

webMethods EntireX

Calling Natural from Integration Server

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-NAT2IS-109-20230403

Table of Contents

1 About this Documentation	1
Document Conventions	2
Online Information and Support	2
Data Protection	3
2 Calling Natural from Integration Server	5
Introduction	6
What do I need to Install for this Scenario?	8
Task 1: Extract the Interface of a Natural Server	8
Task 2: Generate the Connection and Adapter Services in Integration Server	19
Task 3: Execute the Call from Integration Server to Natural	24

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 Natural from Integration Server**

■ Introduction	6
■ What do I need to Install for this Scenario?	8
■ Task 1: Extract the Interface of a Natural Server	8
■ Task 2: Generate the Connection and Adapter Services in Integration Server	19
■ Task 3: Execute the Call from Integration Server to Natural	24

Scenario: “I have a Natural server subprogram and want to call this from the Integration Server.”

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



Introduction

To call a Natural server subprogram from the Integration Server, take an existing Natural server **1** and generate the integration logic **2** to call it from IS platform **3**, as shown below.



- 1** Extract the interface of a Natural server. See *Using the Software AG IDL Extractor for Natural*.
- 2** Generate Integration Server adapter service and adapter connections. See *Using the Integration Server Wrapper*.
- 3** Execute call from Integration Server service to Natural server.

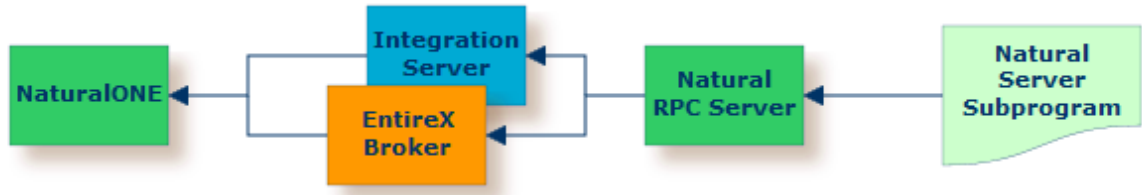
This scenario makes the following important assumptions:

For Task 1:

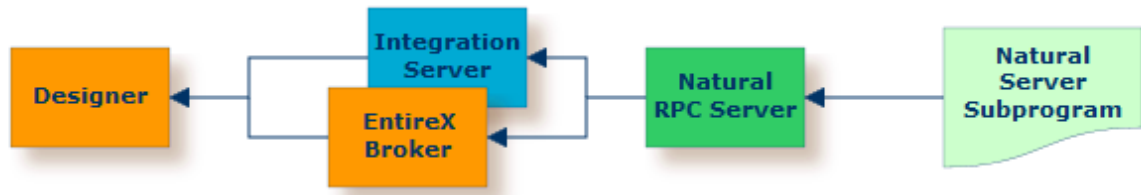
- You have a working Natural subprogram, also known as a CALLNAT program.
- You have access to the sources of this Natural subprogram. These must be stored either
 - *locally*, that is, on the same machine where NaturalONE is running



- or *remotely* and accessed via the Natural RPC Server ⁽¹⁾, and either EntireX Broker or Integration Server.



Instead of NaturalONE, you can also use the Designer.



⁽¹⁾ See your Natural documentation for setting up the Natural RPC Server.

For Tasks 2 and 3:

- You can call the Natural server subprogram 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 Natural RPC Server ⁽¹⁾



- For the *EntireX Direct RPC* connection method you need:
 - the Natural RPC Server ⁽¹⁾



⁽¹⁾ See your Natural documentation for setting up the Natural RPC Server.

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 a Natural RPC Server installed. See your Natural documentation for information on setting up a Natural RPC Server.
- 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 Natural Server

- [Introduction](#)
- [Sample Natural Server used in the Examples](#)
- [Extracting a Natural Server - Fast-track Method](#)
- [Extracting a Natural 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 Natural under *Using the Software AG IDL Extractor for Natural*:

- If your Natural sources are stored *locally* and you are using NaturalONE, see *Extracting IDL from Natural Subprogram Sources in NaturalONE*.
- If your Natural sources are stored *remotely* and this is your first extraction, see *Extracting Software AG IDL File from a New Natural RPC Environment*; for subsequent extractions see *Extracting Software AG IDL File from an Existing Natural RPC Environment*.

This 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 Natural-specific mapping information. See *Server Mapping Files for Natural* in the Designer documentation.

Sample Natural Server used in the Examples

The following Natural server is used to illustrate the features of the IDL Extractor for Natural.

Imagine an existing Natural server called `EMPLOYEE`. It implements the access logic for a `LIST` and `DETAILS` function to a database view `EMPLOYEE`. Its parameter data area is shown below:

```

DEFINE DATA PARAMETER
1 OPERATION          (A1)      /* 'L' => List; 'D' => Details
1 ID                 (A10)     /* Input
1 EMPLOYEE           /* Output
  2 FIRSTNAME        (A20)     /* First name
  2 SURNAME          (A20)     /* Surname
  2 DATE-BIRTH       (D)       /* Date of birth
  2 DETAILS          (A100)
  2 REDEFINE DETAILS
    3 ANNUAL-SALARY(P9) /* Annual salary
    3 VACATION        (N2)     /* Vacation days per year
    3 LANGUAGE        (A3)     /* Language
1 EMPLOYEES          (1:*)     /* Out
  2 IDENT            (A10)     /* Identification number
  2 FIRSTNAME        (A20)     /* First name
  2 SURNAME          (A20)     /* Surname
  2 DATE-BIRTH       (D)       /* Date of birth
END-DEFINE

```

Two approaches for extracting the Natural server are described below:

■ Fast-track

Learn how EntireX helps you to connect the Natural Server with quick results, even without specific Natural knowledge. See [Extracting a Natural Server - Fast-track Method](#).

■ User-defined Mapping

With a user-defined mapping, you can shape the interface to your Natural server. You can minimize the interface by suppressing Natural 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 Natural server. Each function of the Natural server triggered by an operation code or function code can be modelled as a separate IS service. See [Extracting a Natural Server - Modern Method with User-defined Mapping](#).

Extracting a Natural Server - Fast-track Method

➤ To extract the Natural server

- 1 Switch to the EntireX perspective.
- 2 Invoke the IDL Extractor for Natural.
- 3 Select the Natural server subprogram, in this example `EMPLOYEE`, and press **Finish**.

List of Natural subprograms contained in library EMPLAPPL:

Name	Saved	Compiled	Status
<input checked="" type="checkbox"/> EMPLOYEE	016-03-17 15:03:00	2016-03-17 15:03:00	Source a
<input type="checkbox"/> LM0018N	2013-12-04 16:55:02	2016-03-16 14:21:04	Source a
<input type="checkbox"/> LM0315N	2013-12-04 16:59:31	2016-03-16 14:21:04	Source a
<input type="checkbox"/> LM1800N	2013-12-04 17:00:05	2016-03-16 14:21:04	Source a
<input type="checkbox"/> LN0018N	2013-11-04 10:58:07	2016-03-16 14:21:04	Source a

Extraction Settings

☒ Extract from Natural sources (recommended)

☐ Redesign the interfaces (set IN/OUT/INOUT, select REDEFINES, suppress

< Back Next > **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 (A1) /* 'L' => List; 'D' = Details
      1 ID (A10) In /* Input
      1 EMPLOYEE Out /* Output
        2 FIRSTNAME (A20) /* First name
        2 SURNAME (A20) /* Surname
        2 DATE-BIRTH (D) /* Date of birth
        2 DETAILS (A100)
      1 EMPLOYEES (/1:V) Out /* Full usage: (/1:*)
        2 IDENT (A10) /* Identification number
        2 FIRSTNAME (A20) /* First name
        2 SURNAME (A20) /* Surname
        2 DATE-BIRTH (D) /* Date of birth
    end-define

```

By default, the interface name `EMPLOYEE` is taken from the Natural server subprogram name.

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

EntireX allows you to extract all implemented interfaces of the Natural 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 Natural server with a user-defined mapping

- 1 Call the IDL Extractor for Natural, select the Natural server `EMPLOYEE`, check **Redesign the interfaces** and press **Next**.

List of Natural subprograms contained in library EMPLAPPL:

	Name	Saved	Compiled
<input checked="" type="checkbox"/>	EMPLOYEE	2016-03-17 15:03:00	2016-03-17 15:03:00
<input type="checkbox"/>	LM0018N	2013-12-04 16:55:02	2016-03-16 14:21:04
<input type="checkbox"/>	LM0315N	2013-12-04 16:59:31	2016-03-16 14:21:04
<input type="checkbox"/>	LM1800N	2013-12-04 17:00:05	2016-03-16 14:21:04
<input type="checkbox"/>	LN0018N	2013-11-04 10:58:07	2016-03-16 14:21:04

Extraction Settings

☒ Extract from Natural sources (recommended)

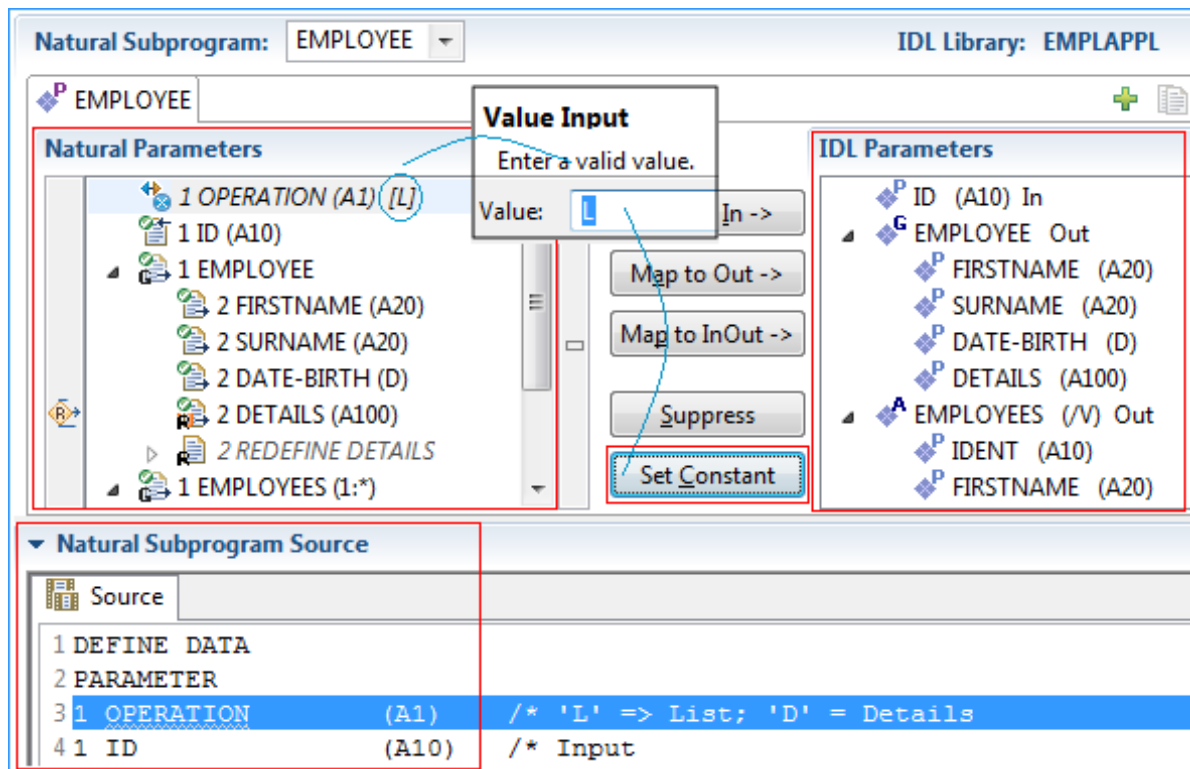
☒ Redesign the interfaces (set IN/OUT/INOUT, select REDEFINES)

< Back Next > Finish

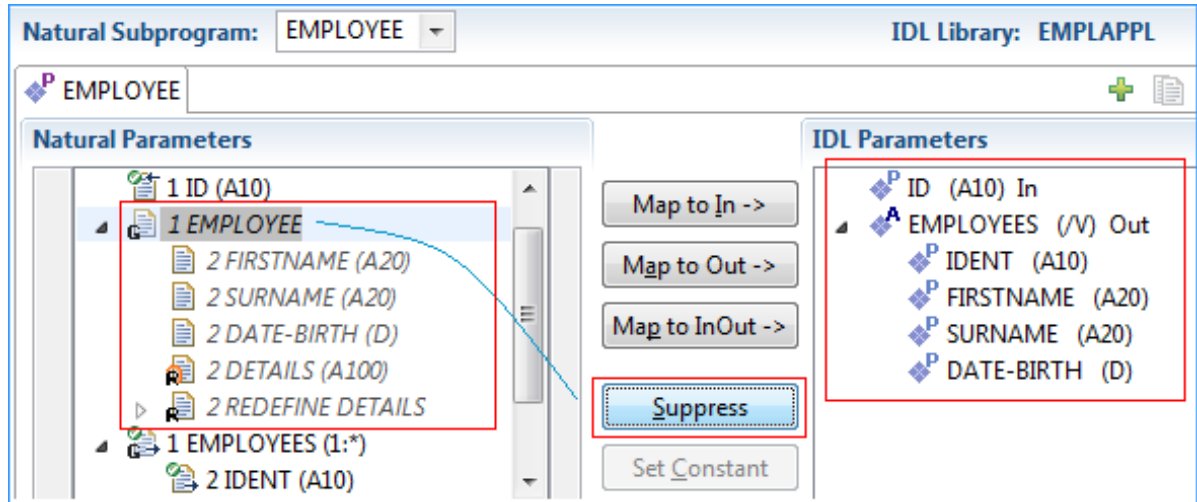
- 2 Model the extracted interface to suit your needs by creating a function `getListOfEmployees`.
 1. Mark the parameter `OPERATION` in the **Natural Subprogram Source** view or the **Natural Parameters** pane
 2. Set Constant 'L' for the parameter `OPERATION`.


In the **Natural Parameters** pane, constant `[L]` is shown in brackets after the Natural parameter. `OPERATION` is removed from the **IDL Parameters** pane.


At runtime, EntireX passes `L` as the value for the `OPERATION` parameter. This forces the `EMPLOYEE LIST` function to be executed.

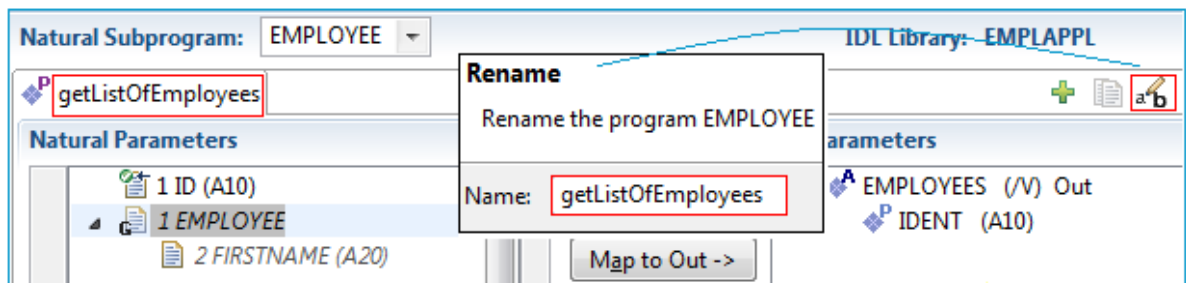



- 3 Suppress the Natural group `EMPLOYEE`, which is unused in the `LIST` function. This removes the IDL group `EMPLOYEE` from the **IDL Parameters** pane. The Natural parameter `EMPLOYEE` remains. Parameters that are suppressed in the **IDL Parameters** pane are displayed in italic font in the **Natural Parameters** pane.

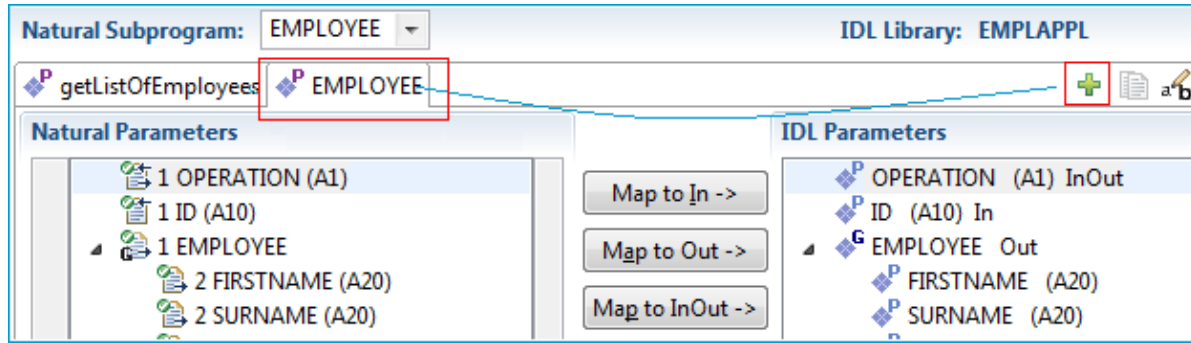


 **Note:** Parameters set to constant are also displayed in italic font, because **Set Constant** suppresses them in the **IDL Parameters** pane too (see Step 2).

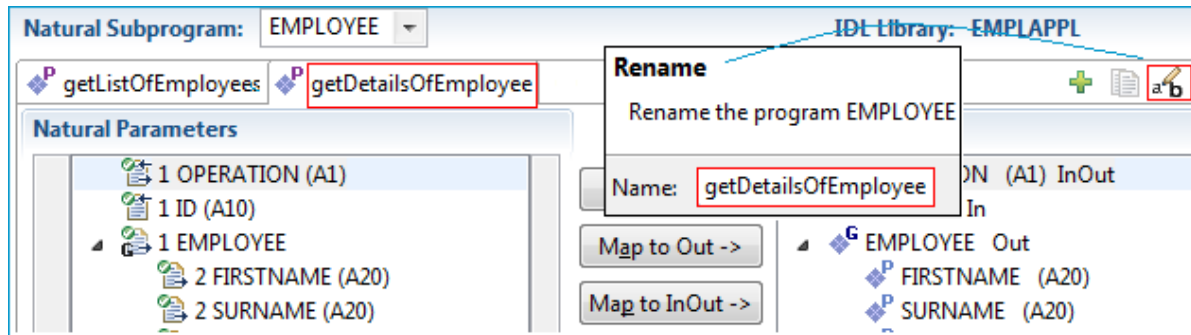
- 4 Also suppress Natural parameter `ID`, similar to what you did for Natural group `EMPLOYEE` in the previous step.
- 5 Specify a readable name `getListOfEmployees` for the **IDL Parameters**, using the toolbar button . The new name appears on the tab.



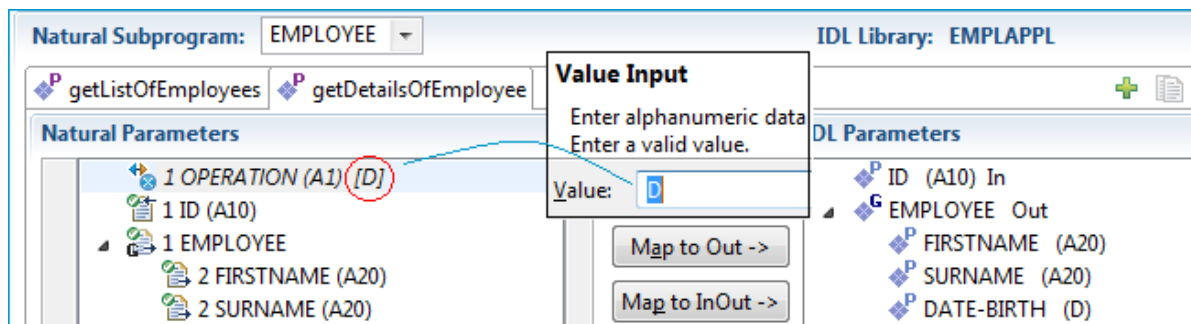
- 6 Create a `getDetailsOfEmployees` function for the `DETAILS` operation. A new interface is needed for this. Use the toolbar button  and open a new tab. The **IDL parameters** are reset to defaults. The previously extracted `getListOfEmployees` interface still exists in the first tab. Once you reactivate the first tab, you will see the interface of `getListOfEmployees` again.



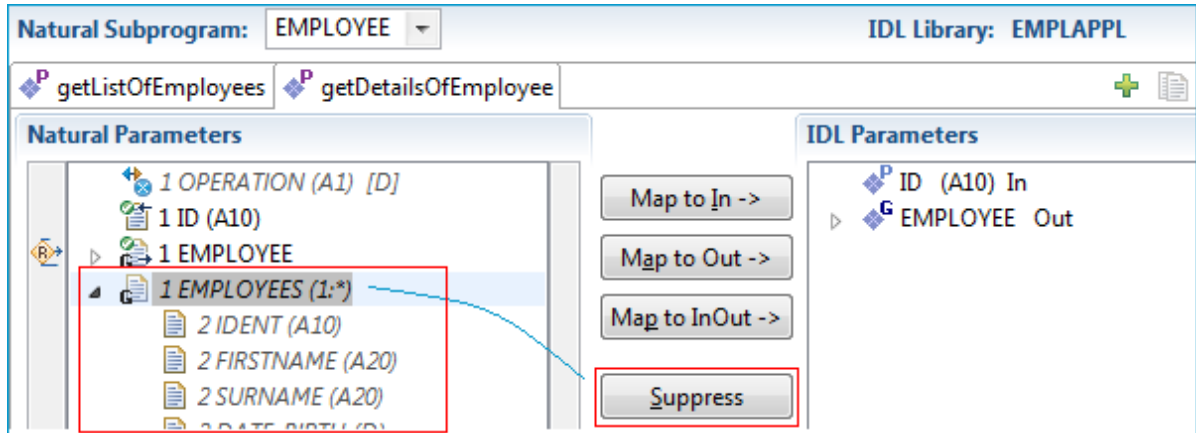
- 7 Specify a readable name `getDetailsOfEmployee` for the **IDL Parameters**, using the toolbar button . The name is displayed on the tab.



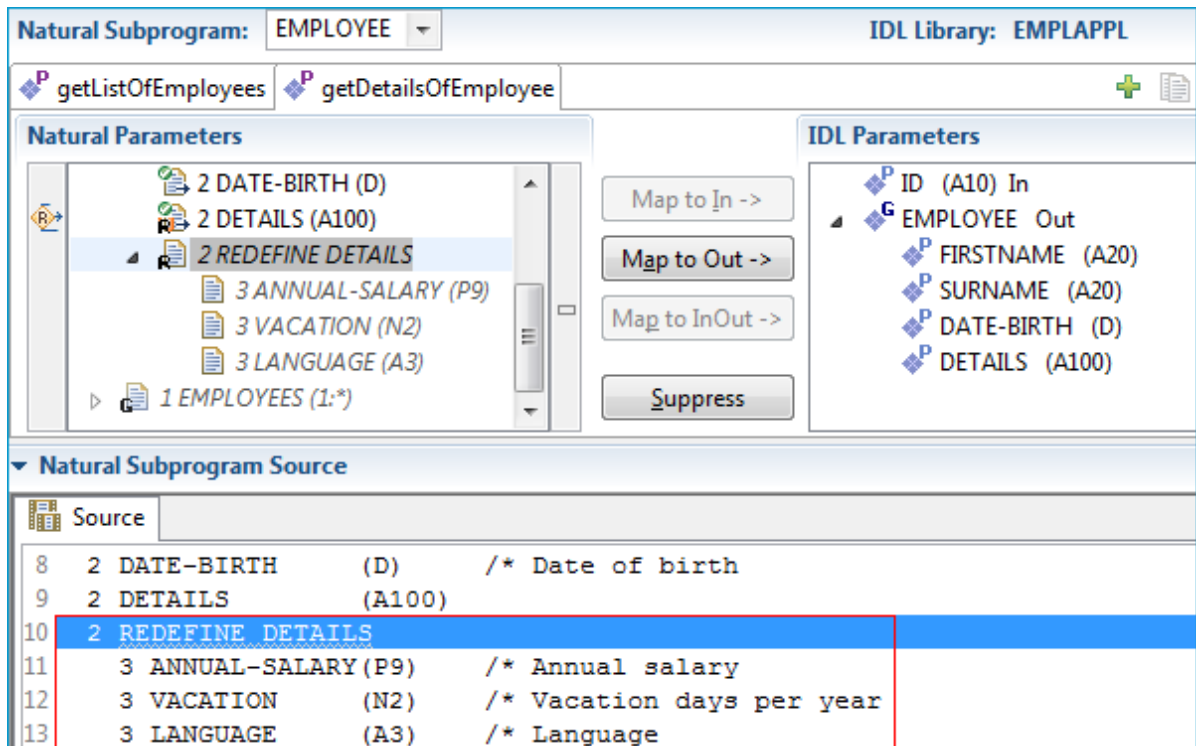
- 8 Set constant 'D' for Natural parameter `OPERATION` to execute the `EMPLOYEE DETAILS` operation at runtime. The IDL parameter `OPERATION` is removed from the **IDL Parameters** pane and displayed in the **Natural Parameters** pane. The italic font indicates suppression.



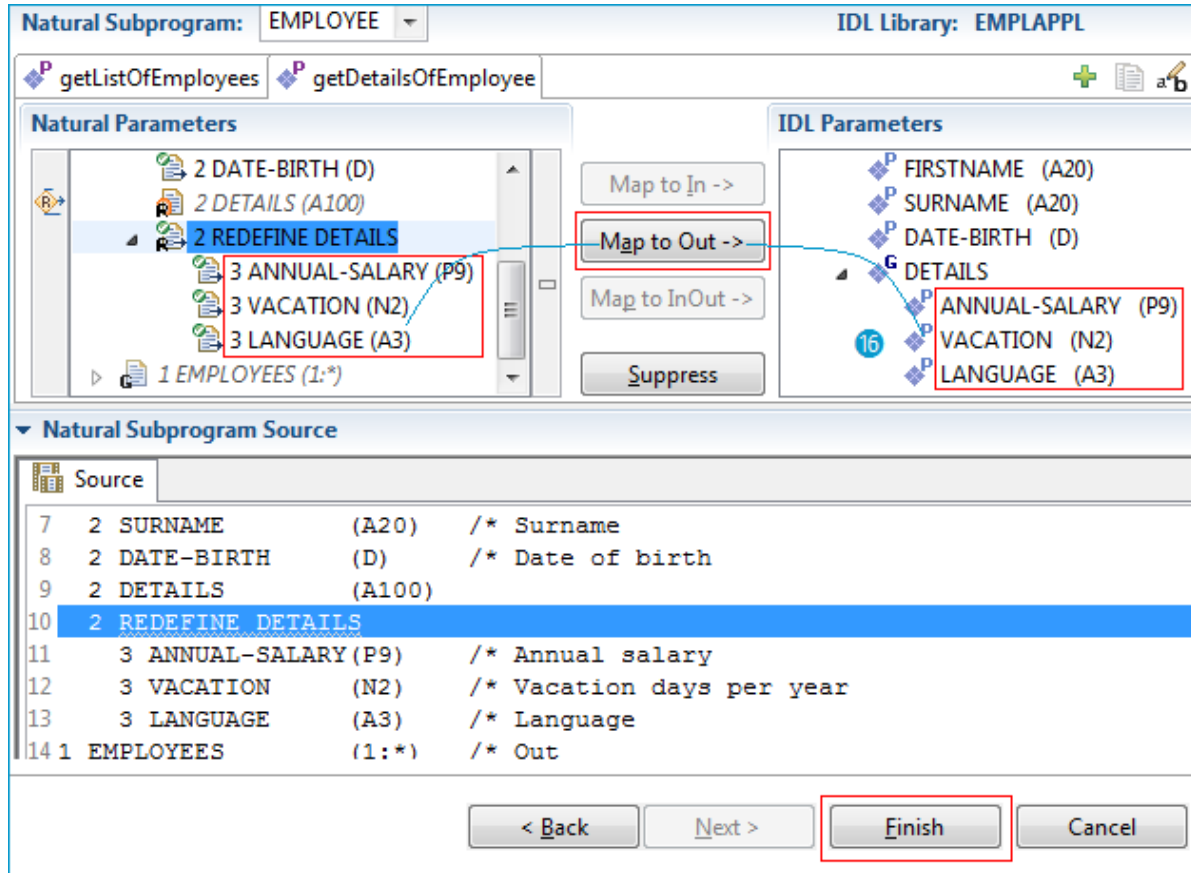
- 9 Suppress the Natural X-array `EMPLOYEES`, which is not needed in the `DETAILS` interface. The IDL parameter `EMPLOYEES` is removed from the **IDL Parameters** pane and displayed in italic font in the **Natural Parameters** pane.



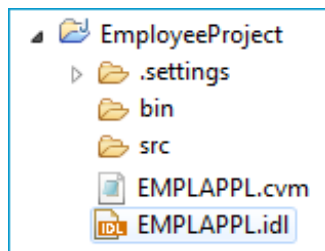
- 10 By default, the Natural parameter DETAILS is mapped. In this case, the redefinition of DETAILS in the **Natural Subprogram Source** pane contains information of more value.



- 11 You can map the redefinition of DETAILS in the **Natural Subprogram** pane if you prefer to use this rather than the DETAILS parameter. Select REDEFINE DETAILS in the **Natural Subprogram** pane and press **Map to Out**. The IDL parameter DETAILS is turned into a group containing parameters that match the Natural redefinition.



- 12 Press **Finish** to retrieve the extraction result in the form of a Software AG IDL file. At the same time, a *Server Mapping Files for Natural* (Designer file with extension .cvm) is created. The *Software AG IDL File* in the IDL Editor documentation describes the interfaces from the , while the server mapping file contains the mapping to the real Natural server. Both of these files must be kept together and in sync, otherwise a call to the Natural 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 4 and 6). Meaningful fields were kept, while superfluous fields were suppressed (Steps 3 and 8). The program `getDetailsOfEmployee` contains the redefined fields of parameter `DETAILS` (see below) mapped during extraction (step 9).



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

```
library 'EMPLAPPL' is
  program 'getListOfEmployees' is
    define data parameter
      1 EMPLOYEES (/V) Out /* Full usage: (/1:*)
      2 IDENT (A10) /* Identification number
      2 FIRSTNAME (A20) /* First name
      2 SURNAME (A20) /* Surname
      2 DATE-BIRTH (D) /* Date of birth
    end-define

    program 'getDetailsOfEmployee' is
      define data parameter
        1 ID (A10) In /* Input
        1 EMPLOYEE Out /* Output
          2 FIRSTNAME (A20) /* First name
          2 SURNAME (A20) /* Surname
          2 DATE-BIRTH (D) /* Date of birth
          2 DETAILS
            3 ANNUAL-SALARY (P9) /* Annual salary
            3 VACATION (N2) /* Vacation days per year
            3 LANGUAGE (A3) /* Language
      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 Natural fields, which are not needed.	Automatic, quick result extraction with Natural-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.

	Interface Shaping with User-defined Mapping	Fast-path Extraction
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 Natural server subprogram parameters match 1:1. As Natural layout descriptions are sometimes used for many different purposes, irrelevant data items appear and clutter up the IS interface.
OPERATIONparameter	Suppressed: the OPERATION parameter does not exist in the IS service as an IS field.	The OPERATION parameter exists in the IS service as an IS field and needs to be filled in by the client endpoint.
OPERATION code	The OPERATION code is provided internally by EntireX runtime in the OPERATION field.	The OPERATION code needs to be specified in the IS service in the OPERATION field by the client endpoint.
REDEFINE 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.

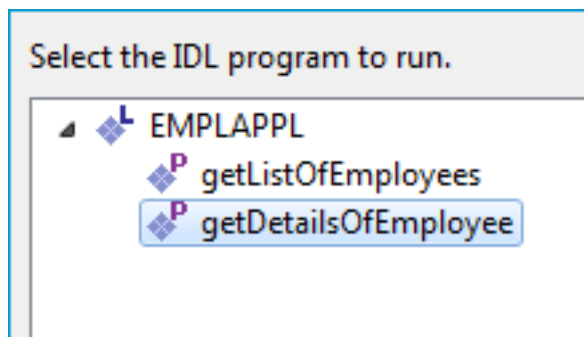
A user-defined mapping enables you also to define alternative mappings for Natural parameters (REDEFINES, etc.). These enable scenarios with Natural servers where data exchange is not fully described by the PDA of the Natural subprogram. For more information refer to the *User-defined Mapping* in the IDL Extractor for Natural documentation.

Testing the Extraction Results

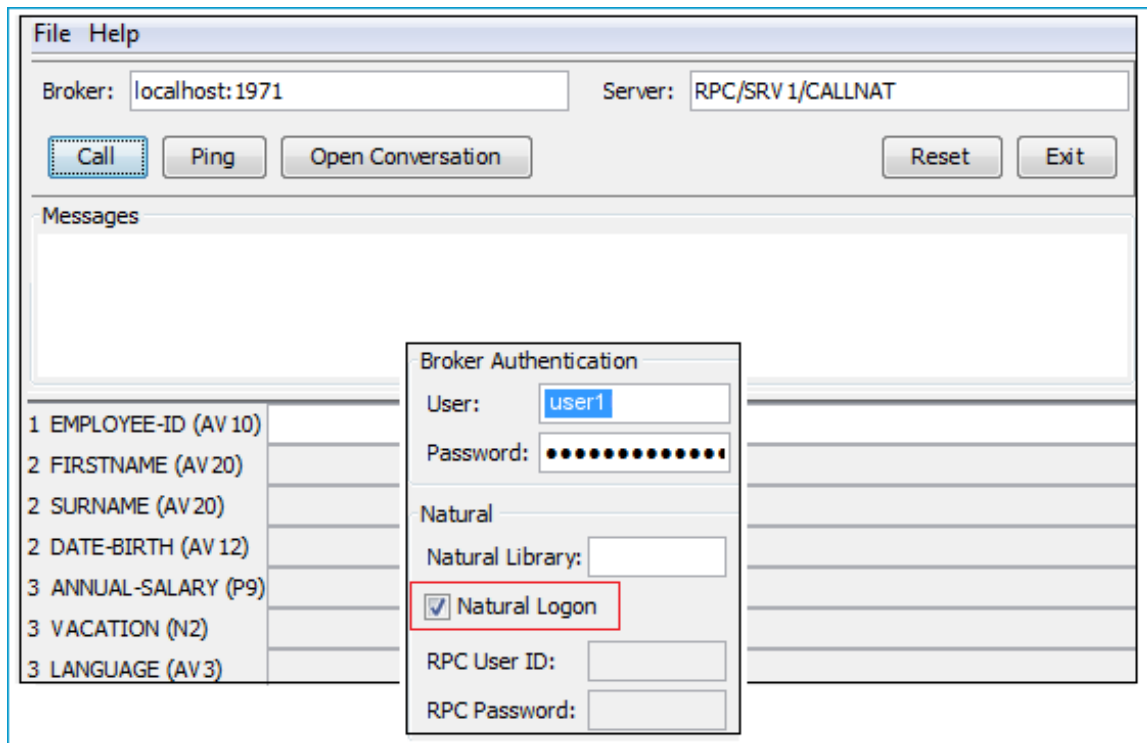
The following pictures use the extraction results described under *Extracting a Natural Server - Modern Method with User-defined Mapping*.

➤ To test the extraction results (optional)

- 1 You can test the results of the extraction operation and the Integration Server 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, check **Natural Logon**. If required 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 an RPC Connection](#)

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**.

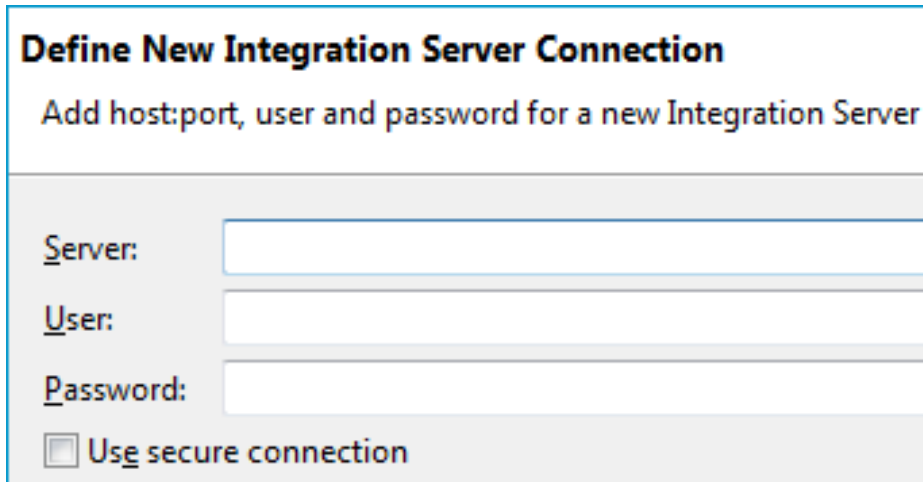
This starts the wizard with a list of existing Integration Server Wrapper connections.



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 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.

Task 3: Execute the Call from Integration Server to Natural

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 following screenshots use the extraction results described in *Extracting a Natural Server - Modern Method with User-defined Mapping*.

We no longer have to specify operation codes to run the services. On output of the `getDetailsOfEmployee` service (see second screen below), the redefinition of the `DETAILS` group is directly available.

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 package structure, a table displays the output of the service.

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

You can use the generated EntireX adapter service like any other IS service - there is no difference. As the webMethods Integration Server developer, you do not require any Natural-specific knowledge on PDAs, LDAs, subprograms, DDMs, CALLNATs, FUSER, FNAT, packed and unpacked formats, X-arrays, etc. EntireX takes care of mapping Natural-specific data types automatically to suitable Integration Server data types. For example, the Natural X-array `EMPLOYEES (1:*)` is mapped to Integration Server document list of `getDetailsOfEmployee` service.

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 *Testing the Extraction Results* above.

