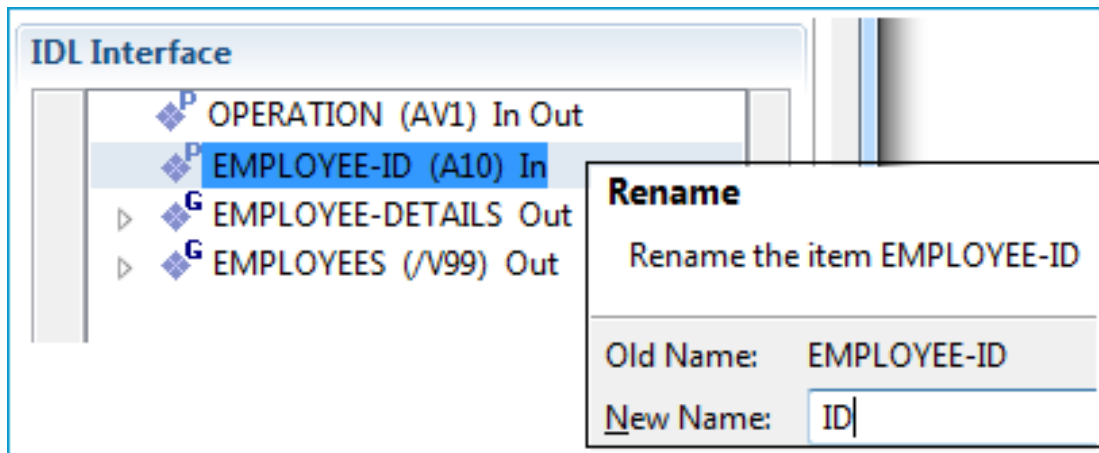
COBOL extraction is a two-step process:

- First, select the COBOL parameters using the **Add to** and **Remove from COBOL Interface** from the context menu on the COBOL data items in the upper window pane (COBOL source); in many situations the COBOL interface is detected by the Mapping Editor directly and you can omit this step.
- Then map the COBOL parameters to IDL directions using the **Map to In/Out/InOut** buttons. COBOL parameters do not have directions, they are always INOUT. This step can be left out for a quick test in non-production scenarios; for real production it is recommended to set IDL directions to reduce data transfer. You also have the option to assign a meaningful name to an IDL parameter if the original COBOL data item is not self-explanatory.

For our example:

- Set **Map to In** for COBOL data item `EMPLOYEE-ID` as IDL direction (see screenshot in previous step). For `EMPLOYEES-DETAILS` and `EMPLOYEES` perform the similar **Map to Out**.

- Use the function **Rename** from the context menu on the IDL parameters in the **IDL Interface** pane to rename COBOL parameter `EMPLOYEE-ID` to IDL parameter `ID`. Do the same for `EMPLOYEE-DETAILS`: Rename to `EMPLOYEE`.



- 5 Press **Finish**. The default extraction settings ensure you get useful results. The outcome is a Software AG IDL file (interface definition language), a Designer file with extension `.idl`:

```
library 'EMPLAPPL' is
program 'EMPLOYEE' is
  define data parameter
    1 OPERATION (AV1) In Out
    1 ID (AV10) In /* Original Name:EMPLOYEE-ID
    1 EMPLOYEE Out /* Original Name:EMPLOYEE-DETAILS
      2 FIRSTNAME (AV20)
      2 SURNAME (AV20)
      2 DATE-BIRTH (AV12)
      2 DETAILS (AV100)
    1 EMPLOYEES (/V99) Out
      2 IDENT (AV10)
      2 FIRSTNAME (AV20)
      2 SURNAME (AV20)
      2 DATE-BIRTH (AV12)
  end-define
```

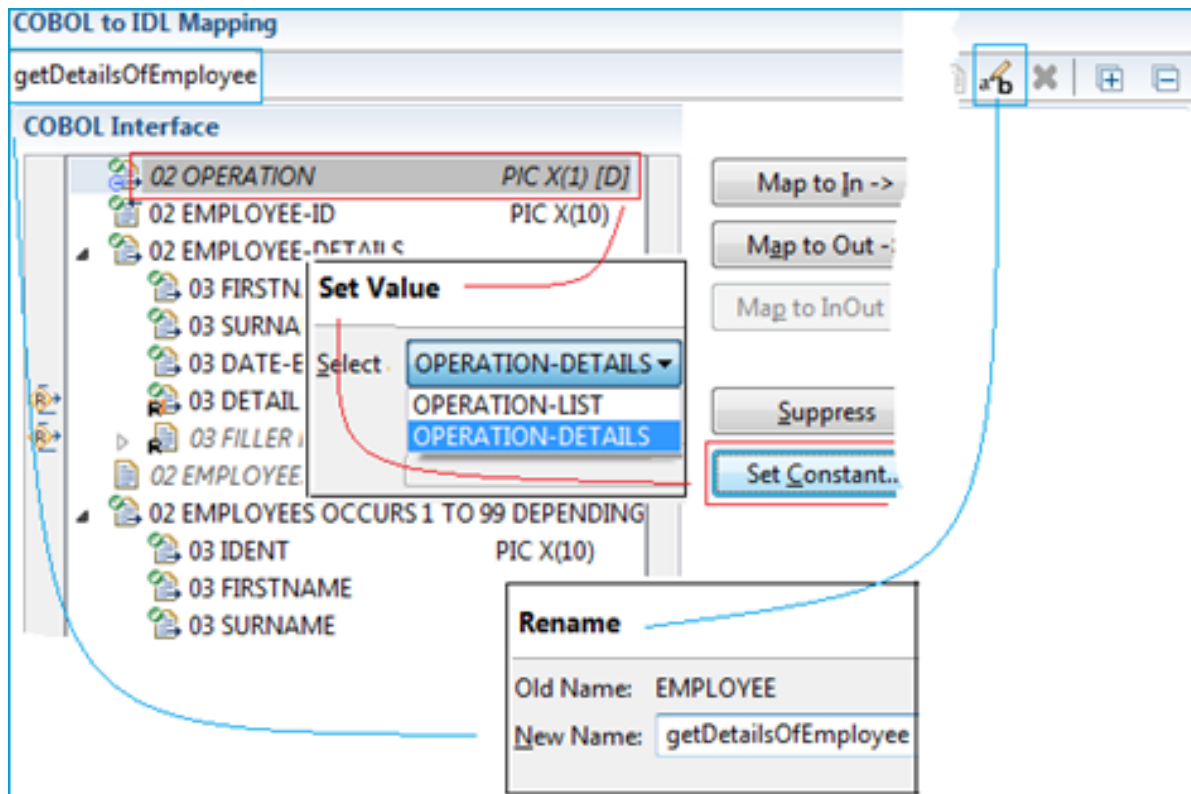
If required, a server mapping file with the same name as the IDL file but the extension `.cvm` may be generated too, for technical reasons. This file must always be stored together with the concomitant IDL file. For more information, see *Server Mapping Files for COBOL* in the Designer documentation. By default, the interface name `EMPLOYEE` is taken from the COBOL server program name.


Extracting a COBOL Server - Modern Method with User-defined Mapping

EntireX allows you to extract all implemented interfaces of the COBOL server separately. Instead of a large `EMPLOYEE` interface, separate interfaces `getListOfEmployees` and `getDetailsOfEmployee` are extracted. Each interface contains required parameters; obsolete parameters for an interface are suppressed, improving its usability.

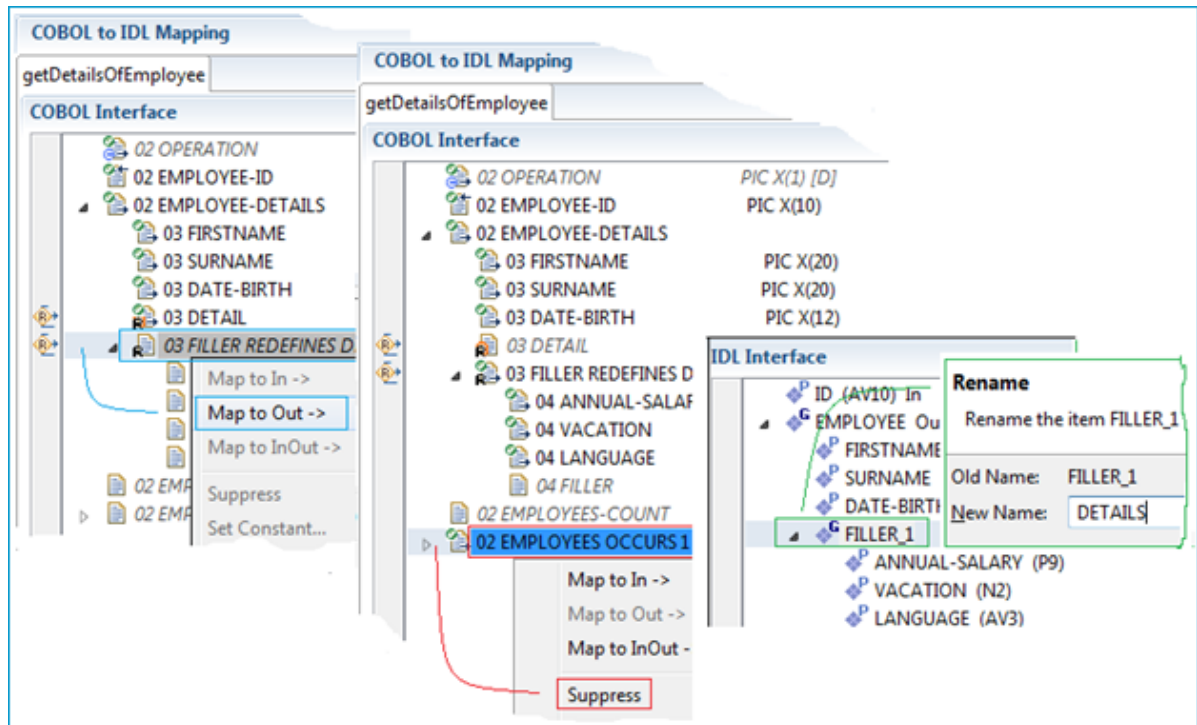
➤ To extract a COBOL server with a user-defined mapping

- 1 Shape `EMPLOYEE` to `getDetailsOfEmployee`:



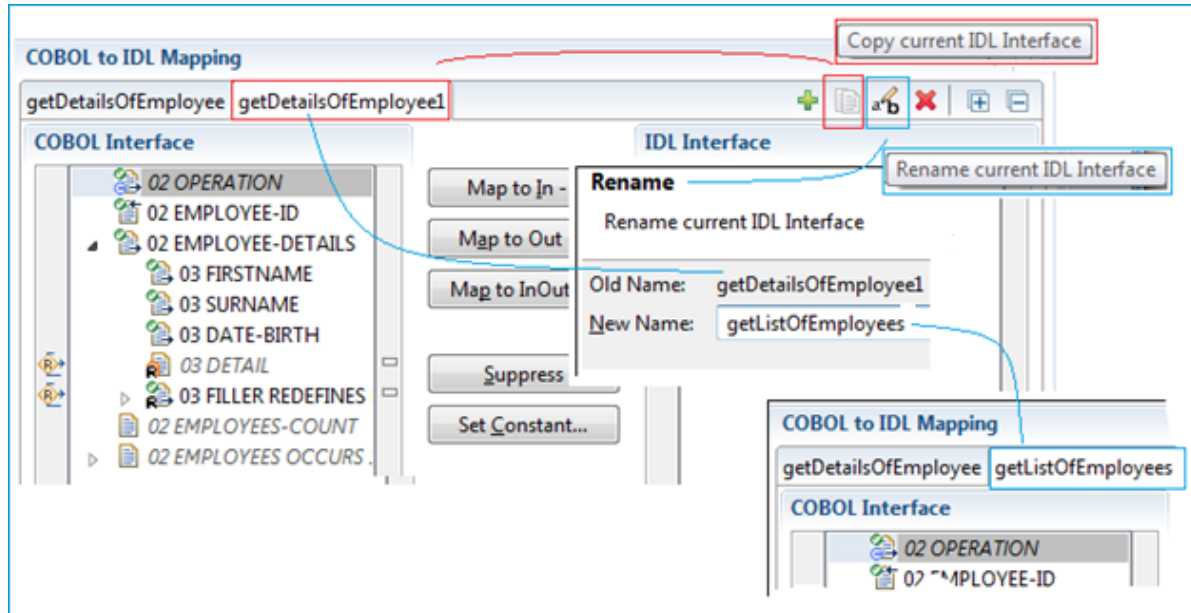
- Mark the `OPERATION` field in the COBOL **Interface** pane and press the **Set Constant** button. In the pop-up window, select `OPERATION-DETAILS` (red markers). The `OPERATION-DETAILS` value originates from COBOL level-88 data item (blue markers). If there is no COBOL level-88 enumeration type in the COBOL program defined, you can enter the function code manually in the pop-up window.
- By default the IDL interface gets the same name as your COBOL server: `EMPLOYEE`. Use the **Rename** button  from the toolbar of the COBOL Mapping Editor and change the IDL interface name to `getDetailsOfEmployee` (blue markers). This name is used as the IS service name later.

- 2 Continue shaping, making `getDetailsOfEmployee` easier to use:




- Use the context menu of the redefinition of the `DETAIL` field in the **COBOL Interface** pane, `FILLER REDEFINES DETAIL` and choose **Map to Out** (blue markers). A redefinition is a secondary layout (field types) of the same memory area. The redefinition contains more specific information (`ANNUAL-SALARY`, `VACATION`, etc.) than character buffer `DETAIL`. This is the reason why it is more useful in your interface.
- Use the context menu of `FILLER_1` in the IDL interface pane and rename `FILLER_1` to `DETAILS` (green markers) for a readable IS service field name later.
- The `DETAILS` function in the COBOL server does not make use of the `EMPLOYEES` field. Therefore you can leave it out of the interface: use the context menu of the `EMPLOYEES` field in the **COBOL Interface** pane and choose **Suppress** (red markers).


3 Shape `EMPLOYEE` to `getListOfEmployees` Function:



The COBOL Mapping Editor allows you to define multiple IDL interfaces to the same COBOL server:

- You create the additional IDL interface using the toolbar button **Copy current IDL interface**

 (red markers). A new tab `getDetailsOfEmployee1` is created. The previously extracted `getDetailsOfEmployee` interface still exists in the first tab. Once you reactivate the first tab, you will see the interface of `getDetailsOfEmployee` again.

- Use the **Rename** button  from the toolbar of the COBOL Mapping Editor and change the IDL interface name to `getListOfEmployees` (blue markers). This name is used as the IS service name later.

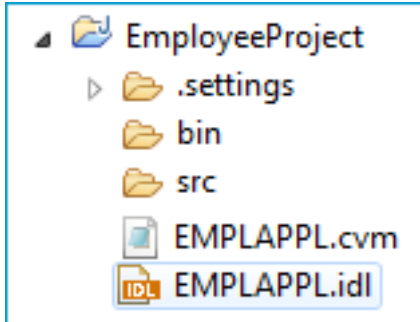
- For the `LIST` function you need the `OPERATION-LIST` value in the `OPERATION` field: Mark the `OPERATION` field in the **COBOL Interface** pane and press the **Set Constant** button. In the pop-up window, select `OPERATION-LIST` (similar to Step 1. above *Shape `EMPLOYEE` to `getDetailsOfEmployee`*; red markers).

- Use the context menu of the `EMPLOYEES` field in the **COBOL Interface** pane, `EMPLOYEES OCCURS 1 TO 99` and choose **Map to Out** (Similar to Step 2. above *Continue shaping, making `getDetailsOfEmployee` easier to use*; blue markers).

- Because the `EMPLOYEE` and `EMPLOYEE-DETAIL` fields are not used in the COBOL server `LIST` function, you leave them out in the IDL interface: use the context menu of `EMPLOYEE-DETAIL` field in the **COBOL Interface** pane and choose **Suppress** (similar to Step 2. above *Continue shaping, making `getDetailsOfEmployee` easier to use*; red markers). Do the same for the `EMPLOYEE-ID` field.

- 4 Press **Finish** to retrieve the extraction result in the form of a Software AG IDL file. At the same time, a server mapping file for COBOL (Designer file with extension `.cvm`) is created.

See *Server Mapping Files for COBOL* in the Designer documentation. The *Software AG IDL File* in the IDL Editor documentation describes the interfaces from the client point of view, while the server mapping file contains the mapping to the real COBOL server. Both of these files must be kept together and in sync, otherwise a call to the COBOL server may fail.



To summarize: We created two IDL interfaces. In the IDL file, these resulted in two IDL programs: `getListOfEmployees` and `getDetailsOfEmployee`. Both IDL programs were given readable names (Steps 1 and 3). Meaningful fields were kept, while superfluous fields were suppressed (Steps 1 and 3). The program `getDetailsOfEmployee` contains the redefined fields of parameter `DETAILS` (see below) mapped during extraction (Step 1).



Note: At runtime, the RPC client generated with the extracted interface will send data for the redesigned interfaces, while your COBOL server still expects `EMPLOYEE` data. The EntireX runtime transforms the incoming data stream from the RPC client, using the server mapping file.

```

program 'getDetailsOfEmployee' is
  define data parameter
    1 ID (AV10) In /* Original Name:EMPLOYEE-ID
    1 EMPLOYEE Out /* Original Name:EMPLOYEE-DETAILS
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
    2 DETAILS /* Original Name:FILLER_1
    3 ANNUAL-SALARY (P9)
    3 VACATION (N2)
    3 LANGUAGE (AV3)
  end-define
  program 'getListOfEmployees' is
    1 EMPLOYEES (/V99) Out
    2 IDENT (AV10)
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
  end-define

```

Comparison Chart: Fast-track versus User-defined Mapping

The following table highlights the differences when shaping an interface with a *user-defined mapping* compared with a *fast-path extraction*.

	Interface Shaping with User-defined Mapping	Fast-path Extraction
General	User-defined interface with dedicated mapping. For our example, the interfaces are small and tidy and more self-explanatory with long, readable interface names. They are easier to use with the hidden <code>OPERATION</code> field and the suppressed COBOL fields, which are not needed.	Automatic, quick result extraction with COBOL-like interfaces on IS. For our example, the IS interface <code>EMPLOYEE</code> matches exactly 1:1, field by field.
IS service(s)	Multiple small and handy IS services; each <code>OPERATION</code> code is mapped to a separate IS service.	One big IS service.
IS service name	Readable long name.	Short Subprogram name; up to 8 characters.
IS fields	Usage of Suppress and Set Constant reduces the message length. This keeps focus on relevant data items and keeps the client's interface clean. It may also improve performance.	The IS fields and COBOL server parameters match 1:1. As COBOL layout descriptions are sometimes used for many different purposes, irrelevant data items appear and clutter up the IS interface.
<code>OPERATION</code> parameter	Suppressed: the <code>OPERATION</code> parameter does not exist in the IS service as an IS field.	The <code>OPERATION</code> parameter exists in the IS service as an IS field and needs to be filled-in by the client endpoint.
<code>OPERATION</code> code	The <code>OPERATION</code> code is provided internally by EntireX runtime in the <code>OPERATION</code> field.	The <code>OPERATION</code> code needs to be specified in the IS service in the <code>OPERATION</code> field by the client endpoint.
<code>REDEFINE</code> parameters	Either the parameter that is redefined or one of its redefinitions is available as an IS field.	Only the parameter that is redefined is available as an IS field. Redefinitions of the parameter are not available as IS fields.

A user-defined mapping enables you also to define alternative mappings for COBOL data items (`REDEFINES`, tables etc.). These enable scenarios with COBOL servers where data exchange is not fully described by the `LINKAGE SECTION` COBOL data items. For more information refer to the IDL Extractor for COBOL documentation:

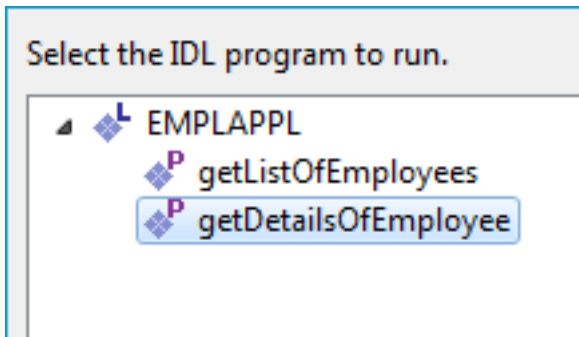
- *Mapping Editor IDL Interface Mapping Functions*
- *User-defined Mapping*

Testing the Extraction Results

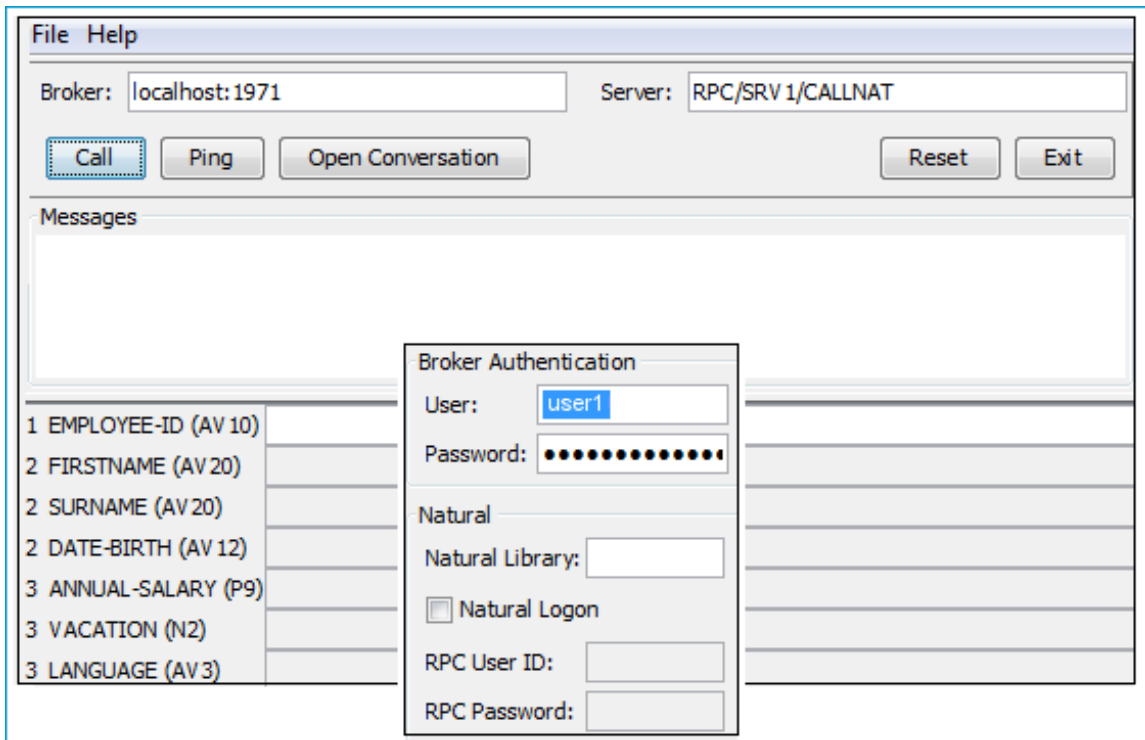
The following pictures use the extraction results described under *Extracting a COBOL Server - Modern Method with User-defined Mapping*.

> To test the extraction results (optional)

- 1 You can test the results of the extraction operation and the REST server back end, using the EntireX IDL Tester. From the context menu of the IDL file in the Designer, choose **Software AG IDL Tester**.



- 2 Select `getDetailsOfEmployee`.



Note that the broker and server parameters contain the explicit route to call the server program, and you can optionally ping the connection from this client. With the **File > Options** dialog, you can set:

- **User** and **Password** for *broker* authentication
- **RPC User ID** and **RPC Password** for *server* authentication

See *EntireX IDL Tester* in the Designer documentation.

- 3 Check the Integration Server log, the EntireX Adapter log or the RPC logs. Applies to all connection methods.

Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server

This section describes your first steps to create a new Integration Server connection. This is described in more detail under *Using the Integration Server Wrapper*, for example working with existing Integration Server connections. This section covers the following topics:

- [Step 1: Start the Integration Server Wrapper Wizard](#)
- [Step 2: Create a New Integration Server Connection](#)
- [Step 3: Select the Connection Type](#)
- [Step 4: Define Adapter Services for an RPC Connection](#)
- [Step 5: Create or Update a REST Resource](#)
- [Step 6: Edit the REST Resource](#)

Step 1: Start the Integration Server Wrapper Wizard

➤ To start the Integration Server Wrapper wizard

- 1 In the context menu of a Software AG IDL file, choose **Integration Server > Generate web-Methods IS Connection**.

This starts the wizard with a list of existing Integration Server Wrapper connections.



Note: If the selected IDL file is not valid because of a syntax error, an error dialog comes up and the wizard does not start.

- 2 Continue with *Step 2: Create a New Integration Server Connection*.

Step 2: Create a New Integration Server Connection

Define New Integration Server Connection

Add host:port, user and password for a new Integration Server

Server:

User:

Password:

☐
Use secure connection

➤ To create a new Integration Server connection

- 1 Define the new Integration Server connection on the wizard page.



Notes:

1. The only required field is **Server**. Enter the hostname of the Integration Server including an optional port number. If no port number is specified, port number defaults to "5555". The **Integration Server Authentication** can be passed with the **User** and **Password** fields.
 2. Optional settings are for secure connections. The **Truststore for HTTPS** contains all signed certificates and must be a valid truststore.
 3. The check box **Verify host name** checks that the hostname is entered in the stored certificate.
 4. When the Integration Server has **Client Authentication** enabled, you can specify your **Keystore** file and keystore **Password**.
 5. For managing Integration Server connections, see [Preferences](#).
- 2 Choose **Next** and continue with *Step 3: Select the Connection Type*.

Step 3: Select the Connection Type

☒ Create a new EntireX Adapter connection of type: **EntireX RPC Connection**

☐ Update Adapter Services or Listeners to an existing Integration Server connection:

List of Connections

Connection Name	Package Name	Connection Type	Enabled
Employees:EmployeesConnection	Employees	EntireX RPC Connection	Yes

Total: 1

☒ Map Software AG IDL data types to String

☒ Create or Update REST resource

> To create a new connection

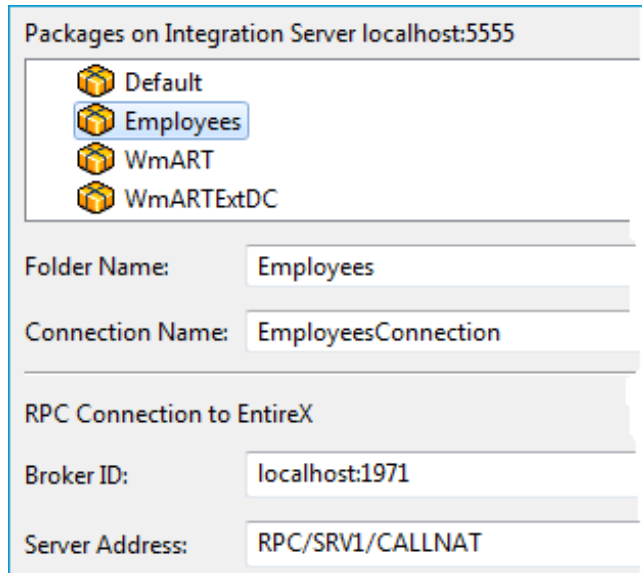
- 1 Select a connection type from the drop down list. Connection types are described under *EntireX Adapter Connections* in the EntireX Adapter documentation and *Introduction to the Integration Server Wrapper*.



Note: The list of connection types is filtered: connection types that require a license are only shown if a corresponding license file is available. Reliable RPC connections are only shown if all IDL programs contain only IN parameters. Also, if a server mapping file is available, only those connection types that support the interface type specified in the server mapping file are shown.

- 2 Check the box **Create or Update REST resource**.
- 3 Click **Next** and continue with *Step 4: Define Adapter Services for an RPC Connection*.

Step 4: Define Adapter Services for an RPC Connection



Packages on Integration Server localhost:5555

- Default
- Employees**
- WmART
- WmARTEExtDC

Folder Name:

Connection Name:

RPC Connection to EntireX

Broker ID:

Server Address:

> To create a connection and related adapter services

- 1 Select an existing package or create a new package for the created objects.
- 2 Define a folder name. If the folder does not exist, it will be created.
- 3 Define a connection name.
- 4 Define the parameters of the connection type. For details, see *Connection Parameters for RPC Connections* in the EntireX Adapter documentation.

As a result, the folder will contain the connection and the adapter services (one for each IDL program). The name of a service is the same as the respective IDL program.

The default settings for the adapter services are:

- the **Default** package; if not available, the first package
- the IDL library name for the **Folder Name**
- the IDL library name with the suffix "Connection" for the **Connection Name**



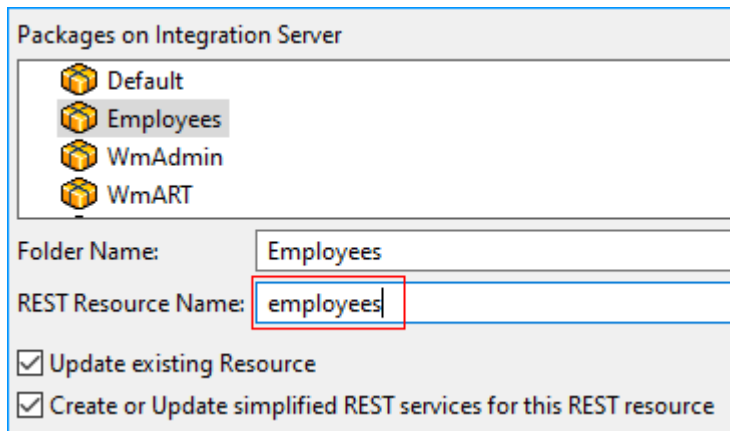
Note: When creating a connection, a package dependency is added such that the selected package depends on webMethods EntireX (the package `WmEntireX`) with the version currently used.

Step 5: Create or Update a REST Resource

➤ To create or update a REST resource

Make sure checkbox **Create or Update REST resource** is checked.

- 1 Select a package.



Packages on Integration Server

- Default
- Employees
- WmAdmin
- WmART

Folder Name: Employees

REST Resource Name: employees

☒ Update existing Resource

☒ Create or Update simplified REST services for this REST resource

- 2 Define a folder name. If the folder does not exist, it will be created.
- 3 Define the REST Resource name.
- 4 Check **Update existing Resource** if you want to add additional services to an existing resource.
- 5 Check **Create or Update simplified REST services for this REST resource** if you want to generate additional services with a simplified REST API. This allows you to use a simple HTTP-GET method without a payload.

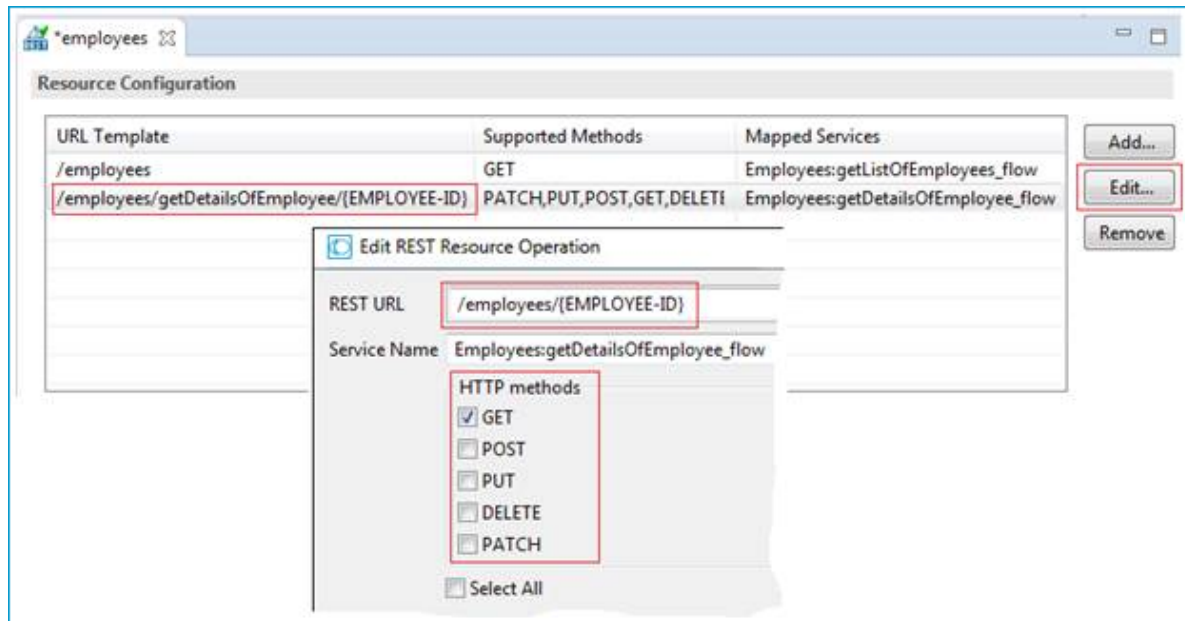


Note: This applies only to APIs where input signature parameters do not contain arrays.

Step 6: Edit the REST Resource

➤ To edit the REST resource

- 1 Switch to the **Service Development** perspective of Software AG Designer.



- 2 Refresh the package where you generated the REST Resource in the previous step.
- 3 Open the REST Resource.
- 4 Edit the REST URLs and HTTP Methods to match the functionality of your application.

For more information see:

- *REST Developer's Guide* in the webMethods Integration Server documentation for information on using the generated REST resources
- *REST Developer's Guide* in the webMethods Integration Server documentation

Task 3: Execute the Call from the REST Client to COBOL

Use an appropriate REST client, specify URL, Method and Headers and invoke your REST resource:

URL http://localhost:5555/restv2/employees

Method GET

Headers Accept application/json

200 OK

```
{
  "errorCode": "00000000",
  "errorFlag": "false",
  "response": {
    "EMPLOYEES-LIST": {
      "EMPLOYEES": [
        {
          "IDENT": "11100102",
          "FIRSTNAME": "Edgar",
          "SURNAME": "Schindler",
          "DATE-BIRTH": "Dec 4, 1962"
        },
        {
          "IDENT": "11100105",
          "FIRSTNAME": "Christian",
          "SURNAME": "Schirm",
          "DATE-BIRTH": "Mar 15, 1961"
        },
        {
          "IDENT": "11100106",
          "FIRSTNAME": "Rainer",
          "SURNAME": "Schmitt",
          "DATE-BIRTH": "Feb 13, 1955"
        },
        {
          "IDENT": "11100107",
          "FIRSTNAME": "Helga",
          "SURNAME": "Schmidt",
          "DATE-BIRTH": "May 26, 1961"
        }
      ]
    }
  }
}
```

URL http://localhost:5555/restv2/employees/111000105

Method GET

Headers Accept application/json

200 OK

```
{
  "errorCode": "00000000",
  "errorFlag": "false",
  "response": {
    "EMPLOYEE-DETAILS": {
      "FIRSTNAME": "Christian",
      "SURNAME": "Schirm",
      "DATE-BIRTH": "Mar 15, 1961",
      "DETAIL": {
        "ANNUAL-SALARY": "68000",
        "VACATION": "21",
        "LANGUAGE": "GER"
      }
    }
  }
}
```


18

Calling COBOL on z/VSE from REST

Under z/VSE, a COBOL server can be called in different environments:



- ❶ Extract the interface of the COBOL server program. See *Using the IDL Extractor for COBOL - Overview*.
- ❷ Generate REST resources, connection and adapter services in Integration Server.
- ❸ Execute the call from the REST client to the COBOL server program.

Continue with the appropriate scenario:

- **CICS** - Choose your CICS calling convention:
 - DFHCOMMAREA using
 - **CICS ECI (zero footprint)**
 - **CICS Socket Listener (minimal footprint)**
 - Channel Container using
 - **CICS Socket Listener (minimal footprint)**
 - Large Buffer using
 - **CICS Socket Listener (minimal footprint)**

19

Calling COBOL on z/VSE CICS from REST

There are different styles (interface types) for calling a COBOL server.



It is important to know the interface type of your COBOL server. If you are unsure, consult a COBOL CICS specialist or see description of interface type in the IDL Extractor for COBOL documentation for details and examples: [DFHCOMMAREA Calling Convention](#) | [Channel Container Calling Convention](#) | [DFHCOMMAREA Large Buffer Interface](#).

When you are sure which programming style you are using, continue with the appropriate scenario:

- Channel Container using
 - [CICS Socket Listener \(minimal footprint\)](#)
- DFHCOMMAREA using
 - [CICS ECI \(zero footprint\)](#)
 - [CICS Socket Listener \(minimal footprint\)](#)
- Large Buffer using
 - [CICS Socket Listener \(minimal footprint\)](#)

20

Calling COBOL DFHCOMMAREA (Zero Footprint using CICS ECI) on z/VSE CICS from REST

■ Introduction	332
■ What do I need to Install for this Scenario?	334
■ Task 1: Extract the Interface of a COBOL Server	335
■ Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server	348
■ Task 3: Execute the Call from the REST Client to COBOL	354

This scenario uses the tools IDL Extractor for COBOL and Integration Server Wrapper of the Designer.

Introduction



- ❶ Extract the interface of the COBOL server program. See *Using the IDL Extractor for COBOL - Overview*.
- ❷ Generate REST resources, connection and adapter services in Integration Server.
- ❸ Execute the call from the REST client to the COBOL server program.

This scenario makes the following important assumptions:

For Step 1

- You have a working COBOL DFHCOMMAREA server. For illustration and examples on such a server, see *CICS with DFHCOMMAREA Calling Convention* under *Introduction to the IDL Extractor for COBOL*.
- You have access to the related COBOL sources and copybooks as files on your local machine. The minimum requirement is the `DATA DIVISION` of the interface.



For Steps 2 and 3

- You have a REST client.
- To call the COBOL server program at runtime using the *EntireX CICS ECI* connection method, you need to configure the CICS ECI TCP/IP service within your CICS region. See *Preparing IBM CICS for ECI* and *Connection Parameters for CICS ECI Connections*.



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.

Task 1: Extract the Interface of a COBOL Server

- [Introduction](#)
- [Sample COBOL Server used in the Examples](#)
- [Extracting a COBOL Server - Fast-track Method](#)
- [Extracting a COBOL Server - Modern Method with User-defined Mapping](#)
- [Comparison Chart: Fast-track versus User-defined Mapping](#)
- [Testing the Extraction Results](#)

Introduction

Follow the instructions for extracting COBOL, see *Using the IDL Extractor for COBOL - Overview* and choose *Scenario I: Create New IDL and Server Mapping Files* if this is your first extraction. Learn how to invoke the IDL Extractor for COBOL, create a COBOL Extractor Environment and about the possibilities to select the COBOL source.

The extraction process creates the following EntireX metafiles:

- IDL file. A Software AG IDL file contains definitions of the interface between client and server. See *Software AG IDL File* in the IDL Editor documentation in the IDL Editor documentation.
- Server mapping file (optional). The mapping file is a Designer file with extension .cvm that contains COBOL-specific mapping information. See *Server Mapping Files for COBOL* in the Designer documentation.

Sample COBOL Server used in the Examples

The following COBOL server is used to illustrate the features of the IDL Extractor for COBOL. Imagine an existing COBOL server called `EMPLOYEE`. It implements the access logic for a `LIST` and `DETAILS` function to a database view `EMPLOYEE`:

```
IDENTIFICATION DIVISION.
PROGRAM-ID.      EMPLOYEE.
*****
*   Program Name      = EMPLOYEE
*   String Literal    = QUOTE
*****
01 DFHCOMMAREA.
  02 OPERATION                                PIC X(1).
    88 OPERATION-LIST      VALUE 'L'.
    88 OPERATION-DETAILS   VALUE 'D'.
  02 EMPLOYEE-ID                                PIC X(10).
  02 EMPLOYEE-DETAILS.
    03 FIRSTNAME                                PIC X(20).
    03 SURNAME                                PIC X(20).
    03 DATE-BIRTH                                PIC X(12).
    03 DETAILS                                PIC X(100).
    03 FILLER REDEFINES DETAILS.
      04 ANNUAL-SALARY      PACKED-DECIMAL PIC S9(9).
      04 VACATION            PIC S9(2).
      04 LANGUAGE            PIC X(3).
      04 FILLER              PIC X(90).
  02 EMPLOYEES-COUNT                                PIC 9(8) BINARY.
  02 EMPLOYEES OCCURS 1 TO 99 DEPENDING ON EMPLOYEES-COUNT.
    03 IDENT                                PIC X(10).
    03 FIRSTNAME                                PIC X(20).
    03 SURNAME                                PIC X(20).
    03 DATE-BIRTH                                PIC X(12).
```

Two approaches for extracting the COBOL server are described below:

- **Fast-track**

Learn how EntireX helps you to connect the COBOL Server with quick results.

- **User-defined Mapping**

The following features are among those available with a user-defined mapping:

- **Shape the interface to your COBOL server**

You can minimize the interface by suppressing COBOL server fields or by providing constant values. Renaming interfaces allows you to specify a readable long name. These two features together - constants and renaming of interfaces - are most powerful when multiple interfaces are implemented in a single COBOL server. Each function of the COBOL server triggered by an operation code or function code can be modelled as a separate REST resource method/operation.

■ Select alternative mappings

You can select alternative mappings for COBOL data items (REDEFINES, tables etc.) to enable scenarios with COBOL servers where data exchange is not fully described by the LINKAGE SECTION COBOL data items.

These approaches are described under [Fast-track Method](#) and [Modern Method](#) below.

Extracting a COBOL Server - Fast-track Method

➤ To extract the COBOL Server

- 1 Switch to the **EntireX** perspective.
- 2 Call the IDL Extractor for COBOL.
- 3 Set the correct interface type as described under *Step 4: Define the Extraction Settings and Start Extraction* under *Scenario I: Create New IDL and Server Mapping Files* in the IDL Extractor for COBOL documentation.

COBOL Source

File Name: EMPLAPPL.c

Operating System: z/VSE

Interface Type: CICS with DFHCOMMAREA calling convention

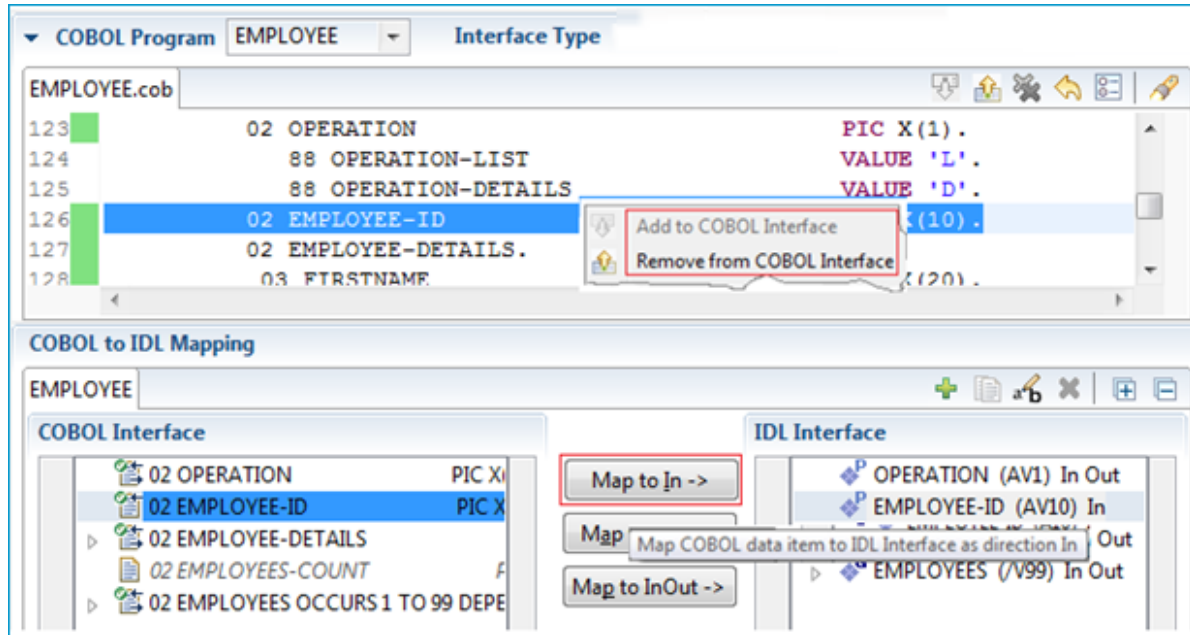
☒ Input Message same as Output Message

- 4 Set the correct value for checkbox **Input Message same as Output Message**.
 - Check the checkbox if the COBOL data structure of the CICS input message is the same as the structure of the CICS output message, that is, there is one COBOL layout for both directions. See the following COBOL server examples in the IDL Extractor for COBOL documentation:
 - Example 1: Redefines
 - Example 2: Buffer Technique
 - Example 3: COBOL SET ADDRESS Statements
 - Clear the checkbox if the COBOL data structure of the CICS input message is different to the structure of the CICS output message, that is, the output overlays the input, which means there is a second COBOL layout for the reply. See the following COBOL server examples in the IDL Extractor for COBOL documentation:
 - Example 1: Redefines
 - Example 2: Buffer Technique

■ *Example 3: COBOL SET ADDRESS Statements*

For our example, check the check box, because the COBOL layout for input is the same as the COBOL layout for output.

- 5 Press **Next** to enter the COBOL Mapping Editor.



The purpose of the COBOL Mapping Editor is to map COBOL data items to IDL (Interface Definition Language) parameters. It consists of three main areas:

- The upper window pane is the COBOL program source.
- The window pane on the left-hand side is the COBOL interface, consisting of COBOL data items representing the parameters in your COBOL server program.
- The window pane on the right-hand side is the IDL interface consisting of IDL parameters.

What you select depends on check-box **Input Message same as Output Message**:

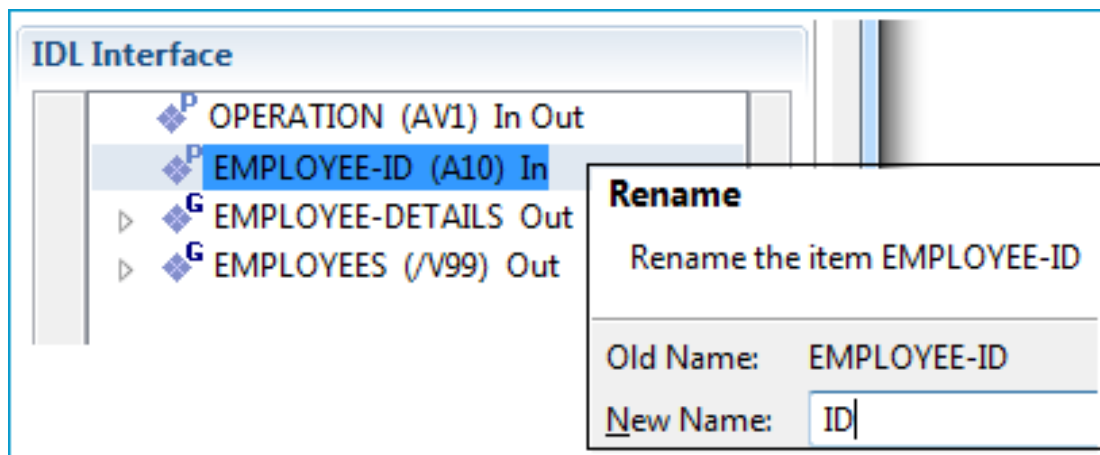
- If there is a single COBOL data structure of the CICS message for input and output - check box **Input Message same as Output Message** is checked - you select a single COBOL data structure in the COBOL Mapping Editor. This structure is used for COBOL input as well as for COBOL output. For more information in this case see *Mapping Editor User Interface* in the IDL Extractor for COBOL documentation.
- If there is a second COBOL data structure for the CICS output message - check box **Input Message same as Output Message** is cleared - you will select two COBOL data structures in the COBOL Mapping Editor. One is used for COBOL input and the other is for COBOL output. For more information in this case see *Mapping Editor User Interface* in the IDL Extractor for COBOL documentation.

COBOL extraction is a two-step process:

- First, select the COBOL parameters using the **Add to** and **Remove from COBOL Interface** from the context menu on the COBOL data items in the upper window pane (COBOL source); in many situations the COBOL interface is detected by the Mapping Editor directly and you can omit this step.
- Then map the COBOL parameters to IDL directions using the **Map to In/Out/InOut** buttons. COBOL parameters do not have directions, they are always INOUT. This step can be left out for a quick test in non-production scenarios; for real production it is recommended to set IDL directions to reduce data transfer. You also have the option to assign a meaningful name to an IDL parameter if the original COBOL data item is not self-explanatory.

For our example:

- Set **Map to In** for COBOL data item `EMPLOYEE-ID` as IDL direction (see screenshot in previous step). For `EMPLOYEES-DETAILS` and `EMPLOYEES` perform the similar **Map to Out**.
- Use the function **Rename** from the context menu on the IDL parameters in the **IDL Interface** pane to rename COBOL parameter `EMPLOYEES-ID` to IDL parameter `ID`. Do the same for `EMPLOYEE-DETAILS`: **Rename** to `EMPLOYEE`.



- 6 Press **Finish**. The default extraction settings ensure you get useful results. The outcome is a Software AG IDL file (interface definition language), a Designer file with extension .idl:

```
library 'EMPLAPPL' is
program 'EMPLOYEE' is
  define data parameter
    1 OPERATION (AV1) In Out
    1 ID (AV10) In /* Original Name:EMPLOYEE-ID
    1 EMPLOYEE Out /* Original Name:EMPLOYEE-DETAILS
      2 FIRSTNAME (AV20)
      2 SURNAME (AV20)
      2 DATE-BIRTH (AV12)
      2 DETAILS (AV100)
    1 EMPLOYEES (/V99) Out
      2 IDENT (AV10)
      2 FIRSTNAME (AV20)
      2 SURNAME (AV20)
      2 DATE-BIRTH (AV12)
  end-define
```

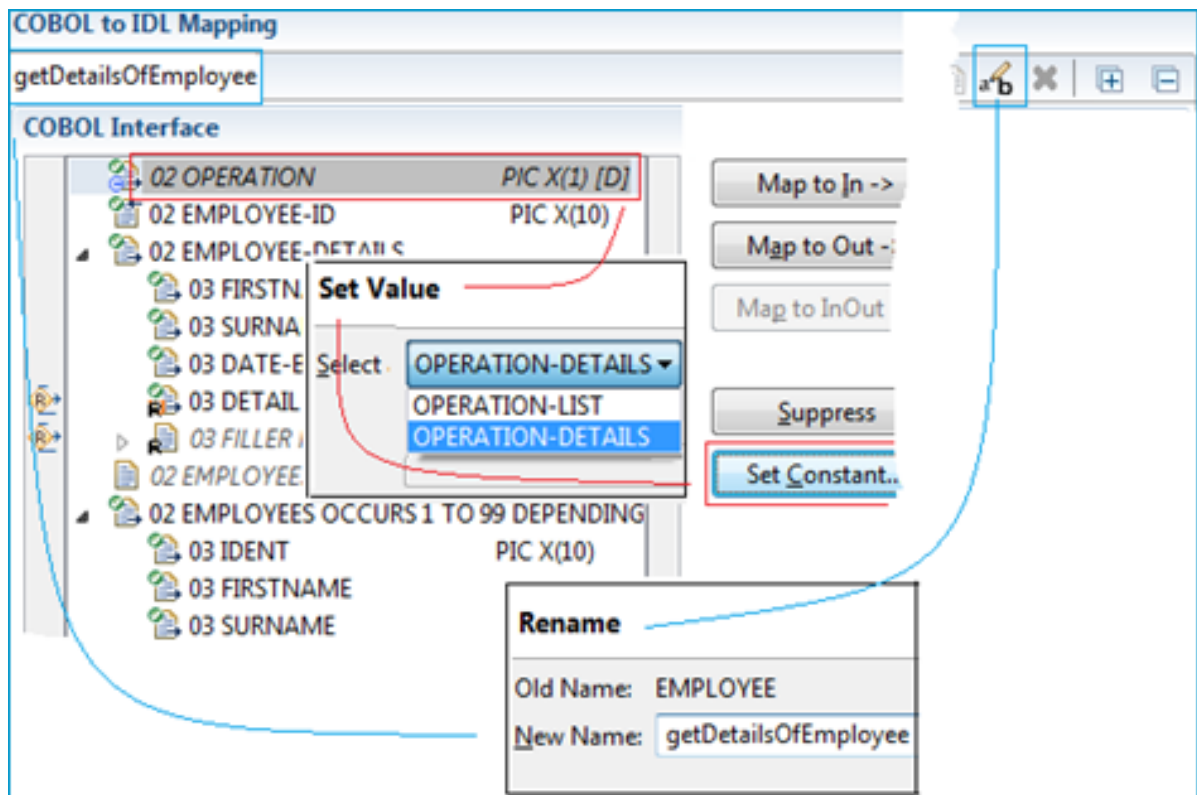
If required, a server mapping file with the same name as the IDL file but the extension .cvm may be generated too, for technical reasons. This file must always be stored together with the concomitant IDL file. For more information, see *Server Mapping Files for COBOL* in the Designer documentation. By default, the interface name `EMPLOYEE` is taken from the COBOL server program name.


Extracting a COBOL Server - Modern Method with User-defined Mapping

EntireX allows you to extract all implemented interfaces of the COBOL server separately. Instead of a large `EMPLOYEE` interface, separate interfaces `getListOfEmployees` and `getDetailsOfEmployee` are extracted. Each interface contains required parameters; obsolete parameters for an interface are suppressed, improving its usability.

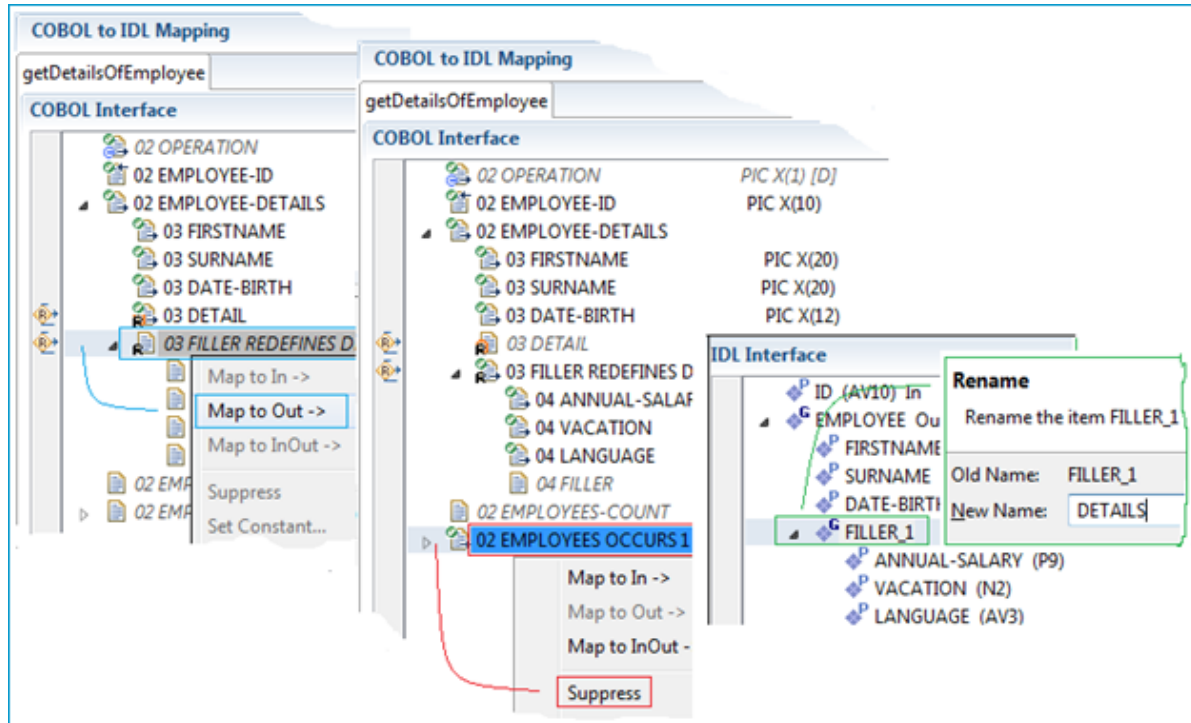
➤ To extract a COBOL server with a user-defined mapping

- 1 Shape `EMPLOYEE` to `getDetailsOfEmployee`:



- Mark the `OPERATION` field in the **COBOL Interface** pane and press the **Set Constant** button. In the pop-up window, select `OPERATION-DETAILS` (red markers). The `OPERATION-DETAILS` value originates from COBOL level-88 data item (blue markers). If there is no COBOL level-88 enumeration type in the COBOL program defined, you can enter the function code manually in the pop-up window.
- By default the IDL interface gets the same name as your COBOL server: `EMPLOYEE`. Use the **Rename** button  from the toolbar of the COBOL Mapping Editor and change the IDL interface name to `getDetailsOfEmployee` (blue markers). This name is used as the IS service name later.

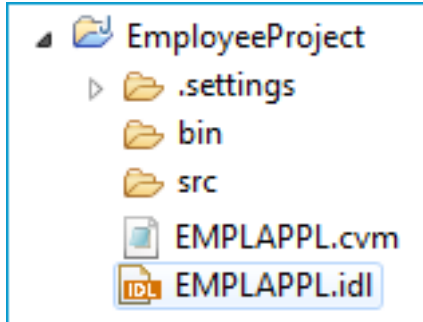
2 Continue shaping, making `getDetailsOfEmployee` easier to use:



- Use the context menu of the redefinition of the `DETAIL` field in the **COBOL Interface** pane, `FILLER REDEFINES DETAIL` and choose **Map to Out** (blue markers). A redefinition is a secondary layout (field types) of the same memory area. The redefinition contains more specific information (`ANNUAL-SALARY`, `VACATION`, etc.) than character buffer `DETAIL`. This is the reason why it is more useful in your interface.
- Use the context menu of `FILLER_1` in the IDL interface pane and rename `FILLER_1` to `DETAILS` (green markers) for a readable IS service field name later.
- The `DETAILS` function in the COBOL server does not make use of the `EMPLOYEES` field. Therefore you can leave it out of the interface: use the context menu of the `EMPLOYEES` field in the **COBOL Interface** pane and choose **Suppress** (red markers).

3 Shape `EMPLOYEE` to `getListOfEmployees` Function:

See *Server Mapping Files for COBOL* in the Designer documentation. The *Software AG IDL File* in the IDL Editor documentation describes the interfaces from the client point of view, while the server mapping file contains the mapping to the real COBOL server. Both of these files must be kept together and in sync, otherwise a call to the COBOL server may fail.



To summarize: We created two IDL interfaces. In the IDL file, these resulted in two IDL programs: `getListOfEmployees` and `getDetailsOfEmployee`. Both IDL programs were given readable names (Steps 1 and 3). Meaningful fields were kept, while superfluous fields were suppressed (Steps 1 and 3). The program `getDetailsOfEmployee` contains the redefined fields of parameter `DETAILS` (see below) mapped during extraction (Step 1).



Note: At runtime, the RPC client generated with the extracted interface will send data for the redesigned interfaces, while your COBOL server still expects `EMPLOYEE` data. The EntireX runtime transforms the incoming data stream from the RPC client, using the server mapping file.

```
program 'getDetailsOfEmployee' is
  define data parameter
    1 ID (AV10) In /* Original Name:EMPLOYEE-ID
    1 EMPLOYEE Out /* Original Name:EMPLOYEE-DETAILS
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
    2 DETAILS /* Original Name:FILLER_1
    3 ANNUAL-SALARY (P9)
    3 VACATION (N2)
    3 LANGUAGE (AV3)
  end-define
  program 'getListOfEmployees' is
    1 EMPLOYEES (/V99) Out
    2 IDENT (AV10)
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
  end-define
```


Comparison Chart: Fast-track versus User-defined Mapping

The following table highlights the differences when shaping an interface with a *user-defined mapping* compared with a *fast-path extraction*.

	Interface Shaping with User-defined Mapping	Fast-path Extraction
General	User-defined interface with dedicated mapping. For our example, the interfaces are small and tidy and more self-explanatory with long, readable interface names. They are easier to use with the hidden <code>OPERATION</code> field and the suppressed COBOL fields, which are not needed.	Automatic, quick result extraction with COBOL-like interfaces on IS. For our example, the IS interface <code>EMPLOYEE</code> matches exactly 1:1, field by field.
IS service(s)	Multiple small and handy IS services; each <code>OPERATION</code> code is mapped to a separate IS service.	One big IS service.
IS service name	Readable long name.	Short Subprogram name; up to 8 characters.
IS fields	Usage of Suppress and Set Constant reduces the message length. This keeps focus on relevant data items and keeps the client's interface clean. It may also improve performance.	The IS fields and COBOL server parameters match 1:1. As COBOL layout descriptions are sometimes used for many different purposes, irrelevant data items appear and clutter up the IS interface.
<code>OPERATION</code> parameter	Suppressed: the <code>OPERATION</code> parameter does not exist in the IS service as an IS field.	The <code>OPERATION</code> parameter exists in the IS service as an IS field and needs to be filled-in by the client endpoint.
<code>OPERATION</code> code	The <code>OPERATION</code> code is provided internally by EntireX runtime in the <code>OPERATION</code> field.	The <code>OPERATION</code> code needs to be specified in the IS service in the <code>OPERATION</code> field by the client endpoint.
<code>REDEFINE</code> parameters	Either the parameter that is redefined or one of its redefinitions is available as an IS field.	Only the parameter that is redefined is available as an IS field. Redefinitions of the parameter are not available as IS fields.

A user-defined mapping enables you also to define alternative mappings for COBOL data items (`REDEFINES`, tables etc.). These enable scenarios with COBOL servers where data exchange is not fully described by the `LINKAGE SECTION` COBOL data items. For more information refer to the IDL Extractor for COBOL documentation:

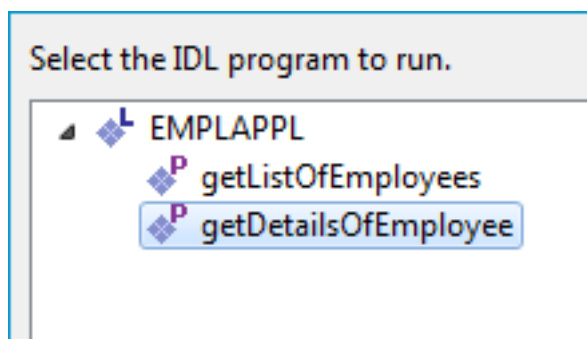
- *Mapping Editor IDL Interface Mapping Functions* if the check-box **Input Message same as Output Message** was checked.
- *Mapping Editor IDL Interface Mapping Functions* if the check-box **Input Message same as Output Message** was cleared.
- *User-defined Mapping*

Testing the Extraction Results

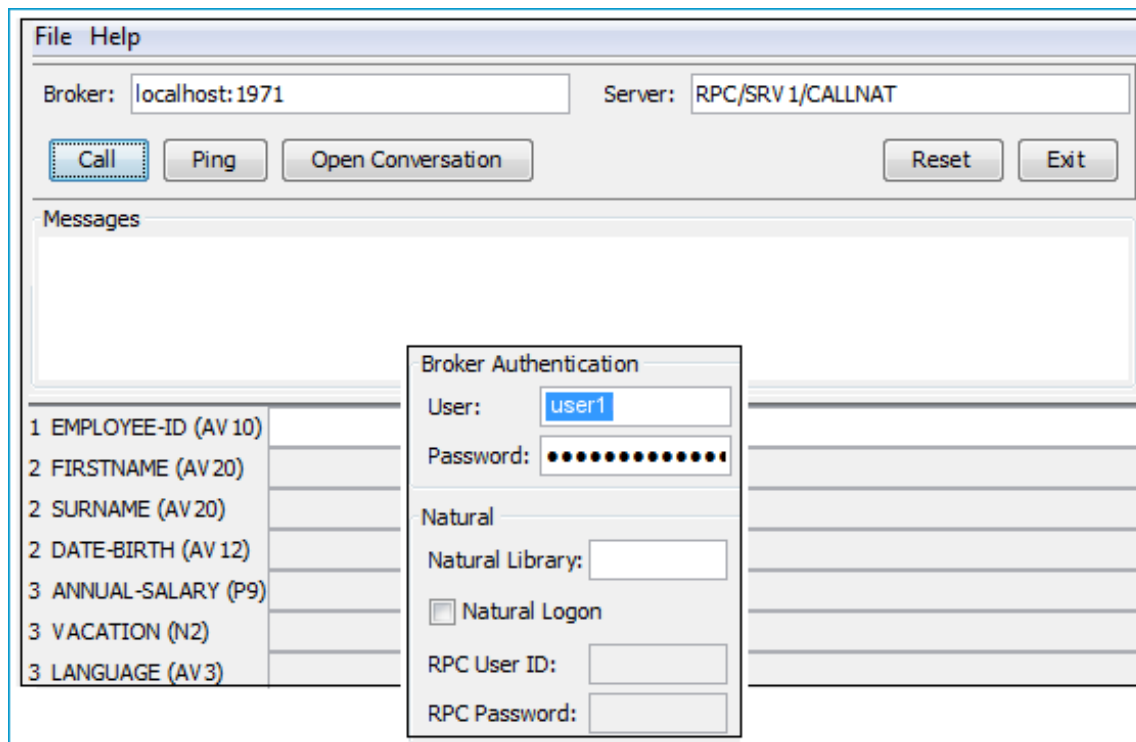
The following pictures use the extraction results described under [Extracting a COBOL Server - Modern Method with User-defined Mapping](#).

> To test the extraction results (optional)

- 1 You can test the results of the extraction operation and the REST server back end, using the EntireX IDL Tester. From the context menu of the IDL file in the Designer, choose **Software AG IDL Tester**.



- 2 Select `getDetailsOfEmployee`.



Note that the broker and server parameters contain the explicit route to call the server program, and you can optionally ping the connection from this client. With the **File > Options** dialog, you can set:

- **User** and **Password** for *broker* authentication
- **RPC User ID** and **RPC Password** for *server* authentication

See *EntireX IDL Tester* in the Designer documentation.

- 3 Check the Integration Server log, the EntireX Adapter log or the RPC logs. Applies to all connection methods.

Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server

This section describes your first steps to create a new Integration Server connection. This is described in more detail under *Using the Integration Server Wrapper*, for example working with existing Integration Server connections. This section covers the following topics:

- [Step 1: Start the Integration Server Wrapper Wizard](#)
- [Step 2: Create a New Integration Server Connection](#)
- [Step 3: Select the Connection Type](#)
- [Step 4: Define Adapter Services for a CICS ECI Connection](#)
- [Step 5: Create or Update a REST Resource](#)
- [Step 6: Edit the REST Resource](#)

Step 1: Start the Integration Server Wrapper Wizard

➤ To start the Integration Server Wrapper wizard

- 1 In the context menu of a Software AG IDL file, choose **Integration Server > Generate web-Methods IS Connection**.

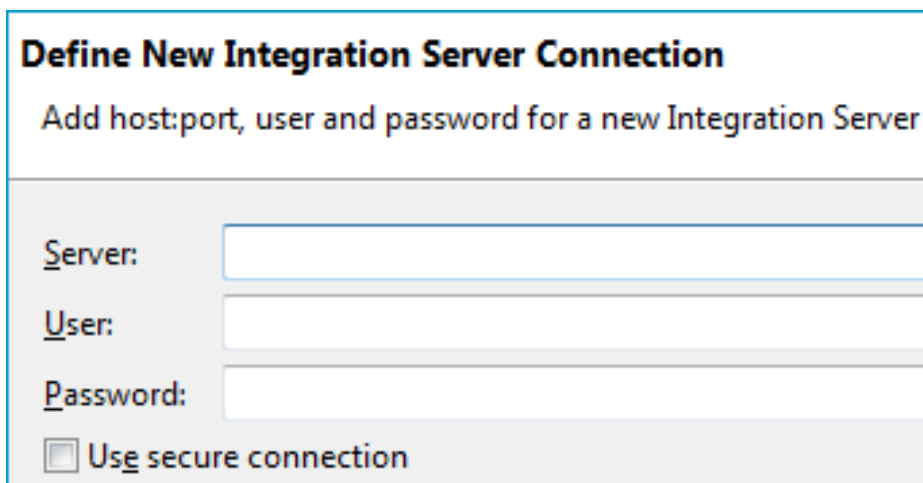
This starts the wizard with a list of existing Integration Server Wrapper connections.



Note: If the selected IDL file is not valid because of a syntax error, an error dialog comes up and the wizard does not start.

- 2 Continue with *Step 2: Create a New Integration Server Connection*.

Step 2: Create a New Integration Server Connection



The screenshot shows a web form titled "Define New Integration Server Connection". Below the title is a subtitle: "Add host:port, user and password for a new Integration Server". The form contains three input fields: "Server:", "User:", and "Password:". Below these fields is a checkbox labeled "Use secure connection".

➤ To create a new Integration Server connection

- 1 Define the new Integration Server connection on the wizard page.



Notes:

1. The only required field is **Server**. Enter the hostname of the Integration Server including an optional port number. If no port number is specified, port number defaults to "5555". The **Integration Server Authentication** can be passed with the **User** and **Password** fields.
2. Optional settings are for secure connections. The **Truststore for HTTPS** contains all signed certificates and must be a valid truststore.
3. The check box **Verify host name** checks that the hostname is entered in the stored certificate.
4. When the Integration Server has **Client Authentication** enabled, you can specify your **Keystore** file and keystore **Password**.
5. For managing Integration Server connections, see [Preferences](#).

- 2 Choose **Next** and continue with *Step 3: Select the Connection Type*.

Step 3: Select the Connection Type

☒ Create a new EntireX Adapter connection of type **CICS ECI Connection**

☐ Update Adapter Services or Listeners to an existing Integration Server connection:

List of Connections

Connection Name	Package Name	Connection Type	Enabled
Employees:EmployeesConnection	Employees	EntireX RPC Connection	Yes

Total: 1

☒ Map Software AG IDL data types to String

☒ Create or Update REST resource

> To create a new connection

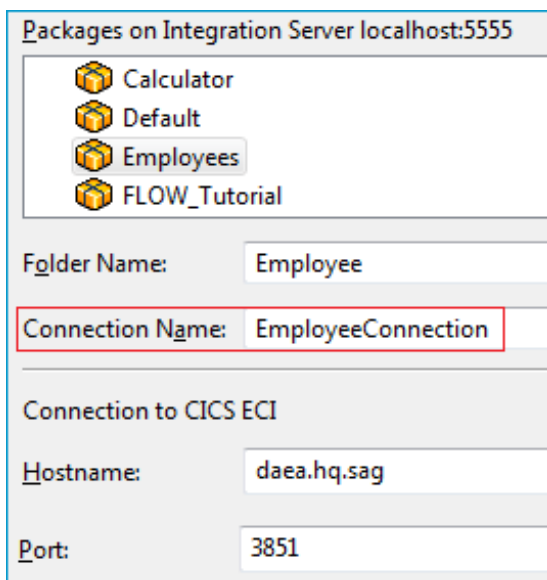
- 1 Select a connection type from the drop down list. Connection types are described under *EntireX Adapter Connections* in the EntireX Adapter documentation and *Introduction to the Integration Server Wrapper*.



Note: The list of connection types is filtered: connection types that require a license are only shown if a corresponding license file is available. Reliable RPC connections are only shown if all IDL programs contain only IN parameters. Also, if a server mapping file is available, only those connection types that support the interface type specified in the server mapping file are shown.

- 2 Check the box **Create or Update REST resource**.
- 3 Click **Next** and continue with *Step 4: Define Adapter Services for a CICS ECI Connection*.

Step 4: Define Adapter Services for a CICS ECI Connection



Packages on Integration Server localhost:5555

- Calculator
- Default
- Employees
- FLOW_Tutorial

Folder Name: Employee

Connection Name: EmployeeConnection

Connection to CICS ECI

Hostname: daea.hq.sag

Port: 3851

➤ To create a connection and related adapter services

- 1 Select a package for the created objects.
- 2 Define a folder name. If the folder does not exist, it will be created.
- 3 Define a connection name.
- 4 Define the parameters of the connection type. For details, see *Connection Parameters for CICS ECI Connections* in the EntireX Adapter documentation.

As a result, the folder will contain the connection and the adapter services (one for each IDL program). The name of a service is the same as the respective IDL program.

The default settings for the adapter services are:

- the **Default** package; if not available, the first package
- the IDL library name for the **Folder Name**
- the IDL library name with the suffix "Connection" for the **Connection Name**



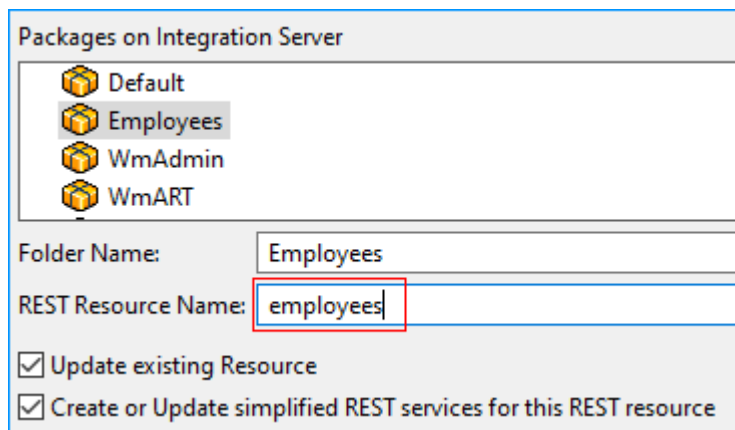
Note: When creating a connection, a package dependency is added such that the selected package depends on webMethods EntireX (the package `WmEntireX`) with the version currently used.

Step 5: Create or Update a REST Resource

➤ To create or update a REST resource

Make sure checkbox **Create or Update REST resource** is checked.

- 1 Select a package.



Packages on Integration Server

- Default
- Employees**
- WmAdmin
- WmART

Folder Name: Employees

REST Resource Name: employees

☒ Update existing Resource

☒ Create or Update simplified REST services for this REST resource

- 2 Define a folder name. If the folder does not exist, it will be created.
- 3 Define the REST Resource name.
- 4 Check **Update existing Resource** if you want to add additional services to an existing resource.
- 5 Check **Create or Update simplified REST services for this REST resource** if you want to generate additional services with a simplified REST API. This allows you to use a simple HTTP-GET method without a payload.

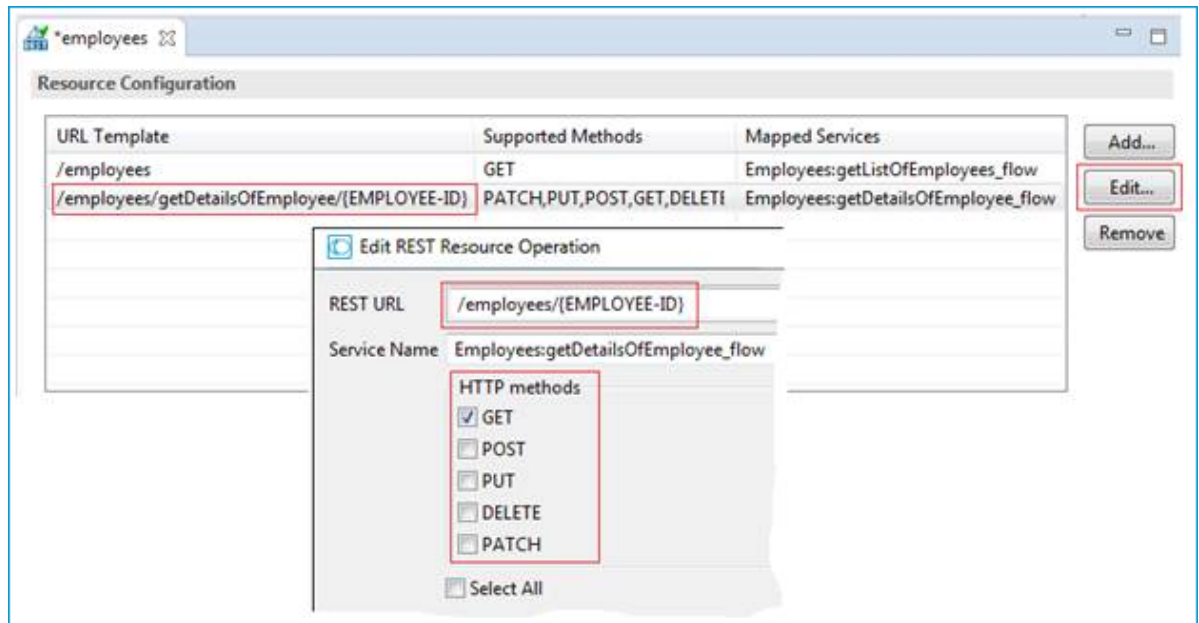


Note: This applies only to APIs where input signature parameters do not contain arrays.

Step 6: Edit the REST Resource

➤ To edit the REST resource

- 1 Switch to the **Service Development** perspective of Software AG Designer.



- 2 Refresh the package where you generated the REST Resource in the previous step.
- 3 Open the REST Resource.
- 4 Edit the REST URLs and HTTP Methods to match the functionality of your application.

For more information see:

- *REST Developer's Guide* in the webMethods Integration Server documentation for information on using the generated REST resources
- *REST Developer's Guide* in the webMethods Integration Server documentation

Task 3: Execute the Call from the REST Client to COBOL

Use an appropriate REST client, specify URL, Method and Headers and invoke your REST resource:

URL http://localhost:5555/restv2/employees	URL http://localhost:5555/restv2/employees/111000105
Method GET	Method GET
Headers Accept application/json	Headers Accept application/json
200 OK	200 OK
<pre>{ "errorCode": "00000000", "errorFlag": "false", "response": { "EMPLOYEES-LIST": { "EMPLOYEES": [{ "IDENT": "11100102", "FIRSTNAME": "Edgar", "SURNAME": "Schindler", "DATE-BIRTH": "Dec 4, 1962" }, { "IDENT": "11100105", "FIRSTNAME": "Christian", "SURNAME": "Schirm", "DATE-BIRTH": "Mar 15, 1961" }, { "IDENT": "11100106", "FIRSTNAME": "Rainer", "SURNAME": "Schmitt", "DATE-BIRTH": "Feb 13, 1955" }, { "IDENT": "11100107", "FIRSTNAME": "Helga", "SURNAME": "Schmidt", "DATE-BIRTH": "May 26, 1961" }] } } }</pre>	<pre>{ "errorCode": "00000000", "errorFlag": "false", "response": { "EMPLOYEE-DETAILS": { "FIRSTNAME": "Christian", "SURNAME": "Schirm", "DATE-BIRTH": "Mar 15, 1961", "DETAIL": { "ANNUAL-SALARY": "68000", "VACATION": "21", "LANGUAGE": "GER" } } } }</pre>

21

Calling COBOL DFHCOMMAREA (Minimal Footprint Using CICS Socket Listener) on z/VSE CICS from REST

■ Introduction	356
■ What do I need to Install for this Scenario?	358
■ Task 1: Extract the Interface of a COBOL Server	359
■ Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server	372
■ Task 3: Execute the Call from the REST Client to COBOL	378

This scenario uses the tools IDL Extractor for COBOL and Integration Server Wrapper of the Designer.

Introduction

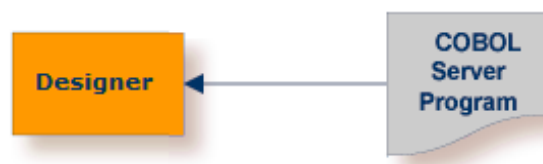


- ❶ Extract the interface of the COBOL server program. See *Using the IDL Extractor for COBOL - Overview*.
- ❷ Generate REST resources, connection and adapter services in Integration Server.
- ❸ Execute the call from the REST client to the COBOL server program.

This scenario makes the following important assumptions:

For Task 1:

- You have a working COBOL DFHCOMMAREA server. For illustration and examples on such a server, see *CICS with DFHCOMMAREA Calling Convention* under *Introduction to the IDL Extractor for COBOL*.
- You have access to the related COBOL sources and copybooks as files on your local machine. The minimum requirement is the `DATA DIVISION` of the interface.



For Tasks 2 and 3:

- You have a REST client.
- To call the COBOL server program at runtime using the *EntireX CICS Socket Listener* connection method, you need to install the CICS Socket Listener within your CICS region: see *Preparing for CICS Socket Listener*. For configuration, see *Connection Parameters for CICS Socket Listener Connections*.



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.
- You have the CICS Socket Listener installed. See *Installing the CICS Socket Listener* in the z/VSE Installation documentation.

Task 1: Extract the Interface of a COBOL Server

- [Introduction](#)
- [Sample COBOL Server used in the Examples](#)
- [Extracting a COBOL Server - Fast-track Method](#)
- [Extracting a COBOL Server - Modern Method with User-defined Mapping](#)
- [Comparison Chart: Fast-track versus User-defined Mapping](#)
- [Testing the Extraction Results](#)

Introduction

Follow the instructions for extracting COBOL, see *Using the IDL Extractor for COBOL - Overview* and choose *Scenario I: Create New IDL and Server Mapping Files* if this is your first extraction. Learn how to invoke the IDL Extractor for COBOL, create a COBOL Extractor Environment and about the possibilities to select the COBOL source.

The extraction process creates the following EntireX metafiles:

- IDL file. A Software AG IDL file contains definitions of the interface between client and server. See *Software AG IDL File* in the IDL Editor documentation in the IDL Editor documentation.
- Server mapping file (optional). The mapping file is a Designer file with extension .cvm that contains COBOL-specific mapping information. See *Server Mapping Files for COBOL* in the Designer documentation.

Sample COBOL Server used in the Examples

The following COBOL server is used to illustrate the features of the IDL Extractor for COBOL. Imagine an existing COBOL server called `EMPLOYEE`. It implements the access logic for a `LIST` and `DETAILS` function to a database view `EMPLOYEE`:

```
IDENTIFICATION DIVISION.
PROGRAM-ID.    EMPLOYEE.
*****
*   Program Name      = EMPLOYEE
*   String Literal    = QUOTE
*****
01 DFHCOMMAREA.
  02 OPERATION                                PIC X(1).
    88 OPERATION-LIST                          VALUE 'L'.
    88 OPERATION-DETAILS                      VALUE 'D'.
  02 EMPLOYEE-ID                                PIC X(10).
  02 EMPLOYEE-DETAILS.
    03 FIRSTNAME                                PIC X(20).
    03 SURNAME                                PIC X(20).
    03 DATE-BIRTH                              PIC X(12).
    03 DETAILS                                PIC X(100).
    03 FILLER REDEFINES DETAILS.
      04 ANNUAL-SALARY          PACKED-DECIMAL PIC S9(9).
      04 VACATION                PIC S9(2).
      04 LANGUAGE                PIC X(3).
      04 FILLER                  PIC X(90).
  02 EMPLOYEES-COUNT                        PIC 9(8) BINARY.
  02 EMPLOYEES OCCURS 1 TO 99 DEPENDING ON EMPLOYEES-COUNT.
    03 IDENT                          PIC X(10).
    03 FIRSTNAME                      PIC X(20).
    03 SURNAME                        PIC X(20).
    03 DATE-BIRTH                     PIC X(12).
```

Two approaches for extracting the COBOL server are described below:

- **Fast-track**

Learn how EntireX helps you to connect the COBOL Server with quick results.

- **User-defined Mapping**

The following features are among those available with a user-defined mapping:

- **Shape the interface to your COBOL server**

You can minimize the interface by suppressing COBOL server fields or by providing constant values. Renaming interfaces allows you to specify a readable long name. These two features together - constants and renaming of interfaces - are most powerful when multiple interfaces are implemented in a single COBOL server. Each function of the COBOL server triggered by an operation code or function code can be modelled as a separate REST resource method/operation.

■ Select alternative mappings

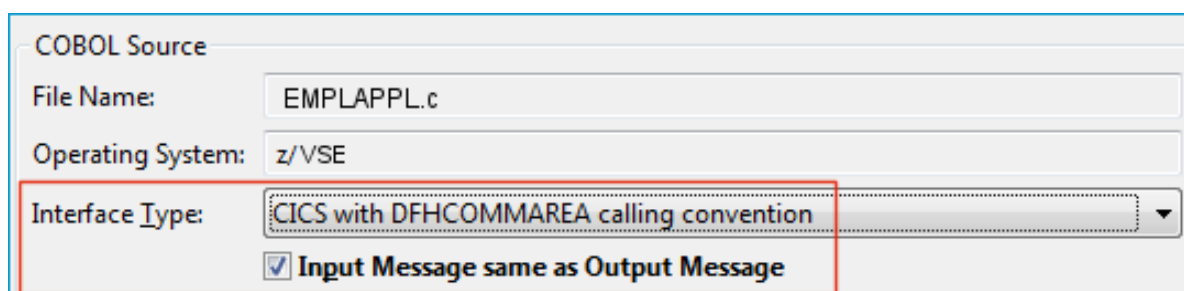
You can select alternative mappings for COBOL data items (REDEFINES, tables etc.) to enable scenarios with COBOL servers where data exchange is not fully described by the LINKAGE SECTION COBOL data items.

These approaches are described under [Fast-track Method](#) and [Modern Method](#) below.

Extracting a COBOL Server - Fast-track Method

➤ To extract the COBOL Server

- 1 Switch to the **EntireX** perspective.
- 2 Call the IDL Extractor for COBOL.
- 3 Set the correct interface type as described under *Step 4: Define the Extraction Settings and Start Extraction* under *Scenario I: Create New IDL and Server Mapping Files* in the IDL Extractor for COBOL documentation.



COBOL Source

File Name: EMPLAPPL.c

Operating System: z/VSE

Interface Type: CICS with DFHCOMMAREA calling convention

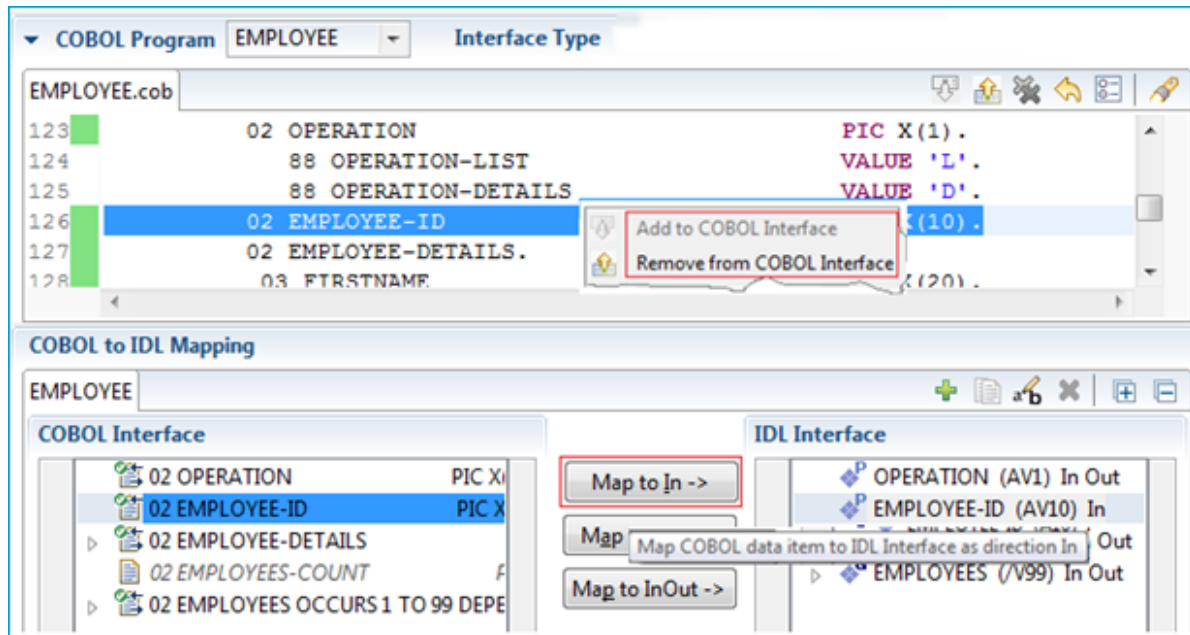
☒ Input Message same as Output Message

- 4 Set the correct value for checkbox **Input Message same as Output Message**.
 - Check the checkbox if the COBOL data structure of the CICS input message is the same as the structure of the CICS output message, that is, there is one COBOL layout for both directions. See the following COBOL server examples in the IDL Extractor for COBOL documentation:
 - Example 1: Redefines
 - Example 2: Buffer Technique
 - Example 3: COBOL SET ADDRESS Statements
 - Clear the checkbox if the COBOL data structure of the CICS input message is different to the structure of the CICS output message, that is, the output overlays the input, which means there is a second COBOL layout for the reply. See the following COBOL server examples in the IDL Extractor for COBOL documentation:
 - Example 1: Redefines
 - Example 2: Buffer Technique

■ *Example 3: COBOL SET ADDRESS Statements*

For our example, check the check box, because the COBOL layout for input is the same as the COBOL layout for output.

- 5 Press **Next** to enter the COBOL Mapping Editor.



The purpose of the COBOL Mapping Editor is to map COBOL data items to IDL (Interface Definition Language) parameters. It consists of three main areas:

- The upper window pane is the COBOL program source.
- The window pane on the left-hand side is the COBOL interface, consisting of COBOL data items representing the parameters in your COBOL server program.
- The window pane on the right-hand side is the IDL interface consisting of IDL parameters.

What you select depends on check-box **Input Message same as Output Message**:

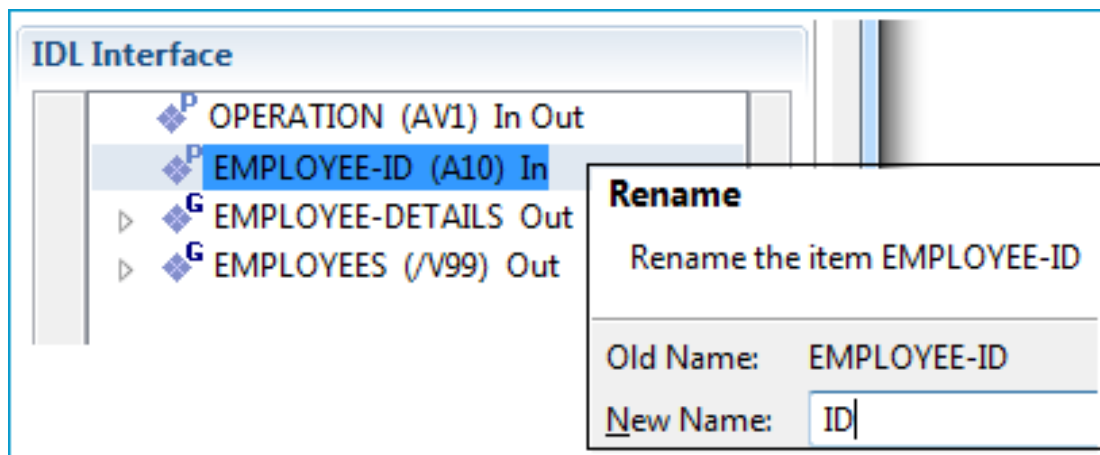
- If there is a single COBOL data structure of the CICS message for input and output - check box **Input Message same as Output Message** is checked - you select a single COBOL data structure in the COBOL Mapping Editor. This structure is used for COBOL input as well as for COBOL output. For more information in this case see *Mapping Editor User Interface* in the IDL Extractor for COBOL documentation.
- If there is a second COBOL data structure for the CICS output message - check box **Input Message same as Output Message** is cleared - you will select two COBOL data structures in the COBOL Mapping Editor. One is used for COBOL input and the other is for COBOL output. For more information in this case see *Mapping Editor User Interface* in the IDL Extractor for COBOL documentation.

COBOL extraction is a two-step process:

- First, select the COBOL parameters using the **Add to** and **Remove from COBOL Interface** from the context menu on the COBOL data items in the upper window pane (COBOL source); in many situations the COBOL interface is detected by the Mapping Editor directly and you can omit this step.
- Then map the COBOL parameters to IDL directions using the **Map to In/Out/InOut** buttons. COBOL parameters do not have directions, they are always INOUT. This step can be left out for a quick test in non-production scenarios; for real production it is recommended to set IDL directions to reduce data transfer. You also have the option to assign a meaningful name to an IDL parameter if the original COBOL data item is not self-explanatory.

For our example:

- Set **Map to In** for COBOL data item `EMPLOYEE-ID` as IDL direction (see screenshot in previous step). For `EMPLOYEES-DETAILS` and `EMPLOYEES` perform the similar **Map to Out**.
- Use the function **Rename** from the context menu on the IDL parameters in the **IDL Interface** pane to rename COBOL parameter `EMPLOYEES-ID` to IDL parameter `ID`. Do the same for `EMPLOYEE-DETAILS`: **Rename** to `EMPLOYEE`.



- 6 Press **Finish**. The default extraction settings ensure you get useful results. The outcome is a Software AG IDL file (interface definition language), a Designer file with extension .idl:

```
library 'EMPLAPPL' is
program 'EMPLOYEE' is
  define data parameter
    1 OPERATION (AV1) In Out
    1 ID (AV10) In /* Original Name:EMPLOYEE-ID
    1 EMPLOYEE Out /* Original Name:EMPLOYEE-DETAILS
      2 FIRSTNAME (AV20)
      2 SURNAME (AV20)
      2 DATE-BIRTH (AV12)
      2 DETAILS (AV100)
    1 EMPLOYEES (/V99) Out
      2 IDENT (AV10)
      2 FIRSTNAME (AV20)
      2 SURNAME (AV20)
      2 DATE-BIRTH (AV12)
  end-define
```

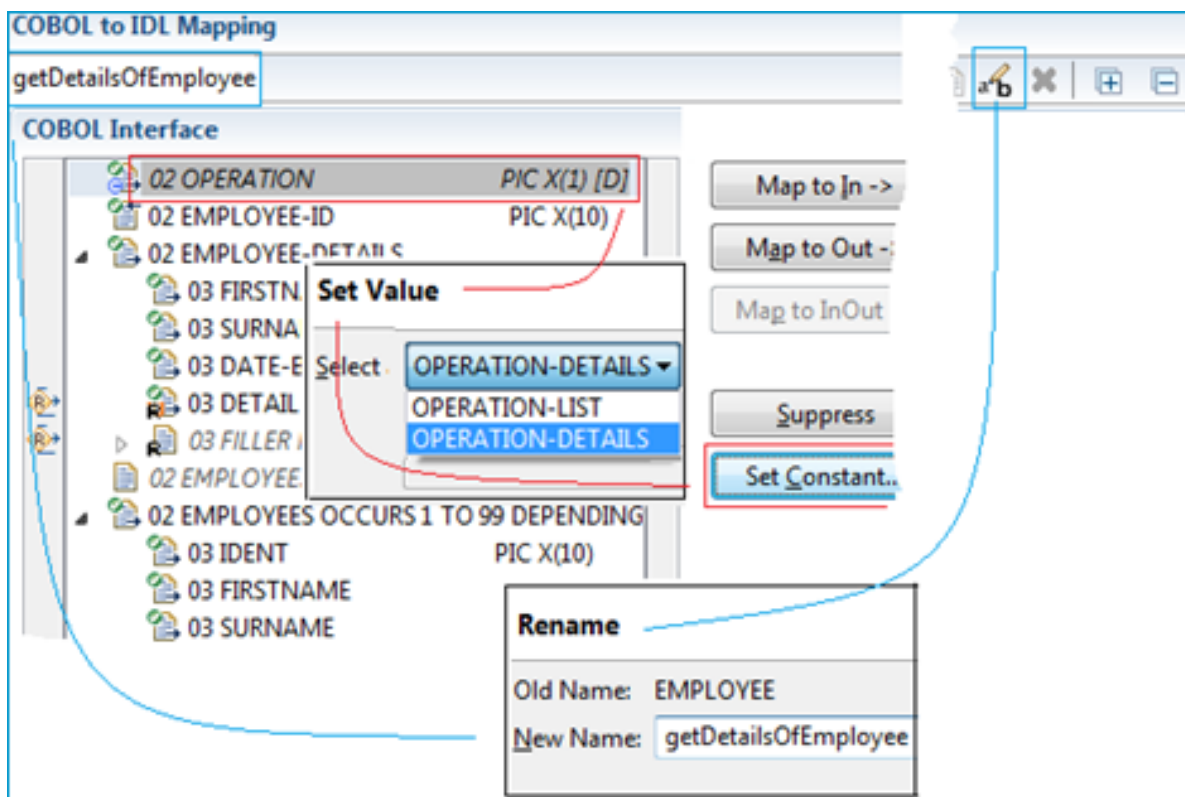
If required, a server mapping file with the same name as the IDL file but the extension .cvm may be generated too, for technical reasons. This file must always be stored together with the concomitant IDL file. For more information, see *Server Mapping Files for COBOL* in the Designer documentation. By default, the interface name `EMPLOYEE` is taken from the COBOL server program name.


Extracting a COBOL Server - Modern Method with User-defined Mapping

EntireX allows you to extract all implemented interfaces of the COBOL server separately. Instead of a large `EMPLOYEE` interface, separate interfaces `getListOfEmployees` and `getDetailsOfEmployee` are extracted. Each interface contains required parameters; obsolete parameters for an interface are suppressed, improving its usability.

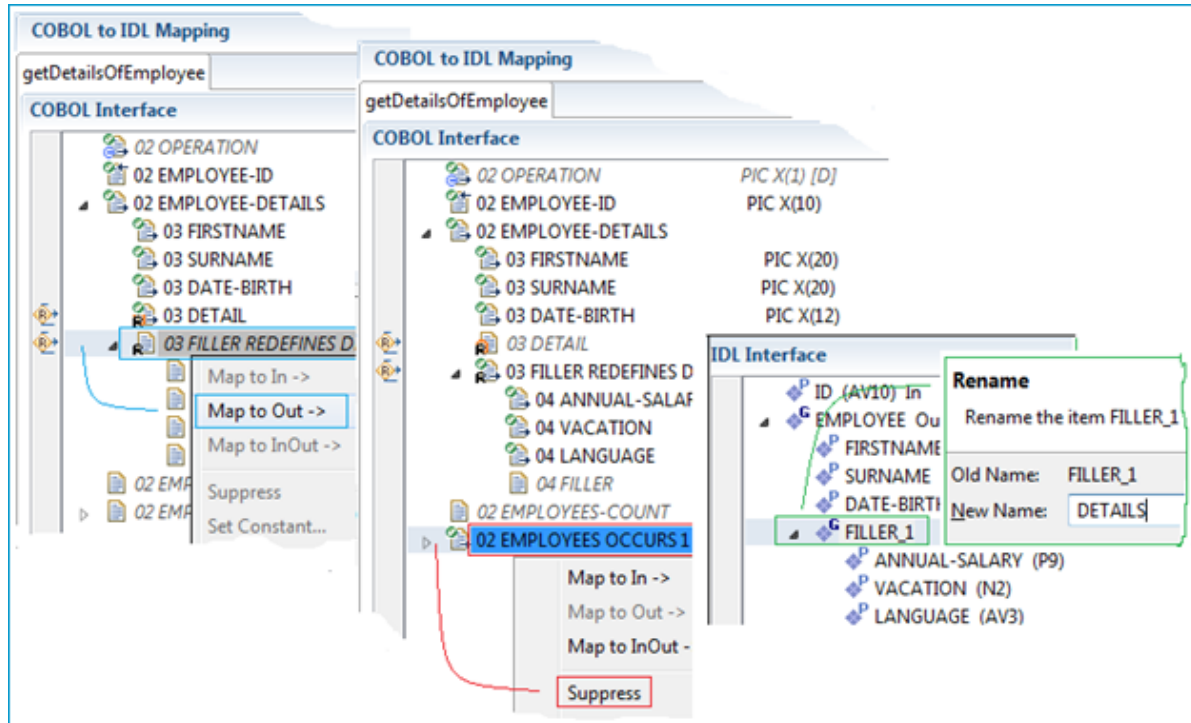
➤ To extract a COBOL server with a user-defined mapping

- 1 Shape `EMPLOYEE` to `getDetailsOfEmployee`:



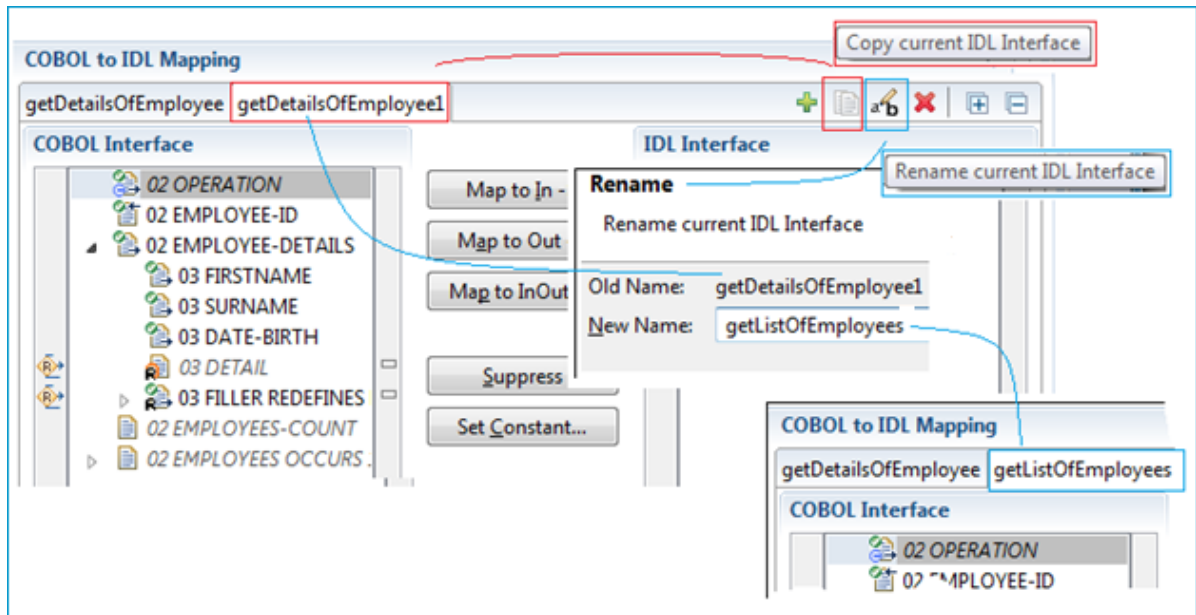
- Mark the `OPERATION` field in the **COBOL Interface** pane and press the **Set Constant** button. In the pop-up window, select `OPERATION-DETAILS` (red markers). The `OPERATION-DETAILS` value originates from COBOL level-88 data item (blue markers). If there is no COBOL level-88 enumeration type in the COBOL program defined, you can enter the function code manually in the pop-up window.
- By default the IDL interface gets the same name as your COBOL server: `EMPLOYEE`. Use the **Rename** button  from the toolbar of the COBOL Mapping Editor and change the IDL interface name to `getDetailsOfEmployee` (blue markers). This name is used as the IS service name later.

2 Continue shaping, making `getDetailsOfEmployee` easier to use:





- Use the context menu of the redefinition of the `DETAIL` field in the **COBOL Interface** pane, `FILLER REDEFINES DETAIL` and choose **Map to Out** (blue markers). A redefinition is a secondary layout (field types) of the same memory area. The redefinition contains more specific information (`ANNUAL-SALARY`, `VACATION`, etc.) than character buffer `DETAIL`. This is the reason why it is more useful in your interface.
- Use the context menu of `FILLER_1` in the IDL interface pane and rename `FILLER_1` to `DETAILS` (green markers) for a readable IS service field name later.
- The `DETAILS` function in the COBOL server does not make use of the `EMPLOYEES` field. Therefore you can leave it out of the interface: use the context menu of the `EMPLOYEES` field in the **COBOL Interface** pane and choose **Suppress** (red markers).

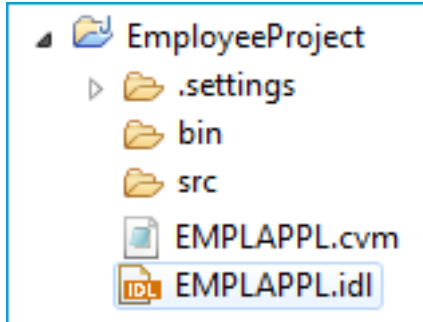
3 Shape `EMPLOYEE` to `getListOfEmployees` Function:



The COBOL Mapping Editor allows you to define multiple IDL interfaces to the same COBOL server:

- You create the additional IDL interface using the toolbar button **Copy current IDL interface**  (red markers). A new tab `getDetailsOfEmployee1` is created. The previously extracted `getDetailsOfEmployee` interface still exists in the first tab. Once you reactivate the first tab, you will see the interface of `getDetailsOfEmployee` again.
 - Use the **Rename** button  from the toolbar of the COBOL Mapping Editor and change the IDL interface name to `getListOfEmployees` (blue markers). This name is used as the IS service name later.
 - For the `LIST` function you need the `OPERATION-LIST` value in the `OPERATION` field: Mark the `OPERATION` field in the **COBOL Interface** pane and press the **Set Constant** button. In the pop-up window, select `OPERATION-LIST` (similar to Step 1. above *Shape `EMPLOYEE` to `getDetailsOfEmployee`*; red markers).
 - Use the context menu of the `EMPLOYEES` field in the **COBOL Interface** pane, `EMPLOYEES OCCURS 1 TO 99` and choose **Map to Out** (Similar to Step 2. above *Continue shaping, making `getDetailsOfEmployee` easier to use*; blue markers).
 - Because the `EMPLOYEE` and `EMPLOYEE-DETAIL` fields are not used in the COBOL server `LIST` function, you leave them out in the IDL interface: use the context menu of `EMPLOYEE-DETAIL` field in the **COBOL Interface** pane and choose **Suppress** (similar to Step 2. above *Continue shaping, making `getDetailsOfEmployee` easier to use*; red markers). Do the same for the `EMPLOYEE-ID` field.
- 4 Press **Finish** to retrieve the extraction result in the form of a Software AG IDL file. At the same time, a server mapping file for COBOL (Designer file with extension `.cvm`) is created.

See *Server Mapping Files for COBOL* in the Designer documentation. The *Software AG IDL File* in the IDL Editor documentation describes the interfaces from the client point of view, while the server mapping file contains the mapping to the real COBOL server. Both of these files must be kept together and in sync, otherwise a call to the COBOL server may fail.



To summarize: We created two IDL interfaces. In the IDL file, these resulted in two IDL programs: `getListOfEmployees` and `getDetailsOfEmployee`. Both IDL programs were given readable names (Steps 1 and 3). Meaningful fields were kept, while superfluous fields were suppressed (Steps 1 and 3). The program `getDetailsOfEmployee` contains the redefined fields of parameter `DETAILS` (see below) mapped during extraction (Step 1).



Note: At runtime, the RPC client generated with the extracted interface will send data for the redesigned interfaces, while your COBOL server still expects `EMPLOYEE` data. The EntireX runtime transforms the incoming data stream from the RPC client, using the server mapping file.

```
program 'getDetailsOfEmployee' is
  define data parameter
    1 ID (AV10) In /* Original Name:EMPLOYEE-ID
    1 EMPLOYEE Out /* Original Name:EMPLOYEE-DETAILS
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
    2 DETAILS /* Original Name:FILLER_1
    3 ANNUAL-SALARY (P9)
    3 VACATION (N2)
    3 LANGUAGE (AV3)
  end-define
  program 'getListOfEmployees' is
    1 EMPLOYEES (/V99) Out
    2 IDENT (AV10)
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
  end-define
```


Comparison Chart: Fast-track versus User-defined Mapping

The following table highlights the differences when shaping an interface with a *user-defined mapping* compared with a *fast-path extraction*.

	Interface Shaping with User-defined Mapping	Fast-path Extraction
General	User-defined interface with dedicated mapping. For our example, the interfaces are small and tidy and more self-explanatory with long, readable interface names. They are easier to use with the hidden <code>OPERATION</code> field and the suppressed COBOL fields, which are not needed.	Automatic, quick result extraction with COBOL-like interfaces on IS. For our example, the IS interface <code>EMPLOYEE</code> matches exactly 1:1, field by field.
IS service(s)	Multiple small and handy IS services; each <code>OPERATION</code> code is mapped to a separate IS service.	One big IS service.
IS service name	Readable long name.	Short Subprogram name; up to 8 characters.
IS fields	Usage of Suppress and Set Constant reduces the message length. This keeps focus on relevant data items and keeps the client's interface clean. It may also improve performance.	The IS fields and COBOL server parameters match 1:1. As COBOL layout descriptions are sometimes used for many different purposes, irrelevant data items appear and clutter up the IS interface.
<code>OPERATION</code> parameter	Suppressed: the <code>OPERATION</code> parameter does not exist in the IS service as an IS field.	The <code>OPERATION</code> parameter exists in the IS service as an IS field and needs to be filled-in by the client endpoint.
<code>OPERATION</code> code	The <code>OPERATION</code> code is provided internally by EntireX runtime in the <code>OPERATION</code> field.	The <code>OPERATION</code> code needs to be specified in the IS service in the <code>OPERATION</code> field by the client endpoint.
<code>REDEFINE</code> parameters	Either the parameter that is redefined or one of its redefinitions is available as an IS field.	Only the parameter that is redefined is available as an IS field. Redefinitions of the parameter are not available as IS fields.

A user-defined mapping enables you also to define alternative mappings for COBOL data items (`REDEFINES`, tables etc.). These enable scenarios with COBOL servers where data exchange is not fully described by the `LINKAGE SECTION` COBOL data items. For more information refer to the IDL Extractor for COBOL documentation:

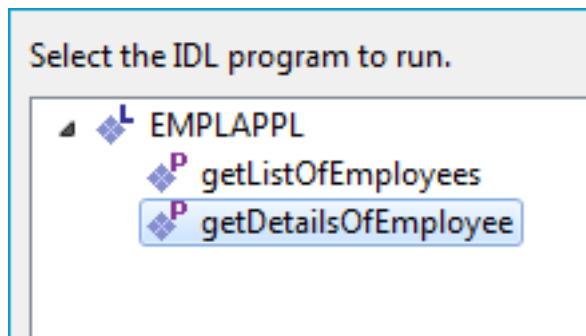
- *Mapping Editor IDL Interface Mapping Functions* if the check-box **Input Message same as Output Message** was checked.
- *Mapping Editor IDL Interface Mapping Functions* if the check-box **Input Message same as Output Message** was cleared.
- *User-defined Mapping*

Testing the Extraction Results

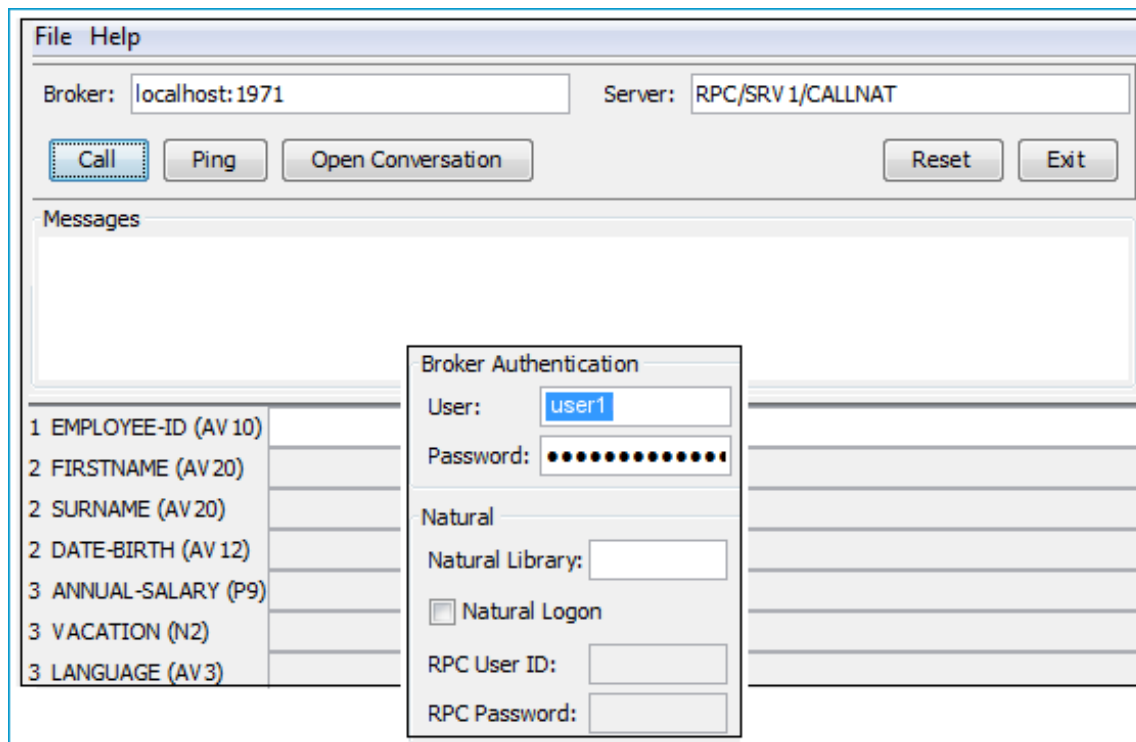
The following pictures use the extraction results described under [Extracting a COBOL Server - Modern Method with User-defined Mapping](#).

> To test the extraction results (optional)

- 1 You can test the results of the extraction operation and the REST server back end, using the EntireX IDL Tester. From the context menu of the IDL file in the Designer, choose **Software AG IDL Tester**.



- 2 Select `getDetailsOfEmployee`.



Note that the broker and server parameters contain the explicit route to call the server program, and you can optionally ping the connection from this client. With the **File > Options** dialog, you can set:

- **User** and **Password** for *broker* authentication
- **RPC User ID** and **RPC Password** for *server* authentication

See *EntireX IDL Tester* in the Designer documentation.

- 3 Check the Integration Server log, the EntireX Adapter log or the RPC logs. Applies to all connection methods.

Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server

This section describes your first steps to create a new Integration Server connection. This is described in more detail under *Using the Integration Server Wrapper*, for example working with existing Integration Server connections. This section covers the following topics:

- [Step 1: Start the Integration Server Wrapper Wizard](#)
- [Step 2: Create a New Integration Server Connection](#)
- [Step 3: Select the Connection Type](#)
- [Step 4: Define Adapter Services for a CICS Socket Listener Connection](#)
- [Step 5: Create or Update a REST Resource](#)
- [Step 6: Edit the REST Resource](#)

Step 1: Start the Integration Server Wrapper Wizard

➤ To start the Integration Server Wrapper wizard

- 1 In the context menu of a Software AG IDL file, choose **Integration Server > Generate web-Methods IS Connection**.

This starts the wizard with a list of existing Integration Server Wrapper connections.



Note: If the selected IDL file is not valid because of a syntax error, an error dialog comes up and the wizard does not start.

- 2 Continue with *Step 2: Create a New Integration Server Connection*.

Step 2: Create a New Integration Server Connection

Define New Integration Server Connection
Add host:port, user and password for a new Integration Server

Server:

User:

Password:

☐ Use secure connection

➤ To create a new Integration Server connection

- 1 Define the new Integration Server connection on the wizard page.



Notes:

1. The only required field is **Server**. Enter the hostname of the Integration Server including an optional port number. If no port number is specified, port number defaults to "5555". The **Integration Server Authentication** can be passed with the **User** and **Password** fields.
2. Optional settings are for secure connections. The **Truststore for HTTPS** contains all signed certificates and must be a valid truststore.
3. The check box **Verify host name** checks that the hostname is entered in the stored certificate.
4. When the Integration Server has **Client Authentication** enabled, you can specify your **Keystore** file and keystore **Password**.
5. For managing Integration Server connections, see [Preferences](#).

- 2 Choose **Next** and continue with *Step 3: Select the Connection Type*.

Step 3: Select the Connection Type

☒ Create a new EntireX Adapter connection of type **CICS Socket Listener Connection**

☐ Update Adapter Services or Listeners to an existing Integration Server connection:

List of Connections

Connection Name	Package Name	Connection Type	Enabled
Employees:EmployeesConnection	Employees	EntireX RPC Connection	Yes

Total: 1

☒ Map Software AG IDL data types to String

☒ Create or Update REST resource

> To create a new connection

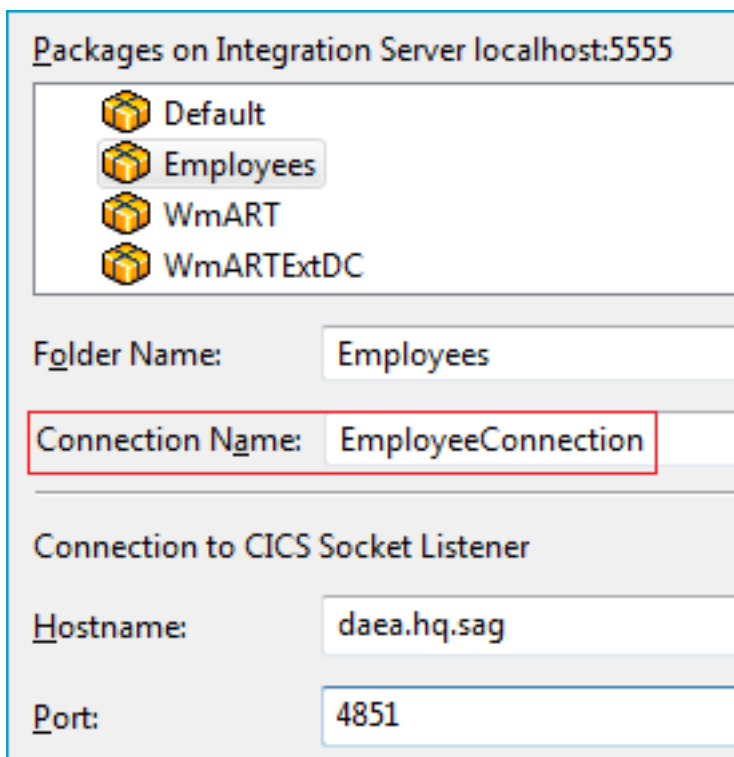
- 1 Select a connection type from the drop down list. Connection types are described under *EntireX Adapter Connections* in the EntireX Adapter documentation and *Introduction to the Integration Server Wrapper*.



Note: The list of connection types is filtered: connection types that require a license are only shown if a corresponding license file is available. Reliable RPC connections are only shown if all IDL programs contain only IN parameters. Also, if a server mapping file is available, only those connection types that support the interface type specified in the server mapping file are shown.

- 2 Check the box **Create or Update REST resource**.
- 3 Click **Next** and continue with *Step 4: Define Adapter Services for a CICS Socket Listener Connection*.

Step 4: Define Adapter Services for a CICS Socket Listener Connection



Packages on Integration Server localhost:5555

- Default
- Employees
- WmART
- WmARTEExtDC

Folder Name: Employees

Connection Name: EmployeeConnection

Connection to CICS Socket Listener

Hostname: daea.hq.sag

Port: 4851

➤ To create a connection and related adapter services

- 1 Select a package for the created objects.
- 2 Define a folder name. If the folder does not exist, it will be created.
- 3 Define a connection name.
- 4 Define the parameters of the connection type. For details, see *Connection Parameters for CICS Socket Listener Connections* in the EntireX Adapter documentation.

As a result, the folder will contain the connection and the adapter services (one for each IDL program). The name of a service is the same as the respective IDL program.

The default settings for the adapter services are:

- the **Default** package; if not available, the first package
- the IDL library name for the **Folder Name**
- the IDL library name with the suffix "Connection" for the **Connection Name**



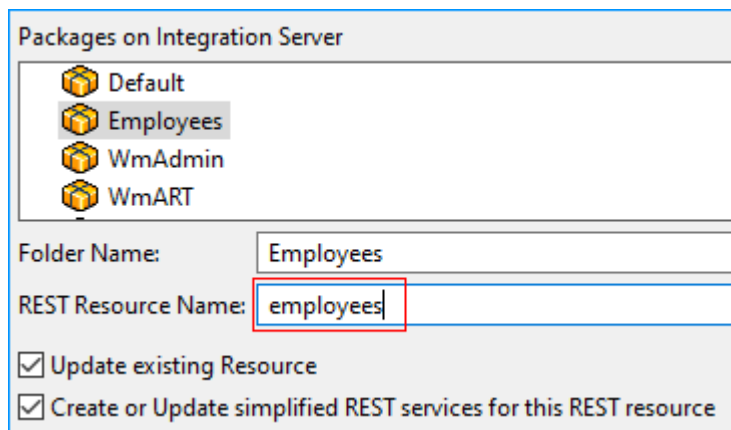
Note: When creating a connection, a package dependency is added such that the selected package depends on webMethods EntireX (the package `WmEntireX`) with the version currently used.

Step 5: Create or Update a REST Resource

➤ To create or update a REST resource

Make sure checkbox **Create or Update REST resource** is checked.

- 1 Select a package.



Packages on Integration Server

- Default
- Employees
- WmAdmin
- WmART

Folder Name: Employees

REST Resource Name: employees

☒ Update existing Resource

☒ Create or Update simplified REST services for this REST resource

- 2 Define a folder name. If the folder does not exist, it will be created.
- 3 Define the REST Resource name.
- 4 Check **Update existing Resource** if you want to add additional services to an existing resource.
- 5 Check **Create or Update simplified REST services for this REST resource** if you want to generate additional services with a simplified REST API. This allows you to use a simple HTTP-GET method without a payload.



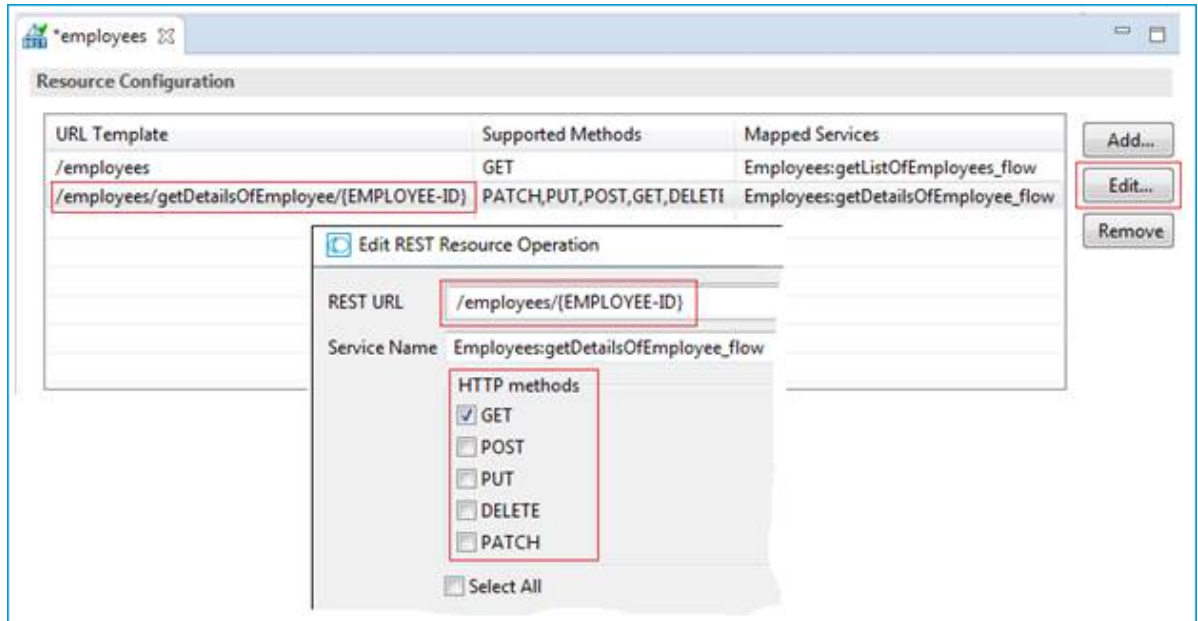
Note: This applies only to APIs where input signature parameters do not contain arrays.

;

Step 6: Edit the REST Resource

➤ To edit the REST resource

- 1 Switch to the **Service Development** perspective of Software AG Designer.



- 2 Refresh the package where you generated the REST Resource in the previous step.
- 3 Open the REST Resource.
- 4 Edit the REST URLs and HTTP Methods to match the functionality of your application.

For more information see:

- *REST Developer's Guide* in the webMethods Integration Server documentation for information on using the generated REST resources
- *REST Developer's Guide* in the webMethods Integration Server documentation

Task 3: Execute the Call from the REST Client to COBOL

Use an appropriate REST client, specify URL, Method and Headers and invoke your REST resource:

URL http://localhost:5555/restv2/employees	URL http://localhost:5555/restv2/employees/111000105
Method GET	Method GET
Headers Accept application/json	Headers Accept application/json
200 OK	200 OK
<pre>{ "errorCode": "00000000", "errorFlag": "false", "response": { "EMPLOYEES-LIST": { "EMPLOYEES": [{ "IDENT": "11100102", "FIRSTNAME": "Edgar", "SURNAME": "Schindler", "DATE-BIRTH": "Dec 4, 1962" }, { "IDENT": "11100105", "FIRSTNAME": "Christian", "SURNAME": "Schirm", "DATE-BIRTH": "Mar 15, 1961" }, { "IDENT": "11100106", "FIRSTNAME": "Rainer", "SURNAME": "Schmitt", "DATE-BIRTH": "Feb 13, 1955" }, { "IDENT": "11100107", "FIRSTNAME": "Helga", "SURNAME": "Schmidt", "DATE-BIRTH": "May 26, 1961" }] } } }</pre>	<pre>{ "errorCode": "00000000", "errorFlag": "false", "response": { "EMPLOYEE-DETAILS": { "FIRSTNAME": "Christian", "SURNAME": "Schirm", "DATE-BIRTH": "Mar 15, 1961", "DETAIL": { "ANNUAL-SALARY": "68000", "VACATION": "21", "LANGUAGE": "GER" } } } }</pre>

22

Calling COBOL Channel Container (Minimal Footprint Using CICS Socket Listener) on z/VSE CICS from REST

■ Introduction	381
■ What do I need to Install for this Scenario?	383
■ Task 1: Extract the Interface of a COBOL Server	384
■ Task 2: Generate the Connection and Adapter Services in Integration Server	396
■ Task 3: Execute the Call from the REST Client to COBOL	402

This scenario uses the tools IDL Extractor for COBOL and Integration Server Wrapper of the Designer.

Introduction



- ❶ Extract the interface of the COBOL server program. See *Using the IDL Extractor for COBOL - Overview*.
- ❷ Generate REST resources, connection and adapter services in Integration Server.
- ❸ Execute the call from the REST client to the COBOL server program.

This scenario makes the following important assumptions:

For Task 1:

- You have a working COBOL Channel Container server. For illustration and examples on such a server, see *CICS with Channel Container Calling Convention* under *Introduction to the IDL Extractor for COBOL*.
- You have access to the related COBOL sources and copybooks as files on your local machine. The minimum requirement is the `DATA DIVISION` of the interface.



For Tasks 2 and 3:

- You have a REST client.
- To call the COBOL server program at runtime using the *EntireX CICS Socket Listener* connection method, you need to install the CICS Socket Listener within your CICS region: see *Preparing for CICS Socket Listener*. For configuration, see *Connection Parameters for CICS Socket Listener Connections*.



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.
- You have the CICS Socket Listener installed. See *Installing the CICS Socket Listener* in the z/VSE Installation documentation.

Task 1: Extract the Interface of a COBOL Server

- [Introduction](#)
- [Sample COBOL Server used in the Examples](#)
- [Extracting a COBOL Server - Fast-track Method](#)
- [Extracting a COBOL Server - Modern Method with User-defined Mapping](#)
- [Comparison Chart: Fast-track versus User-defined Mapping](#)
- [Testing the Extraction Results](#)

Introduction

Follow the instructions for extracting COBOL, see *Using the IDL Extractor for COBOL - Overview* and choose *Scenario I: Create New IDL and Server Mapping Files* if this is your first extraction. Learn how to invoke the IDL Extractor for COBOL, create a COBOL Extractor Environment and about the possibilities to select the COBOL source.

The extraction process creates the following EntireX metafiles:

- IDL file. A Software AG IDL file contains definitions of the interface between client and server. See *Software AG IDL File* in the IDL Editor documentation in the IDL Editor documentation.
- Server mapping file (optional). The mapping file is a Designer file with extension .cvm that contains COBOL-specific mapping information. See *Server Mapping Files for COBOL* in the Designer documentation.

Sample COBOL Server used in the Examples

The following COBOL server is used to illustrate the features of the IDL Extractor for COBOL. Imagine an existing COBOL server called `EMPLOYEE`. It implements the access logic for a `LIST` and `DETAILS` function to a database view `EMPLOYEE`:


```

IDENTIFICATION DIVISION.
    PROGRAM-ID.      EMPLOYEE.
*****
*   Program Name    = EMPLOYEE
*   String Literal  = QUOTE
*****
01 DFHCOMMAREA.
    02 OPERATION                                PIC X(1).
        88 OPERATION-LIST                      VALUE 'L'.
        88 OPERATION-DETAILS                  VALUE 'D'.
    02 EMPLOYEE-ID                                PIC X(10).
    02 EMPLOYEE-DETAILS.
        03 FIRSTNAME                            PIC X(20).
        03 SURNAME                              PIC X(20).
        03 DATE-BIRTH                          PIC X(12).
        03 DETAILS                             PIC X(100).
        03 FILLER REDEFINES DETAILS.
            04 ANNUAL-SALARY          PACKED-DECIMAL PIC S9(9).
            04 VACATION                PIC S9(2).
            04 LANGUAGE                PIC X(3).
            04 FILLER                  PIC X(90).
    02 EMPLOYEES-COUNT                          PIC 9(8) BINARY.
    02 EMPLOYEES OCCURS 1 TO 99 DEPENDING ON EMPLOYEES-COUNT.
        03 IDENT                          PIC X(10).
        03 FIRSTNAME                      PIC X(20).
        03 SURNAME                        PIC X(20).
        03 DATE-BIRTH                    PIC X(12).

```

Two approaches for extracting the COBOL server are described below:

- **Fast-track**

Learn how EntireX helps you to connect the COBOL Server with quick results.

- **User-defined Mapping**

The following features are among those available with a user-defined mapping:

- **Shape the interface to your COBOL server**

You can minimize the interface by suppressing COBOL server fields or by providing constant values. Renaming interfaces allows you to specify a readable long name. These two features together - constants and renaming of interfaces - are most powerful when multiple interfaces are implemented in a single COBOL server. Each function of the COBOL server triggered by an operation code or function code can be modelled as a separate REST resource method/operation.

■ **Select alternative mappings**

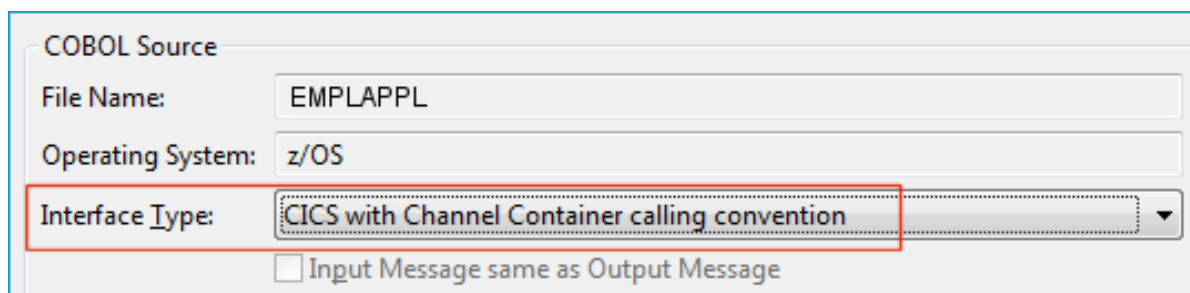
You can select alternative mappings for COBOL data items (REDEFINES, tables etc.) to enable scenarios with COBOL servers where data exchange is not fully described by the LINKAGE SECTION COBOL data items.

These approaches are described under *Fast-track Method* and *Modern Method* below.

Extracting a COBOL Server - Fast-track Method

➤ **To extract the COBOL Server**

- 1 Switch to the **EntireX** perspective.
- 2 Call the IDL Extractor for COBOL.
- 3 Set the correct interface type as described under *Step 4: Define the Extraction Settings and Start Extraction* under *Scenario I: Create New IDL and Server Mapping Files* in the IDL Extractor for COBOL documentation.



COBOL Source

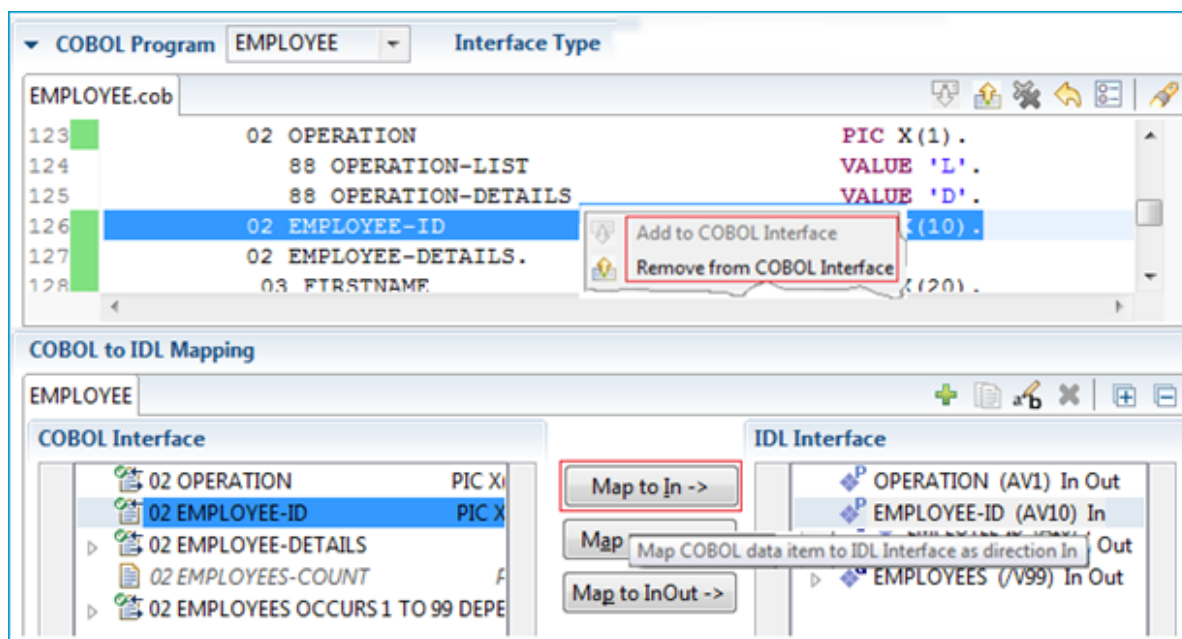
File Name: EMPLAPPL

Operating System: z/OS

Interface Type: CICS with Channel Container calling convention

☐ Input Message same as Output Message

- 4 Press **Next** to enter the COBOL Mapping Editor.



The purpose of the COBOL Mapping Editor is to map COBOL data items to IDL (Interface Definition Language) parameters. It consists of three main areas:

- The upper window pane is the COBOL program source.
- The window pane on the left-hand side is the COBOL interface, consisting of COBOL data items representing the parameters in your COBOL server program.
- The window pane on the right-hand side is the IDL interface consisting of IDL parameters.

For more information see *Mapping Editor User Interface* in the IDL Extractor for COBOL documentation.

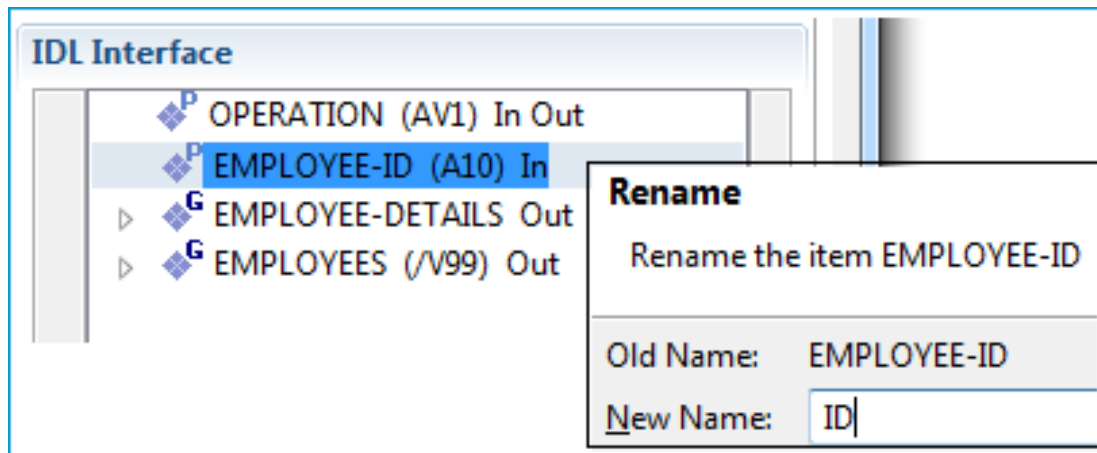
COBOL extraction is a two-step process:

- First, select the COBOL parameters using the **Add to** and **Remove from COBOL Interface** from the context menu on the COBOL data items in the upper window pane (COBOL source); in many situations the COBOL interface is detected by the Mapping Editor directly and you can omit this step.
- Then map the COBOL parameters to IDL directions using the **Map to In/Out/InOut** buttons. COBOL parameters do not have directions, they are always INOUT. This step can be left out for a quick test in non-production scenarios; for real production it is recommended to set IDL directions to reduce data transfer. You also have the option to assign a meaningful name to an IDL parameter if the original COBOL data item is not self-explanatory.

For our example:

- Set **Map to In** for COBOL data item `EMPLOYEE-ID` as IDL direction (see screenshot in previous step). For `EMPLOYEES-DETAILS` and `EMPLOYEES` perform the similar **Map to Out**.

- Use the function **Rename** from the context menu on the IDL parameters in the **IDL Interface** pane to rename COBOL parameter `EMPLOYEE- ID` to IDL parameter `ID`. Do the same for `EMPLOYEE- DETAILS`: Rename to `EMPLOYEE`.



- 5 Press **Finish**. The default extraction settings ensure you get useful results. The outcome is a Software AG IDL file (interface definition language), a Designer file with extension `.idl`:

```
library 'EMPLAPPL' is
program 'EMPLOYEE' is
  define data parameter
    1 OPERATION (AV1) In Out
    1 ID (AV10) In /* Original Name:EMPLOYEE-ID
    1 EMPLOYEE Out /* Original Name:EMPLOYEE-DETAILS
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
    2 DETAILS (AV100)
    1 EMPLOYEES (/V99) Out
    2 IDENT (AV10)
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
  end-define
```

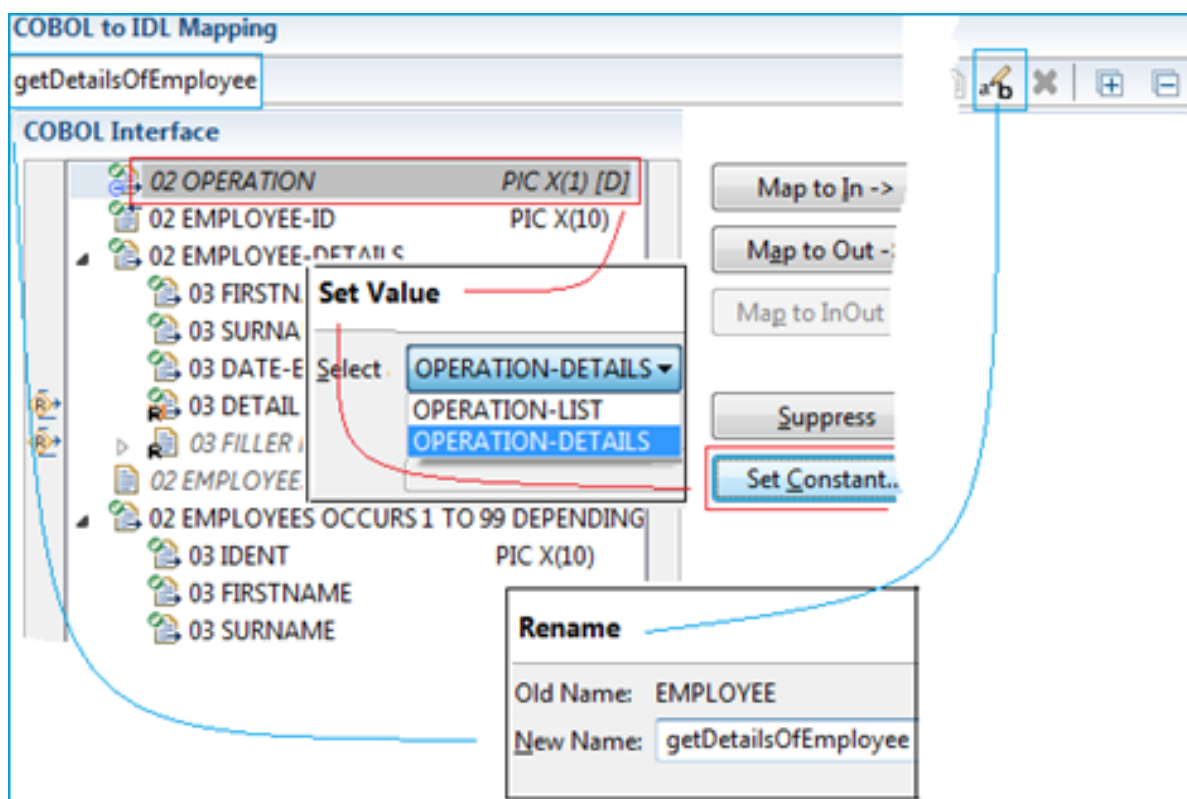
If required, a server mapping file with the same name as the IDL file but the extension `.cvm` may be generated too, for technical reasons. This file must always be stored together with the concomitant IDL file. For more information, see *Server Mapping Files for COBOL* in the Designer documentation. By default, the interface name `EMPLOYEE` is taken from the COBOL server program name.


Extracting a COBOL Server - Modern Method with User-defined Mapping

EntireX allows you to extract all implemented interfaces of the COBOL server separately. Instead of a large `EMPLOYEE` interface, separate interfaces `getListOfEmployees` and `getDetailsOfEmployee` are extracted. Each interface contains required parameters; obsolete parameters for an interface are suppressed, improving its usability.

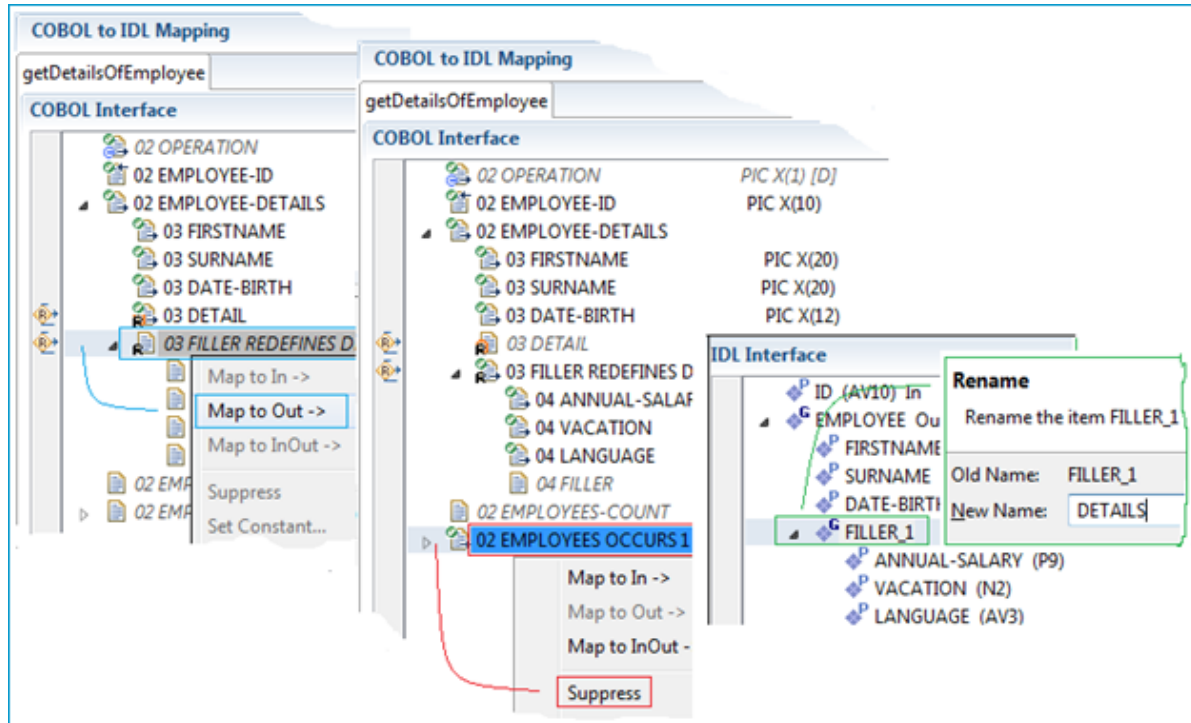
➤ To extract a COBOL server with a user-defined mapping

- 1 Shape `EMPLOYEE` to `getDetailsOfEmployee`:



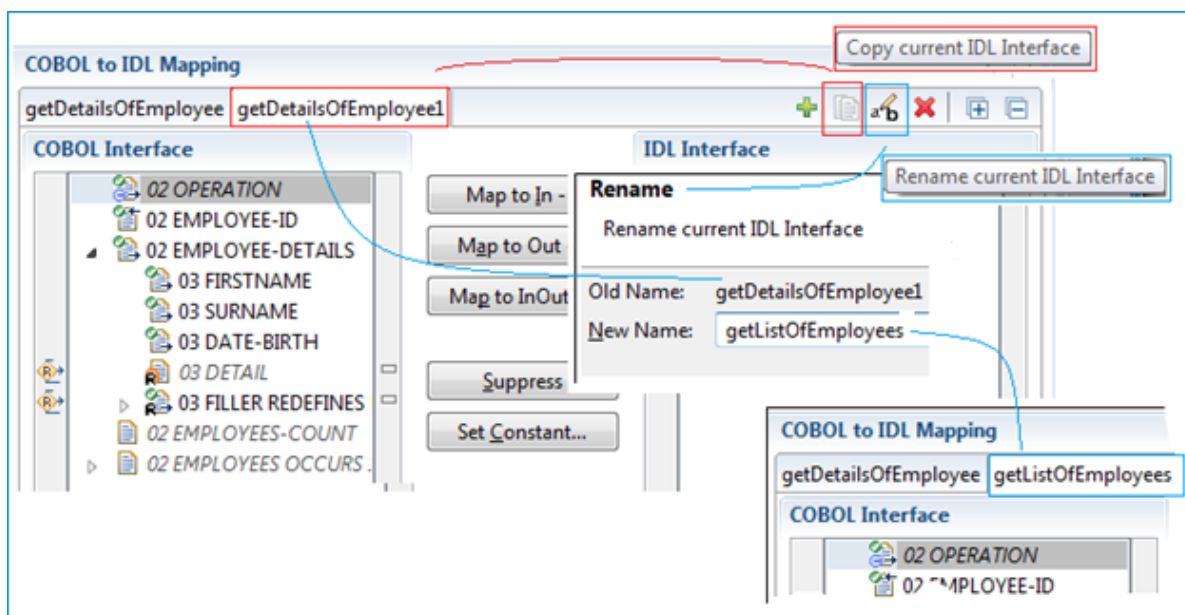
- Mark the `OPERATION` field in the COBOL **Interface** pane and press the **Set Constant** button. In the pop-up window, select `OPERATION-DETAILS` (red markers). The `OPERATION-DETAILS` value originates from COBOL level-88 data item (blue markers). If there is no COBOL level-88 enumeration type in the COBOL program defined, you can enter the function code manually in the pop-up window.
- By default the IDL interface gets the same name as your COBOL server: `EMPLOYEE`. Use the **Rename** button  from the toolbar of the COBOL Mapping Editor and change the IDL interface name to `getDetailsOfEmployee` (blue markers). This name is used as the IS service name later.

- 2 Continue shaping, making `getDetailsOfEmployee` easier to use:



- Use the context menu of the redefinition of the `DETAIL` field in the **COBOL Interface** pane, `FILLER REDEFINES DETAIL` and choose **Map to Out** (blue markers). A redefinition is a secondary layout (field types) of the same memory area. The redefinition contains more specific information (`ANNUAL-SALARY`, `VACATION`, etc.) than character buffer `DETAIL`. This is the reason why it is more useful in your interface.
- Use the context menu of `FILLER_1` in the IDL interface pane and rename `FILLER_1` to `DETAILS` (green markers) for a readable IS service field name later.
- The `DETAILS` function in the COBOL server does not make use of the `EMPLOYEES` field. Therefore you can leave it out of the interface: use the context menu of the `EMPLOYEES` field in the **COBOL Interface** pane and choose **Suppress** (red markers).


3 Shape `EMPLOYEE` to `getListOfEmployees` Function:



The COBOL Mapping Editor allows you to define multiple IDL interfaces to the same COBOL server:

- You create the additional IDL interface using the toolbar button **Copy current IDL interface**

 (red markers). A new tab `getDetailsOfEmployee1` is created. The previously extracted `getDetailsOfEmployee` interface still exists in the first tab. Once you reactivate the first tab, you will see the interface of `getDetailsOfEmployee` again.

- Use the **Rename** button  from the toolbar of the COBOL Mapping Editor and change the IDL interface name to `getListOfEmployees` (blue markers). This name is used as the IS service name later.

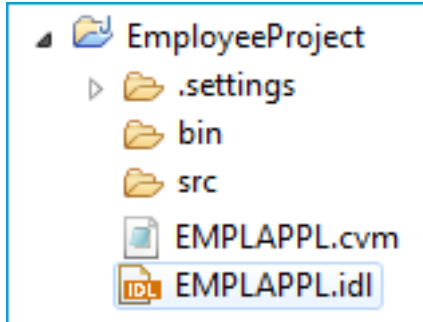
- For the `LIST` function you need the `OPERATION-LIST` value in the `OPERATION` field: Mark the `OPERATION` field in the **COBOL Interface** pane and press the **Set Constant** button. In the pop-up window, select `OPERATION-LIST` (similar to Step 1. above *Shape EMPLOYEE to getDetailsOfEmployee*; red markers).

- Use the context menu of the `EMPLOYEES` field in the **COBOL Interface** pane, `EMPLOYEES OCCURS 1 TO 99` and choose **Map to Out** (Similar to Step 2. above *Continue shaping, making getDetailsOfEmployee easier to use*; blue markers).

- Because the `EMPLOYEE` and `EMPLOYEE-DETAIL` fields are not used in the COBOL server `LIST` function, you leave them out in the IDL interface: use the context menu of `EMPLOYEE-DETAIL` field in the **COBOL Interface** pane and choose **Suppress** (similar to Step 2. above *Continue shaping, making getDetailsOfEmployee easier to use*; red markers). Do the same for the `EMPLOYEE-ID` field.

- 4 Press **Finish** to retrieve the extraction result in the form of a Software AG IDL file. At the same time, a server mapping file for COBOL (Designer file with extension `.cvm`) is created.

See *Server Mapping Files for COBOL* in the Designer documentation. The *Software AG IDL File* in the IDL Editor documentation describes the interfaces from the client point of view, while the server mapping file contains the mapping to the real COBOL server. Both of these files must be kept together and in sync, otherwise a call to the COBOL server may fail.



To summarize: We created two IDL interfaces. In the IDL file, these resulted in two IDL programs: `getListOfEmployees` and `getDetailsOfEmployee`. Both IDL programs were given readable names (Steps 1 and 3). Meaningful fields were kept, while superfluous fields were suppressed (Steps 1 and 3). The program `getDetailsOfEmployee` contains the redefined fields of parameter `DETAILS` (see below) mapped during extraction (Step 1).



Note: At runtime, the RPC client generated with the extracted interface will send data for the redesigned interfaces, while your COBOL server still expects `EMPLOYEE` data. The EntireX runtime transforms the incoming data stream from the RPC client, using the server mapping file.

```
program 'getDetailsOfEmployee' is
  define data parameter
    1 ID (AV10) In /* Original Name:EMPLOYEE-ID
    1 EMPLOYEE Out /* Original Name:EMPLOYEE-DETAILS
      2 FIRSTNAME (AV20)
      2 SURNAME (AV20)
      2 DATE-BIRTH (AV12)
      2 DETAILS /* Original Name:FILLER_1
        3 ANNUAL-SALARY (P9)
        3 VACATION (N2)
        3 LANGUAGE (AV3)
  end-define
  program 'getListOfEmployees' is
    1 EMPLOYEES (/V99) Out
      2 IDENT (AV10)
      2 FIRSTNAME (AV20)
      2 SURNAME (AV20)
      2 DATE-BIRTH (AV12)
  end-define
```


Comparison Chart: Fast-track versus User-defined Mapping

The following table highlights the differences when shaping an interface with a *user-defined mapping* compared with a *fast-path extraction*.

	Interface Shaping with User-defined Mapping	Fast-path Extraction
General	User-defined interface with dedicated mapping. For our example, the interfaces are small and tidy and more self-explanatory with long, readable interface names. They are easier to use with the hidden <code>OPERATION</code> field and the suppressed COBOL fields, which are not needed.	Automatic, quick result extraction with COBOL-like interfaces on IS. For our example, the IS interface <code>EMPLOYEE</code> matches exactly 1:1, field by field.
IS service(s)	Multiple small and handy IS services; each <code>OPERATION</code> code is mapped to a separate IS service.	One big IS service.
IS service name	Readable long name.	Short Subprogram name; up to 8 characters.
IS fields	Usage of Suppress and Set Constant reduces the message length. This keeps focus on relevant data items and keeps the client's interface clean. It may also improve performance.	The IS fields and COBOL server parameters match 1:1. As COBOL layout descriptions are sometimes used for many different purposes, irrelevant data items appear and clutter up the IS interface.
<code>OPERATION</code> parameter	Suppressed: the <code>OPERATION</code> parameter does not exist in the IS service as an IS field.	The <code>OPERATION</code> parameter exists in the IS service as an IS field and needs to be filled-in by the client endpoint.
<code>OPERATION</code> code	The <code>OPERATION</code> code is provided internally by EntireX runtime in the <code>OPERATION</code> field.	The <code>OPERATION</code> code needs to be specified in the IS service in the <code>OPERATION</code> field by the client endpoint.
<code>REDEFINE</code> parameters	Either the parameter that is redefined or one of its redefinitions is available as an IS field.	Only the parameter that is redefined is available as an IS field. Redefinitions of the parameter are not available as IS fields.

A user-defined mapping enables you also to define alternative mappings for COBOL data items (`REDEFINES`, tables etc.). These enable scenarios with COBOL servers where data exchange is not fully described by the `LINKAGE SECTION` COBOL data items. For more information refer to the IDL Extractor for COBOL documentation:

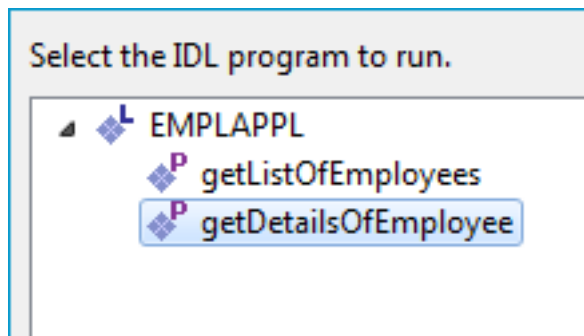
- *Mapping Editor IDL Interface Mapping Functions*
- *User-defined Mapping*

Testing the Extraction Results

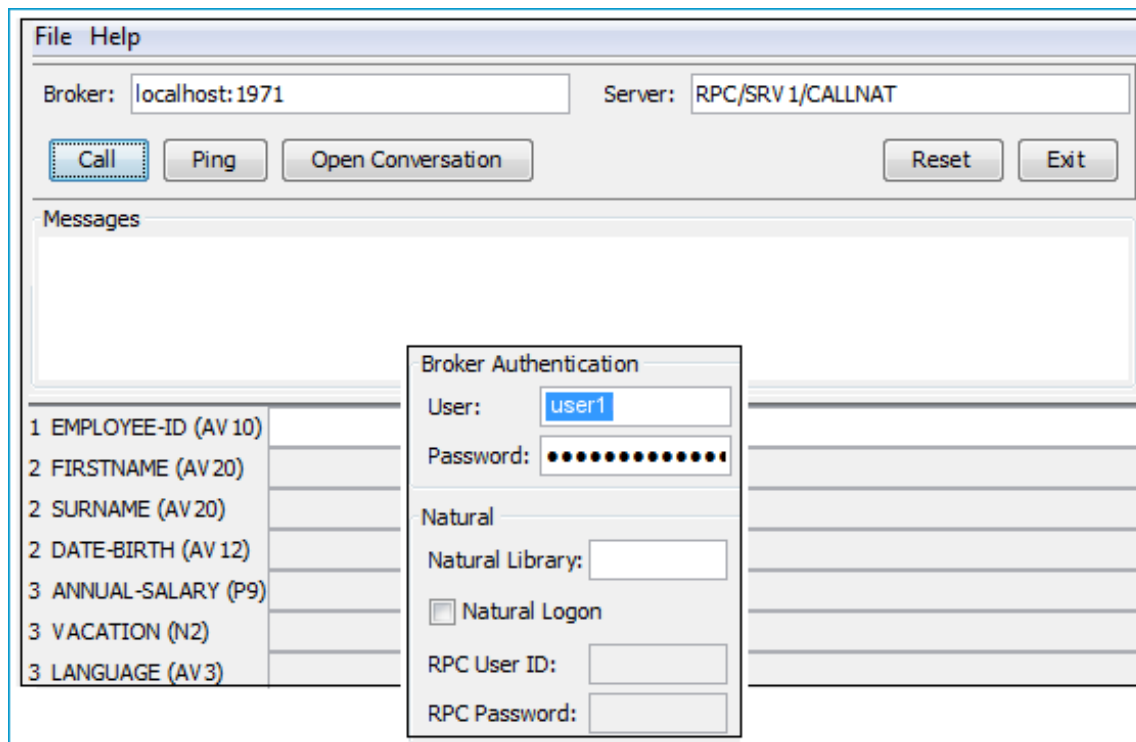
The following pictures use the extraction results described under [Extracting a COBOL Server - Modern Method with User-defined Mapping](#).

> To test the extraction results (optional)

- 1 You can test the results of the extraction operation and the REST server back end, using the EntireX IDL Tester. From the context menu of the IDL file in the Designer, choose **Software AG IDL Tester**.



- 2 Select `getDetailsOfEmployee`.



Note that the broker and server parameters contain the explicit route to call the server program, and you can optionally ping the connection from this client. With the **File > Options** dialog, you can set:

- **User** and **Password** for *broker* authentication
- **RPC User ID** and **RPC Password** for *server* authentication

See *EntireX IDL Tester* in the Designer documentation.

- 3 Check the Integration Server log, the EntireX Adapter log or the RPC logs. Applies to all connection methods.

Task 2: Generate the Connection and Adapter Services in Integration Server

This section describes your first steps to create a new Integration Server connection. This is described in more detail under *Using the Integration Server Wrapper*, for example working with existing Integration Server connections. This section covers the following topics:

- [Step 1: Start the Integration Server Wrapper Wizard](#)
- [Step 2: Create a New Integration Server Connection](#)
- [Step 3: Select the Connection Type](#)
- [Step 4: Define Adapter Services for a CICS Socket Listener Connection](#)
- [Step 5: Create or Update a REST Resource](#)
- [Step 6: Edit the REST Resource](#)

Step 1: Start the Integration Server Wrapper Wizard

➤ To start the Integration Server Wrapper wizard

- 1 In the context menu of a Software AG IDL file, choose **Integration Server > Generate web-Methods IS Connection**.

This starts the wizard with a list of existing Integration Server Wrapper connections.



Note: If the selected IDL file is not valid because of a syntax error, an error dialog comes up and the wizard does not start.

- 2 Continue with *Step 2: Create a New Integration Server Connection*.

Step 2: Create a New Integration Server Connection

Define New Integration Server Connection
Add host:port, user and password for a new Integration Server

Server:

User:

Password:

☐ Use secure connection

➤ To create a new Integration Server connection

- 1 Define the new Integration Server connection on the wizard page.



Notes:

1. The only required field is **Server**. Enter the hostname of the Integration Server including an optional port number. If no port number is specified, port number defaults to "5555". The **Integration Server Authentication** can be passed with the **User** and **Password** fields.
2. Optional settings are for secure connections. The **Truststore for HTTPS** contains all signed certificates and must be a valid truststore.
3. The check box **Verify host name** checks that the hostname is entered in the stored certificate.
4. When the Integration Server has **Client Authentication** enabled, you can specify your **Keystore** file and keystore **Password**.
5. For managing Integration Server connections, see [Preferences](#).

- 2 Choose **Next** and continue with *Step 3: Select the Connection Type*.

Step 3: Select the Connection Type

☒ Create a new EntireX Adapter connection of type **CICS Socket Listener Connection**

☐ Update Adapter Services or Listeners to an existing Integration Server connection:

List of Connections

Connection Name	Package Name	Connection Type	Enabled
Employees:EmployeesConnection	Employees	EntireX RPC Connection	Yes

Total: 1

☒ Map Software AG IDL data types to String

☒ Create or Update REST resource

> To create a new connection

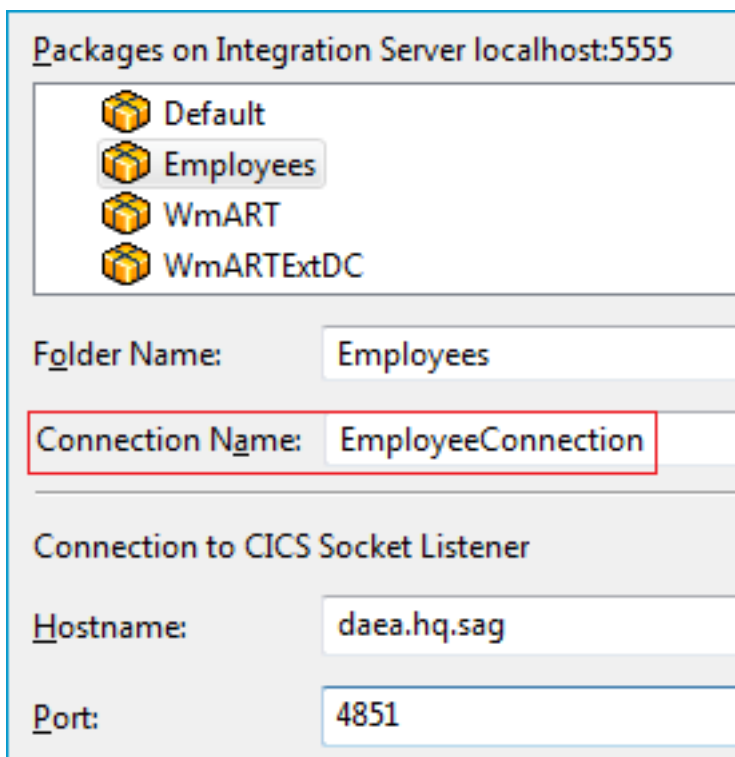
- 1 Select a connection type from the drop down list. Connection types are described under *EntireX Adapter Connections* in the EntireX Adapter documentation and *Introduction to the Integration Server Wrapper*.



Note: The list of connection types is filtered: connection types that require a license are only shown if a corresponding license file is available. Reliable RPC connections are only shown if all IDL programs contain only IN parameters. Also, if a server mapping file is available, only those connection types that support the interface type specified in the server mapping file are shown.

- 2 Check the box **Create or Update REST resource**.
- 3 Click **Next** and continue with *Step 4: Define Adapter Services for a CICS Socket Listener Connection*.

Step 4: Define Adapter Services for a CICS Socket Listener Connection



Packages on Integration Server localhost:5555

- Default
- Employees
- WmART
- WmARTEExtDC

Folder Name: Employees

Connection Name: EmployeeConnection

Connection to CICS Socket Listener

Hostname: daea.hq.sag

Port: 4851

➤ To create a connection and related adapter services

- 1 Select a package for the created objects.
- 2 Define a folder name. If the folder does not exist, it will be created.
- 3 Define a connection name.
- 4 Define the parameters of the connection type. For details, see *Connection Parameters for CICS Socket Listener Connections* in the EntireX Adapter documentation.

As a result, the folder will contain the connection and the adapter services (one for each IDL program). The name of a service is the same as the respective IDL program.

The default settings for the adapter services are:

- the **Default** package; if not available, the first package
- the IDL library name for the **Folder Name**
- the IDL library name with the suffix "Connection" for the **Connection Name**



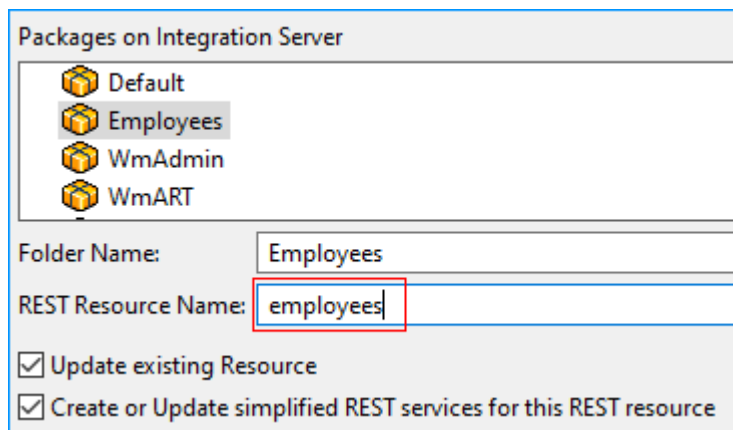
Note: When creating a connection, a package dependency is added such that the selected package depends on webMethods EntireX (the package `WmEntireX`) with the version currently used.

Step 5: Create or Update a REST Resource

➤ To create or update a REST resource

Make sure checkbox **Create or Update REST resource** is checked.

- 1 Select a package.



Packages on Integration Server

- Default
- Employees
- WmAdmin
- WmART

Folder Name: Employees

REST Resource Name: employees

☒ Update existing Resource

☒ Create or Update simplified REST services for this REST resource

- 2 Define a folder name. If the folder does not exist, it will be created.
- 3 Define the REST Resource name.
- 4 Check **Update existing Resource** if you want to add additional services to an existing resource.
- 5 Check **Create or Update simplified REST services for this REST resource** if you want to generate additional services with a simplified REST API. This allows you to use a simple HTTP-GET method without a payload.



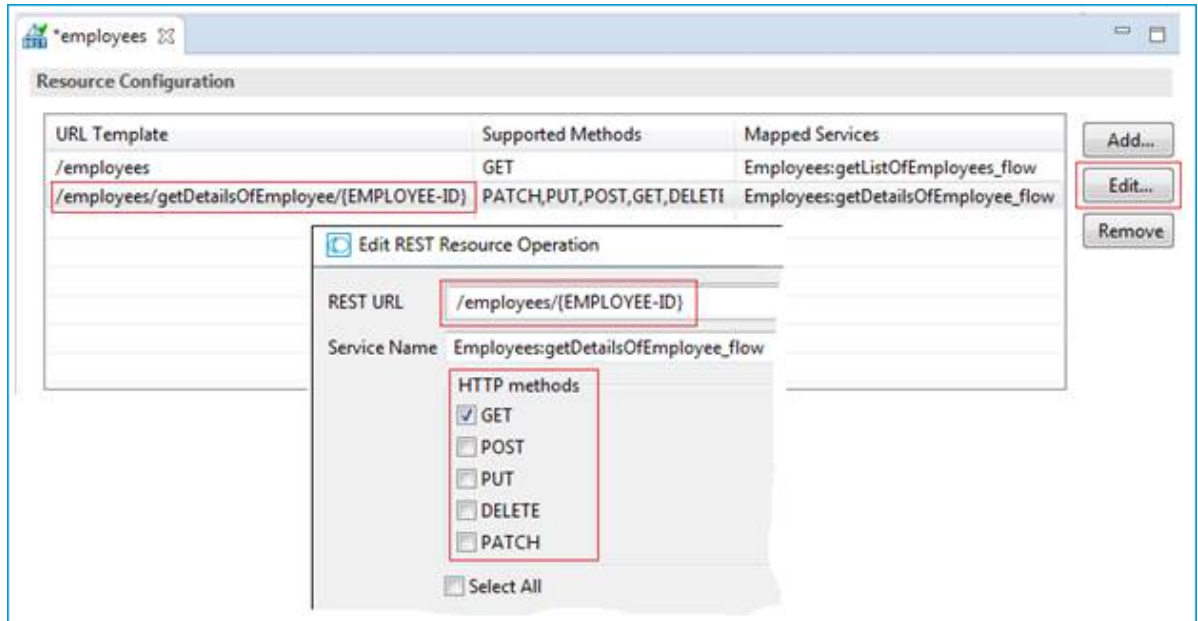
Note: This applies only to APIs where input signature parameters do not contain arrays.

;

Step 6: Edit the REST Resource

➤ To edit the REST resource

- 1 Switch to the **Service Development** perspective of Software AG Designer.



- 2 Refresh the package where you generated the REST Resource in the previous step.
- 3 Open the REST Resource.
- 4 Edit the REST URLs and HTTP Methods to match the functionality of your application.

For more information see:

- *REST Developer's Guide* in the webMethods Integration Server documentation for information on using the generated REST resources
- *REST Developer's Guide* in the webMethods Integration Server documentation

Task 3: Execute the Call from the REST Client to COBOL

Use an appropriate REST client, specify URL, Method and Headers and invoke your REST resource:

URL http://localhost:5555/restv2/employees	URL http://localhost:5555/restv2/employees/111000105
Method GET	Method GET
Headers Accept application/json	Headers Accept application/json
200 OK	200 OK
<pre>{ "errorCode": "00000000", "errorFlag": "false", "response": { "EMPLOYEES-LIST": { "EMPLOYEES": [{ "IDENT": "11100102", "FIRSTNAME": "Edgar", "SURNAME": "Schindler", "DATE-BIRTH": "Dec 4, 1962" }, { "IDENT": "11100105", "FIRSTNAME": "Christian", "SURNAME": "Schirm", "DATE-BIRTH": "Mar 15, 1961" }, { "IDENT": "11100106", "FIRSTNAME": "Rainer", "SURNAME": "Schmitt", "DATE-BIRTH": "Feb 13, 1955" }, { "IDENT": "11100107", "FIRSTNAME": "Helga", "SURNAME": "Schmidt", "DATE-BIRTH": "May 26, 1961" }] } } }</pre>	<pre>{ "errorCode": "00000000", "errorFlag": "false", "response": { "EMPLOYEE-DETAILS": { "FIRSTNAME": "Christian", "SURNAME": "Schirm", "DATE-BIRTH": "Mar 15, 1961", "DETAIL": { "ANNUAL-SALARY": "68000", "VACATION": "21", "LANGUAGE": "GER" } } } }</pre>

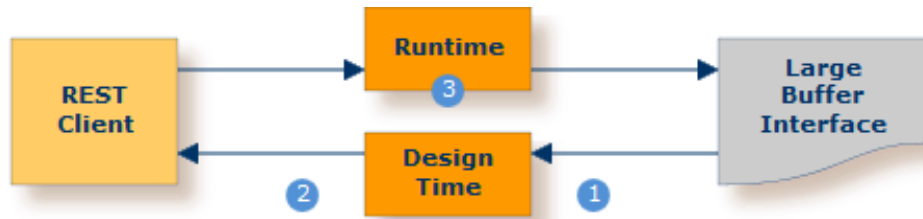
23

Calling COBOL Large Buffer (Minimal Footprint Using CICS Socket Listener) from REST

■ Introduction	405
■ What do I need to Install for this Scenario?	407
■ Task 1: Extract the Interface of a COBOL Server	408
■ Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server	421
■ Task 3: Execute the Call from the REST Client to COBOL	427

This scenario uses the tools IDL Extractor for COBOL and Integration Server Wrapper of the Designer.

Introduction



- ① Extract the interface of the COBOL server program. See *Using the IDL Extractor for COBOL - Overview*.
- ② Generate REST resources, connection and adapter services in Integration Server.
- ③ Execute the call from the REST client to the COBOL server program.

This scenario makes the following important assumptions:

For Task 1:

- You have a working COBOL Large Buffer server. For illustration and examples on such a server, see *CICS with DFHCOMMAREA Large Buffer Interface* under *Introduction to the IDL Extractor for COBOL*.
- You have access to the related COBOL sources and copybooks as files on your local machine. The minimum requirement is the `DATA DIVISION` of the interface.



For Tasks 2 and 3:

- You have a REST client.
- To call the COBOL server program at runtime using the *EntireX CICS Socket Listener* connection method, you need to install the CICS Socket Listener within your CICS region: see *Preparing for CICS Socket Listener*. For configuration, see *Connection Parameters for CICS Socket Listener Connections*.



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.
- You have the CICS Socket Listener installed. See *Installing the CICS Socket Listener* in the z/VSE Installation documentation.

Task 1: Extract the Interface of a COBOL Server

- [Introduction](#)
- [Sample COBOL Server used in the Examples](#)
- [Extracting a COBOL Server - Fast-track Method](#)
- [Extracting a COBOL Server - Modern Method with User-defined Mapping](#)
- [Comparison Chart: Fast-track versus User-defined Mapping](#)
- [Testing the Extraction Results](#)

Introduction

Follow the instructions for extracting COBOL, see *Using the IDL Extractor for COBOL - Overview* and choose *Scenario I: Create New IDL and Server Mapping Files* if this is your first extraction. Learn how to invoke the IDL Extractor for COBOL, create a COBOL Extractor Environment and about the possibilities to select the COBOL source.

The extraction process creates the following EntireX metafiles:

- IDL file. A Software AG IDL file contains definitions of the interface between client and server. See *Software AG IDL File* in the IDL Editor documentation in the IDL Editor documentation.
- Server mapping file (optional). The mapping file is a Designer file with extension .cvm that contains COBOL-specific mapping information. See *Server Mapping Files for COBOL* in the Designer documentation.

Sample COBOL Server used in the Examples

The following COBOL server is used to illustrate the features of the IDL Extractor for COBOL. Imagine an existing COBOL server called `EMPLOYEE`. It implements the access logic for a `LIST` and `DETAILS` function to a database view `EMPLOYEE`:


```

IDENTIFICATION DIVISION.
    PROGRAM-ID.      EMPLOYEE.
*****
*   Program Name    = EMPLOYEE
*   String Literal  = QUOTE
*****
01 DFHCOMMAREA.
    02 OPERATION                                PIC X(1).
        88 OPERATION-LIST                      VALUE 'L'.
        88 OPERATION-DETAILS                   VALUE 'D'.
    02 EMPLOYEE-ID                             PIC X(10).
    02 EMPLOYEE-DETAILS.
        03 FIRSTNAME                          PIC X(20).
        03 SURNAME                            PIC X(20).
        03 DATE-BIRTH                         PIC X(12).
        03 DETAILS                            PIC X(100).
        03 FILLER REDEFINES DETAILS.
            04 ANNUAL-SALARY                    PACKED-DECIMAL PIC S9(9).
            04 VACATION                          PIC S9(2).
            04 LANGUAGE                         PIC X(3).
            04 FILLER                            PIC X(90).
    02 EMPLOYEES-COUNT                         PIC 9(8) BINARY.
    02 EMPLOYEES OCCURS 1 TO 99 DEPENDING ON EMPLOYEES-COUNT.
        03 IDENT                              PIC X(10).
        03 FIRSTNAME                          PIC X(20).
        03 SURNAME                            PIC X(20).
        03 DATE-BIRTH                         PIC X(12).

```

Two approaches for extracting the COBOL server are described below:

- **Fast-track**

Learn how EntireX helps you to connect the COBOL Server with quick results.

- **User-defined Mapping**

The following features are among those available with a user-defined mapping:

- **Shape the interface to your COBOL server**

You can minimize the interface by suppressing COBOL server fields or by providing constant values. Renaming interfaces allows you to specify a readable long name. These two features together - constants and renaming of interfaces - are most powerful when multiple interfaces are implemented in a single COBOL server. Each function of the COBOL server triggered by an operation code or function code can be modelled as a separate REST resource method/operation.

■ Select alternative mappings

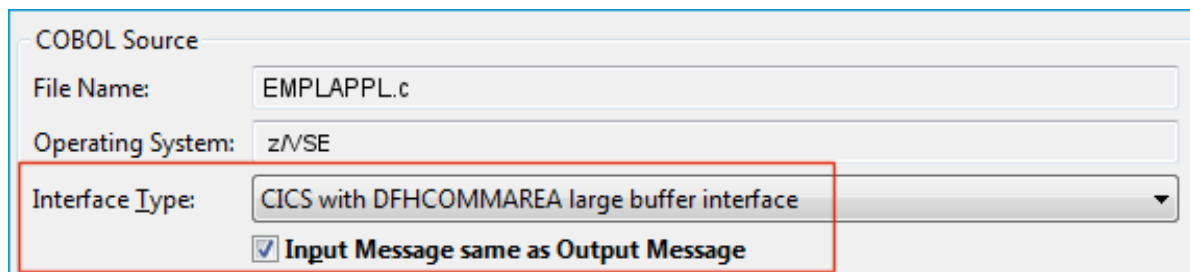
You can select alternative mappings for COBOL data items (REDEFINES, tables etc.) to enable scenarios with COBOL servers where data exchange is not fully described by the LINKAGE SECTION COBOL data items.

These approaches are described under [Fast-track Method](#) and [Modern Method](#) below.

Extracting a COBOL Server - Fast-track Method

> To extract the COBOL Server

- 1 Switch to the **EntireX** perspective.
- 2 Call the IDL Extractor for COBOL.
- 3 Set the correct interface type as described under *Step 4: Define the Extraction Settings and Start Extraction* under *Scenario I: Create New IDL and Server Mapping Files* in the IDL Extractor for COBOL documentation.



COBOL Source

File Name: EMPLAPPL.c

Operating System: z/VSE

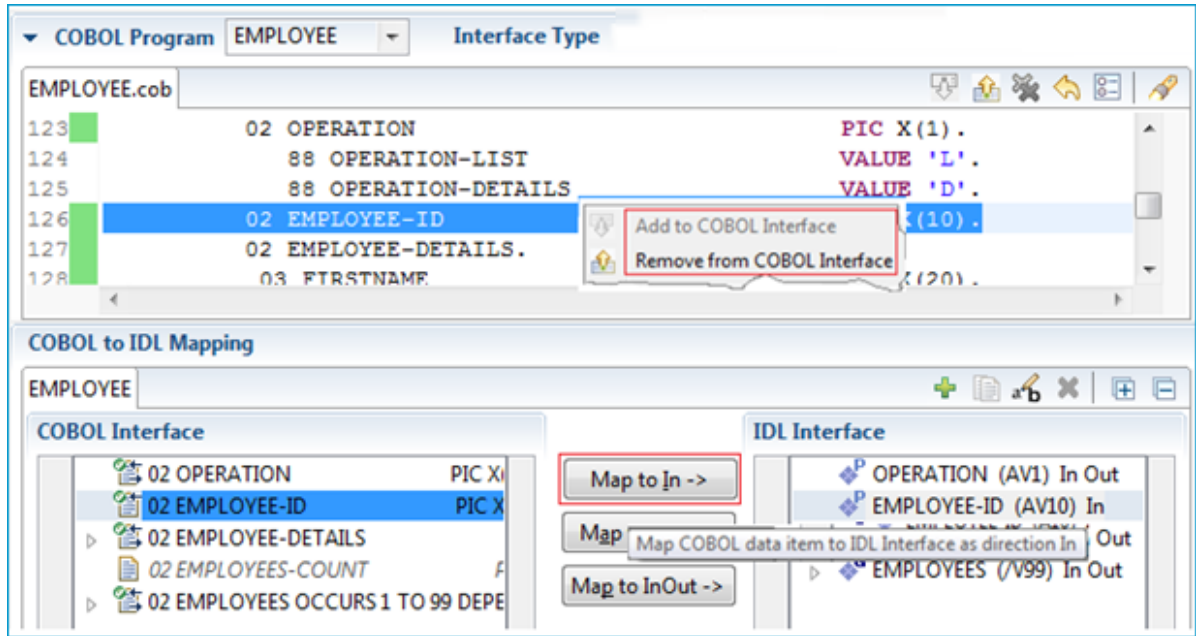
Interface Type: CICS with DFHCOMMAREA large buffer interface

☒ Input Message same as Output Message

- 4 Set the correct value for checkbox **Input Message same as Output Message**.
 - *Check* the checkbox if the COBOL data structure of the CICS input message is the same as the structure of the CICS output message, that is, there is one COBOL layout for both directions.
 - *Clear* the checkbox if the COBOL data structure of the CICS input message is different to the structure of the CICS output message, that is, the output overlays the input, which means there is a second COBOL layout for the reply.

For our example, check the check box, because the COBOL layout for input is the same as the COBOL layout for output.

- 5 Press **Next** to enter the COBOL Mapping Editor.



The purpose of the COBOL Mapping Editor is to map COBOL data items to IDL (Interface Definition Language) parameters. It consists of three main areas:

- The upper window pane is the COBOL program source.
- The window pane on the left-hand side is the COBOL interface, consisting of COBOL data items representing the parameters in your COBOL server program.
- The window pane on the right-hand side is the IDL interface consisting of IDL parameters.

What you select depends on check-box **Input Message same as Output Message**:

- If there is a single COBOL data structure of the CICS message for input and output - check box **Input Message same as Output Message** is checked - you select a single COBOL data structure in the COBOL Mapping Editor. This structure is used for COBOL input as well as for COBOL output. For more information in this case see *Mapping Editor User Interface* in the IDL Extractor for COBOL documentation.
- If there is a second COBOL data structure for the CICS output message - check box **Input Message same as Output Message** is cleared - you will select two COBOL data structures in the COBOL Mapping Editor. One is used for COBOL input and the other is for COBOL output. For more information in this case see *Mapping Editor User Interface* in the IDL Extractor for COBOL documentation.

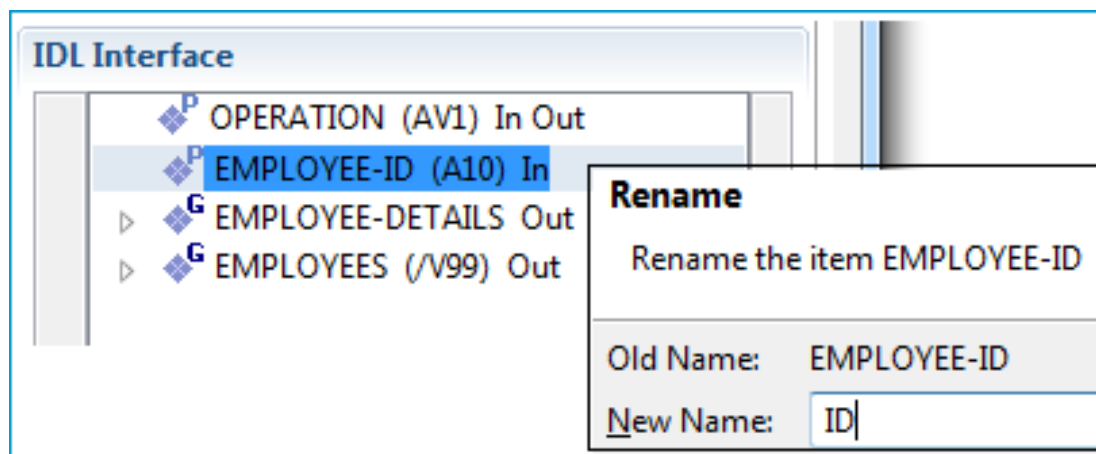
COBOL extraction is a two-step process:

- First, select the COBOL parameters using the **Add to** and **Remove from COBOL Interface** from the context menu on the COBOL data items in the upper window pane (COBOL source); in many situations the COBOL interface is detected by the Mapping Editor directly and you can omit this step.

- Then map the COBOL parameters to IDL directions using the **Map to In/Out/InOut** buttons. COBOL parameters do not have directions, they are always INOUT. This step can be left out for a quick test in non-production scenarios; for real production it is recommended to set IDL directions to reduce data transfer. You also have the option to assign a meaningful name to an IDL parameter if the original COBOL data item is not self-explanatory.

For our example:

- Set **Map to In** for COBOL data item `EMPLOYEE-ID` as IDL direction (see screenshot in previous step). For `EMPLOYEES-DETAILS` and `EMPLOYEES` perform the similar **Map to Out**.
- Use the function **Rename** from the context menu on the IDL parameters in the **IDL Interface** pane to rename COBOL parameter `EMPLOYEES-ID` to IDL parameter `ID`. Do the same for `EMPLOYEE-DETAILS`: Rename to `EMPLOYEE`.



- 6 Press **Finish**. The default extraction settings ensure you get useful results. The outcome is a Software AG IDL file (interface definition language), a Designer file with extension `.idl`:

```

library 'EMPLAPPL' is
program 'EMPLOYEE' is
  define data parameter
    1 OPERATION (AV1) In Out
    1 ID (AV10) In /* Original Name:EMPLOYEE-ID
    1 EMPLOYEE Out /* Original Name:EMPLOYEE-DETAILS
      2 FIRSTNAME (AV20)
      2 SURNAME (AV20)
      2 DATE-BIRTH (AV12)
      2 DETAILS (AV100)
    1 EMPLOYEES (/V99) Out
      2 IDENT (AV10)
      2 FIRSTNAME (AV20)
      2 SURNAME (AV20)
      2 DATE-BIRTH (AV12)
  end-define

```

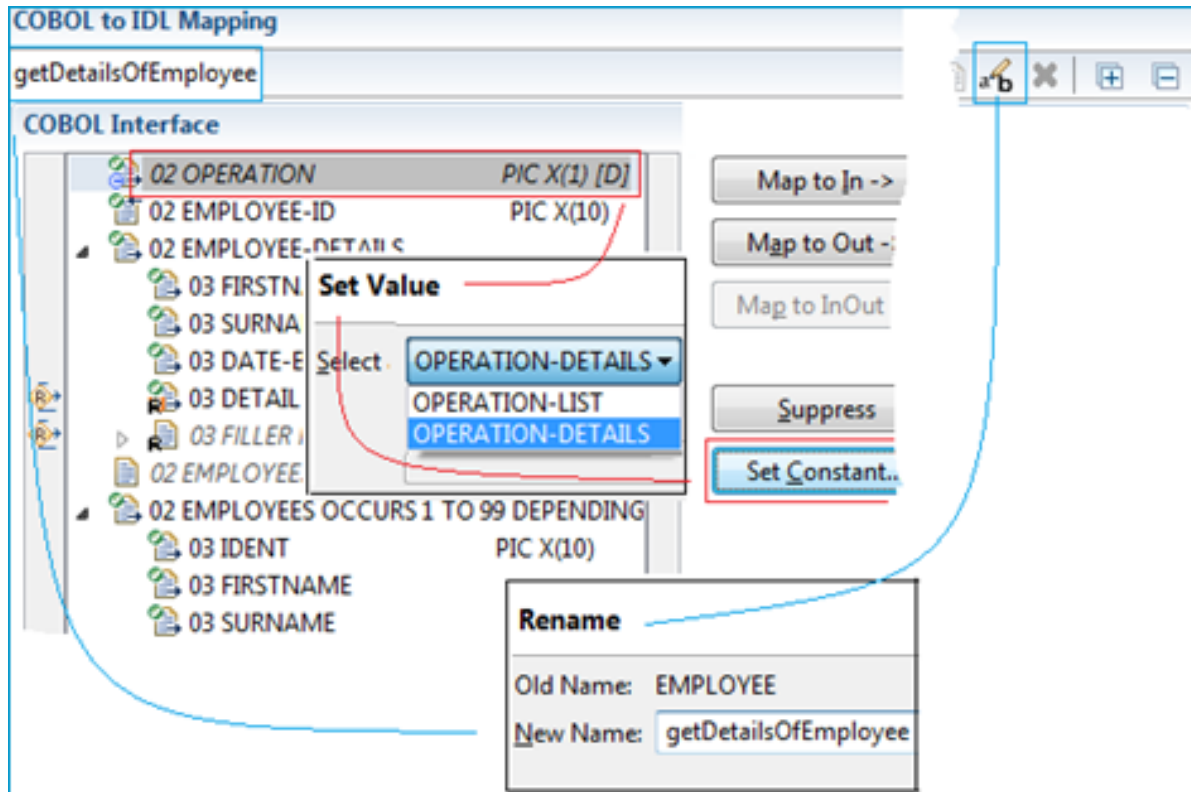
If required, a server mapping file with the same name as the IDL file but the extension .cvm may be generated too, for technical reasons. This file must always be stored together with the concomitant IDL file. For more information, see *Server Mapping Files for COBOL* in the Designer documentation. By default, the interface name `EMPLOYEE` is taken from the COBOL server program name.


Extracting a COBOL Server - Modern Method with User-defined Mapping

EntireX allows you to extract all implemented interfaces of the COBOL server separately. Instead of a large `EMPLOYEE` interface, separate interfaces `getListOfEmployees` and `getDetailsOfEmployee` are extracted. Each interface contains required parameters; obsolete parameters for an interface are suppressed, improving its usability.

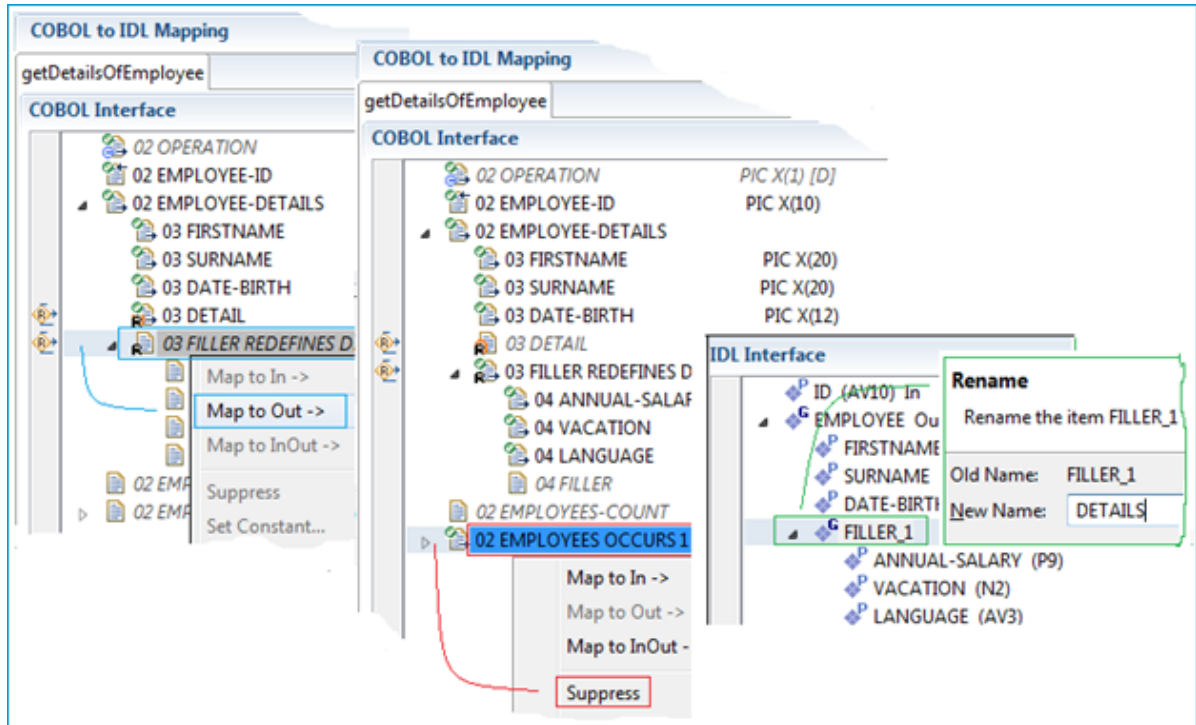
» To extract a COBOL server with a user-defined mapping

- 1 Shape `EMPLOYEE` to `getDetailsOfEmployee`:



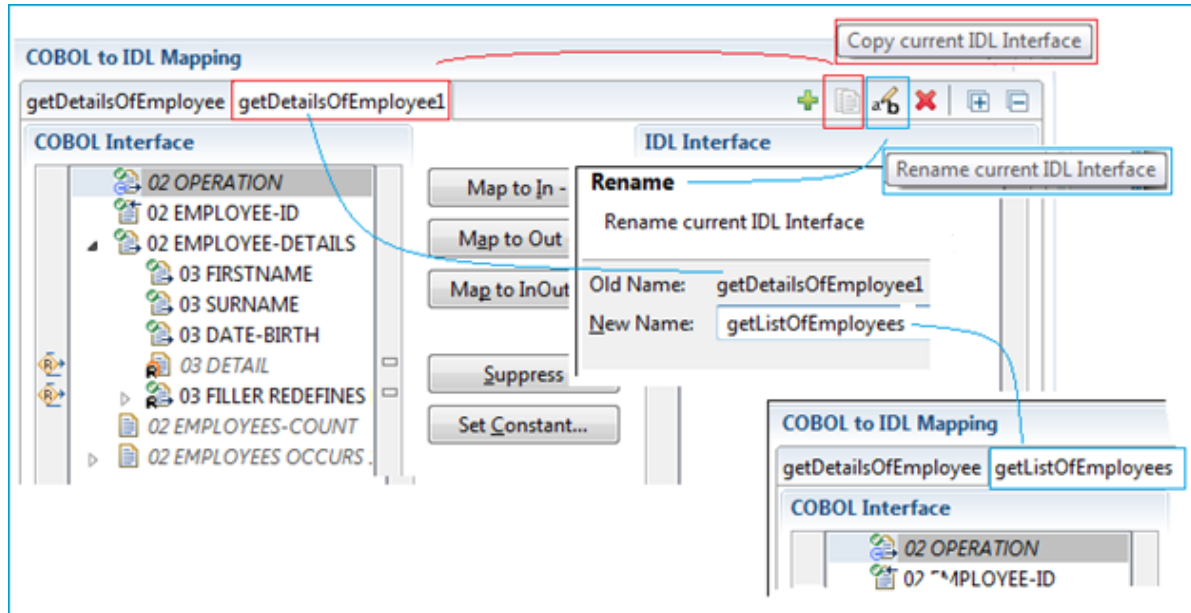
- Mark the `OPERATION` field in the **COBOL Interface** pane and press the **Set Constant** button. In the pop-up window, select `OPERATION-DETAILS` (red markers). The `OPERATION-DETAILS` value originates from COBOL level-88 data item (blue markers). If there is no COBOL level-88 enumeration type in the COBOL program defined, you can enter the function code manually in the pop-up window.
- By default the IDL interface gets the same name as your COBOL server: `EMPLOYEE`. Use the **Rename** button  from the toolbar of the COBOL Mapping Editor and change the IDL interface name to `getDetailsOfEmployee` (blue markers). This name is used as the IS service name later.

2 Continue shaping, making `getDetailsOfEmployee` easier to use:




- Use the context menu of the redefinition of the `DETAIL` field in the **COBOL Interface** pane, `FILLER REDEFINES DETAIL` and choose **Map to Out** (blue markers). A redefinition is a secondary layout (field types) of the same memory area. The redefinition contains more specific information (`ANNUAL-SALARY`, `VACATION`, etc.) than character buffer `DETAIL`. This is the reason why it is more useful in your interface.
- Use the context menu of `FILLER_1` in the IDL interface pane and rename `FILLER_1` to `DETAILS` (green markers) for a readable IS service field name later.
- The `DETAILS` function in the COBOL server does not make use of the `EMPLOYEES` field. Therefore you can leave it out of the interface: use the context menu of the `EMPLOYEES` field in the **COBOL Interface** pane and choose **Suppress** (red markers).


3 Shape `EMPLOYEE` to `getListOfEmployees` Function:



The COBOL Mapping Editor allows you to define multiple IDL interfaces to the same COBOL server:

- You create the additional IDL interface using the toolbar button **Copy current IDL interface**

 (red markers). A new tab `getDetailsOfEmployee1` is created. The previously extracted `getDetailsOfEmployee` interface still exists in the first tab. Once you reactivate the first tab, you will see the interface of `getDetailsOfEmployee` again.

- Use the **Rename** button  from the toolbar of the COBOL Mapping Editor and change the IDL interface name to `getListOfEmployees` (blue markers). This name is used as the IS service name later.

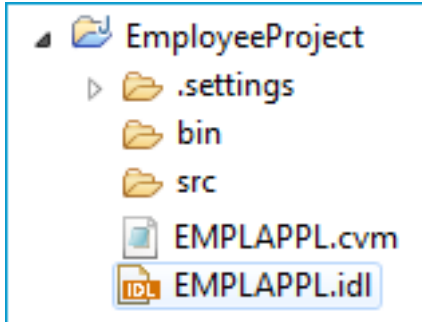
- For the `LIST` function you need the `OPERATION-LIST` value in the `OPERATION` field: Mark the `OPERATION` field in the **COBOL Interface** pane and press the **Set Constant** button. In the pop-up window, select `OPERATION-LIST` (similar to Step 1. above *Shape `EMPLOYEE` to `getDetailsOfEmployee`*; red markers).

- Use the context menu of the `EMPLOYEES` field in the **COBOL Interface** pane, `EMPLOYEES OCCURS 1 TO 99` and choose **Map to Out** (Similar to Step 2. above *Continue shaping, making `getDetailsOfEmployee` easier to use*; blue markers).

- Because the `EMPLOYEE` and `EMPLOYEE-DETAIL` fields are not used in the COBOL server `LIST` function, you leave them out in the IDL interface: use the context menu of `EMPLOYEE-DETAIL` field in the **COBOL Interface** pane and choose **Suppress** (similar to Step 2. above *Continue shaping, making `getDetailsOfEmployee` easier to use*; red markers). Do the same for the `EMPLOYEE-ID` field.

- 4 Press **Finish** to retrieve the extraction result in the form of a Software AG IDL file. At the same time, a server mapping file for COBOL (Designer file with extension `.cvm`) is created.

See *Server Mapping Files for COBOL* in the Designer documentation. The *Software AG IDL File* in the IDL Editor documentation describes the interfaces from the client point of view, while the server mapping file contains the mapping to the real COBOL server. Both of these files must be kept together and in sync, otherwise a call to the COBOL server may fail.



To summarize: We created two IDL interfaces. In the IDL file, these resulted in two IDL programs: `getListOfEmployees` and `getDetailsOfEmployee`. Both IDL programs were given readable names (Steps 1 and 3). Meaningful fields were kept, while superfluous fields were suppressed (Steps 1 and 3). The program `getDetailsOfEmployee` contains the redefined fields of parameter `DETAILS` (see below) mapped during extraction (Step 1).



Note: At runtime, the RPC client generated with the extracted interface will send data for the redesigned interfaces, while your COBOL server still expects `EMPLOYEE` data. The EntireX runtime transforms the incoming data stream from the RPC client, using the server mapping file.

```

program 'getDetailsOfEmployee' is
  define data parameter
    1 ID (AV10) In /* Original Name:EMPLOYEE-ID
    1 EMPLOYEE Out /* Original Name:EMPLOYEE-DETAILS
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
    2 DETAILS /* Original Name:FILLER_1
    3 ANNUAL-SALARY (P9)
    3 VACATION (N2)
    3 LANGUAGE (AV3)
  end-define
  program 'getListOfEmployees' is
    1 EMPLOYEES (/V99) Out
    2 IDENT (AV10)
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
  end-define

```

Comparison Chart: Fast-track versus User-defined Mapping

The following table highlights the differences when shaping an interface with a *user-defined mapping* compared with a *fast-path extraction*.

	Interface Shaping with User-defined Mapping	Fast-path Extraction
General	User-defined interface with dedicated mapping. For our example, the interfaces are small and tidy and more self-explanatory with long, readable interface names. They are easier to use with the hidden <code>OPERATION</code> field and the suppressed COBOL fields, which are not needed.	Automatic, quick result extraction with COBOL-like interfaces on IS. For our example, the IS interface <code>EMPLOYEE</code> matches exactly 1:1, field by field.
IS service(s)	Multiple small and handy IS services; each <code>OPERATION</code> code is mapped to a separate IS service.	One big IS service.
IS service name	Readable long name.	Short Subprogram name; up to 8 characters.
IS fields	Usage of Suppress and Set Constant reduces the message length. This keeps focus on relevant data items and keeps the client's interface clean. It may also improve performance.	The IS fields and COBOL server parameters match 1:1. As COBOL layout descriptions are sometimes used for many different purposes, irrelevant data items appear and clutter up the IS interface.
<code>OPERATION</code> parameter	Suppressed: the <code>OPERATION</code> parameter does not exist in the IS service as an IS field.	The <code>OPERATION</code> parameter exists in the IS service as an IS field and needs to be filled-in by the client endpoint.
<code>OPERATION</code> code	The <code>OPERATION</code> code is provided internally by EntireX runtime in the <code>OPERATION</code> field.	The <code>OPERATION</code> code needs to be specified in the IS service in the <code>OPERATION</code> field by the client endpoint.
<code>REDEFINE</code> parameters	Either the parameter that is redefined or one of its redefinitions is available as an IS field.	Only the parameter that is redefined is available as an IS field. Redefinitions of the parameter are not available as IS fields.

A user-defined mapping enables you also to define alternative mappings for COBOL data items (`REDEFINES`, tables etc.). These enable scenarios with COBOL servers where data exchange is not fully described by the `LINKAGE SECTION` COBOL data items. For more information refer to the IDL Extractor for COBOL documentation:

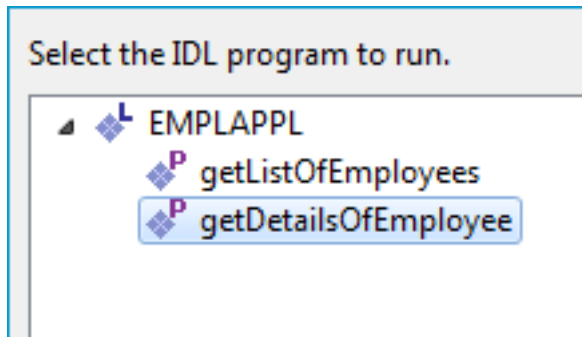
- *Mapping Editor IDL Interface Mapping Functions* if check-box **Input Message same as Output Message** was checked.
- *Mapping Editor IDL Interface Mapping Functions* if check-box **Input Message same as Output Message** was cleared.
- *User-defined Mapping*

Testing the Extraction Results

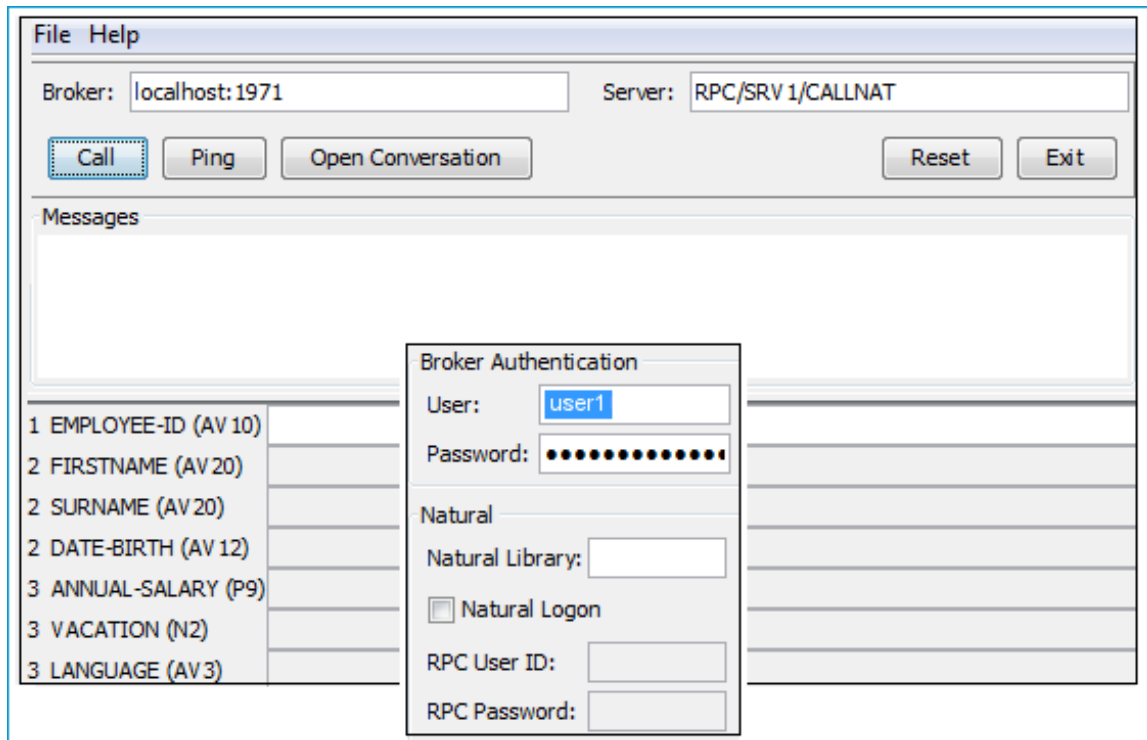
The following pictures use the extraction results described under *Extracting a COBOL Server - Modern Method with User-defined Mapping*.

> To test the extraction results (optional)

- 1 You can test the results of the extraction operation and the REST server back end, using the EntireX IDL Tester. From the context menu of the IDL file in the Designer, choose **Software AG IDL Tester**.



- 2 Select `getDetailsOfEmployee`.



Note that the broker and server parameters contain the explicit route to call the server program, and you can optionally ping the connection from this client. With the **File > Options** dialog, you can set:

- **User** and **Password** for *broker* authentication
- **RPC User ID** and **RPC Password** for *server* authentication

See *EntireX IDL Tester* in the Designer documentation.

- 3 Check the Integration Server log, the EntireX Adapter log or the RPC logs. Applies to all connection methods.

Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server

This section describes your first steps to create a new Integration Server connection. This is described in more detail under *Using the Integration Server Wrapper*, for example working with existing Integration Server connections. This section covers the following topics:

- [Step 1: Start the Integration Server Wrapper Wizard](#)
- [Step 2: Create a New Integration Server Connection](#)
- [Step 3: Select the Connection Type](#)
- [Step 4: Define Adapter Services for a CICS Socket Listener Connection](#)
- [Step 5: Create or Update a REST Resource](#)
- [Step 6: Edit the REST Resource](#)

Step 1: Start the Integration Server Wrapper Wizard

➤ To start the Integration Server Wrapper wizard

- 1 In the context menu of a Software AG IDL file, choose **Integration Server > Generate web-Methods IS Connection**.

This starts the wizard with a list of existing Integration Server Wrapper connections.



Note: If the selected IDL file is not valid because of a syntax error, an error dialog comes up and the wizard does not start.

- 2 Continue with *Step 2: Create a New Integration Server Connection*.

Step 2: Create a New Integration Server Connection

Define New Integration Server Connection

Add host:port, user and password for a new Integration Server

Server:

User:

Password:

☐
Use secure connection

➤ To create a new Integration Server connection

- 1 Define the new Integration Server connection on the wizard page.



Notes:

1. The only required field is **Server**. Enter the hostname of the Integration Server including an optional port number. If no port number is specified, port number defaults to "5555". The **Integration Server Authentication** can be passed with the **User** and **Password** fields.
 2. Optional settings are for secure connections. The **Truststore for HTTPS** contains all signed certificates and must be a valid truststore.
 3. The check box **Verify host name** checks that the hostname is entered in the stored certificate.
 4. When the Integration Server has **Client Authentication** enabled, you can specify your **Keystore** file and keystore **Password**.
 5. For managing Integration Server connections, see [Preferences](#).
- 2 Choose **Next** and continue with *Step 3: Select the Connection Type*.

Step 3: Select the Connection Type

☒ Create a new EntireX Adapter connection of type CICS Socket Listener Connection

☐ Update Adapter Services or Listeners to an existing Integration Server connection:

List of Connections

Connection Name	Package Name	Connection Type	Enabled
Employees:EmployeesConnection	Employees	EntireX RPC Connection	Yes

Total: 1

☒ Map Software AG IDL data types to String
☒ Create or Update REST resource

> To create a new connection

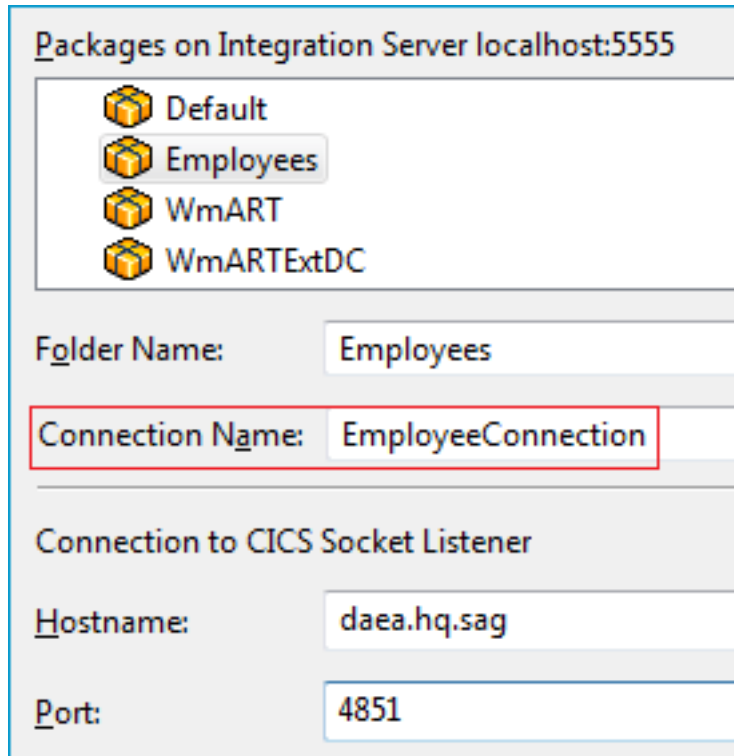
- 1 Select a connection type from the drop down list. Connection types are described under *EntireX Adapter Connections* in the EntireX Adapter documentation and *Introduction to the Integration Server Wrapper*.



Note: The list of connection types is filtered: connection types that require a license are only shown if a corresponding license file is available. Reliable RPC connections are only shown if all IDL programs contain only IN parameters. Also, if a server mapping file is available, only those connection types that support the interface type specified in the server mapping file are shown.

- 2 Check the box **Create or Update REST resource**.
- 3 Click **Next** and continue with *Step 4: Define Adapter Services for a CICS Socket Listener Connection*.

Step 4: Define Adapter Services for a CICS Socket Listener Connection



Packages on Integration Server localhost:5555

- Default
- Employees
- WmART
- WmARTEExtDC

Folder Name:

Connection Name:

Connection to CICS Socket Listener

Hostname:

Port:

➤ To create a connection and related adapter services

- 1 Select a package for the created objects.
- 2 Define a folder name. If the folder does not exist, it will be created.
- 3 Define a connection name.
- 4 Define the parameters of the connection type. For details, see *Connection Parameters for CICS Socket Listener Connections* in the EntireX Adapter documentation.

As a result, the folder will contain the connection and the adapter services (one for each IDL program). The name of a service is the same as the respective IDL program.

The default settings for the adapter services are:

- the **Default** package; if not available, the first package
- the IDL library name for the **Folder Name**
- the IDL library name with the suffix "Connection" for the **Connection Name**



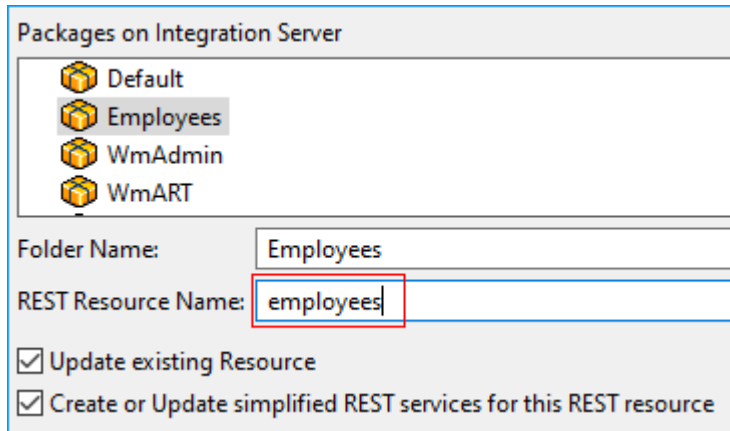
Note: When creating a connection, a package dependency is added such that the selected package depends on webMethods EntireX (the package `WmEntireX`) with the version currently used.

Step 5: Create or Update a REST Resource

➤ To create or update a REST resource

Make sure checkbox **Create or Update REST resource** is checked.

- 1 Select a package.



Packages on Integration Server

- Default
- Employees
- WmAdmin
- WmART

Folder Name: Employees

REST Resource Name: employees

☒ Update existing Resource

☒ Create or Update simplified REST services for this REST resource

- 2 Define a folder name. If the folder does not exist, it will be created.
- 3 Define the REST Resource name.
- 4 Check **Update existing Resource** if you want to add additional services to an existing resource.
- 5 Check **Create or Update simplified REST services for this REST resource** if you want to generate additional services with a simplified REST API. This allows you to use a simple HTTP-GET method without a payload.



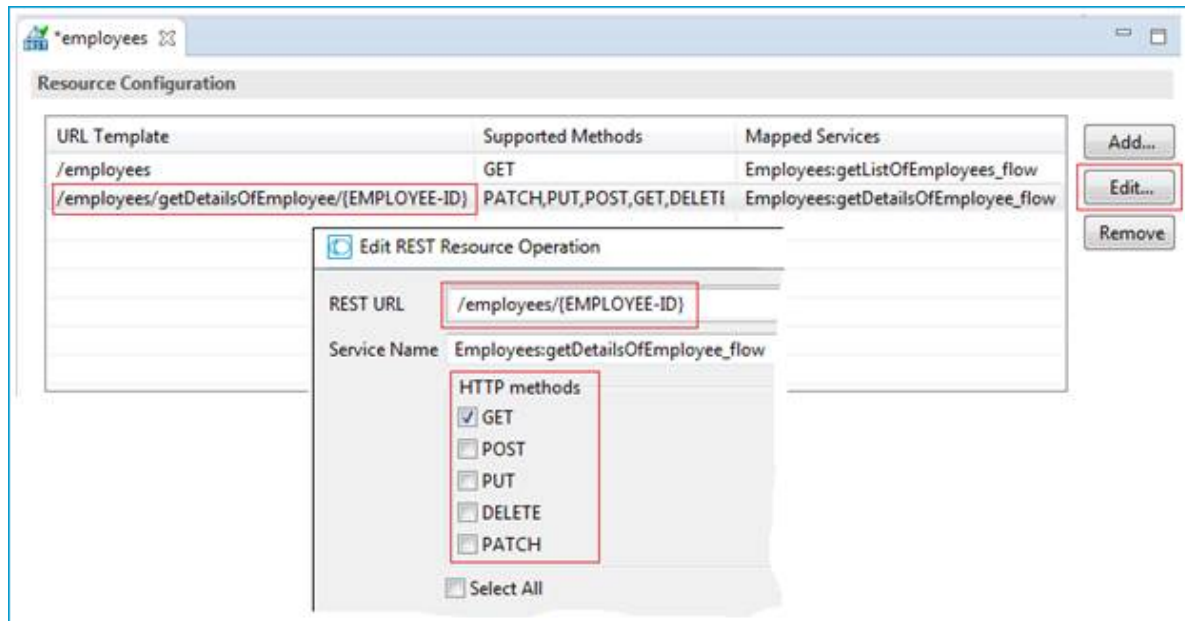
Note: This applies only to APIs where input signature parameters do not contain arrays.

;

Step 6: Edit the REST Resource

➤ To edit the REST resource

- 1 Switch to the **Service Development** perspective of Software AG Designer.



- 2 Refresh the package where you generated the REST Resource in the previous step.
- 3 Open the REST Resource.
- 4 Edit the REST URLs and HTTP Methods to match the functionality of your application.

For more information see:

- *REST Developer's Guide* in the webMethods Integration Server documentation for information on using the generated REST resources
- *REST Developer's Guide* in the webMethods Integration Server documentation

Task 3: Execute the Call from the REST Client to COBOL

Use an appropriate REST client, specify URL, Method and Headers and invoke your REST resource:

URL http://localhost:5555/restv2/employees

Method GET

Headers Accept application/json

200 OK

```
{
  "errorCode": "00000000",
  "errorFlag": "false",
  "response": {
    "EMPLOYEES-LIST": {
      "EMPLOYEES": [
        {
          "IDENT": "11100102",
          "FIRSTNAME": "Edgar",
          "SURNAME": "Schindler",
          "DATE-BIRTH": "Dec 4, 1962"
        },
        {
          "IDENT": "11100105",
          "FIRSTNAME": "Christian",
          "SURNAME": "Schirm",
          "DATE-BIRTH": "Mar 15, 1961"
        },
        {
          "IDENT": "11100106",
          "FIRSTNAME": "Rainer",
          "SURNAME": "Schmitt",
          "DATE-BIRTH": "Feb 13, 1955"
        },
        {
          "IDENT": "11100107",
          "FIRSTNAME": "Helga",
          "SURNAME": "Schmidt",
          "DATE-BIRTH": "May 26, 1961"
        }
      ]
    }
  }
}
```

URL http://localhost:5555/restv2/employees/111000105

Method GET

Headers Accept application/json

200 OK

```
{
  "errorCode": "00000000",
  "errorFlag": "false",
  "response": {
    "EMPLOYEE-DETAILS": {
      "FIRSTNAME": "Christian",
      "SURNAME": "Schirm",
      "DATE-BIRTH": "Mar 15, 1961",
      "DETAIL": {
        "ANNUAL-SALARY": "68000",
        "VACATION": "21",
        "LANGUAGE": "GER"
      }
    }
  }
}
```

24

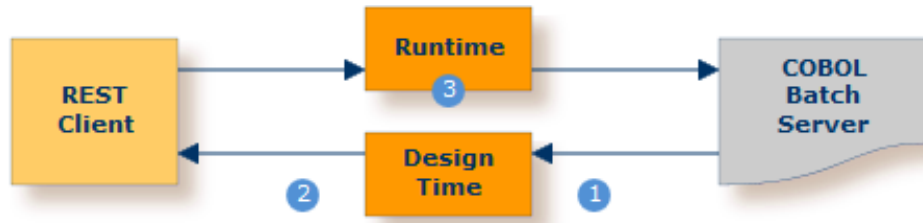
Calling COBOL on IBM i from REST

■ Introduction	431
■ What do I need to Install for this Scenario?	433
■ Task 1: Extract the Interface of a COBOL Server	434
■ Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server	446
■ Task 3: Execute the Call from the REST Client to COBOL	452

This scenario uses the tools IDL Extractor for COBOL and Integration Server Wrapper of the Designer.

Introduction

Under IBM i, a COBOL server with a standard call interface can be called.



- ① Extract the interface of the COBOL server program. See *Using the IDL Extractor for COBOL - Overview*.
- ② Generate REST resources, connection and adapter services in Integration Server.
- ③ Execute the call from the REST client to the COBOL server program.

This scenario makes the following important assumptions:

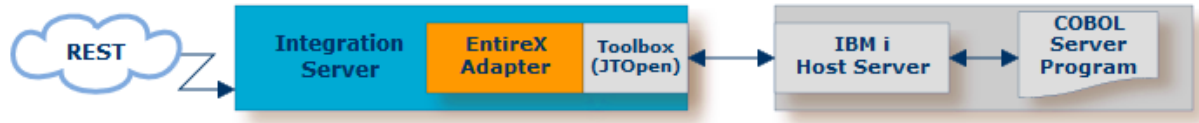
For Task 1:

- You have a working COBOL batch server. For illustration and examples on such a server, see *Batch with Standard Linkage Calling Convention* under *Introduction to the IDL Extractor for COBOL*.
- You have access to the related COBOL sources and copybooks as files on your local machine. The minimum requirement is the `DATA DIVISION` of the interface.



For Tasks 2 and 3:

- You have a REST client.
- You have installed the IBM ToolBox for Java (JTOpen). See *Post-installation Steps for AS/400*.
- With interface type AS/400 connection you can remotely call server programs written in COBOL. See *Connection Parameters for AS/400 Connections* in the EntireX Adapter documentation.



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.

Task 1: Extract the Interface of a COBOL Server

- [Introduction](#)
- [Sample COBOL Server used in the Examples](#)
- [Extracting a COBOL Server - Fast-track Method](#)
- [Extracting a COBOL Server - Modern Method with User-defined Mapping](#)
- [Comparison Chart: Fast-track versus User-defined Mapping](#)
- [Testing the Extraction Results](#)

Introduction

Follow the instructions for extracting COBOL, see *Using the IDL Extractor for COBOL - Overview* and choose *Scenario I: Create New IDL and Server Mapping Files* if this is your first extraction. Learn how to invoke the IDL Extractor for COBOL, create a COBOL Extractor Environment and about the possibilities to select the COBOL source.

The extraction process creates the following EntireX metafiles:

- IDL file. A Software AG IDL file contains definitions of the interface between client and server. See *Software AG IDL File* in the IDL Editor documentation in the IDL Editor documentation.
- Server mapping file (optional). The mapping file is a Designer file with extension .cvm that contains COBOL-specific mapping information. See *Server Mapping Files for COBOL* in the Designer documentation.

Sample COBOL Server used in the Examples

The following COBOL server is used to illustrate the features of the IDL Extractor for COBOL. Imagine an existing COBOL server called `EMPLOYEE`. It implements the access logic for a `LIST` and `DETAILS` function to a database view `EMPLOYEE`:

```

IDENTIFICATION DIVISION.
    PROGRAM-ID.      EMPLOYEE.
*****
*   Program Name    = EMPLOYEE
*   String Literal  = QUOTE
*****
01 DFHCOMMAREA.
    02 OPERATION                                PIC X(1).
        88 OPERATION-LIST                      VALUE 'L'.
        88 OPERATION-DETAILS                   VALUE 'D'.
    02 EMPLOYEE-ID                             PIC X(10).
    02 EMPLOYEE-DETAILS.
        03 FIRSTNAME                          PIC X(20).
        03 SURNAME                            PIC X(20).
        03 DATE-BIRTH                         PIC X(12).
        03 DETAILS                            PIC X(100).
        03 FILLER REDEFINES DETAILS.
            04 ANNUAL-SALARY                    PACKED-DECIMAL PIC S9(9).
            04 VACATION                          PIC S9(2).
            04 LANGUAGE                         PIC X(3).
            04 FILLER                           PIC X(90).
    02 EMPLOYEES-COUNT                         PIC 9(8) BINARY.
    02 EMPLOYEES OCCURS 1 TO 99 DEPENDING ON EMPLOYEES-COUNT.
        03 IDENT                              PIC X(10).
        03 FIRSTNAME                          PIC X(20).
        03 SURNAME                            PIC X(20).
        03 DATE-BIRTH                         PIC X(12).

```

Two approaches for extracting the COBOL server are described below:

- **Fast-track**

Learn how EntireX helps you to connect the COBOL Server with quick results.

- **User-defined Mapping**

The following features are among those available with a user-defined mapping:

- **Shape the interface to your COBOL server**

You can minimize the interface by suppressing COBOL server fields or by providing constant values. Renaming interfaces allows you to specify a readable long name. These two features together - constants and renaming of interfaces - are most powerful when multiple interfaces are implemented in a single COBOL server. Each function of the COBOL server triggered by an operation code or function code can be modelled as a separate REST resource method/operation.

■ **Select alternative mappings**

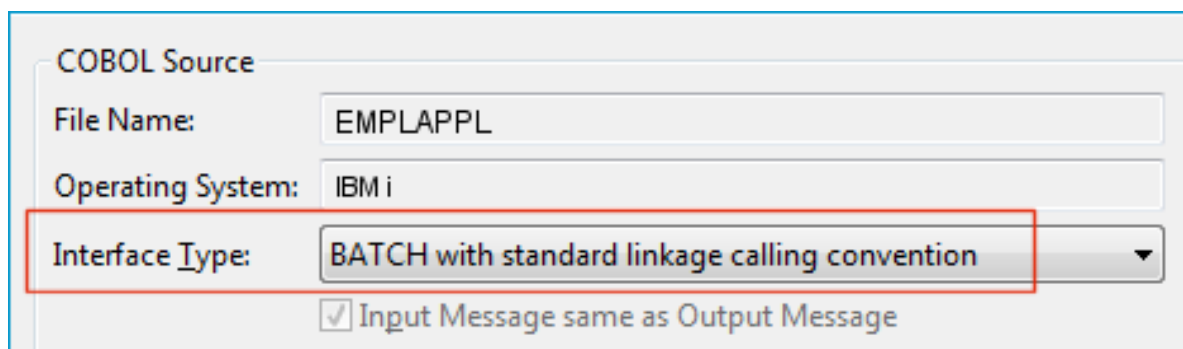
You can select alternative mappings for COBOL data items (REDEFINES, tables etc.) to enable scenarios with COBOL servers where data exchange is not fully described by the LINKAGE SECTION COBOL data items.

These approaches are described under [Fast-track Method](#) and [Modern Method](#) below.

Extracting a COBOL Server - Fast-track Method

> To extract the COBOL Server

- 1 Switch to the **EntireX** perspective.
- 2 Call the IDL Extractor for COBOL.
- 3 Set the correct interface type as described under *Step 4: Define the Extraction Settings and Start Extraction* under *Scenario I: Create New IDL and Server Mapping Files* in the IDL Extractor for COBOL documentation.



COBOL Source

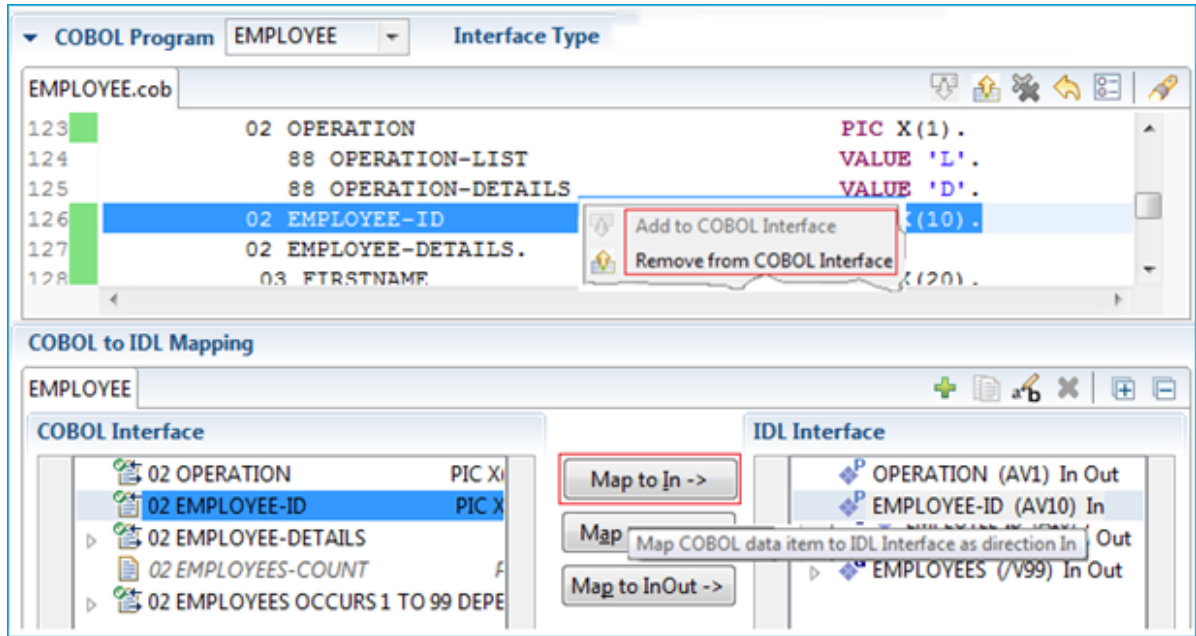
File Name: EMPLAPPL

Operating System: IBM i

Interface Type: BATCH with standard linkage calling convention ▼

☒ Input Message same as Output Message

- 4 Press **Next** to enter the COBOL Mapping Editor.



The purpose of the COBOL Mapping Editor is to map COBOL data items to IDL (Interface Definition Language) parameters. It consists of three main areas:

- The upper window pane is the COBOL program source.
- The window pane on the left-hand side is the COBOL interface, consisting of COBOL data items representing the parameters in your COBOL server program.
- The window pane on the right-hand side is the IDL interface consisting of IDL parameters.

For more information see *Mapping Editor User Interface* in the IDL Extractor for COBOL documentation.

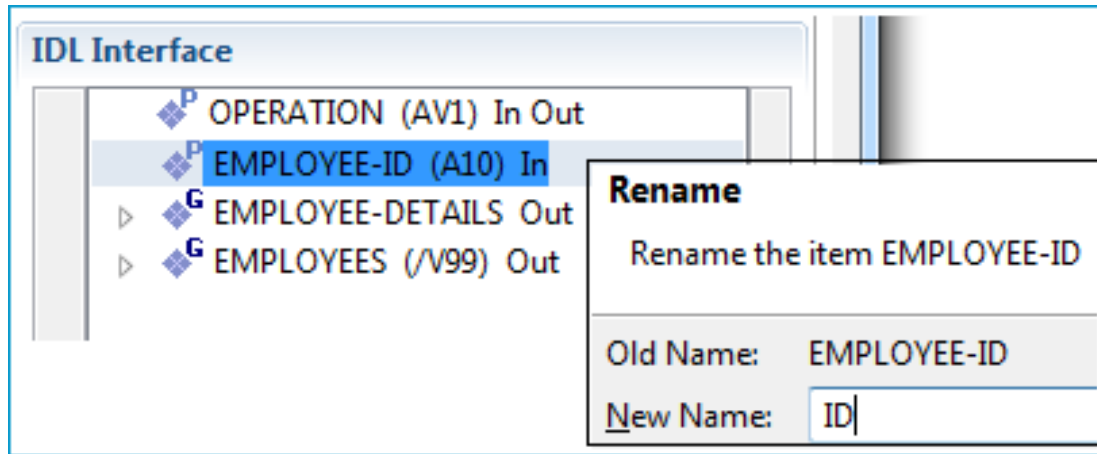
COBOL extraction is a two-step process:

- First, select the COBOL parameters using the **Add to** and **Remove from COBOL Interface** from the context menu on the COBOL data items in the upper window pane (COBOL source); in many situations the COBOL interface is detected by the Mapping Editor directly and you can omit this step.
- Then map the COBOL parameters to IDL directions using the **Map to In/Out/InOut** buttons. COBOL parameters do not have directions, they are always INOUT. This step can be left out for a quick test in non-production scenarios; for real production it is recommended to set IDL directions to reduce data transfer. You also have the option to assign a meaningful name to an IDL parameter if the original COBOL data item is not self-explanatory.

For our example:

- Set **Map to In** for COBOL data item `EMPLOYEE-ID` as IDL direction (see screenshot in previous step). For `EMPLOYEES-DETAILS` and `EMPLOYEES` perform the similar **Map to Out**.

- Use the function **Rename** from the context menu on the IDL parameters in the **IDL Interface** pane to rename COBOL parameter `EMPLOYEE-ID` to IDL parameter `ID`. Do the same for `EMPLOYEE-DETAILS`: Rename to `EMPLOYEE`.



- 5 Press **Finish**. The default extraction settings ensure you get useful results. The outcome is a Software AG IDL file (interface definition language), a Designer file with extension `.idl`:

```
library 'EMPLAPPL' is
program 'EMPLOYEE' is
define data parameter
  1 OPERATION (AV1) In Out
  1 ID (AV10) In /* Original Name:EMPLOYEE-ID
  1 EMPLOYEE Out /* Original Name:EMPLOYEE-DETAILS
  2 FIRSTNAME (AV20)
  2 SURNAME (AV20)
  2 DATE-BIRTH (AV12)
  2 DETAILS (AV100)
  1 EMPLOYEES (/V99) Out
  2 IDENT (AV10)
  2 FIRSTNAME (AV20)
  2 SURNAME (AV20)
  2 DATE-BIRTH (AV12)
end-define
```

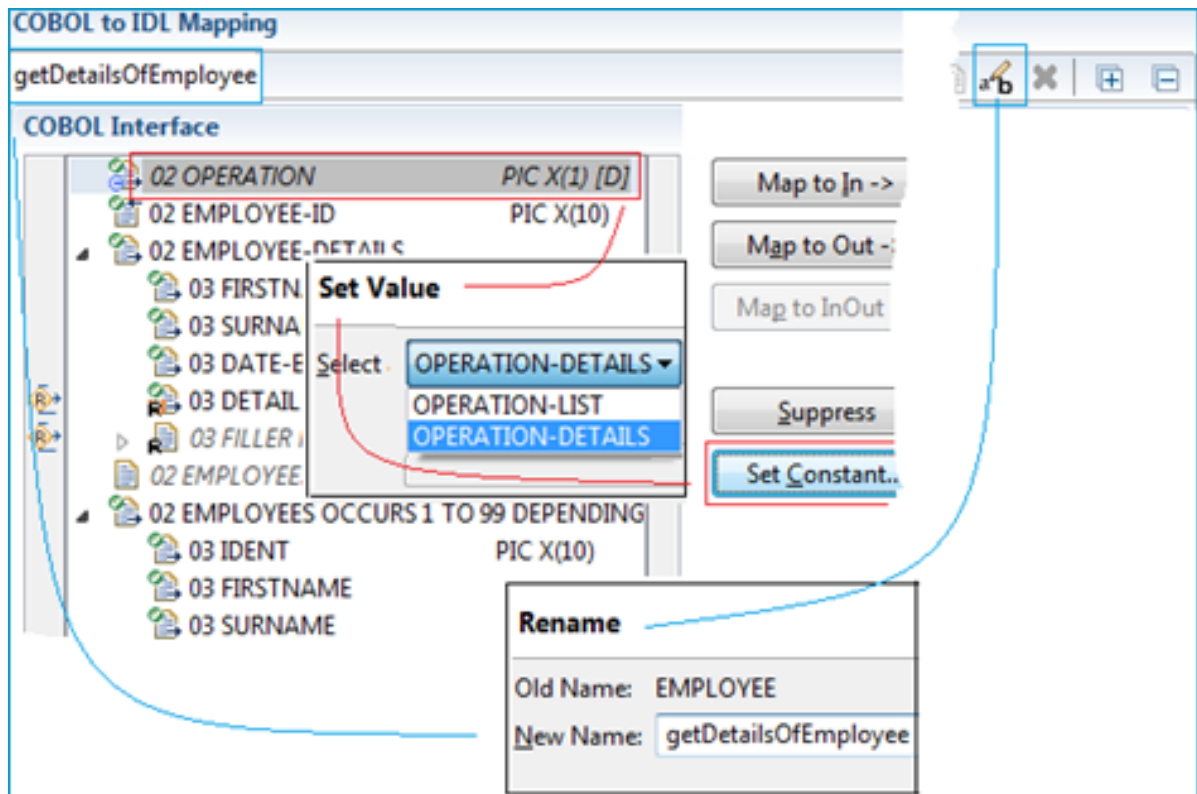
If required, a server mapping file with the same name as the IDL file but the extension `.cvm` may be generated too, for technical reasons. This file must always be stored together with the concomitant IDL file. For more information, see *Server Mapping Files for COBOL* in the Designer documentation. By default, the interface name `EMPLOYEE` is taken from the COBOL server program name.


Extracting a COBOL Server - Modern Method with User-defined Mapping

EntireX allows you to extract all implemented interfaces of the COBOL server separately. Instead of a large `EMPLOYEE` interface, separate interfaces `getListOfEmployees` and `getDetailsOfEmployee` are extracted. Each interface contains required parameters; obsolete parameters for an interface are suppressed, improving its usability.

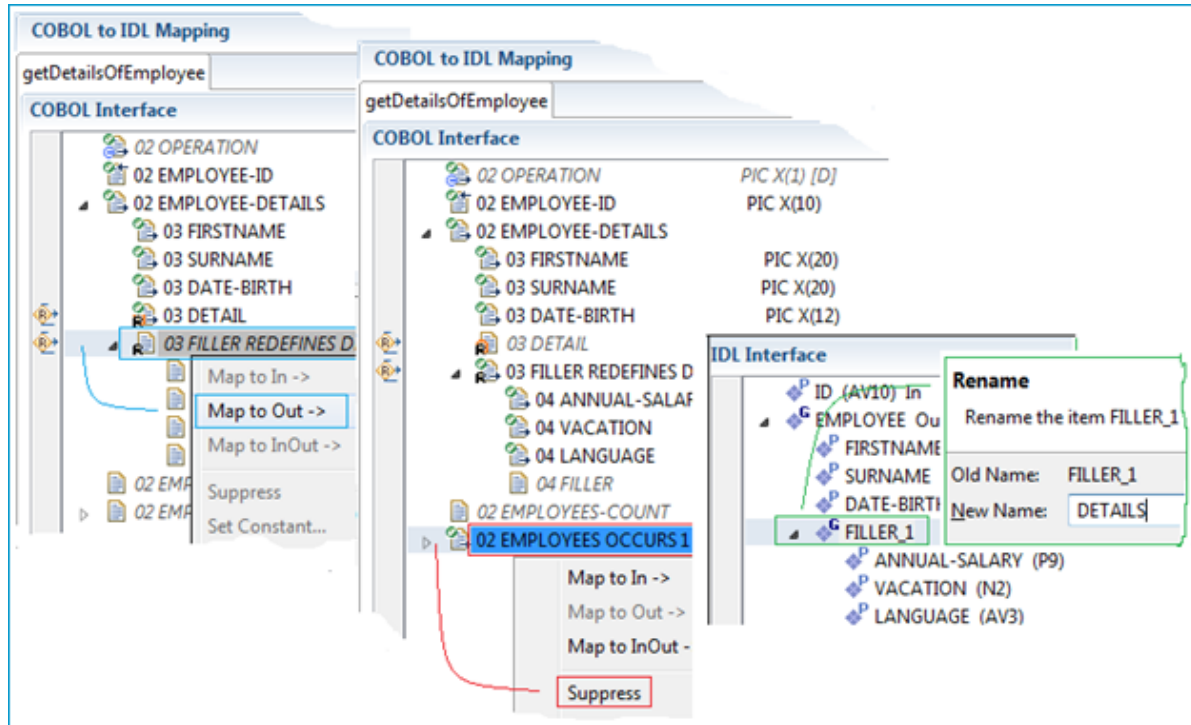
➤ To extract a COBOL server with a user-defined mapping

- 1 Shape `EMPLOYEE` to `getDetailsOfEmployee`:



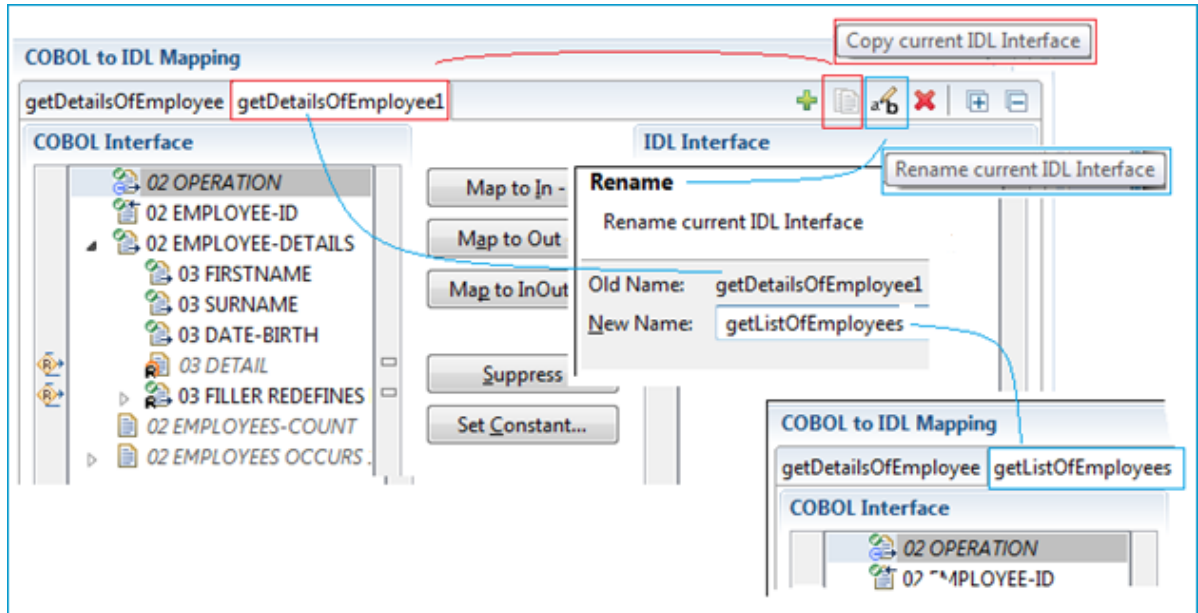
- Mark the `OPERATION` field in the COBOL **Interface** pane and press the **Set Constant** button. In the pop-up window, select `OPERATION-DETAILS` (red markers). The `OPERATION-DETAILS` value originates from COBOL level-88 data item (blue markers). If there is no COBOL level-88 enumeration type in the COBOL program defined, you can enter the function code manually in the pop-up window.
- By default the IDL interface gets the same name as your COBOL server: `EMPLOYEE`. Use the **Rename** button  from the toolbar of the COBOL Mapping Editor and change the IDL interface name to `getDetailsOfEmployee` (blue markers). This name is used as the IS service name later.

- 2 Continue shaping, making `getDetailsOfEmployee` easier to use:





- Use the context menu of the redefinition of the `DETAIL` field in the **COBOL Interface** pane, `FILLER REDEFINES DETAIL` and choose **Map to Out** (blue markers). A redefinition is a secondary layout (field types) of the same memory area. The redefinition contains more specific information (`ANNUAL-SALARY`, `VACATION`, etc.) than character buffer `DETAIL`. This is the reason why it is more useful in your interface.
- Use the context menu of `FILLER_1` in the IDL interface pane and rename `FILLER_1` to `DETAILS` (green markers) for a readable IS service field name later.
- The `DETAILS` function in the COBOL server does not make use of the `EMPLOYEES` field. Therefore you can leave it out of the interface: use the context menu of the `EMPLOYEES` field in the **COBOL Interface** pane and choose **Suppress** (red markers).

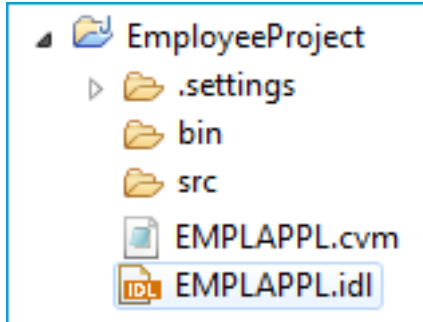
3 Shape `EMPLOYEE` to `getListOfEmployees` Function:



The COBOL Mapping Editor allows you to define multiple IDL interfaces to the same COBOL server:

- You create the additional IDL interface using the toolbar button **Copy current IDL interface**  (red markers). A new tab `getDetailsOfEmployee1` is created. The previously extracted `getDetailsOfEmployee` interface still exists in the first tab. Once you reactivate the first tab, you will see the interface of `getDetailsOfEmployee` again.
 - Use the **Rename** button  from the toolbar of the COBOL Mapping Editor and change the IDL interface name to `getListOfEmployees` (blue markers). This name is used as the IS service name later.
 - For the `LIST` function you need the `OPERATION-LIST` value in the `OPERATION` field: Mark the `OPERATION` field in the **COBOL Interface** pane and press the **Set Constant** button. In the pop-up window, select `OPERATION-LIST` (similar to Step 1. above *Shape `EMPLOYEE` to `getDetailsOfEmployee`*; red markers).
 - Use the context menu of the `EMPLOYEES` field in the **COBOL Interface** pane, `EMPLOYEES OCCURS 1 TO 99` and choose **Map to Out** (Similar to Step 2. above *Continue shaping, making `getDetailsOfEmployee` easier to use*; blue markers).
 - Because the `EMPLOYEE` and `EMPLOYEE-DETAIL` fields are not used in the COBOL server `LIST` function, you leave them out in the IDL interface: use the context menu of `EMPLOYEE-DETAIL` field in the **COBOL Interface** pane and choose **Suppress** (similar to Step 2. above *Continue shaping, making `getDetailsOfEmployee` easier to use*; red markers). Do the same for the `EMPLOYEE-ID` field.
- 4 Press **Finish** to retrieve the extraction result in the form of a Software AG IDL file. At the same time, a server mapping file for COBOL (Designer file with extension `.cvm`) is created.

See *Server Mapping Files for COBOL* in the Designer documentation. The *Software AG IDL File* in the IDL Editor documentation describes the interfaces from the client point of view, while the server mapping file contains the mapping to the real COBOL server. Both of these files must be kept together and in sync, otherwise a call to the COBOL server may fail.



To summarize: We created two IDL interfaces. In the IDL file, these resulted in two IDL programs: `getListOfEmployees` and `getDetailsOfEmployee`. Both IDL programs were given readable names (Steps 1 and 3). Meaningful fields were kept, while superfluous fields were suppressed (Steps 1 and 3). The program `getDetailsOfEmployee` contains the redefined fields of parameter `DETAILS` (see below) mapped during extraction (Step 1).



Note: At runtime, the RPC client generated with the extracted interface will send data for the redesigned interfaces, while your COBOL server still expects `EMPLOYEE` data. The EntireX runtime transforms the incoming data stream from the RPC client, using the server mapping file.

```

program 'getDetailsOfEmployee' is
  define data parameter
    1 ID (AV10) In /* Original Name:EMPLOYEE-ID
    1 EMPLOYEE Out /* Original Name:EMPLOYEE-DETAILS
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
    2 DETAILS /* Original Name:FILLER_1
    3 ANNUAL-SALARY (P9)
    3 VACATION (N2)
    3 LANGUAGE (AV3)
  end-define
  program 'getListOfEmployees' is
    1 EMPLOYEES (/V99) Out
    2 IDENT (AV10)
    2 FIRSTNAME (AV20)
    2 SURNAME (AV20)
    2 DATE-BIRTH (AV12)
  end-define

```

Comparison Chart: Fast-track versus User-defined Mapping

The following table highlights the differences when shaping an interface with a *user-defined mapping* compared with a *fast-path extraction*.

	Interface Shaping with User-defined Mapping	Fast-path Extraction
General	User-defined interface with dedicated mapping. For our example, the interfaces are small and tidy and more self-explanatory with long, readable interface names. They are easier to use with the hidden <code>OPERATION</code> field and the suppressed COBOL fields, which are not needed.	Automatic, quick result extraction with COBOL-like interfaces on IS. For our example, the IS interface <code>EMPLOYEE</code> matches exactly 1:1, field by field.
IS service(s)	Multiple small and handy IS services; each <code>OPERATION</code> code is mapped to a separate IS service.	One big IS service.
IS service name	Readable long name.	Short Subprogram name; up to 8 characters.
IS fields	Usage of Suppress and Set Constant reduces the message length. This keeps focus on relevant data items and keeps the client's interface clean. It may also improve performance.	The IS fields and COBOL server parameters match 1:1. As COBOL layout descriptions are sometimes used for many different purposes, irrelevant data items appear and clutter up the IS interface.
<code>OPERATION</code> parameter	Suppressed: the <code>OPERATION</code> parameter does not exist in the IS service as an IS field.	The <code>OPERATION</code> parameter exists in the IS service as an IS field and needs to be filled-in by the client endpoint.
<code>OPERATION</code> code	The <code>OPERATION</code> code is provided internally by EntireX runtime in the <code>OPERATION</code> field.	The <code>OPERATION</code> code needs to be specified in the IS service in the <code>OPERATION</code> field by the client endpoint.
<code>REDEFINE</code> parameters	Either the parameter that is redefined or one of its redefinitions is available as an IS field.	Only the parameter that is redefined is available as an IS field. Redefinitions of the parameter are not available as IS fields.

A user-defined mapping enables you also to define alternative mappings for COBOL data items (`REDEFINES`, tables etc.). These enable scenarios with COBOL servers where data exchange is not fully described by the `LINKAGE SECTION` COBOL data items. For more information refer to the IDL Extractor for COBOL documentation:

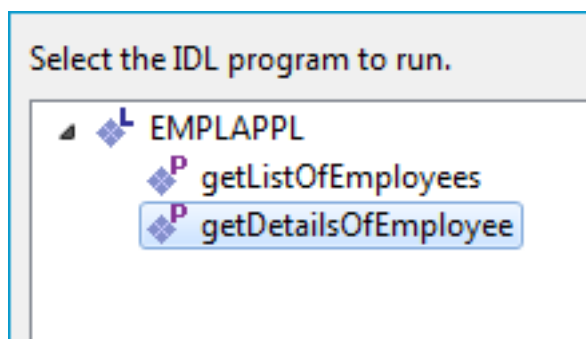
- *Mapping Editor IDL Interface Mapping Functions*
- *User-defined Mapping*

Testing the Extraction Results

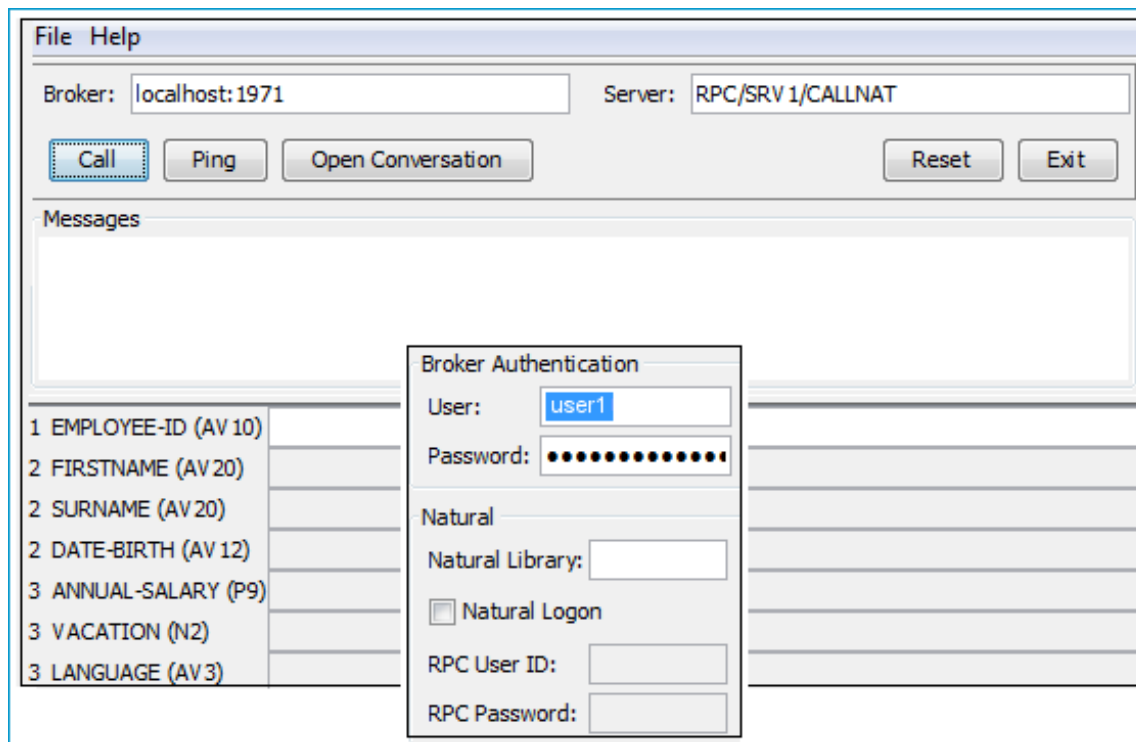
The following pictures use the extraction results described under *Extracting a COBOL Server - Modern Method with User-defined Mapping*.

> To test the extraction results (optional)

- 1 You can test the results of the extraction operation and the REST server back end, using the EntireX IDL Tester. From the context menu of the IDL file in the Designer, choose **Software AG IDL Tester**.



- 2 Select `getDetailsOfEmployee`.



Note that the broker and server parameters contain the explicit route to call the server program, and you can optionally ping the connection from this client. With the **File > Options** dialog, you can set:

- **User** and **Password** for *broker* authentication
- **RPC User ID** and **RPC Password** for *server* authentication

See *EntireX IDL Tester* in the Designer documentation.

- 3 Check the Integration Server log, the EntireX Adapter log or the RPC logs. Applies to all connection methods.

Task 2: Generate the REST Resources, Connection and Adapter Services in Integration Server

This section describes your first steps to create a new Integration Server connection. This is described in more detail under *Using the Integration Server Wrapper*, for example working with existing Integration Server connections. This section covers the following topics:

- [Step 1: Start the Integration Server Wrapper Wizard](#)
- [Step 2: Create a New Integration Server Connection](#)
- [Step 3: Select the Connection Type](#)
- [Step 4: Define Adapter Services for an AS/400 Connection](#)
- [Step 5: Create or Update a REST Resource](#)
- [Step 6: Edit the REST Resource](#)

Step 1: Start the Integration Server Wrapper Wizard

➤ To start the Integration Server Wrapper wizard

- 1 In the context menu of a Software AG IDL file, choose **Integration Server > Generate web-Methods IS Connection**.

This starts the wizard with a list of existing Integration Server Wrapper connections.



Note: If the selected IDL file is not valid because of a syntax error, an error dialog comes up and the wizard does not start.

- 2 Continue with *Step 2: Create a New Integration Server Connection*.

Step 2: Create a New Integration Server Connection

Define New Integration Server Connection

Add host:port, user and password for a new Integration Server

Server:

User:

Password:

☐ Use secure connection

➤ To create a new Integration Server connection

- 1 Define the new Integration Server connection on the wizard page.



Notes:

1. The only required field is **Server**. Enter the hostname of the Integration Server including an optional port number. If no port number is specified, port number defaults to "5555". The **Integration Server Authentication** can be passed with the **User** and **Password** fields.
 2. Optional settings are for secure connections. The **Truststore for HTTPS** contains all signed certificates and must be a valid truststore.
 3. The check box **Verify host name** checks that the hostname is entered in the stored certificate.
 4. When the Integration Server has **Client Authentication** enabled, you can specify your **Keystore** file and keystore **Password**.
 5. For managing Integration Server connections, see [Preferences](#).
- 2 Choose **Next** and continue with *Step 3: Select the Connection Type*.

Step 3: Select the Connection Type

☒ Create a new EntireX Adapter connection of type **AS/400 Connection**

☐ Update Adapter Services or Listeners to an existing Integration Server connection:

List of Connections

Connection Name	Package Name	Connection Type	Enabled
Employees:EmployeesConnection	Employees	EntireX RPC Connection	Yes

Total: 1

☒ Map Software AG IDL data types to String

☒ Create or Update REST resource

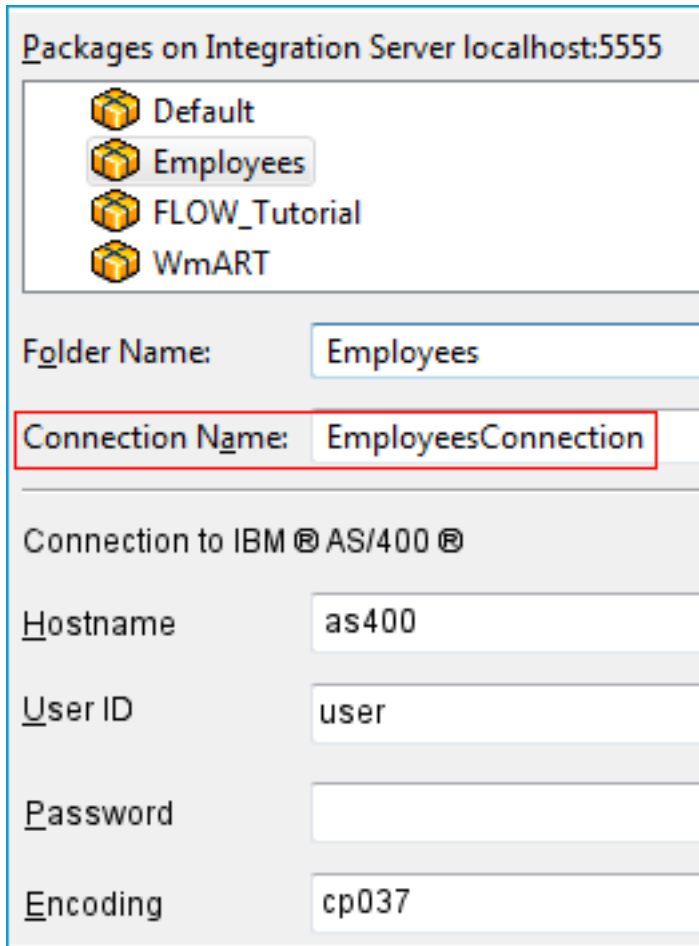
> To create a new connection

- 1 Select a connection type from the drop down list. Connection types are described under *EntireX Adapter Connections* in the EntireX Adapter documentation and *Introduction to the Integration Server Wrapper*.



Note: The list of connection types is filtered: connection types that require a license are only shown if a corresponding license file is available. Reliable RPC connections are only shown if all IDL programs contain only IN parameters. Also, if a server mapping file is available, only those connection types that support the interface type specified in the server mapping file are shown.

- 2 Click **Next** and continue with *Step 4: Define Adapter Services for an AS/400 Connection*.

Step 4: Define Adapter Services for an AS/400 Connection


Packages on Integration Server localhost:5555

- Default
- Employees
- FLOW_Tutorial
- WmART

Folder Name: Employees

Connection Name: EmployeesConnection

Connection to IBM® AS/400®

Hostname: as400

User ID: user

Password:

Encoding: cp037

➤ **To create a connection and related adapter services**

- 1 Select a package for the created objects.
- 2 Define a folder name. If the folder does not exist, it will be created.
- 3 Define a connection name.
- 4 Define the parameters of the connection type. For details, see *Connection Parameters for AS/400 Connections* in the EntireX Adapter documentation.

As a result, the folder will contain the connection and the adapter services (one for each IDL program). The name of a service is the same as the respective IDL program.

The default settings for the adapter services are:

- the **Default** package; if not available, the first package
- the IDL library name for the **Folder Name**

- the IDL library name with the suffix "Connection" for the **Connection Name**



Note: 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.

Step 5: Create or Update a REST Resource

> To create or update a REST resource

Make sure checkbox **Create or Update REST resource** is checked.

- 1 Select a package.

The screenshot shows a dialog box titled "Packages on Integration Server". It contains a list of packages: Default, Employees (selected), WmAdmin, and WmART. Below the list, there are two text input fields: "Folder Name:" with the value "Employees" and "REST Resource Name:" with the value "employees". At the bottom, there are two checked checkboxes: "Update existing Resource" and "Create or Update simplified REST services for this REST resource".

- 2 Define a folder name. If the folder does not exist, it will be created.
- 3 Define the REST Resource name.
- 4 Check **Update existing Resource** if you want to add additional services to an existing resource.
- 5 Check **Create or Update simplified REST services for this REST resource** if you want to generate additional services with a simplified REST API. This allows you to use a simple HTTP-GET method without a payload.

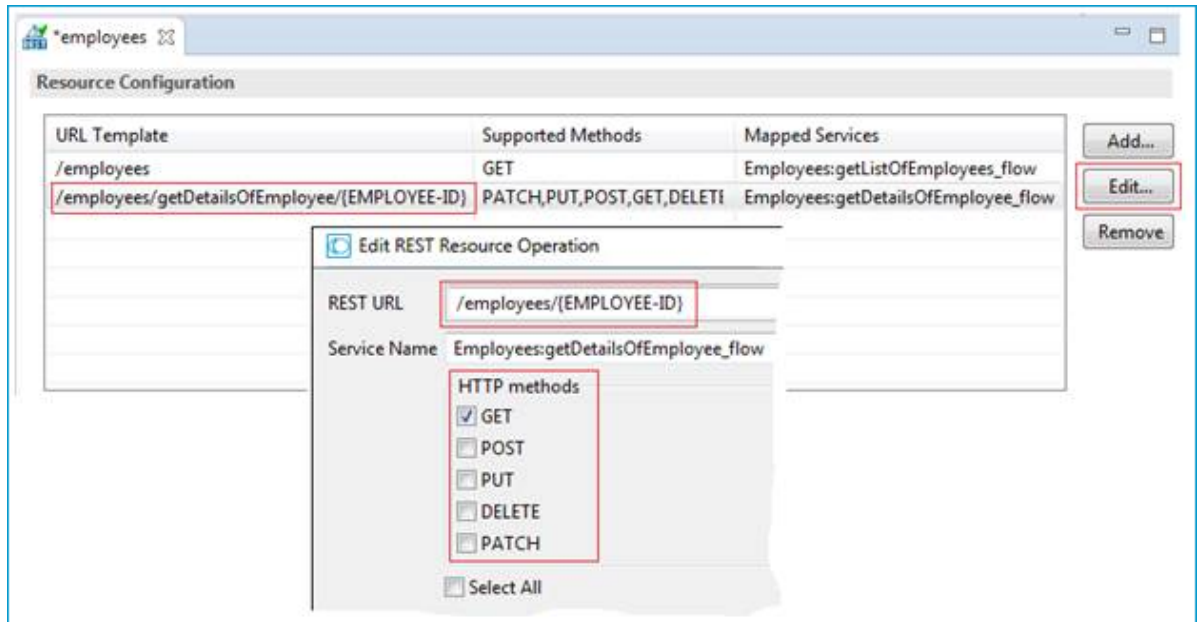


Note: This applies only to APIs where input signature parameters do not contain arrays.

Step 6: Edit the REST Resource

> To edit the REST resource

- 1 Switch to the **Service Development** perspective of Software AG Designer.



- 2 Refresh the package where you generated the REST Resource in the previous step.
- 3 Open the REST Resource.
- 4 Edit the REST URLs and HTTP Methods to match the functionality of your application.

For more information see:

- *REST Developer's Guide* in the webMethods Integration Server documentation for information on using the generated REST resources
- *REST Developer's Guide* in the webMethods Integration Server documentation

Task 3: Execute the Call from the REST Client to COBOL

Use an appropriate REST client, specify URL, Method and Headers and invoke your REST resource:

URL http://localhost:5555/restv2/employees

Method GET

Headers Accept application/json

200 OK

```
{
  "errorCode": "00000000",
  "errorFlag": "false",
  "response": {
    "EMPLOYEES-LIST": {
      "EMPLOYEES": [
        {
          "IDENT": "11100102",
          "FIRSTNAME": "Edgar",
          "SURNAME": "Schindler",
          "DATE-BIRTH": "Dec 4, 1962"
        },
        {
          "IDENT": "11100105",
          "FIRSTNAME": "Christian",
          "SURNAME": "Schirm",
          "DATE-BIRTH": "Mar 15, 1961"
        },
        {
          "IDENT": "11100106",
          "FIRSTNAME": "Rainer",
          "SURNAME": "Schmitt",
          "DATE-BIRTH": "Feb 13, 1955"
        },
        {
          "IDENT": "11100107",
          "FIRSTNAME": "Helga",
          "SURNAME": "Schmidt",
          "DATE-BIRTH": "May 26, 1961"
        }
      ]
    }
  }
}
```

URL http://localhost:5555/restv2/employees/111000105

Method GET

Headers Accept application/json

200 OK

```
{
  "errorCode": "00000000",
  "errorFlag": "false",
  "response": {
    "EMPLOYEE-DETAILS": {
      "FIRSTNAME": "Christian",
      "SURNAME": "Schirm",
      "DATE-BIRTH": "Mar 15, 1961",
      "DETAIL": {
        "ANNUAL-SALARY": "68000",
        "VACATION": "21",
        "LANGUAGE": "GER"
      }
    }
  }
}
```