

## **webMethods EntireX**

**Software AG IDL Extractor for Natural**

Version 10.7

October 2020

This document applies to webMethods EntireX Version 10.7 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 © 1997-2020 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-EEXXNATEXTRACTOR-107-20220422**

## Table of Contents

1 About this Documentation .....	1
Document Conventions .....	2
Online Information and Support .....	2
Data Protection .....	3
2 Introduction to the IDL Extractor for Natural .....	5
Scope .....	6
Extractor Wizard .....	7
Server Mapping File .....	8
3 Using the Software AG IDL Extractor for Natural .....	9
Extracting IDL from Natural Subprogram Sources in NaturalONE .....	10
Extracting Software AG IDL File from a New Natural RPC Environment .....	17
Extracting Software AG IDL File from an Existing Natural RPC Environment .....	27
Extraction Result .....	29
Preferences .....	30
4 RPC Environment Manager for Natural RPC Server .....	31
5 RPC Environment Monitor .....	35
6 Using the IDL Extractor for Natural in Command-line Mode .....	37
Command-line Options .....	38
Example .....	39
7 Natural to IDL Mapping .....	41
Mapping Natural Data Types to Software AG IDL .....	42
User-defined Mapping .....	43
Extracting the IDL Library Name .....	43
Extracting the IDL Program Name .....	43
Extracting IDL Parameter Names .....	43
Extracting IDL Directions (IN,OUT,INOUT) .....	44
Extracting Natural REDEFINES .....	45
Extracting Multiple Interfaces .....	45
Extracting Natural Arrays, Groups, X-Arrays and Variable Arrays .....	46
Extracting Natural Structure Information (IDL Levels) .....	48
Extracting Parameters defined with OPTIONAL .....	49
Setting Natural Parameters to Constants .....	49
Suppressing Natural Parameters .....	49
Renaming a Program .....	50
8 Examples for Redesigning the Extracted Interfaces .....	51
Set Constant .....	52
Extracting Multiple Interfaces .....	54
Extracting Natural REDEFINES .....	56
Extracting a Natural Server with User-defined Mapping .....	57



# 1 About this Documentation

---

▪ Document Conventions .....	2
▪ Online Information and Support .....	2
▪ Data Protection .....	3

## Document Conventions

---

Convention	Description
<b>Bold</b>	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 Introduction to the IDL Extractor for Natural

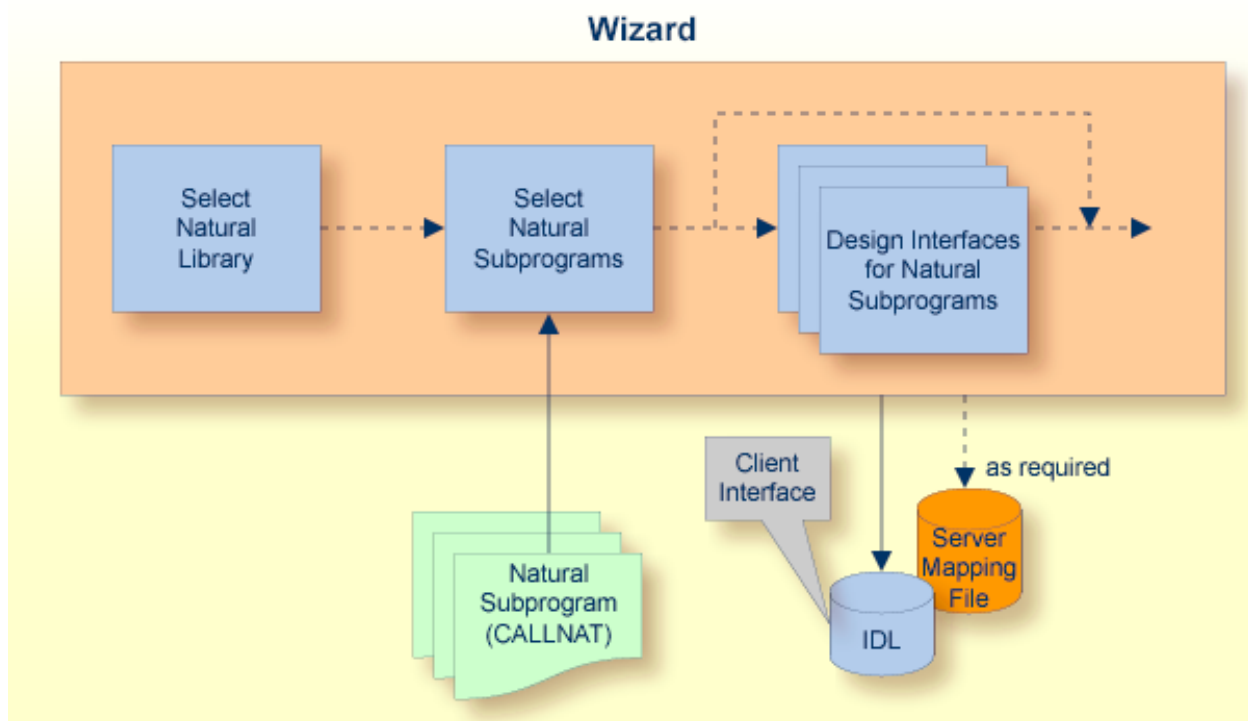
---

▪ Scope .....	6
▪ Extractor Wizard .....	7
▪ Server Mapping File .....	8

## Scope

The Software AG IDL Extractor for Natural extracts a Software AG IDL definition from a Natural source in a Natural project in Eclipse, or from an object within a Natural RPC environment.

In a user-driven process supported by an extractor wizard (see [Using the Software AG IDL Extractor for Natural](#)), the interface of a Natural subprogram (CALLNAT) is extracted, optionally along with various features offered by the **Design Interface for Natural Subprogram** page modelled to a client interface.



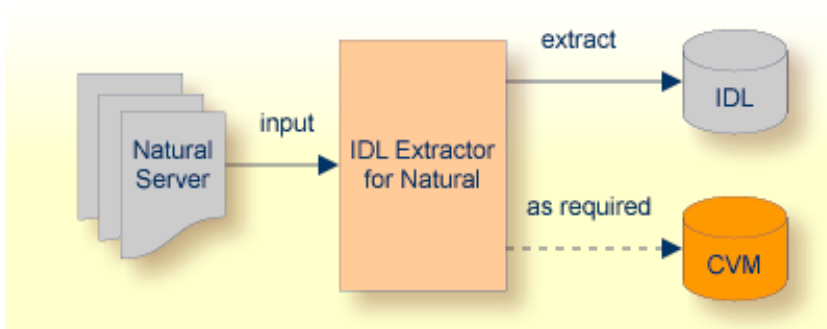
The result of this process is an IDL file and, optionally, a related server mapping file mapping the client interface to the Natural subprograms (CALLNATs):

### ■ IDL File

The Software AG IDL (interface definition language) file contains the modelled interface of the Natural server. In a follow-up step, the IDL file is the starting point for the RPC client-side wrapping generation tools to generate client interface objects. See *EntireX Wrappers* in the Designer documentation.

### ■ Server Mapping File

A server mapping file (extension .cvm) to complete the mapping is generated only if it is required by the RPC server at runtime to call the Natural server. See *Server Mapping Files for Natural* in the Designer documentation.



## Extractor Wizard

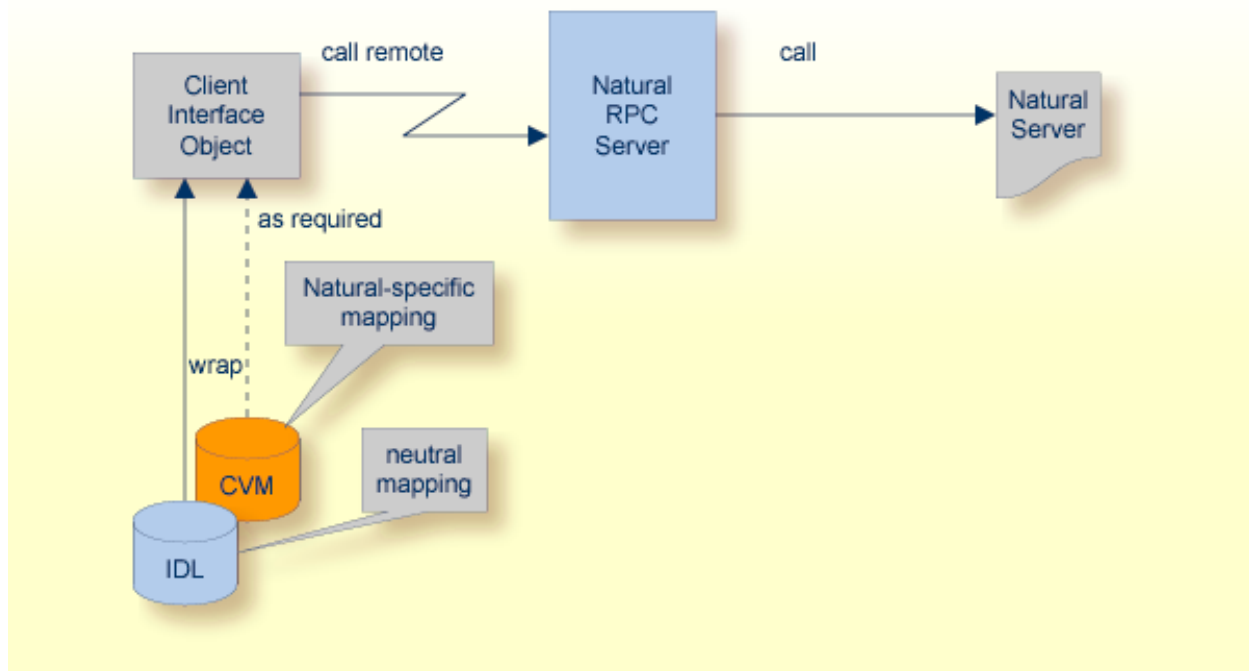
The extractor wizard guides you through the extraction process and supports the following tasks:

- Accessing Natural subprograms (CALLNATs), sources or object files, either in the local file system where the Designer is running, or remotely from the host computer with the RPC server extractor service.
- Selecting multiple Natural subprograms for IDL extraction.
- Redesign the extracted interface. See [User-defined Mapping](#).
  - Provide IDL directions (IN, OUT, INOUT) for parameters of the Natural subprogram, see [Extracting IDL Directions \(IN,OUT,INOUT\)](#).
  - Selecting the REDEFINE to be used in the IDL, see [Extracting Natural REDEFINES](#).
  - Suppress or hide unneeded fields of the Natural subprogram. This keeps the IDL client interface lean, and also minimizes the amount of data to be transferred at runtime.

The extractor wizard is described in a step-by-step tutorial; see [Using the Software AG IDL Extractor for Natural](#).

## Server Mapping File

The server mapping file completes the IDL file with a mapping from the programming-language-neutral parameter definition in the IDL file to the parameters and data types expected by the Natural subprograms (CALLNATs). Extraction is only performed if it is needed. For information on when a server mapping file is required, see *When is a Server Mapping File Required?* in the Designer documentation.



If a server mapping file (extension `.cvm`) has to be extracted, a correct interpretation of the IDL file is only possible if the server mapping file is also available. See *Server Mapping Files for Natural* in the Designer documentation. Always keep the server mapping files in the same folder as the related IDL file.

A server mapping file contains:

- constructs specific to Natural such as `REDEFINE`
- suppress parameters
- constants
- multiple interfaces
- IDL program names mapped to customized Natural names; see *Using the Natural Wrapper for the Server Side*.
- etc.

# 3 Using the Software AG IDL Extractor for Natural

---

- Extracting IDL from Natural Subprogram Sources in NaturalONE ..... 10
- Extracting Software AG IDL File from a New Natural RPC Environment ..... 17
- Extracting Software AG IDL File from an Existing Natural RPC Environment ..... 27
- Extraction Result ..... 29
- Preferences ..... 30

This chapter describes how to use the Software AG IDL Extractor for Natural.

## Extracting IDL from Natural Subprogram Sources in NaturalONE

---

In a NaturalONE project in Software AG Designer, IDL can be extracted directly from Natural subprogram sources (CALLNATs), using the following steps:

- [Step 1: Start the IDL Extractor for Natural Wizard](#)
- [Step 2: Select the Natural Library from NaturalONE Project \(Optional\)](#)
- [Step 3: Select the Natural Subprograms from NaturalONE Project](#)

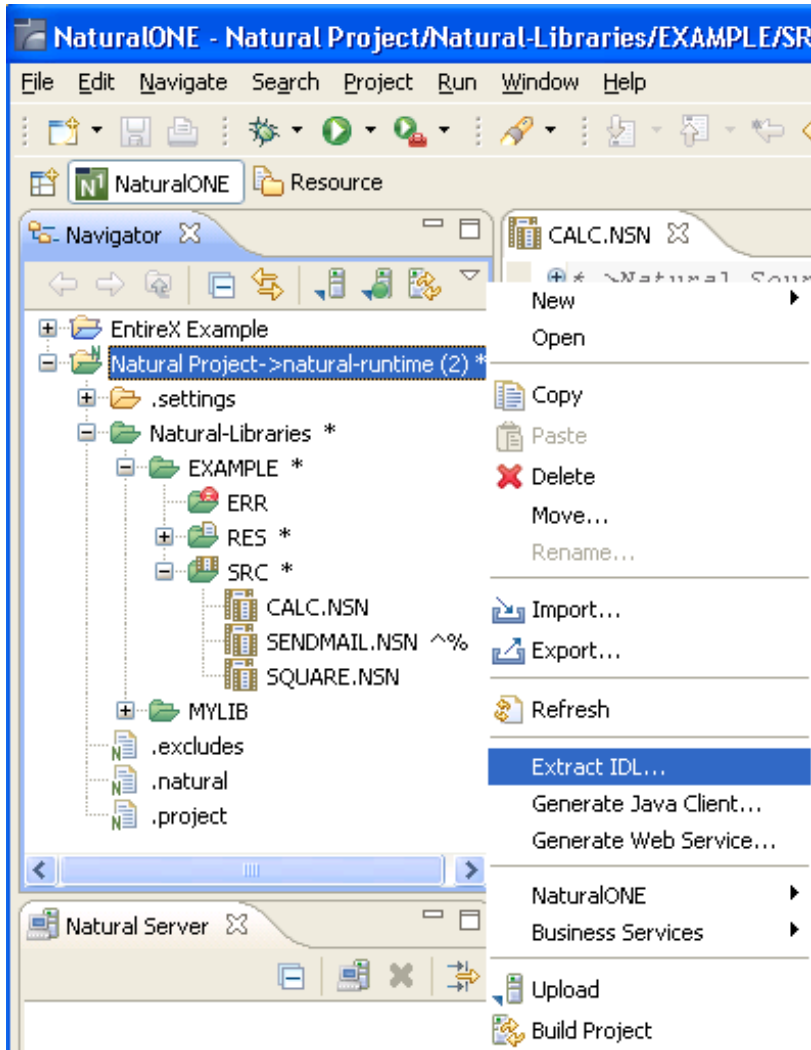
### Step 1: Start the IDL Extractor for Natural Wizard

There are various ways of starting the IDL Extractor for Natural Wizard:

- [Select a NaturalONE Project](#)
- [Select a Natural Library in a NaturalONE Project](#)
- [Select Natural Subprograms in a NaturalONE Project](#)

#### Select a NaturalONE Project

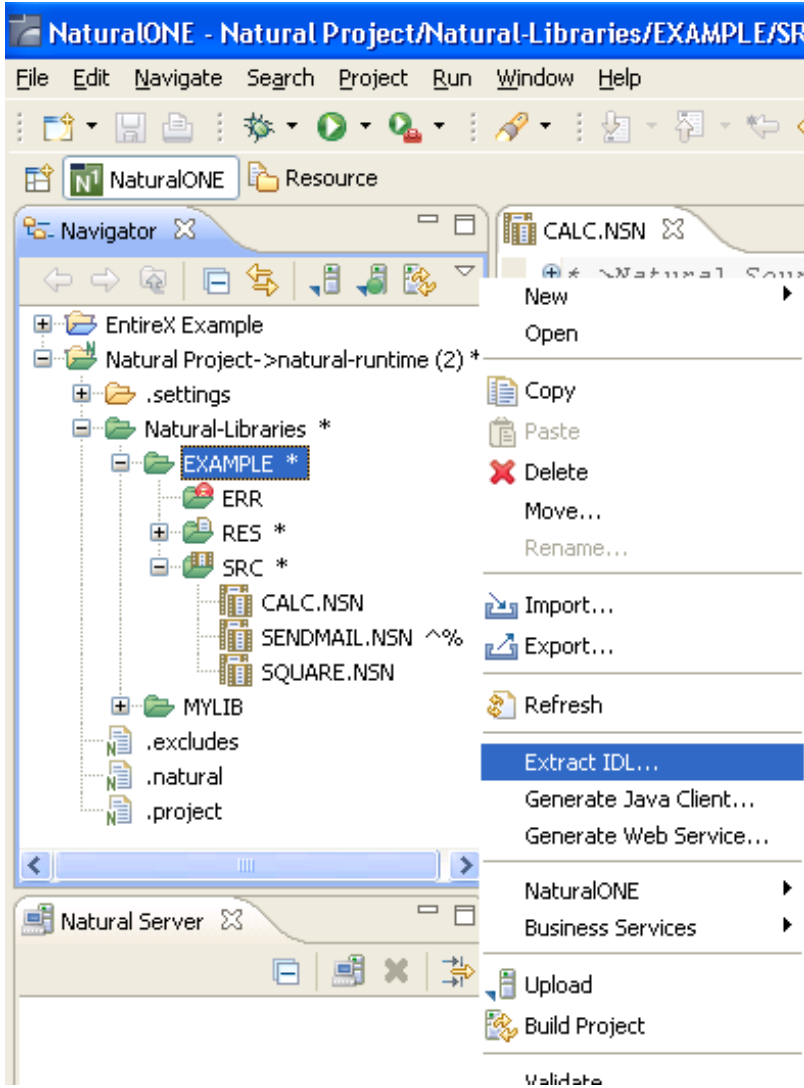
To extract the IDL, select the NaturalONE project in the Designer.



From the context menu, choose **Extract IDL...** and continue with [Step 2: Select the Natural Library from NaturalONE Project \(Optional\)](#).

### Select a Natural Library in a NaturalONE Project

To extract the IDL, select the NaturalONE library in the Designer.

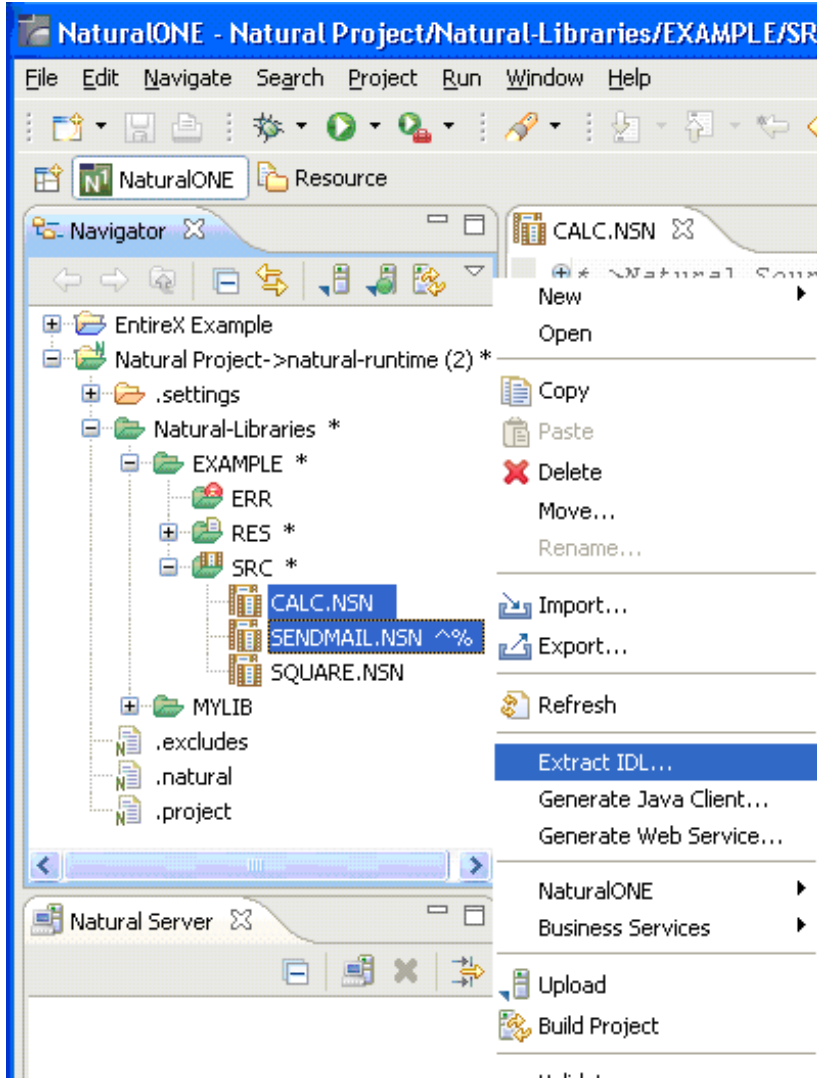


From the context menu, choose **Extract IDL...** and continue with [Step 3: Select the Natural Subprograms from NaturalONE Project](#).

### Select Natural Subprograms in a NaturalONE Project

To extract the IDL, select one or multiple Natural subprogram sources (.NSN) in the Designer.

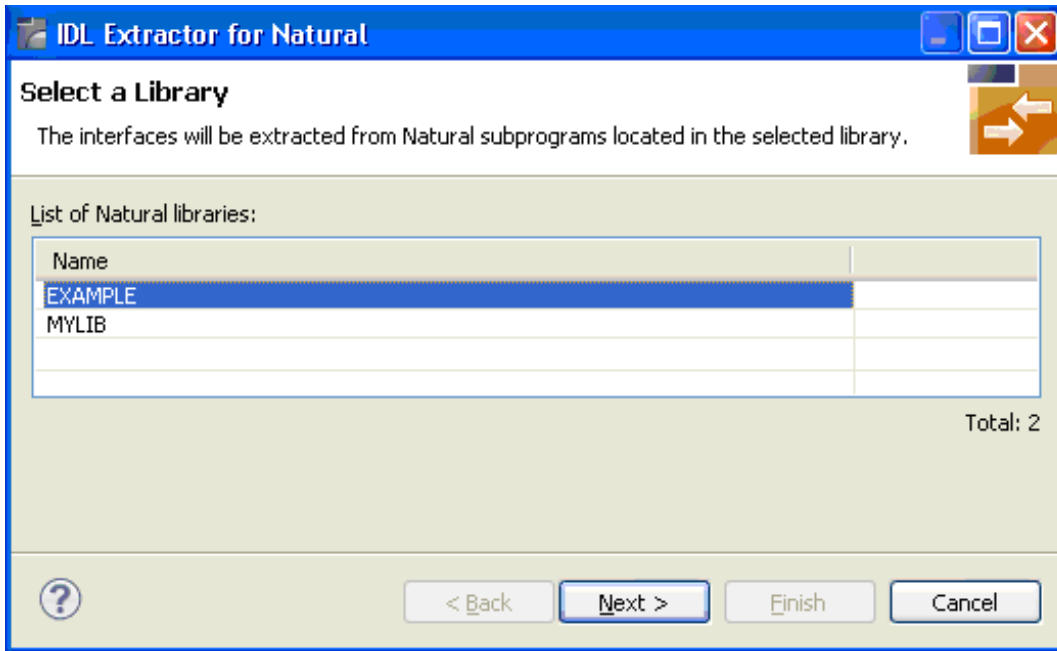




From the context menu, choose **Extract IDL...** and continue with [Step 3: Select the Natural Subprograms from NaturalONE Project](#).

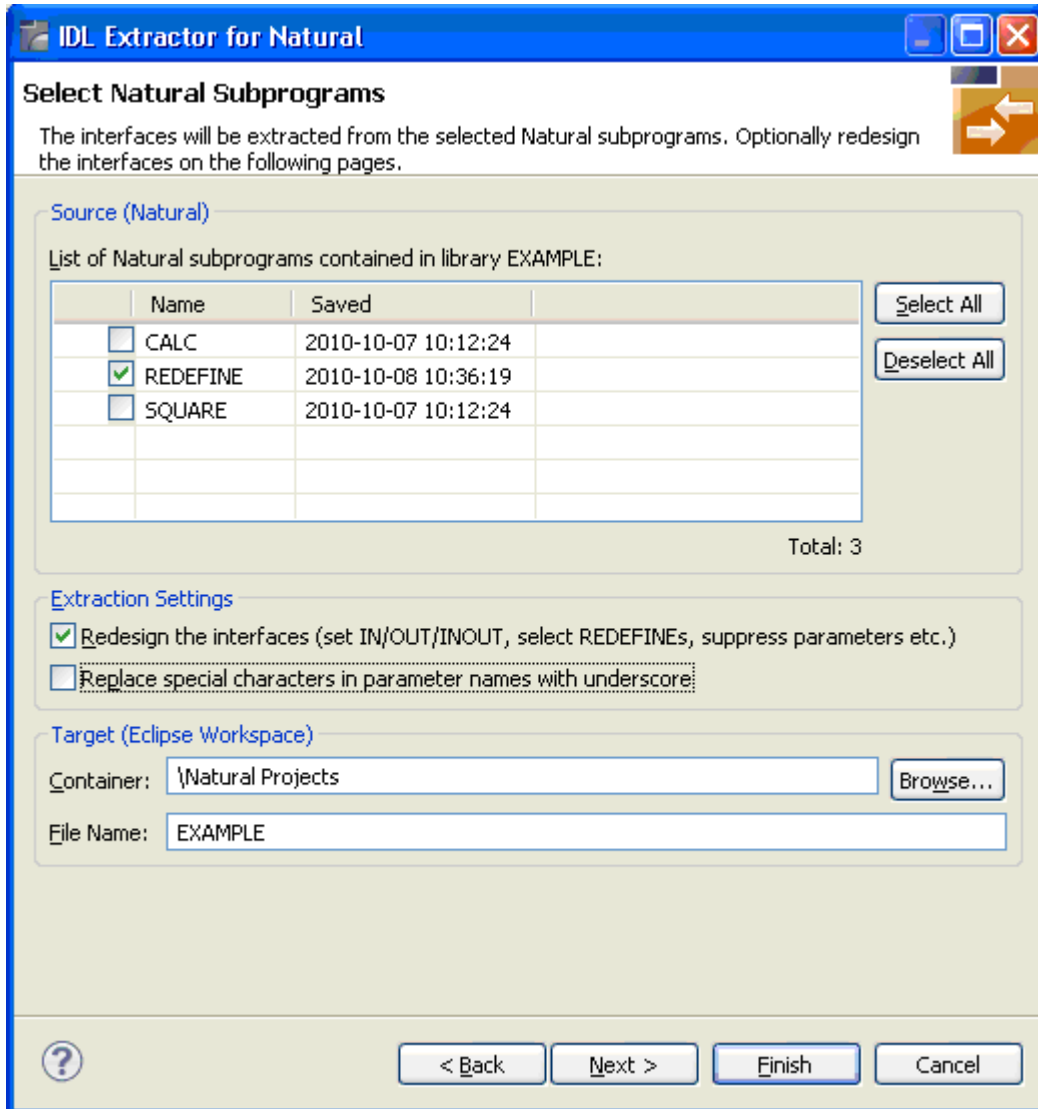
Alternatively, you can start the IDL Extractor for Natural from the context menu of the Natural source folder or any parent folder in the project, including the Natural library and the Project folder.

**Step 2: Select the Natural Library from NaturalONE Project (Optional)**



Select the Natural library from the list and choose **Next** to continue with [Step 3: Select the Natural Subprograms from NaturalONE Project](#).

**Step 3: Select the Natural Subprograms from NaturalONE Project**



In the **Source** pane, select at least one program from the list of Natural subprograms (CALLNATs). You can also choose **Select All** or **Deselect All**.

In the **Extraction Setting** pane, check **Redesign the interfaces** if you want to design the extracted interfaces to the Natural subprograms. The **Next** button will be enabled. See [User-defined Mapping](#). If you do not check **Redesign the interfaces**, see [Natural to IDL Mapping](#) for default mappings.

By checking **Replace special characters in parameter names with underscore** under **Extraction Setting**, the special characters ("\$", "#", "&", "@", "/") allowed in Natural parameters can be replaced by underscores, see also [Extracting IDL Parameter Names](#).

In the **Target** pane, select the container where the IDL file will be stored. Enter the file name of the new IDL file. If the IDL file exists, it will be overwritten. Preserving the IDL File and optional mapping file allows you to customize the IDL (alias names, in/out/inout modifiers) for subsequent optimized client generation steps. See [Server Mapping Files for Natural](#) in the Designer documentation.

Choose **Next** to redesign the interfaces. See [Step 6: Redesign the Interface for Natural Subprograms \(Optional\)](#). To enable the **Next** button, check **Redesign the interfaces**.

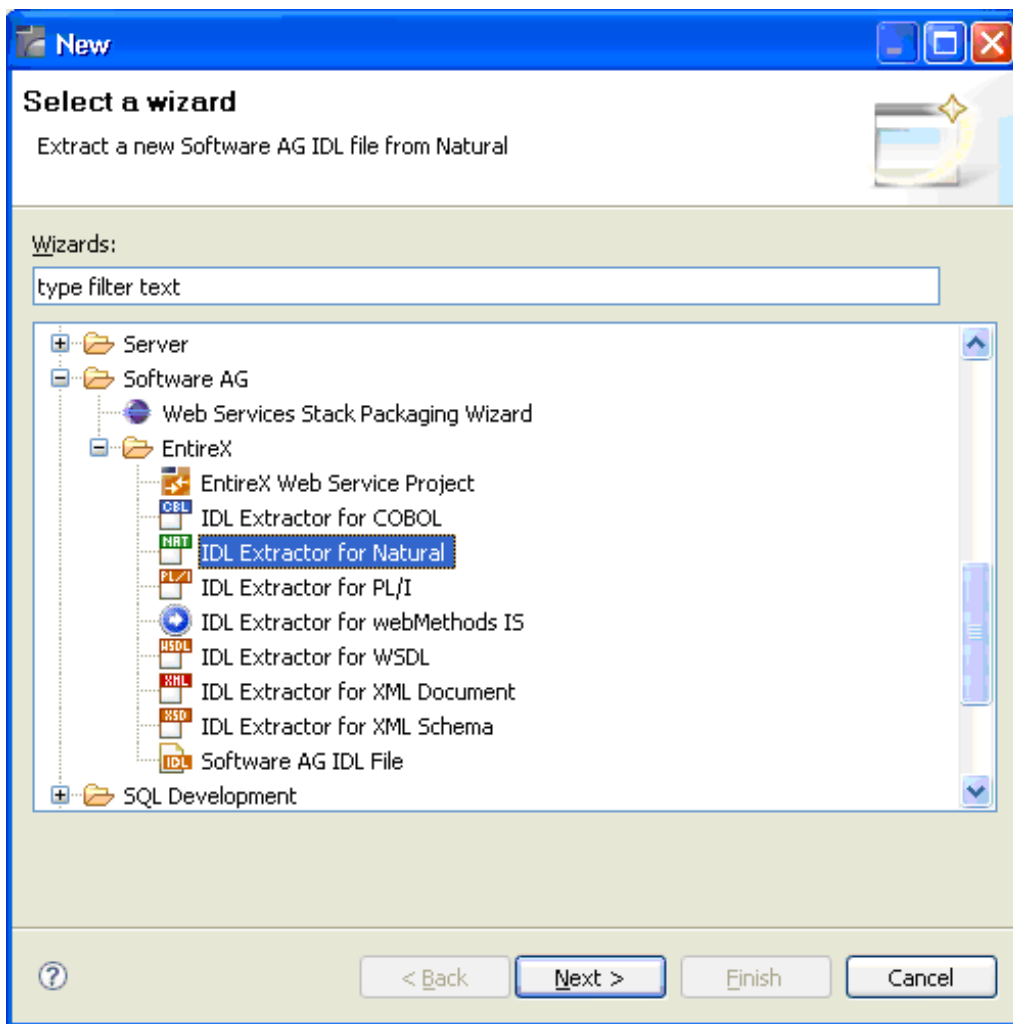
Choose **Finish** to start extraction with a default mapping. For more information see [Extraction Result](#).

## Extracting Software AG IDL File from a New Natural RPC Environment

This section covers the following topics:

- Step 1: Start the IDL Extractor for Natural Wizard
- Step 2: Create a New RPC Environment
- Step 3: Edit RPC Environment
- Step 4: Select Natural Library from RPC Environment (Optional)
- Step 5: Select Natural Subprograms from RPC Environment
- Step 6: Redesign the Interface for Natural Subprograms (Optional)

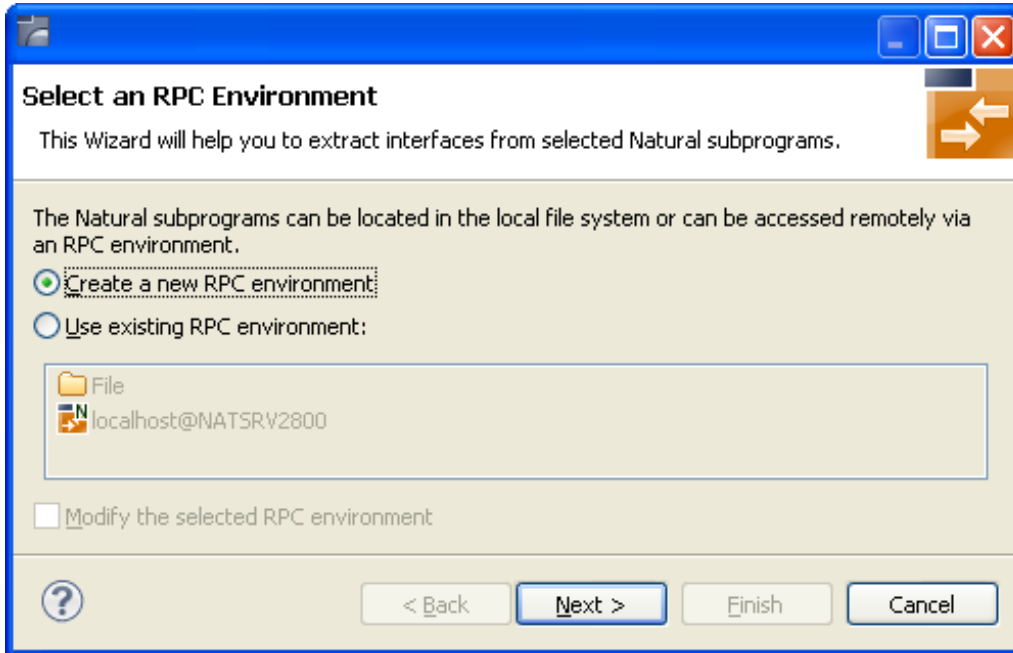
### Step 1: Start the IDL Extractor for Natural Wizard




Press **Next** and continue with [Step 3: Edit RPC Environment](#).

## Step 2: Create a New RPC Environment

If no RPC environments are defined, you only have the option to **Create a New RPC Environment**. The RPC environments are managed in the [Preferences](#).



Choose **Create a new RPC Environment** and press **Next** to create a new RPC environment. Continue with [Step 3: Edit RPC Environment](#).

-  **Note:** The folder **File** and the RPC environment **localhost@NATSRV2800** (default RPC server for NaturalONE) are available only if NaturalONE plugins are installed.

**Step 3: Edit RPC Environment**

**New RPC Environment for Natural RPC Server**  
Define a new RPC Environment.

**Broker Parameters**

Broker ID: myHost:1971

Server Address: \*RPC/NATSRV1/CALLNAT Edit...

Timeout (Seconds): 60

**EntireX Authentication**

User ID:

Password:

**RPC Server Authentication**

RPC User ID:

RPC Password:

**Extractor Settings**

Enter names, or use filter for range of values (wildcards \* and ? on any position, < and > as final character only).

Library Name: NAT\*

Program Name: CO>

**Wrapper Settings**

Save in workspace project

Save in Natural FUSER via RPC

Target Library Name: \*

**Environment Name**

Default (myHost:1971@RPC/NATSRV1/CALLNAT)

Other: myHost:1971@RPC/NATSRV1/CALLNAT

? < Back Next > Finish Cancel

Define the new RPC environment on this page. Required fields are **Broker ID**, **Server Address** and the **Environment Name**. The timeout value must be in the range 1-9999 seconds (default: 60).

The **EntireX Authentication** fields apply to the broker.

The **RPC Server Authentication** fields apply to the RPC server. If the Natural RPC Server operates under Natural Security,

- your user ID and password must be defined in Natural Security. If the Natural Security user ID or password differs from the broker user ID and password, use **RPC Server Authentication** - otherwise use **EntireX Authentication** for both.
- access to the Natural system library SYSIDL is required



**Note:** Users who do not have access rights to the Natural system library SYSIDL are not allowed to extract IDL from your Natural environment.

With **Extractor Settings**, specify the Natural library and program name from which you want to extract. You have the following options to select a range of Natural libraries and programs:

Wildcard	Position	Extract from
* (asterisk)	any	Libraries or program names that match any sequence of characters.
? (question mark)	any	Libraries or program names that match any single character.
> (greater than)	final character only	Libraries or program names after.
< (less than)	final character only	Libraries or program names before.
none		Specified library or program. If the library or program does not exist, an error message is given.

Any Natural RPC Server can be used. Only Natural libraries that reside in the FUSER system file of the Natural RPC Server can be accessed.

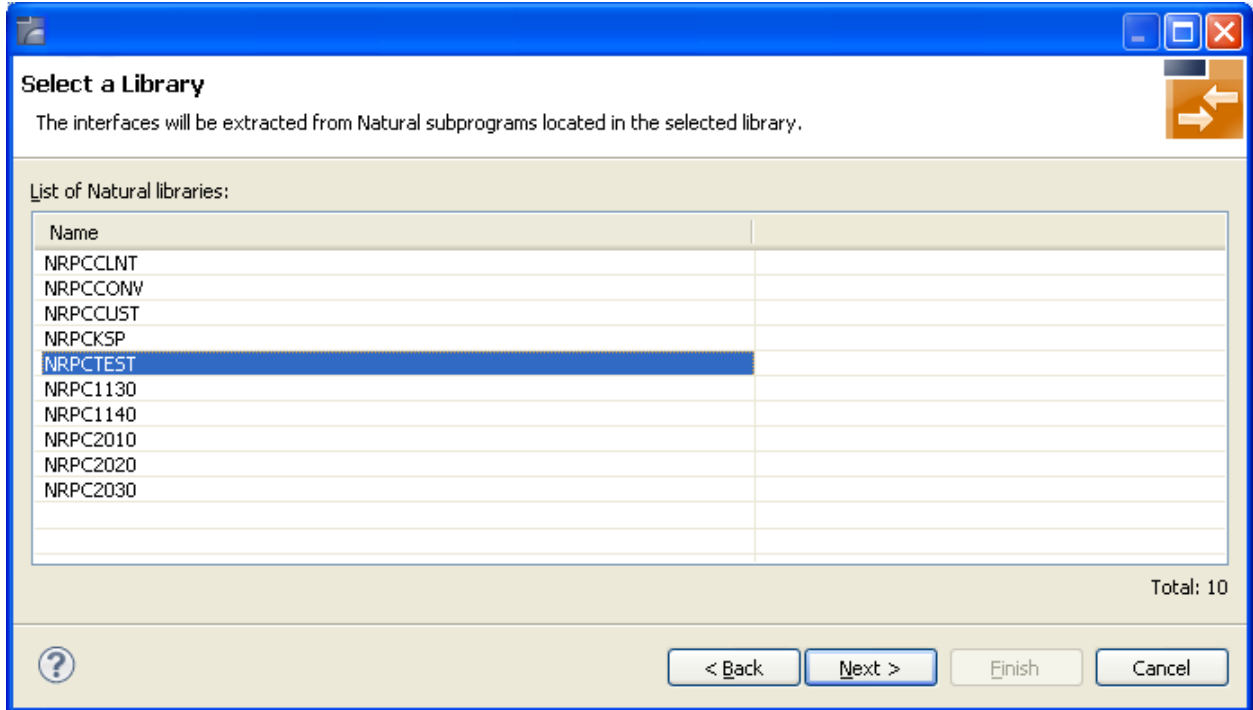
Press **Next** to continue.

- If **Library Name** (using any wildcards) matches one Natural library only, continue with [Step 5: Select Natural Subprograms from RPC Environment](#).
- If **Library Name** matches more than one Natural library, continue with [Step 4: Select Natural Library from RPC Environment \(Optional\)](#).

#### Step 4: Select Natural Library from RPC Environment (Optional)

All Natural libraries that reside in the FUSER system file of the Natural RPC Server and that match the specification in the **Extractor Settings** are listed here. See [Step 3: Edit RPC Environment](#). This step is skipped if exactly one Natural library matches the specification. In this case continue with [Step 5: Select Natural Subprograms from RPC Environment](#).



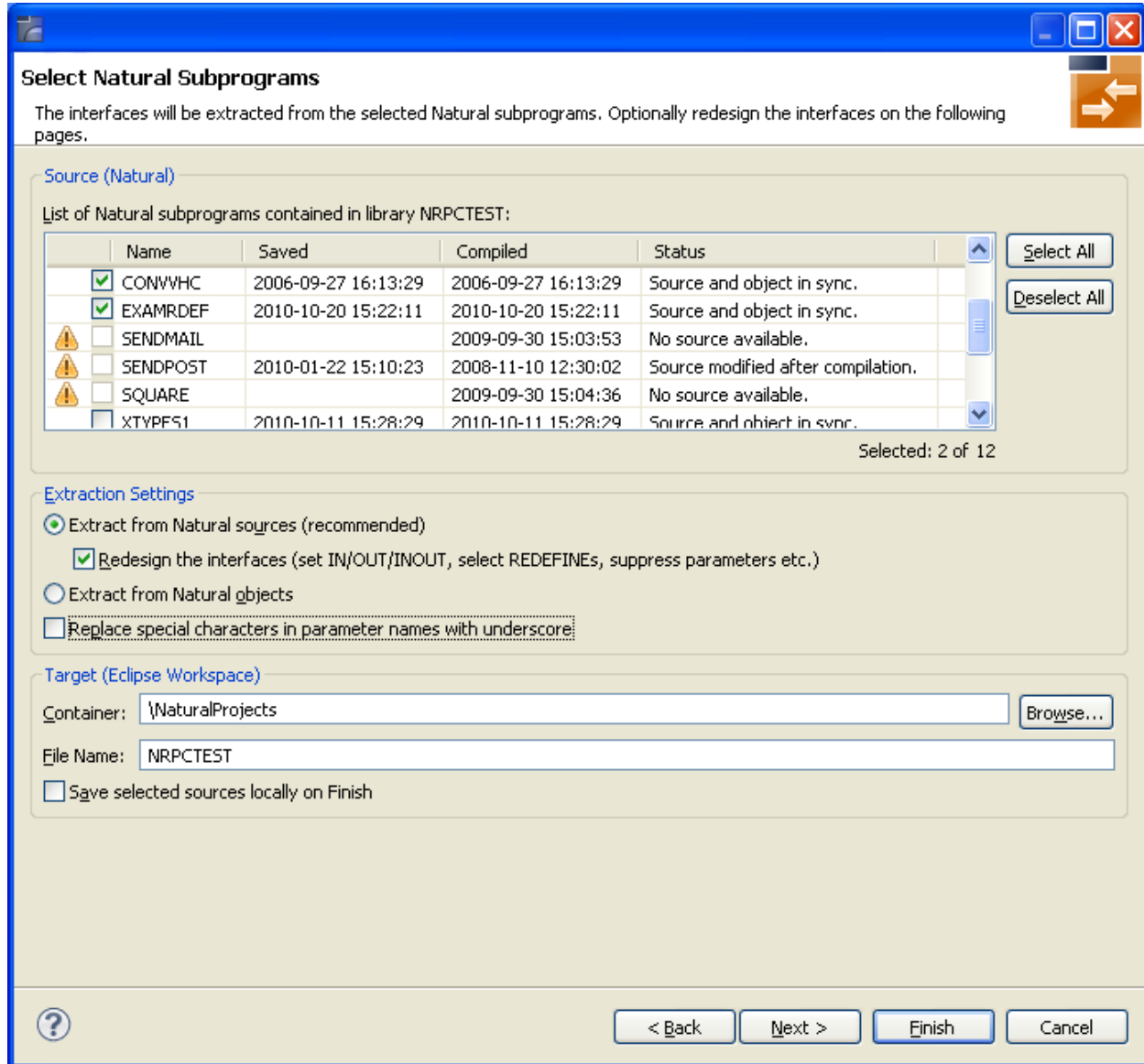


Only Natural libraries from the Natural User System file (*FUSER*) are displayed. If the Natural RPC server operates under Natural Security, Natural libraries are displayed only if you are allowed to access the library. This means that if a library is people-protected and you do not have access rights, it is not displayed. See also **RPC Server Authentication** under [Step 3: Edit RPC Environment](#).

Select the Natural library from the list and choose **Next** to continue with [Step 5: Select Natural Subprograms from RPC Environment](#).

## Step 5: Select Natural Subprograms from RPC Environment

All Natural subprograms that match the specification in the Extractor Settings are listed here. See [Step 3: Edit RPC Environment](#).



In the **Source** pane select at least one program from the list of Natural subprograms (CALLNATs). You can also choose **Select All** or **Deselect All**.

In the **Extraction Settings** pane, specify whether the IDL is to be extracted from Natural subprogram sources or compiled objects. The following restrictions apply:

- **Extract from Natural sources (recommended)**

Extracting from a source is only possible if the compiled object and source are in sync. For example, in the screen above, source extraction is not possible for

- SENDMAIL, because there is no source, only an object
- SENDPOST, because the source is modified after last compilation

- **Extract from Natural objects**

Extracting from an object is always possible.

Check **Redesign the interfaces** if you want to design the extracted interfaces to the Natural subprograms. The **Next** button will be enabled. See [User-defined Mapping](#). If you do not check **Redesign the interfaces**, refer to [Natural to IDL Mapping](#) for default mappings.

Check **Replace special characters in parameter names with underscore** to substitute the special characters '\$', '#', '&', '@', '/' by underscores. See also [Extracting IDL Parameter Names](#) in [Natural to IDL Mapping](#).

Choose **Next** to continue. If multiple Natural subprograms have been selected in the Natural subprogram selection step, redesign the next interface. You can see the current and total number of the subprogram you are extracting from in the title (n/m).

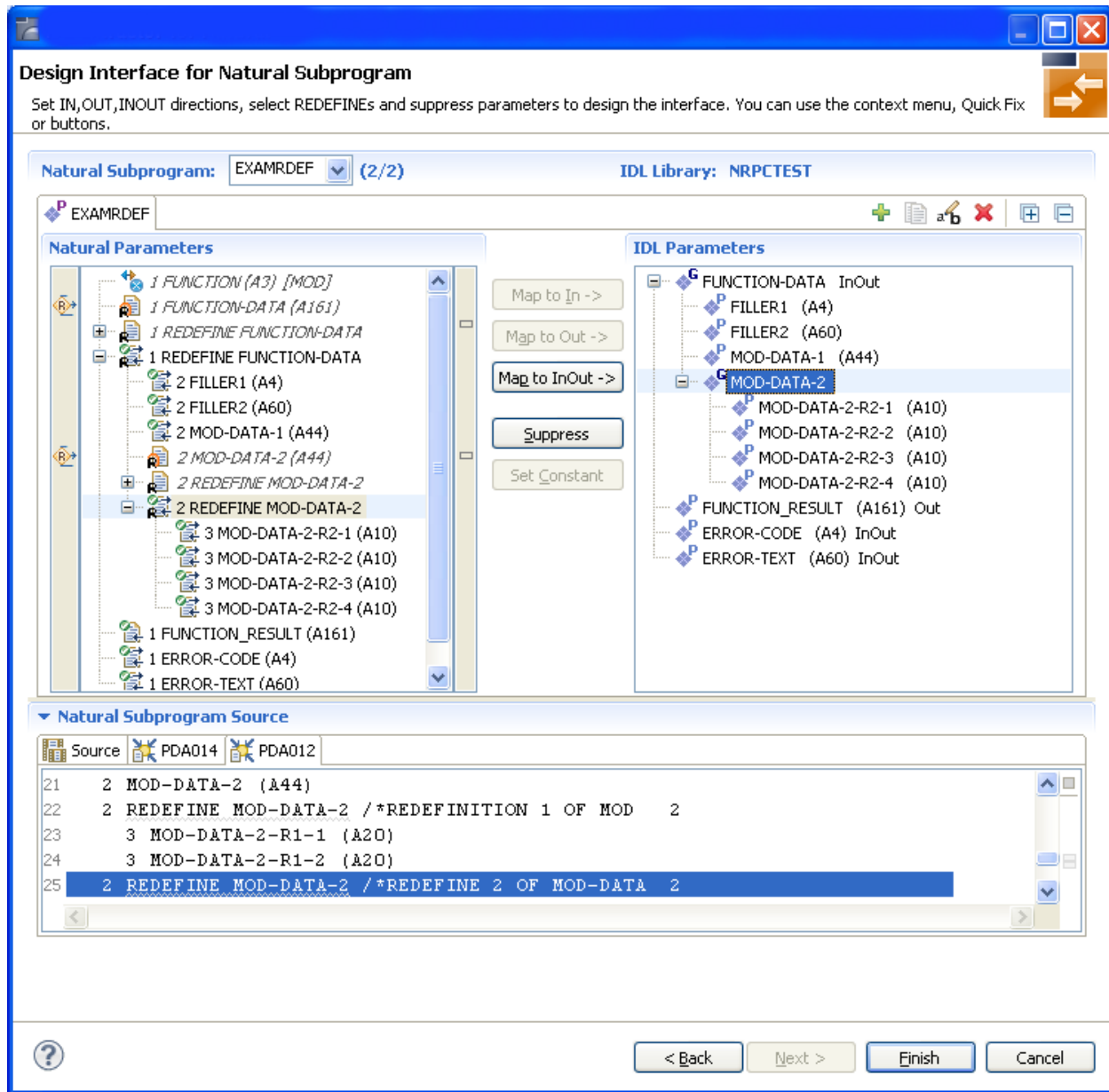
Choose **Finish** to extract the interface. For further subprograms that have been selected, a default mapping is created, see [Extraction Result](#).

## **Step 6: Redesign the Interface for Natural Subprograms (Optional)**

In this step, you can redesign the interface. This includes:

- [Extracting Multiple Interfaces](#)
- [Extracting Natural REDEFINES](#)
- [Extracting IDL Directions \(IN,OUT,INOUT\)](#)
- [Setting Natural Parameters to Constants](#)
- [Suppressing Natural Parameters](#)
- [Renaming a Program](#)

These tasks are described in more detail under [Step 6: Redesign the Interface for Natural Subprograms \(Optional\)](#).



Use this page for the following tasks:

- Define the direction of parameters in the extracted interface. Choose **Map to In**, **Map to Out** or **Map to InOut** for each parameter on level 1.
- Define which parameters redefined in the Natural PDA are part of the extracted interface. Choose **Map to In**, **Map to Out** or **Map to InOut** for the **REDEFINE** base parameter or any **REDEFINE** path.
- Hide or suppress unneeded parameters in the extracted interface. Choose **Suppress**.
- Set parameters to constants and hide or suppress them in the extracted interface. Choose **Set Constant**.

This page consists of the following main parts:

### ■ Top line

The top line contains the current Natural subprogram and the IDL library name. The combo box can be used as quick navigation if more than one Natural subprogram is selected.

### ■ Middle

The middle part contains a tab item for each interface (IDL program) extracted from the Natural subprogram.



**Note:** It is possible to extract more than one interface (IDL program) from a Natural subprogram. To create, rename and remove interfaces, use the toolbar on the right side of tab folder.

Icon	Function	Description
	Create	Create a new interface (IDL program) based on the original parameters of the Natural subprogram.
	Duplicate	Create a new interface (IDL program) based on the current interface (active tab). All modifications of the current interface are copied.
	Rename	Change the name of the current interface (active tab). The name must be unique.
	Remove	Remove the current interface (active tab). At least one interface must exist.
	Expand All	Expand the Natural and IDL tree.
	Collapse All	Collapse the Natural and IDL tree.

### ■ Middle left

Input pane. The parameters of the Natural subprogram to extract from. For each Natural subprogram parameter you can choose one of the operations **Map to In**, **Map to Out**, **Map to InOut**, **Suppress** and **Set Constant**. Additionally for **REDEFINES**, a quick fix is available (icons on the left side of the pane) to choose which parameters redefined in the Natural PDA are part of the extracted interface.



#### Notes:

1. The mapping operations **Map to In**, **Map to Out**, **Map to InOut**, **Suppress** and **Set Constant** are also available in the context menu of the Natural parameter tree.
2. Natural parameters that are suppressed or set to constant in the interface are rendered in italic type. For example, in the screen above, *FUNCTION (A3)* is set to constant; *FILLER1(A4)* and *FILLER2(A60)* are suppressed; *FUNCTION-DATA(A161)* and its first **REDEFINE** path are implicitly suppressed because the second **REDEFINE** path with prefix **MOD-DATA-2-R2** is selected.
3. The value for Natural parameters set to constant are displayed behind the parameter in the Natural parameter tree (e.g. in the screen above, *FUNCTION (A3) [MOD]*).
4. Natural parameters mapped in the interface are displayed with a green tick (☑).

■ **Middle right**

Output pane. The extracted interface (IDL).

■ **Bottom**

Reference. The Natural subprogram source and its PDA sources, each displayed in a separate tab.

**Tips:**

- The panes can be resized.
- To enlarge parameter lists, use the vertical bars on the side.
- You can close the bottom pane if it is not needed by clicking on the triangle next to **Natural Subprogram Source**. In this way, you have more space for viewing the upper panes.

Use the quick navigation or choose **Next** to continue. If multiple Natural subprograms have been selected in the Natural subprogram selection step, redesign the next interface. The amount of subprograms extracted so far is indicated by the fraction next to the title (current/total).

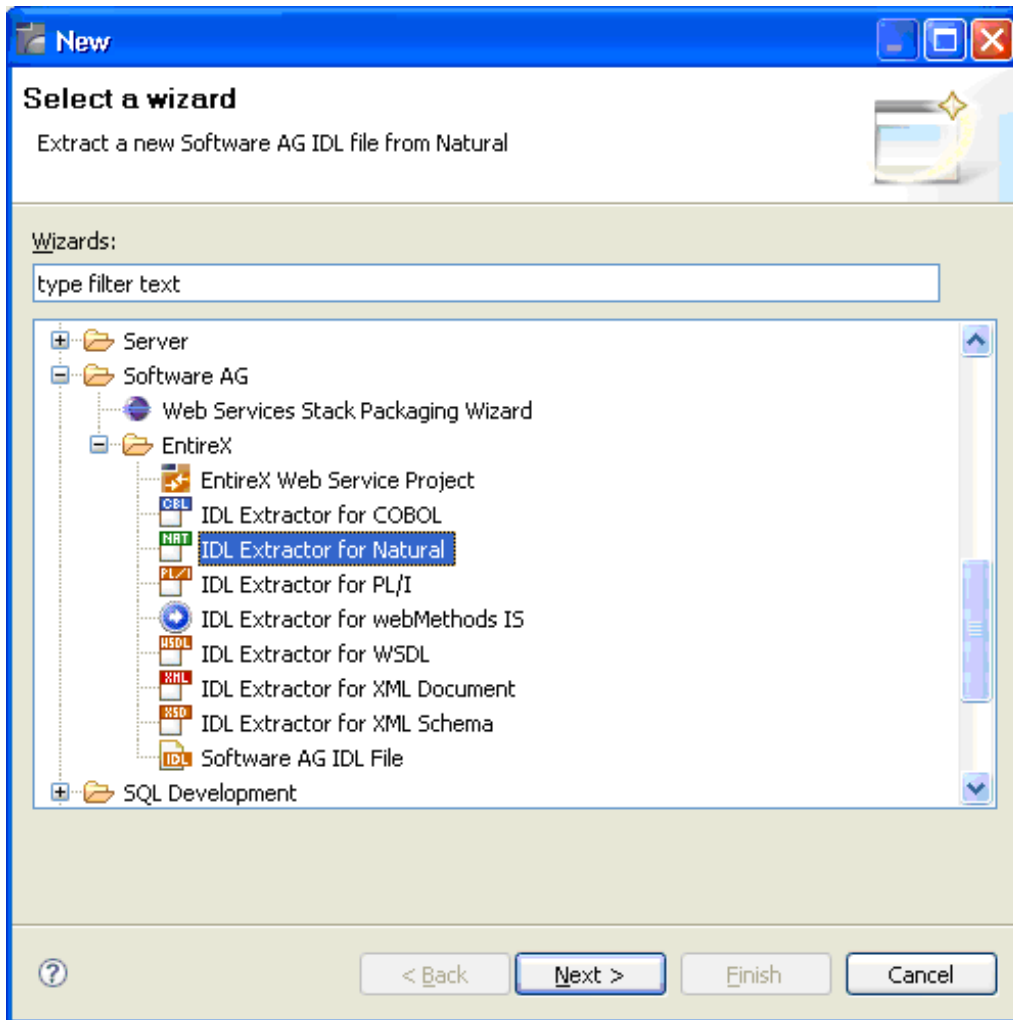
Choose **Next** to continue. If multiple Natural subprograms have been selected in the Natural subprogram selection step, redesign the next interface. You can see the current and total number of the subprogram you are extracting from in the title (n/m).

Choose **Finish** to extract the interface. For further subprograms that have been selected, a default mapping is created, see [Extraction Result](#).

## Extracting Software AG IDL File from an Existing Natural RPC Environment

- Step 1: Start the IDL Extractor for Natural Wizard
- Step 2: Select an RPC Environment

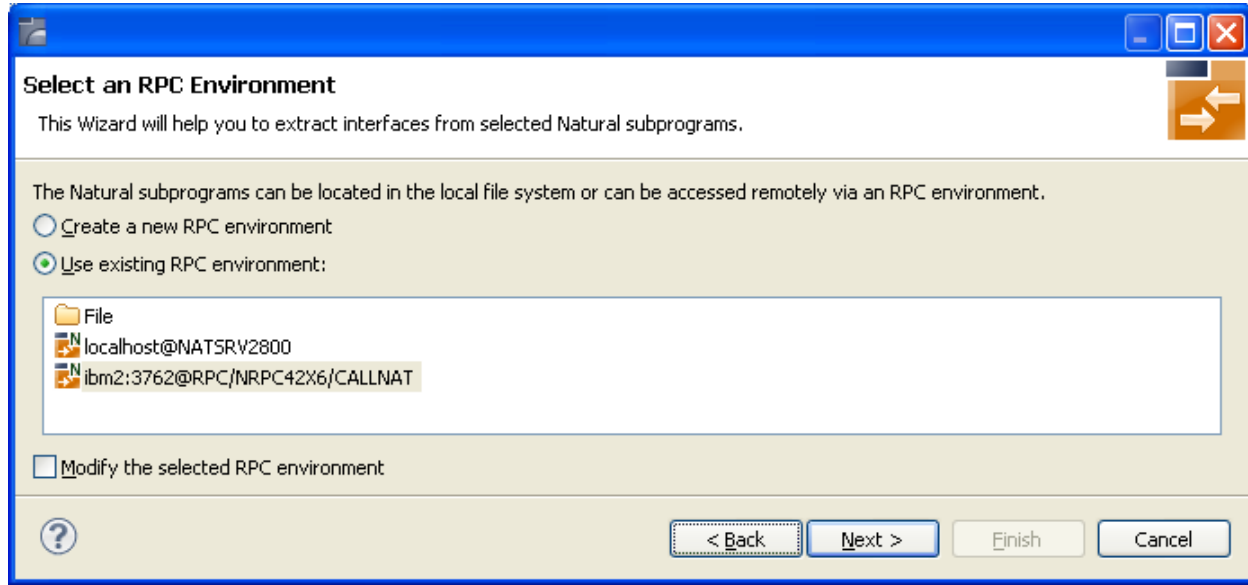
### Step 1: Start the IDL Extractor for Natural Wizard



Choose Next and continue with [Step 6: Redesign the Interface for Natural Subprograms \(Optional\)](#).

## Step 2: Select an RPC Environment

If RPC environments are defined, you can select an existing one. RPC environments are managed in the *Preferences*.



Select **Use existing RPC environment** and select the appropriate RPC environment from the list below.

If **Modify the selected RPC environment** is checked, the selected RPC environment can be modified before extraction starts.

Press **Next** to continue.

- If **Modify the selected RPC environment** is *checked*, continue with [Step 3: Edit RPC Environment](#) above.
- If **Modify the selected RPC environment** is *not checked* and the Natural library name matches *one* Natural library only (using any wildcard defined in [Step 3: Edit RPC Environment](#)), continue with [Step 5: Select Natural Subprograms from RPC Environment](#) above.
- If **Modify the selected RPC environment** is *not checked* and the Natural library name matches *more than one* Natural library, continue with [Step 4: Select Natural Library from RPC Environment \(Optional\)](#) above.



**Note:** The folder **File** and the RPC environment **localhost@NATSRV2800** (default RPC server for NaturalONE) are available only if NaturalONE plug-ins are installed.



## Extraction Result

---

When the wizard has finished successfully, you can see two additional files in your Eclipse project:

- **.idl file**

The IDL file that describes the RPC interface of the Natural server component that implements the service's business logic. The IDL file is opened with the IDL Editor. See *Software AG IDL File* in the IDL Editor documentation in the IDL Editor documentation.

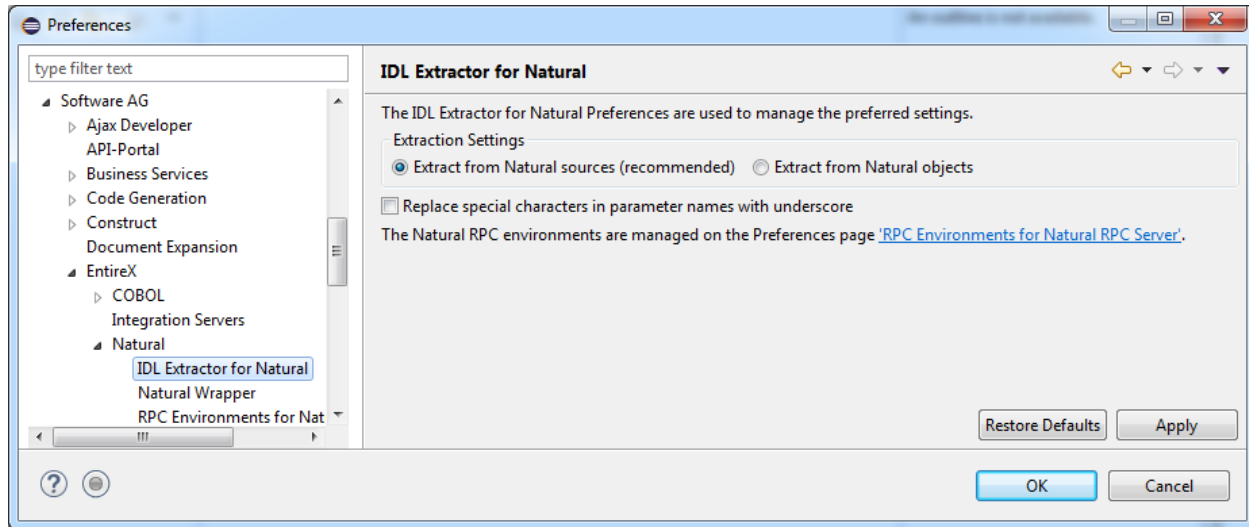
- **.cvm file (optional)**

This server mapping file completes the IDL file with a mapping from the programming-language-neutral parameter definition in the IDL file to the parameters and data types expected by the Natural subprograms (CALLNATs). Always keep the server mapping in the same folder as its related IDL file. See *Server Mapping Files for Natural* in the Designer documentation.

For more information on how Natural programs are extracted, refer to [Natural to IDL Mapping](#).

## Preferences

Use the preference page for IDL Extractor for Natural to manage the default values for the IDL Extractor for Natural wizard.



The option **IDL Extraction from Source** or **from Object** determines from which code type the IDL file is to be extracted.

With the check box **Replace special characters in parameter names with underscore**, the special characters ("\$", "#", "&", "@", "/") allowed in Natural parameters can be replaced by underscores. See also *Rules for Coding Group and Parameter Names* under *Software AG IDL File* in the IDL Editor documentation.

For more information on object and source extraction and character replacing, see [Step 5: Select Natural Subprograms from RPC Environment](#).

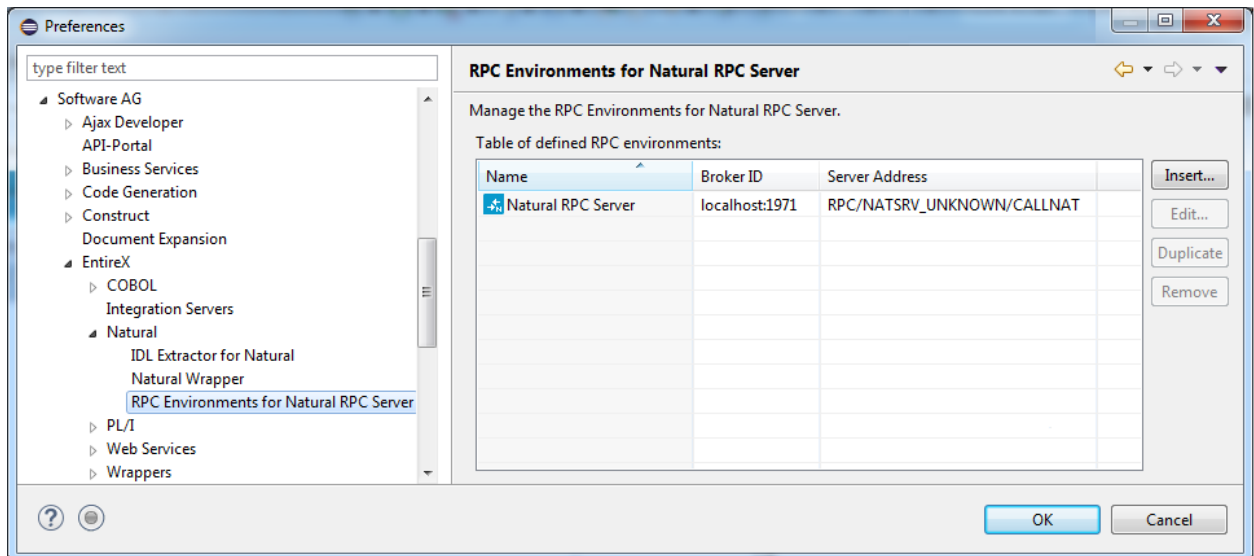
RPC environments are managed with the [RPC Environment Manager](#).

## 4 RPC Environment Manager for Natural RPC Server

The RPC environment for a Natural RPC Server is managed on the **Preferences** page.

Use the *RPC Environment Monitor* to check the availability of each RPC environment.

Using this wizard, you can add new RPC environments for a Natural RPC Server. To manage these RPC environments, open the **Preferences** page.



**Note:** If NaturalONE is installed, a predefined RPC environment is provided for the default Natural server. This RPC environment cannot be removed. The server address of the Natural server is managed by NaturalONE, but other settings can be changed.

### > To edit an existing RPC environment

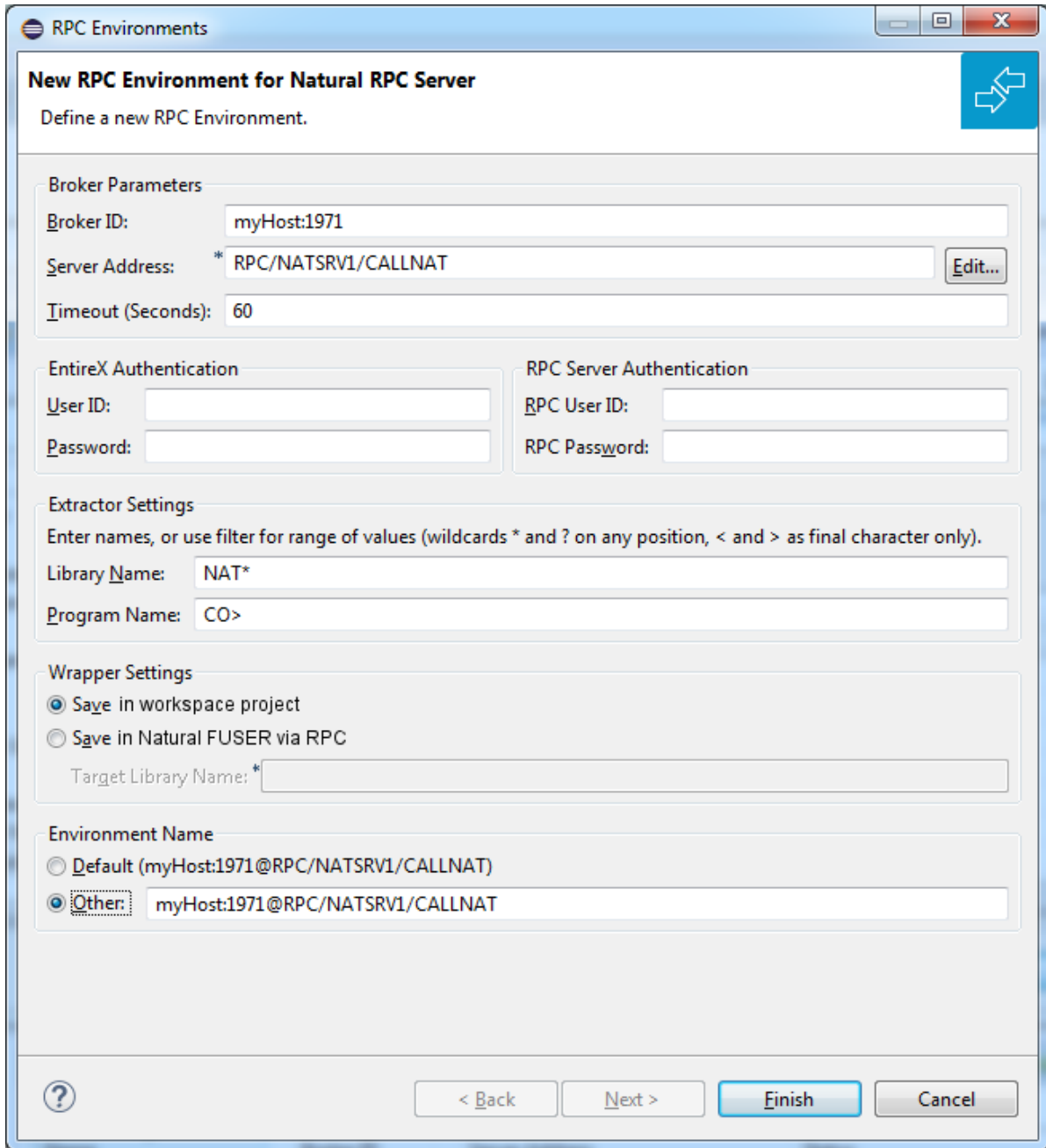
- Select the table row and press **Edit...** If multiple entries are selected, the first entry is used.

> To remove an RPC environment

- Select the table row and press **Remove**. You can select multiple environments.

> To create a new RPC environment

- 1 Choose **Insert...** to call the following screen:



- 2 Enter the required fields: **Broker ID**, **Server Address** and a unique **Environment Name**, which will have the default format *brokerID@serverAddress*. The given **Timeout** value must be in the range from 1 to 9999 seconds (default: 60).

**EntireX Authentication** describes the settings for the broker, and **RPC Server Authentication** describes the settings for the RPC server.

The **Wrapper Settings** are used for Natural Wrapper. You can specify the operation type “Save in workspace project” (save locally) or “Save in Natural FUSEr via RPC” (save remotely). If you save remotely you can specify the target library name.



# 5 RPC Environment Monitor

The RPC Environment Monitor is part of the Software AG Designer. It is an Eclipse view that provides a quick overview of the availability of the defined RPC environments in your workspace.

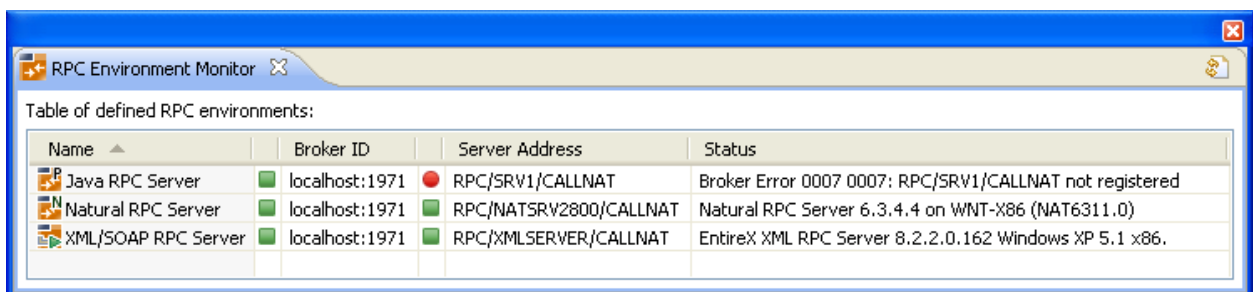
➤ To open the RPC Environment Monitor from the EntireX perspective

- Choose **Window > Show View > RPC Environment Monitor**.

➤ To open the RPC Environment Monitor from a non-EntireX perspective

- Choose **Window > Show View > Other > Software AG > RPC Environment Monitor**.




The RPC environments are managed on the **Preferences** page. See [RPC Environment Manager for Natural RPC Server](#).



The screenshot shows the 'RPC Environment Monitor' window with a table titled 'Table of defined RPC environments:'. The table has four columns: Name, Broker ID, Server Address, and Status. It lists three RPC servers: Java RPC Server, Natural RPC Server, and XML/SOAP RPC Server. The Java RPC Server is marked with a red circle, indicating an error. The Natural RPC Server and XML/SOAP RPC Server are marked with green squares, indicating they are available.

Name	Broker ID	Server Address	Status
Java RPC Server	localhost:1971	RPC/SRV1/CALLNAT	Broker Error 0007 0007: RPC/SRV1/CALLNAT not registered
Natural RPC Server	localhost:1971	RPC/NATSRV2800/CALLNAT	Natural RPC Server 6.3.4.4 on WNT-X86 (NAT6311.0)
XML/SOAP RPC Server	localhost:1971	RPC/XMLSERVER/CALLNAT	EntireX XML RPC Server 8.2.2.0.162 Windows XP 5.1 x86.

The status check starts when the view is opened. To force an additional check, choose **Refresh** from the **Views** toolbar. The status check can be cancelled in the dialog that appears or within the Eclipse progress view. When the check is complete or if it cancelled, the following symbols indicate the status of the corresponding item. The table will be reloaded every time a status check is started to make sure all stored RPC environments are available.

Symbol	Status
	Running.
	Not running.
	Unknown (at the beginning of the check or if the check was cancelled).



**Note:** Additional status information (including error messages) is displayed when refreshing the view (by a ping command to all specified RPC servers).



# 6 Using the IDL Extractor for Natural in Command-line Mode

---

- Command-line Options ..... 38
- Example ..... 39

## Command-line Options

See *Using EntireX in the Designer Command-line Mode* for the general command-line syntax. The table below shows the command-line options for the IDL Extractor for Natural.

Task	Command	Option	Description
Extract the Natural objects from a Natural RPC server.	-extract:natural	-brokerpassword	Password used for broker authentication.
		-brokeruser	User used for broker authentication.
		-environment	Name of the environment or an RPC server description.
		-filter	Filter the Natural source objects. Show those objects which match the pattern.
		-help	Display this usage message.
		-library	The library defined within the Natural RPC Server.
		-object	A Natural object in the library.
		-project	Name of the project or subfolder where the IDL file is stored.
		-rpcpassword	Password used for RPC server authentication.
		-rpcuser	User used for RPC server authentication.
		-timeout	Timeout in seconds (default is 60).
List the Natural objects from a Natural RPC server.	-list:natural	-brokerpassword	Password used for broker authentication.
		-brokeruser	User used for broker authentication.
		-environment	Name of the environment or an RPC server description.
		-filter	Filter the Natural source objects. Show those objects which match the pattern.
		-help	Display this usage message.
		-library	The library defined within the Natural RPC Server.
		-object	A Natural object in the library.
		-rpcpassword	Password used for RPC server authentication.
		-rpcuser	User used for RPC server authentication.
		-timeout	Timeout in seconds (default is 60).

## Example

---

```
<workbench> -extract:natural -environment natBroker:2006@RPC/NATSRV1/CALLNAT -project ↵  
/Demo -library NATDEMO -object CALC
```

where *<workbench>* is a placeholder for the actual EntireX design-time starter as described under *Using EntireX in the Designer Command-line Mode*.

The extracted IDL file will be stored in the project *Demo*.

If the environment name is not a defined RPC Environment in the current workspace, the name will be interpreted as a Broker ID and RPC Server Address (*brokerID@serverAddress*).

The library specifies a library name and the optional object defines the program name. Wildcard notation with asterisk (\*) can be used at the end of these names.

Status and processing messages are written to standard output (stdout), which is normally set to the executing shell window.



# 7 Natural to IDL Mapping

---

▪ Mapping Natural Data Types to Software AG IDL .....	42
▪ User-defined Mapping .....	43
▪ Extracting the IDL Library Name .....	43
▪ Extracting the IDL Program Name .....	43
▪ Extracting IDL Parameter Names .....	43
▪ Extracting IDL Directions (IN,OUT,INOUT) .....	44
▪ Extracting Natural REDEFINES .....	45
▪ Extracting Multiple Interfaces .....	45
▪ Extracting Natural Arrays, Groups, X-Arrays and Variable Arrays .....	46
▪ Extracting Natural Structure Information (IDL Levels) .....	48
▪ Extracting Parameters defined with OPTIONAL .....	49
▪ Setting Natural Parameters to Constants .....	49
▪ Suppressing Natural Parameters .....	49
▪ Renaming a Program .....	50

This chapter describes how Natural data types are mapped to Software AG IDL files by the Software AG IDL Extractor for Natural.

For more information on Natural syntax, refer to the Natural documentation.

## Mapping Natural Data Types to Software AG IDL

The IDL Extractor for Natural maps the following subset of Natural data types to Software AG IDL data types.

In the table below, the following metasympols and informal terms are used for the IDL.

- The metasympols "[" and "]" enclose optional lexical entities.
- The informal term *number* (or in some cases *number1.number2*) is a sequence of numeric characters, for example 123.

Natural Data Type	Software AG IDL Data Type	Description
<i>Anumber</i>	<i>An</i>	Alphanumeric
A DYNAMIC	<i>AVn</i>	Alphanumeric variable length
<i>Bnumber</i>	<i>Bnumber</i>	Binary
B DYNAMIC	<i>BV</i>	Binary variable length
C	not supported	
D	D	Date
F4	F4	Floating point (small)
F8	F8	Floating point (large)
I1	I1	Integer (small)
I2	I2	Integer (medium)
I4	I4	Integer (large)
L	L	Logical
<i>Nnumber[.number]</i>	<i>Nnumber[.number]</i>	Unpacked decimal
<i>Pnumber[.number]</i>	<i>Pnumber[.number]</i>	Packed decimal
T	T	Time
Unumber	Unumber	Unicode
U DYNAMIC	<i>UV</i>	Unicode variable length

---

## User-defined Mapping

---

Depending on the Natural syntax and the Natural server implementation, user interaction may be required to get correct extraction results. User interaction can also simplify or modernize the extracted IDL. As a result, the user-defined mapping is contained in a Designer file with extension `.cvm` that contains Natural-specific mapping information. See *Server Mapping Files for Natural* in the Designer documentation. The following is covered:

- *Extracting Multiple Interfaces*
- *Extracting Natural REDEFINES*
- *Extracting IDL Directions (IN,OUT,INOUT)*
- *Setting Natural Parameters to Constants*
- *Suppressing Natural Parameters*
- *Renaming a Program*

See *Step 6: Redesign the Interface for Natural Subprograms (Optional)* for more information.

---

## Extracting the IDL Library Name

---

The Natural library from where Natural programs are extracted is used as the IDL library name. See `library-definition` under *Software AG IDL Grammar* in the IDL Editor documentation.

---

## Extracting the IDL Program Name

---

The Natural program name is used as the IDL program name, see `program-definition` under *Software AG IDL Grammar* in the IDL Editor documentation.

---

## Extracting IDL Parameter Names

---

For **source extractions**, Natural parameter names are kept and used as IDL parameters, see `simple-parameter-definition` under *Software AG IDL Grammar* in the IDL Editor documentation and `group-parameter-definition` under *Software AG IDL Grammar* in the IDL Editor documentation.

For **object extractions**, Natural programs must be compiled (cataloged) with the compiler option SYMGEN=ON to keep original Natural parameter names. Otherwise, generic parameter names are generated (PARAMETER-1, PARAMETER-2, etc.).

In **Select Natural Sources** (see [Step 3: Select the Natural Subprograms from NaturalONE Project](#) if you are extracting from NaturalONE projects or [Step 5: Select Natural Subprograms from RPC Environment](#) if you are extracting from a Natural RPC environment), you can choose special characters (\$, #, &, @, /, £, æ, Æ, ø, Ø, å, Å) in Natural parameter names to be replaced by under-scores. See *Rules for Coding Group and Parameter Names* under *Software AG IDL File* in the IDL Editor documentation.

## Extracting IDL Directions (IN,OUT,INOUT)

In most Natural subprograms, parameters have no specification for a direction. Missing a direction is unproblematic for local calls. For remote RPC calls, however, specifying the direction helps to reduce data sizes.

If you redesign the interface, you can define IDL directions in [Step 6: Redesign the Interface for Natural Subprograms \(Optional\)](#) using the mapping operations **Map to In**, **Map to Out**, **Map to InOut**.

Otherwise, IDL directions can be inserted at top-level parameters (level 1) using a Natural line comment in the Natural subprogram (CALLNAT) interface definition (DEFINE DATA PARAMETER), example:

```
DEFINE DATA PARAMETER
1 #IN-FIELD-1          (P9) /* IN
1 #OUT-FIELD-1        (P9) /* OUT
1 #INOUT-FIELD-1     (P9) /* INOUT
1 #INOUT-FIELD-2     (P9)
1 #IN-GROUP-1        /* IN
  2 #IN-GROUP-FIELD-1 (A10)
1 #OUT-GROUP-1        /* OUT
  2 #OUT-GROUP-FIELD-1 (A10)
1 #INOUT-GROUP-1     /* INOUT
  2 #INOUT-GROUP-FIELD-1 (A10)
1 #INOUT-GROUP-2
  2 #INOUT-GROUP-FIELD-2 (A10)
1 #INOUT-GROUP-3
  2 #INOUT-GROUP-FIELD-3 (A10) /* OUT
END-DEFINE
```

If no direction is specified (such as in #INOUT-FIELD-2 and #INOUT-GROUP-2 in the example above), the default direction INOUT applies.



Specifications on a level greater than 1 (such as `#INOUT-GROUP-FIELD-3` in the example above) are ignored. Note that in IDL directions are specified on top-level fields (level 1), see `attribute-list` under *Software AG IDL Grammar* in the IDL Editor documentation.

Specifications on IDL directions are only considered when extracting from a source. If you are extracting from an object (compiled), as described in [Step 5: Select Natural Subprograms from RPC Environment](#), the default direction `INOUT` always applies.

## Extracting Natural REDEFINES

A redefinition is a second parameter layout of the same memory portion. The parameter `#BASE-FIELD` is redefined by the fields `FILLER-1` thru `R-P3-01`.

```
DEFINE DATA PARAMETER
1 #BASE-FIELD          (A161)
1 REDEFINE #BASE-FIELD
  2 FILLER-1          (A4)
  2 FILLER-2          (A60)
  2 R-P1-01           (A1)
  2 R-P2-01           (A10)
  2 R-P3-01           (I4)
END-DEFINE
```

With the extractor wizard you can select a single redefine path for IDL usage (here the fields `FILLER-1` thru `R-P3-01`) if you redesign the interface. See [Extracting Natural REDEFINES](#) and [Step 6: Redesign the Interface for Natural Subprograms \(Optional\)](#).

## Extracting Multiple Interfaces

Legacy Natural subprograms often implement multiple functions in a single Natural subprogram. The function executed is often controlled by a so-called function code or operation-code field. See [Extracting Multiple Interfaces](#) under *Examples for Redesigning the Extracted Interfaces*.

With the extractor wizard you can extract the functions from the server as separate interfaces (IDL programs). In this way, the legacy server with a single physical interface can be

- turned into a web service with operations, where the legacy functions match operations.
- called with an object-oriented wrapper such as the Java Wrapper, the .NET Wrapper or the DCOM Wrapper, where the legacy functions match methods.

Note that every function in the Natural subprogram may have a different interface described with `REDEFINE` syntax. Therefore, multiple interface extraction is often combined with [Extracting Natural REDEFINES](#).

For more information, see [Step 6: Redesign the Interface for Natural Subprograms \(Optional\)](#).

## Extracting Natural Arrays, Groups, X-Arrays and Variable Arrays

---

This section describes IDL mapping for Natural arrays and groups:

- [Arrays and Groups with Fixed upper Limits](#)
- [X-Arrays and X-Groups](#)
- [Variable Arrays and Variable Groups](#)
- [Arrays and Groups with Mixed Dimensions \(X, Variable and Fixed\)](#)

### Arrays and Groups with Fixed upper Limits

Ordinary Natural arrays and groups with fixed/bound upper limits are mapped to Software AG IDL fixed-bound-array definitions, see `array-definition` under *Software AG IDL Grammar* in the IDL Editor documentation.

#### Natural syntax example:

```
DEFINE DATA PARAMETER
1 #ARRAY1 (I4/1:10) /* lower bound is fixed at 1, upper bound is 10
1 #ARRAY2 (A5/10) /* shortcut for (A5/1:10)
1 #GROUP1 (10) /* shortcut for (1:10)
  2 #FIELD1 (I2)
  2 #FIELD2 (A10)
. . .
END-DEFINE
```

### X-Arrays and X-Groups

For X-arrays (eXtensible arrays) the number of occurrences is flexible at runtime. The number of occurrences can be resized, that is, increased or reduced. It is defined by specifying an asterisk (\*) for index bounds.

#### Natural syntax example:

```
DEFINE DATA LOCAL
1 #X-ARRAY1 (A5/1:*) /* lower bound is fixed, upper bound is variable
1 #X-ARRAY2 (A5/*) /* shortcut for (A5/1:*)
. . .
END-DEFINE
```

Natural X-arrays are mapped to Software AG IDL unbounded-array definitions, see `array-definition` under *Software AG IDL Grammar* in the IDL Editor documentation.

Natural X-arrays with variable lower bounds are *not* supported by Software AG RPC technology, example:

```
DEFINE DATA PARAMETER
1 #X-ARRAY1 (A5/*:10) /* lower bound is variable, upper bound is fixed
. . .
END-DEFINE
```

### Variable Arrays and Variable Groups

In a Natural parameter data area (PDA), you can specify an array or group with a variable number of occurrences. This is done with the index notation 1:V. The maximum number of occurrences for such an array is either passed to the subprogram using an extra parameter such as #ARRAY1-LIMIT (see example below), or it can be accessed using the system variable \*OCCURRENCE.

#### Natural syntax example:

```
DEFINE DATA PARAMETER
1 #ARRAY1-LIMIT (I4) /* extra parameter to pass the upper limit
1 #ARRAY1      (I4/1:V)
. . .
END-DEFINE
```

Natural variable arrays are mapped to Software AG IDL unbounded-array definitions, see `array-definition` under *Software AG IDL Grammar* in the IDL Editor documentation.

If the Natural server program uses a separate parameter such as #ARRAY1-LIMIT (see the example above) instead of \*OCCURRENCE to determine the upper bound limit, it is required to extract this extra parameter, too. At runtime, it is also required to specify the number of occurrences in a calling RPC client.

In a Natural server program, Natural variable arrays

- cannot be resized for direction INOUT, which means you can only reply the same number of occurrences to the RPC client.
- cannot be used for direction OUT either, because they cannot be created (instantiated). You may get error 10204008 during extraction.

## Arrays and Groups with Mixed Dimensions (X, Variable and Fixed)

Natural arrays and groups with a mixture of fixed variable and eXtensible dimensions are *not* supported by Software AG RPC technology, example:

```
DEFINE DATA PARAMETER
1 #ARRAY1 (I4/1:10,1:*) /* first dimension fixed and second eXtensible
1 #ARRAY2 (I4/1:10,1:V) /* first dimension fixed and second variable
1 #ARRAY3 (I4/1:V,1:*) /* first dimension variable and second eXtensible
. . .
END-DEFINE
```

## Extracting Natural Structure Information (IDL Levels)

---

### Source Extractions

Natural levels are always kept. This means that the structure in the extracted IDL is the same as in the original Natural program.

### Object Extractions

#### ■ UNIX or Windows

In UNIX or Windows RPC environments, Natural levels are *not* kept. The IDL is extracted in a flat way, where

- all IDL parameters are at level 1
- all Natural groups are removed
- Natural fields within groups using repetition (PERIODIC GROUPS) are mapped to IDL arrays
- the dimension of Natural arrays within groups using repetition (PERIODIC GROUPS) is increased in the IDL. For example, a one-dimensional array may become a two-dimensional or three-dimensional IDL array depending on the dimension of the group

#### ■ z/OS

In z/OS RPC environments, the Natural programs must be compiled (cataloged) with the compiler option SYMGEN=ON to keep Natural levels, otherwise flat extraction is carried out.

---

## Extracting Parameters defined with OPTIONAL

---

For a parameter defined without `OPTIONAL`, a value must be passed from the invoking Natural object, that is, the caller.

For a parameter defined with `OPTIONAL`, a value can, but need not be passed from the invoking Natural object to this parameter. With the `SPECIFIED` option, a Natural server can find out at runtime whether an optional parameter has been defined or not.

The IDL Extractor for Natural ignores the `OPTIONAL` specification, that is, the parameter is extracted as without the `OPTIONAL` specification. See the Natural documentation for more information.

EntireX RPC technology does *not* support optional IDL parameters. Using pure Natural RPC (Natural client to Natural server), Natural optional parameters are supported.

---

## Setting Natural Parameters to Constants

---

Setting parameters to constant values and suppressing them in the IDL is part of the redesign process of the extracted interface. This keeps the IDL client interface lean. See [Set Constant](#) under *Examples for Redesigning the Extracted Interfaces*.

EntireX and Natural RPC make sure the constant value is passed to the Natural server at runtime.

For more information, see [Step 6: Redesign the Interface for Natural Subprograms \(Optional\)](#).

---

## Suppressing Natural Parameters

---

Hiding or suppressing unneeded parameters in the IDL is part of the redesign process of the extracted interface. This keeps the IDL client interface lean and minimizes the amount of data to be transferred during runtime.

EntireX and Natural RPC make sure to provide low values as input for suppressed parameters to the Natural server called (blank for IDL type `A`, zero for numeric data types such as IDL `I`, `N` and `P`). No data is transferred between an RPC client and the RPC server.

For more information, see [Step 6: Redesign the Interface for Natural Subprograms \(Optional\)](#).

## Renaming a Program

---

Renaming a program to a different name in the IDL is part of the redesign process of the extracted interface. You can adjust the short Natural name to a meaningful longer name for better readability. See [Extracting Multiple Interfaces](#) where the original Natural name `CALC` is renamed to IDL names `ADD`, `SUBTRACT`, `MULTIPLY` etc.

EntireX and Natural RPC make sure the original Natural server is called at runtime. For more information, see [Step 6: Redesign the Interface for Natural Subprograms \(Optional\)](#).

# 8

## Examples for Redesigning the Extracted Interfaces

---

- Set Constant ..... 52
- Extracting Multiple Interfaces ..... 54
- Extracting Natural REDEFINES ..... 56
- Extracting a Natural Server with User-defined Mapping ..... 57

## Set Constant

This example turns the CALC server into an ADD server. The CALC server used here can be found in *Basic RPC Server Examples - CALC, SQUARE*.

```
*****
*
*   CALC RPC Server Example for Natural
*
*****
DEFINE DATA
PARAMETER
  1 #OPERATION          (A1)
  1 #OPERAND1           (I4)
  1 #OPERAND2           (I4)
  1 #FUNCTION-RESULT    (I4)
LOCAL
  1 #TMP                (I4)
END-DEFINE
*
MOVE 0 TO #FUNCTION-RESULT
DECIDE ON FIRST VALUE OF #OPERATION
  VALUE '+'
    COMPUTE #FUNCTION-RESULT = #OPERAND1 + #OPERAND2
  VALUE '-'
    COMPUTE #FUNCTION-RESULT = #OPERAND1 - #OPERAND2
  VALUE '*'
    COMPUTE #FUNCTION-RESULT = #OPERAND1 * #OPERAND2
  VALUE '/'
    IF #OPERAND2 NE 0 THEN
      COMPUTE #FUNCTION-RESULT = #OPERAND1 / #OPERAND2
    END-IF
  VALUE '%'
    IF #OPERAND2 NE 0 THEN
      DIVIDE #OPERAND2 INTO #OPERAND1 GIVING #TMP
      REMAINDER #FUNCTION-RESULT
    END-IF
  NONE VALUE
    IGNORE
END-DECIDE
END
```

Extraction without a redesign leads to an interface (IDL program) in which extracted parameters and Natural subprogram parameters are identical:



```

Library 'EXAMPLE' Is
  Program 'CALC' Is
    Define Data Parameter
      1 Operation      (A1) In
      1 Operand1      (I4) In
      1 Operand2      (I4) In
      1 Function_Result (I4) Out
    End-Define

```

Calling such an interface with an RPC client means providing the ADD function as a value on the operation parameter in the same way legacy callers do.

After the redesign, the IDL looks as follows:

```

Library 'EXAMPLE' Is
  Program 'ADD' Is
    Define Data Parameter
      1 Operand1      (I4) In
      1 Operand2      (I4) In
      1 Function_Result (I4) Out
    End-Define

```

This process is called “Redesigning the interface”, which can be seen as manual step in the overall extraction process. See [Step 6: Redesign the Interface for Natural Subprograms \(Optional\)](#).

Setting parameters to constant values and suppressing them in the IDL is often done together with multiple interfaces, where the function code or operation-code field has to be set to a constant. See [Extracting Multiple Interfaces](#) for a more advanced example.

#### ➤ To redesign the interface and to set a constant for the parameter operation

- 1 Make sure to check **Redesign the interfaces** in [Step 5: Select Natural Subprograms from RPC Environment](#).
- 2 In [Step 6: Redesign the Interface for Natural Subprograms \(Optional\)](#), proceed as follows:

##### **Design interface for ADD Server**

- Choose **Set Constant** for parameter Operation and enter "+" as value.
- Choose **Rename** from the toolbar and change the name to "ADD".

## Extracting Multiple Interfaces

---

Multiple functions are often implemented in a single Natural subprogram, for example in [Set Constant](#). The CALC server implements the functions ADD, SUBTRACT, MULTIPLY, DIVIDE and MODULO in one source.

Extraction without a redesign leads to an interface (IDL program) in which extracted parameters and Natural subprogram parameters are identical:

```
Library 'EXAMPLE' Is
  Program 'CALC' Is
    Define Data Parameter
      1 Operation      (A1) In
      1 Operand1      (I4) In
      1 Operand2      (I4) In
      1 Function_Result (I4) Out
    End-Define
```

Calling such an interface with an RPC client means providing the function (ADD, SUBTRACT, MULTIPLY, DIVIDE or MODULO) as a value on the operation parameter in the same way legacy callers do.

The interface above would

- result in a class with a single method if it was wrapped with an object-oriented wrapper (Java Wrapper, .NET Wrapper, DCOM Wrapper).
- offer CALC as only operation if it was used as web service.

A modern design of the same interface would provide more specialized interfaces. Each interface may have fewer parameters compared to the original Natural subprogram. Parameters that are obsolete for a function are not offered in the corresponding interface, for example the parameter `Operation` in our example. This keeps the interface lean and clear and makes it easier to use from the perspective of an RPC client.

This process is called “Redesigning the interface”, which can be seen as manual step in the overall extraction process. See [Step 6: Redesign the Interface for Natural Subprograms \(Optional\)](#).

After the redesign, the IDL looks as follows:

```

Library 'EXAMPLE' Is
  Program 'ADD' Is
    Define Data Parameter
      1 Operand1      (I4) In
      1 Operand2      (I4) In
      1 Function_Result (I4) Out
    End-Define
  Program 'SUBTRACT'
    Define Data Parameter
      1 Operand1      (I4) In
      1 Operand2      (I4) In
      1 Function_Result (I4) Out
    End-Define
  Program 'MULTIPLY'
    Define Data Parameter
      1 Operand1      (I4) In
      1 Operand2      (I4) In
      1 Function_Result (I4) Out
    End-Define
  Program 'DIVIDE'
    Define Data Parameter
      1 Operand1      (I4) In
      1 Operand2      (I4) In
      1 Function_Result (I4) Out
    End-Define
  Program 'MODULO'
    Define Data Parameter
      1 Operand1      (I4) In
      1 Operand2      (I4) In
      1 Function_Result (I4) Out
    End-Define

```

By redesigning the interface you can turn the legacy interface of the CALC server into a more modern, object-oriented interface. In case the redesigned interface above is wrapped

- with an object-oriented wrapper (Java Wrapper, .NET Wrapper, DCOM Wrapper), this results in a class with five methods - ADD, SUBTRACT, MULTIPLY, DIVDE and MODULO, a real class/method design.
- as web service, also five operations are offered.

#### ➤ To Redesign the interface and extract multiple interfaces

- 1 Make sure to check **Redesign the interfaces** in *Step 5: Select Natural Subprograms from RPC Environment* in the extractor wizard (see *Extracting Software AG IDL File from an Existing Natural RPC Environment*).
- 2 In *Step 6: Redesign the Interface for Natural Subprograms (Optional)*, proceed as follows:

- **Design interface for ADD**
  - Choose **Set Constant** for parameter **Operation** and enter '+' as value.
  - Choose **Rename** from the toolbar and change the name to ADD
- **Design interface for SUBTRACT**
  - Choose **Set Constant** for parameter **Operation** and enter '-' as value.
  - Choose **Rename** from the toolbar and change the name to SUBTRACT
- **To design MULTIPLY, DIVIDE or MODULO, proceed as for SUBTRACT**
  - Enter '\*' as value for **Operation** and rename it to MULTIPLY.
  - Enter '/' as value for **Operation** and rename it to DIVIDE.
  - Enter '%' as value for **Operation** and rename it to MODULO.

## Extracting Natural REDEFINES

---

REDEFINES are often used in Natural. The example below illustrates a simple redefinition where the parameter #BASE-FIELD is redefined by the fields FILLER-1 thru R-P3-01.

```

DEFINE DATA PARAMETER
1 #BASE-FIELD          (A161) /* first parameter
1 REDEFINE #BASE-FIELD /* second parameter
  2 FILLER-1           (A4)
  2 FILLER-2           (A60)
  2 R-P1-01            (A1)
  2 R-P2-01            (A10)
  2 R-P3-01            (I4)
END-DEFINE
    
```

You can select a single REDEFINE path of the same base field for one interface only. If you require more than one REDEFINE path of the same base field, extract multiple interfaces - for each REDEFINE path a separate interface, see [Extracting Multiple Interfaces](#).

### ➤ To redesign the interface and extract a REDEFINE path

- Choose **Map to In**, **Map to Out** or **Map to InOut** either for
  - the REDEFINE base parameter, here parameter #BASE-FIELD (A161) /\* first parameter, or
  - any REDEFINE path, here REDEFINE#BASE-FIELD /\* second parameter

## Extracting a Natural Server with User-defined Mapping

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:

```

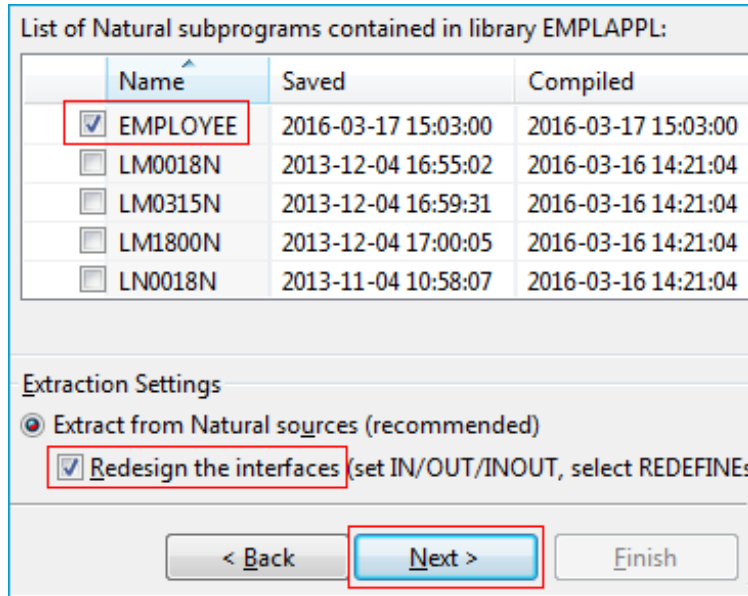
DEFINE DATA PARAMETER
1 OPERATION          (A1)      /* 'L' => List; 'D' => Details
1 ID                 (A10)     /* Input
1 EMPLOYEE           /* Output
  2 FIRSTNAME        (A20)     /* First name
  2 SURNAME          (A20)     /* Surname
  2 DATE-BIRTH       (D)       /* Date of birth
  2 DETAILS          (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

```

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**.

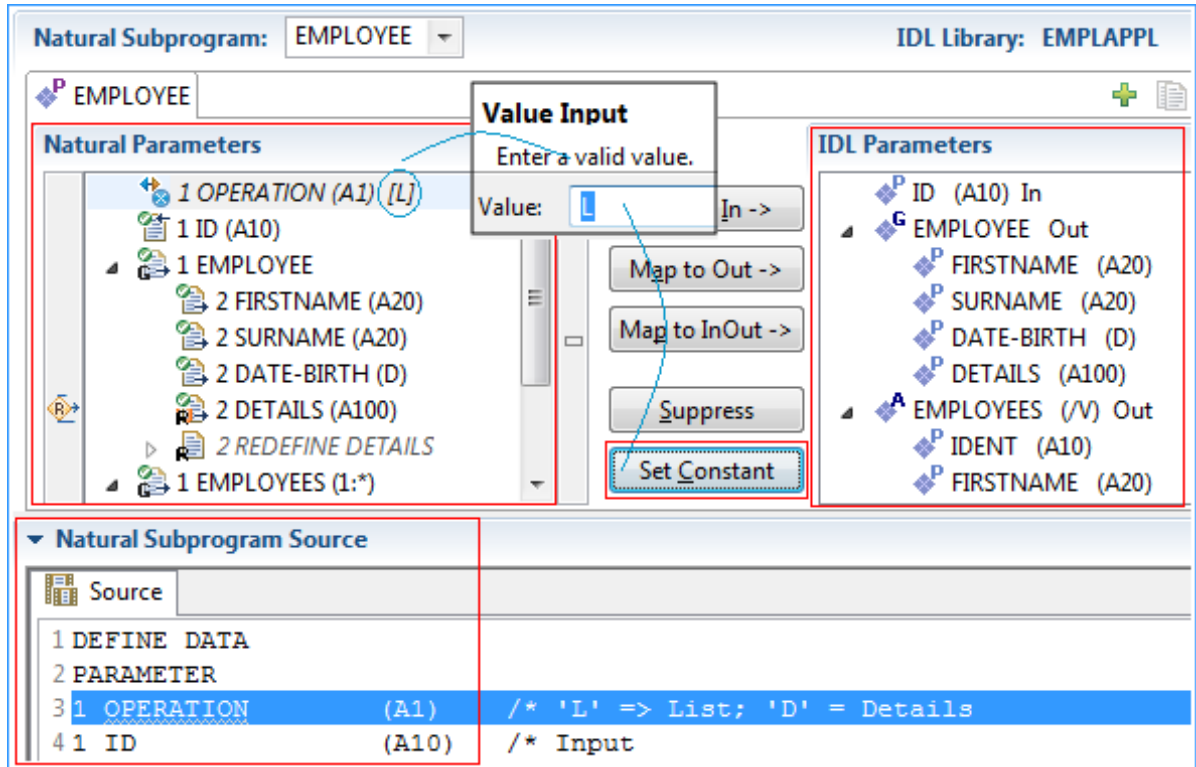


2. Model the extracted interface to suit your needs by creating a function `getListOfEmployees`.

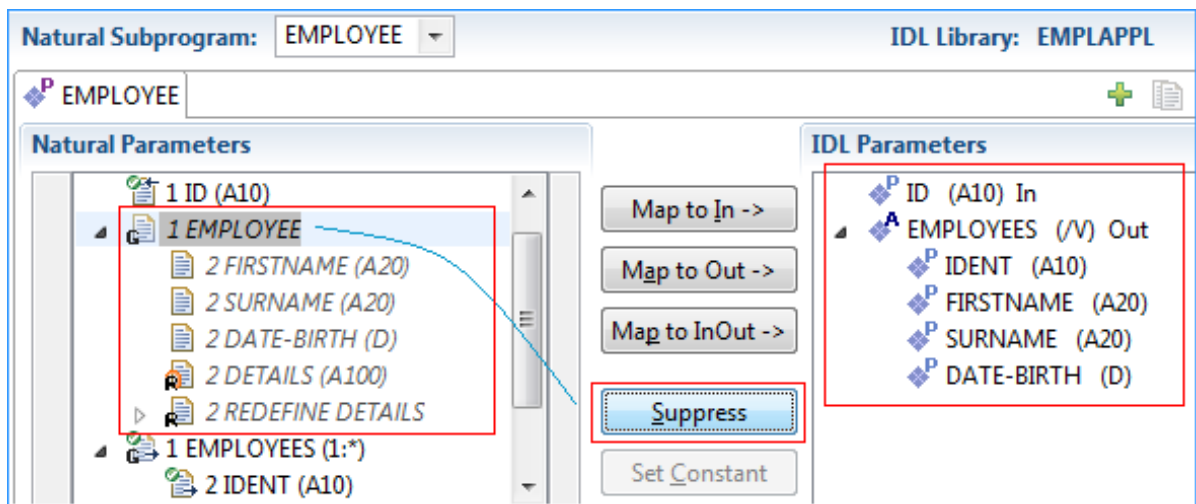
1. Mark the parameter `OPERATION` in the **Natural Subprogram Source** view or the **Natural Parameters** pane
2. Set Constant 'L' for the parameter `OPERATION`.

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.

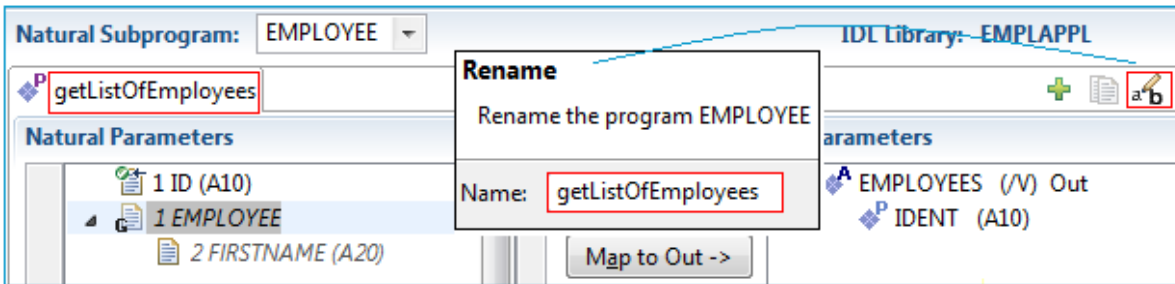



- 3 Suppress the Natural group `EMPLOYEE`, which is unused in the `LIST` function. This removes the IDL group `EMPLOYEE` from the **IDL Parameters** pane. The Natural parameter `EMPLOYEE` remains. Parameters that are suppressed in the **IDL Parameters** pane are displayed in italic font in the **Natural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

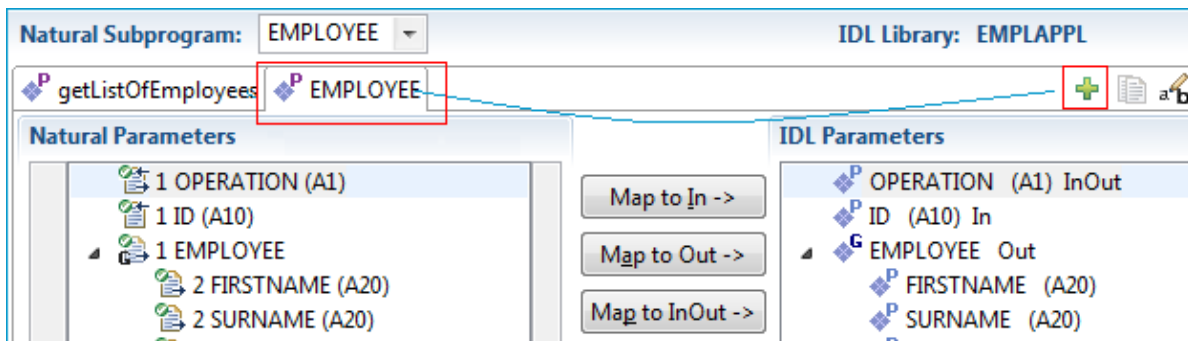



- Note:** Parameters set to constant are also displayed in italic font, because **Set Constant** suppresses them in the **IDL Parameters** pane too (see Step 2).

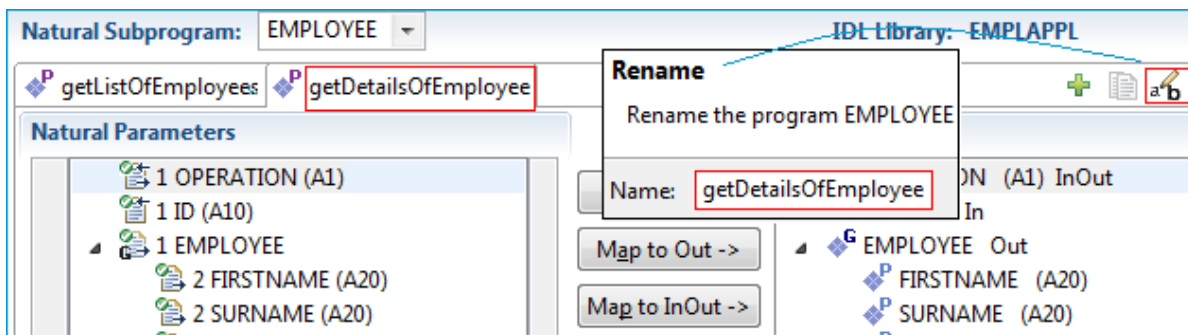
- 4 Also suppress Natural parameter ID, similar as already done for Natural group EMPLOYEE in the previous step.
- 5 Specify a readable name `getListOfEmployees` for the **IDL Parameters**, using the toolbar button . The new name appears on the tab.



- 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.

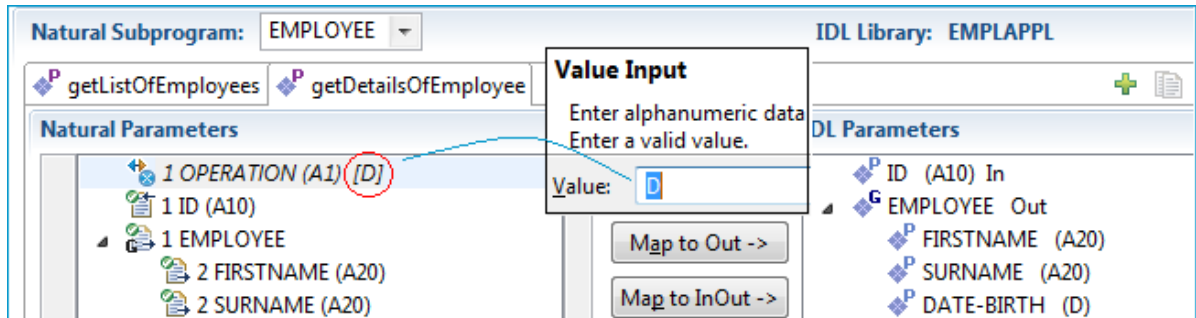


- 7 Specify a readable name `getDetailsOfEmployee` for the **IDL Parameters**, using the toolbar button . The name is displayed on the tab.

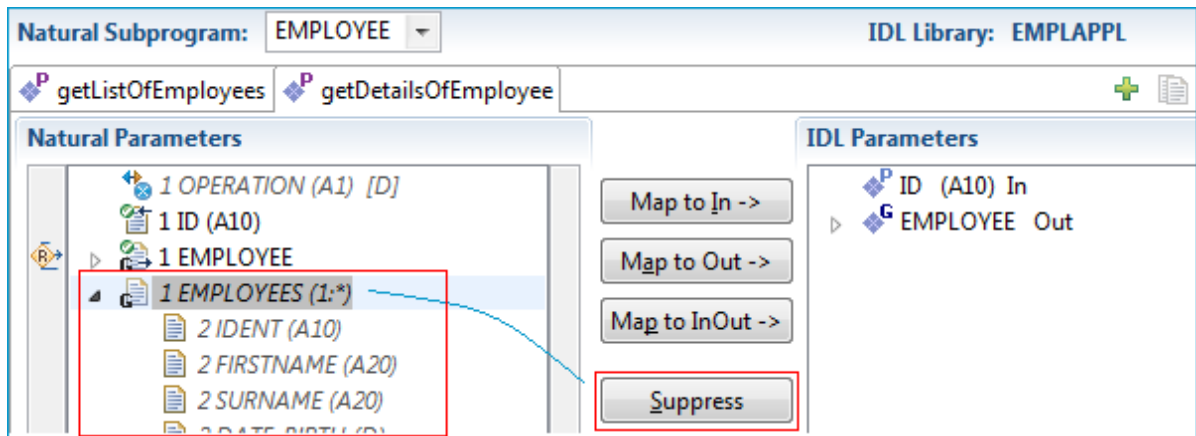




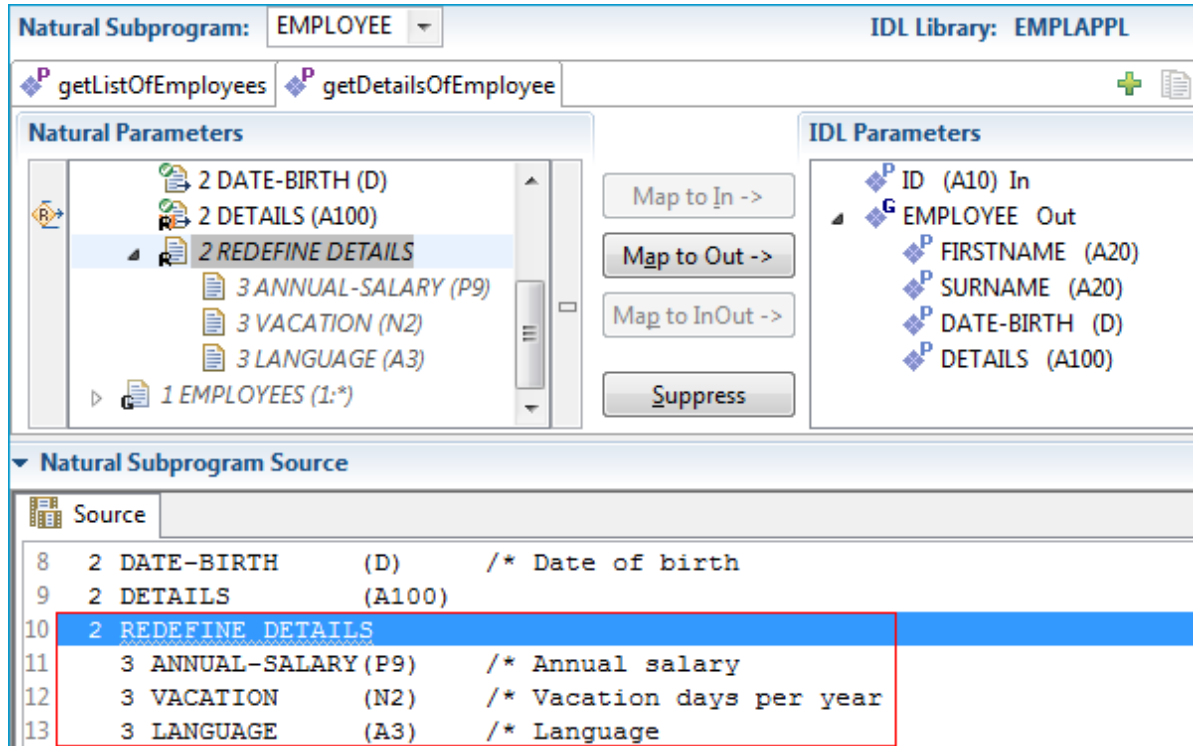
- 8 Set constant 'D' for Natural parameter `OPERATION` to execute the `EMPLOYEE DETAILS` operation at runtime. The IDL parameter `OPERATION` is removed from the **IDL Parameters** pane and displayed in the **Natural Parameters** pane. The italic font indicates suppression.



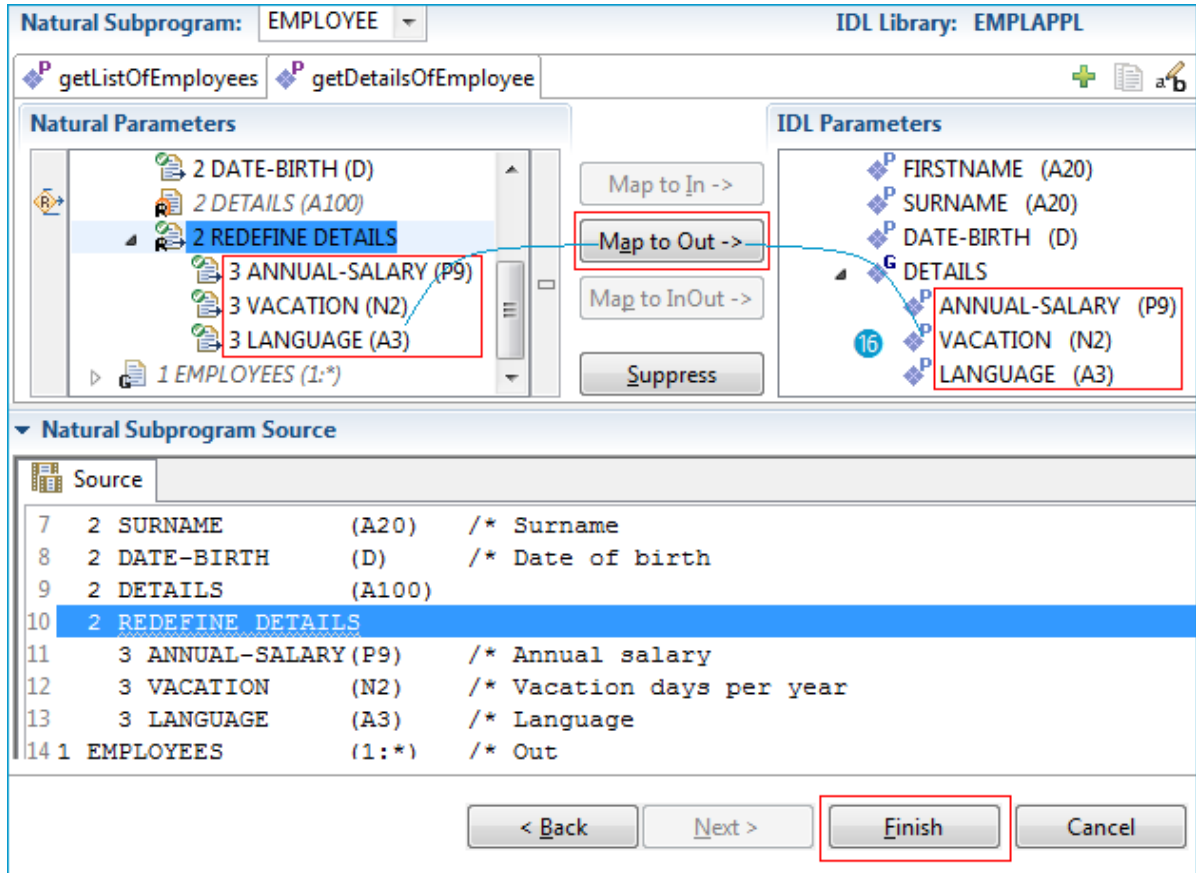
- 9 Suppress the Natural X-array `EMPLOYEES`, which is not needed in the `DETAILS` interface. The IDL parameter `EMPLOYEES` is removed from the **IDL Parameters** pane and displayed in italic font in the **Natural Parameters** pane.



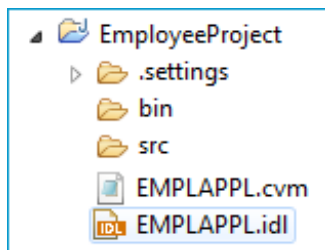
- 10 By default, the Natural parameter `DETAILS` is mapped. In this case, the redefinition of `DETAILS` in the **Natural Subprogram Source** pane contains information of more value.



- 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 RPC client view, 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
```