webMethods EntireX

Software AG IDL Extractor for Natural

Version 9.5 SP1

November 2013

Document ID: EXX-EEXXNATEXTRACTOR-95SP1-20140628

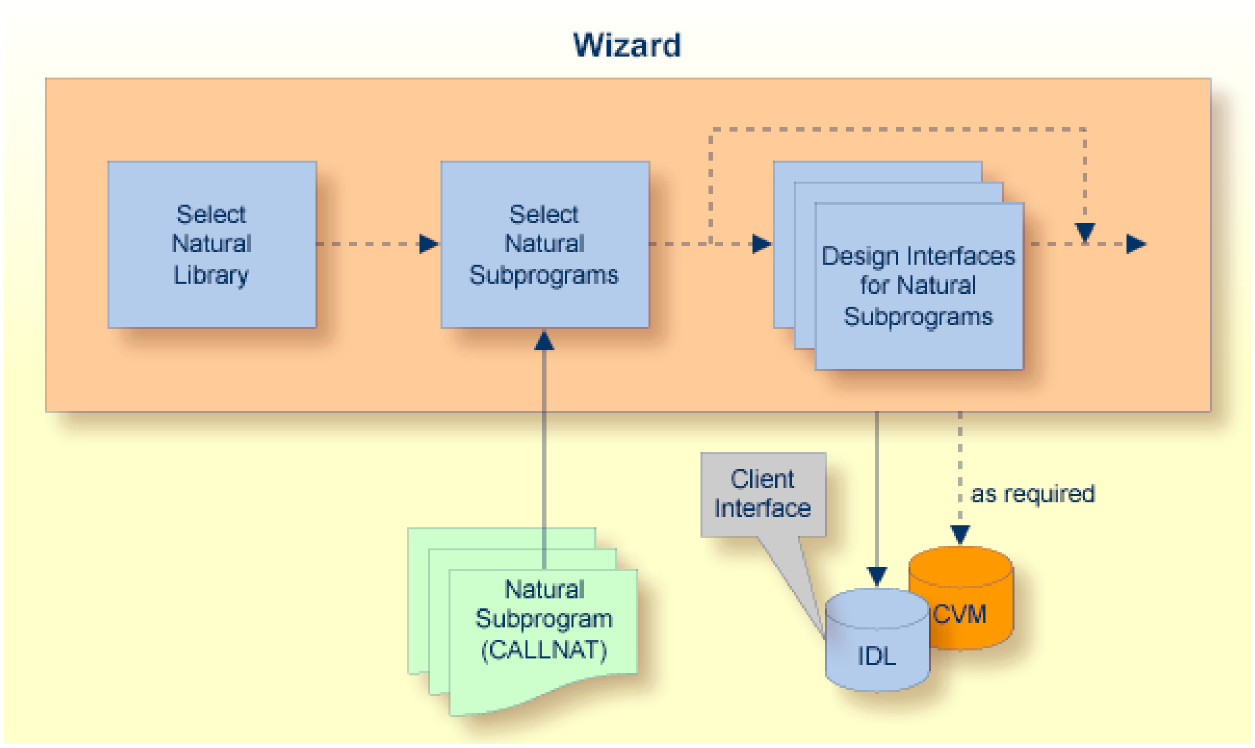
Table of Contents

1 Introduction to the Software AG IDL Extractor for Natural

Scope

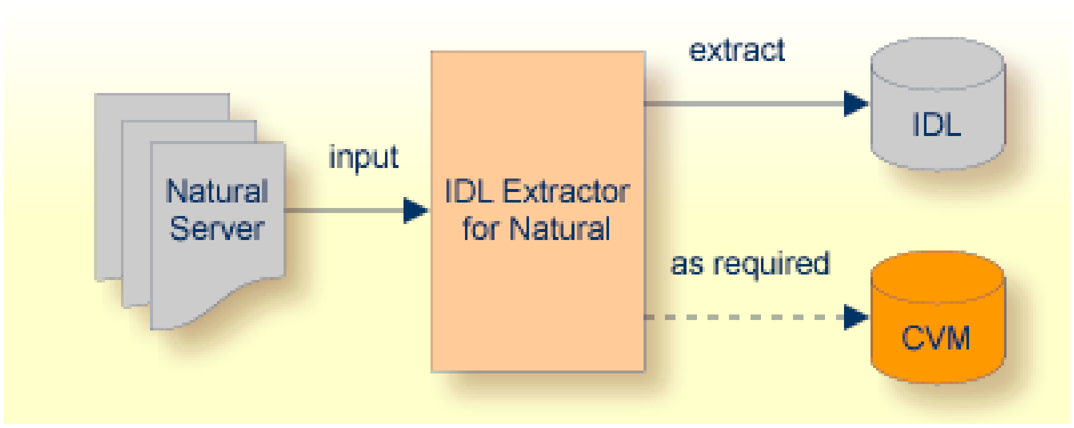
The Software AG IDL Extractor for Natural extracts a Software AG IDL definition from a Natural source in a Natural project in Eclipse, or from an object within a Natural RPC environment.

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.


Wizard
Select Natural Library
Select Natural Subprograms
Design Interfaces for Natural Subprograms
Client Interface
as required
Natural Subprogram (CALLNAT)
IDL
CVM

The result of this process is an IDL file and, optionally, a related client-side server mapping file (CVM) 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*.
- **CVM File**
  A CVM file (client-side server mapping file) to complete the mapping is generated only if it is required by the RPC server during runtime to call the Natural server. For information on when a CVM file is required, see *When is a CVM File Required?*



Extractor Wizard

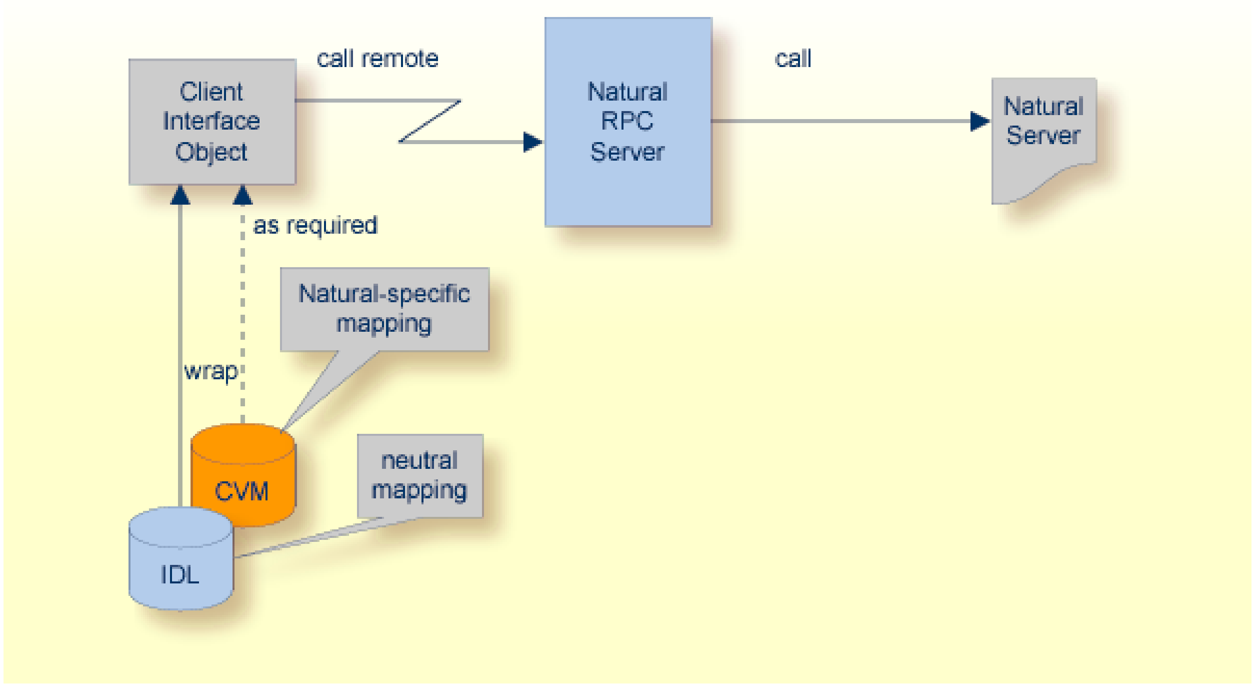
The extractor wizard guides you through the extraction process and supports the following tasks:

- Accessing Natural subprograms (CALLNATs), sources or object files, either in the local file system where the EntireX Workbench 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 ***Redesigning the Extracted Interface***.
  - 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 during runtime.

The extractor wizard is described in a step-by-step tutorial; see ***Using the Software AG IDL Extractor for Natural***.

CVM File

The CVM file (client-side 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 CVM file is required, see ***When is a CVM File Required?***.



If a CVM file has to be extracted, a correct interpretation of the IDL file is only possible if the CVM file is also available. Therefore, a CVM file always has to kept in the same folder as its related the IDL file.

A CVM 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.

2 Using the Software AG IDL Extractor for Natural

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

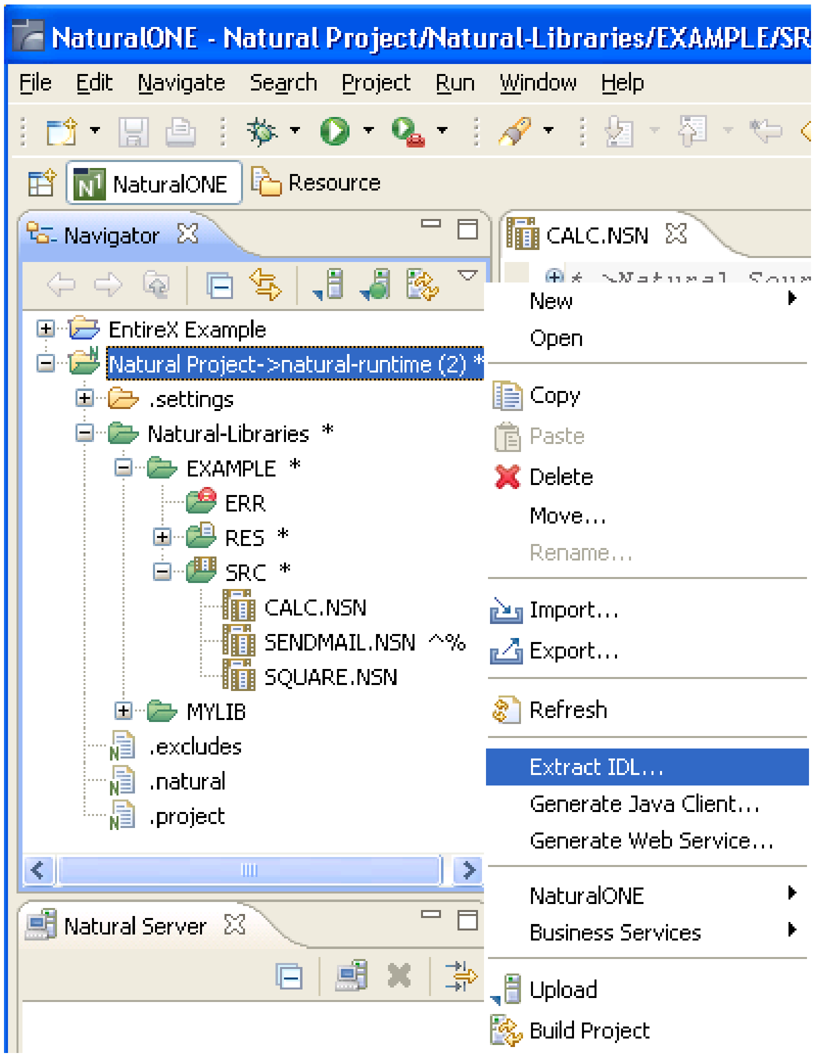
Step 1: Start the IDL Extractor for Natural Wizard

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

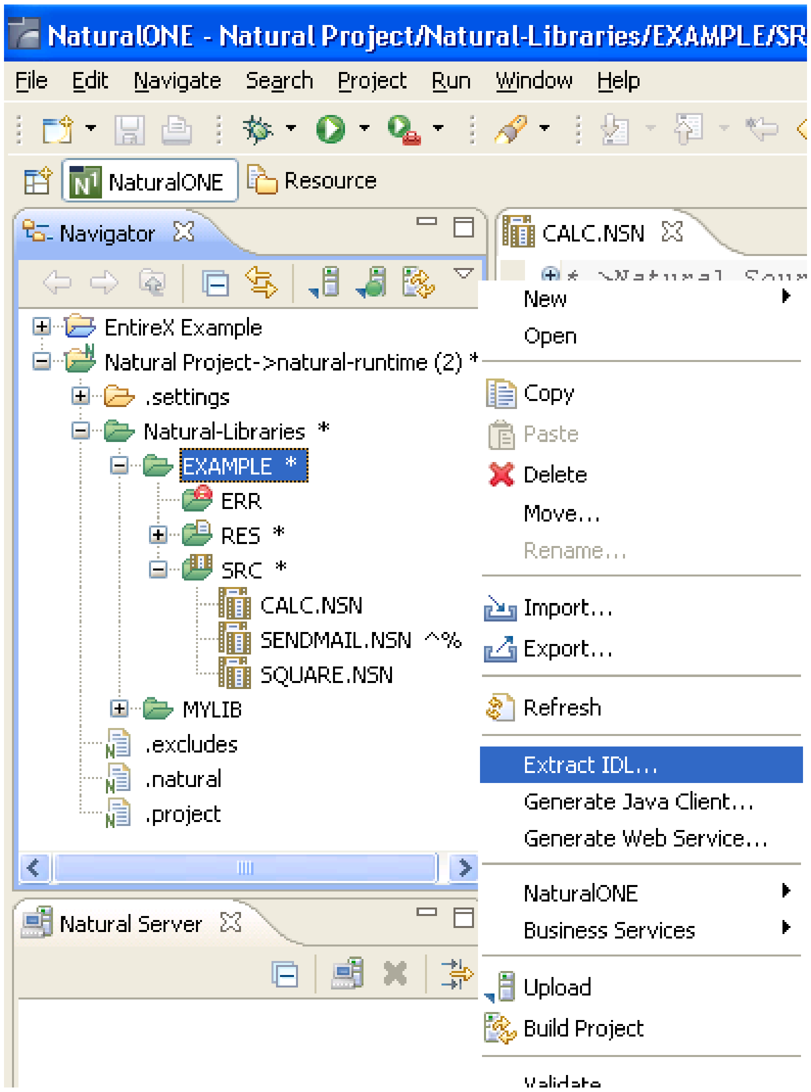
To extract the IDL, select the NaturalONE project in the Software AG Designer.



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

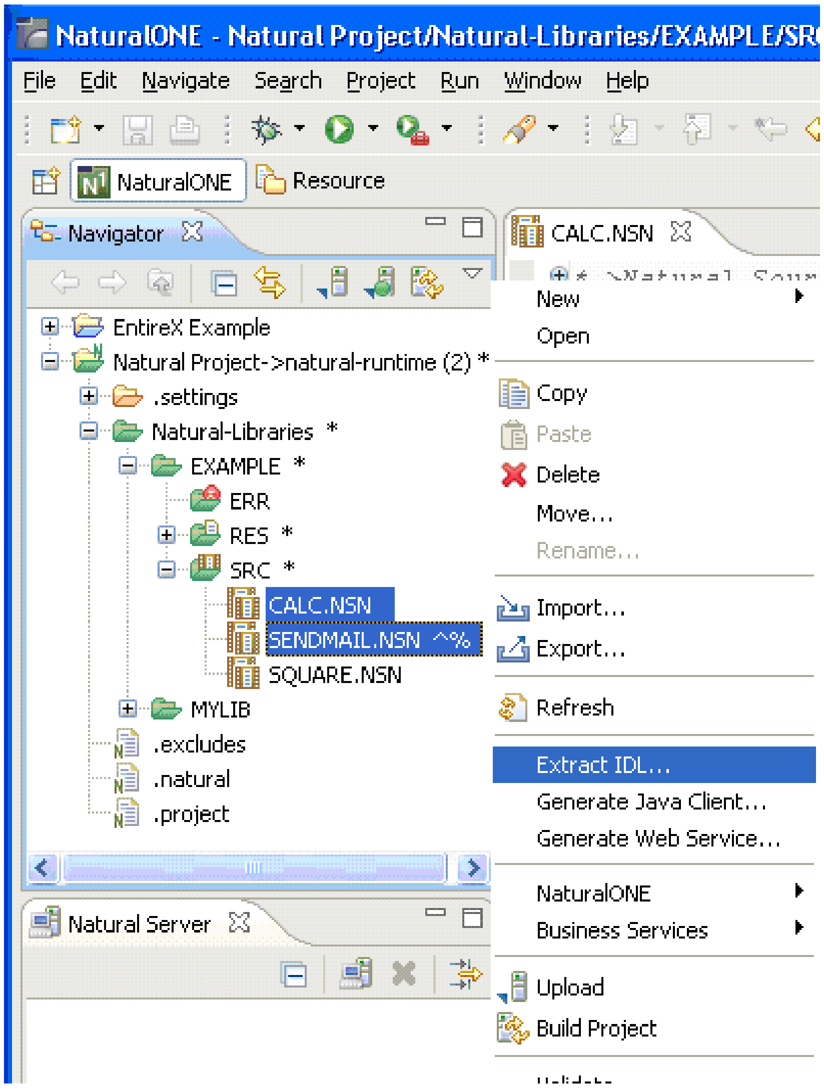
To extract the IDL, select the NaturalONE project in the Software AG Designer.



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

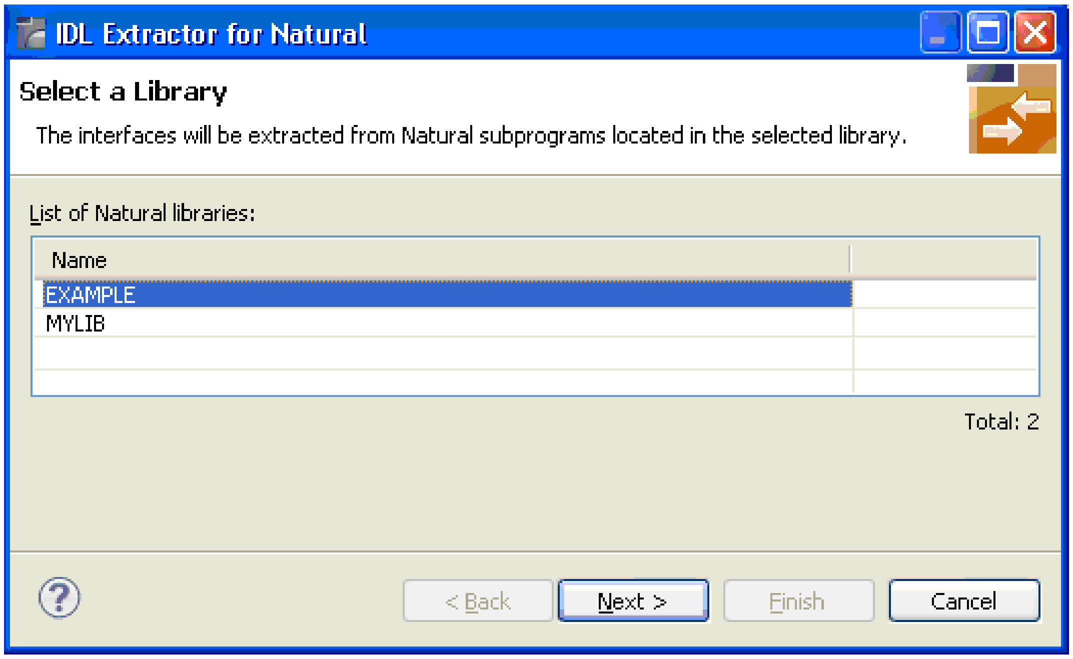
To extract the IDL, select one or multiple Natural subprogram sources (.NSN) in the Software AG Designer.



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 a Library

The interfaces will be extracted from Natural subprograms located in the selected library.

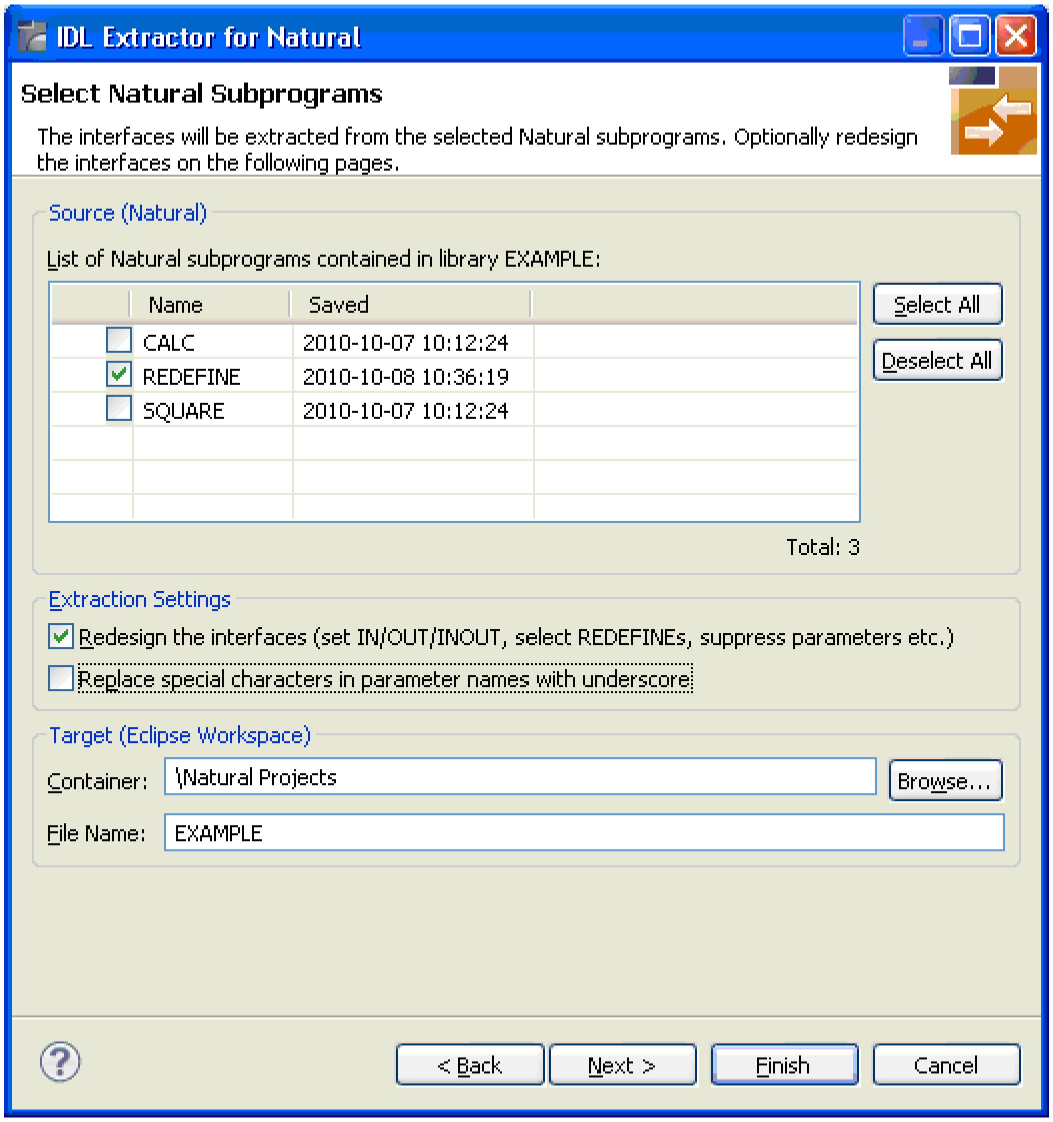
List of Natural libraries:

Name
EXAMPLE
MYLIB

Total: 2

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



Select Natural Subprograms

The interfaces will be extracted from the selected Natural subprograms. Optionally redesign the interfaces on the following pages.

Source (Natural)

List of Natural subprograms contained in library EXAMPLE:

	Name	Saved
☐	CALC	2010-10-07 10:12:24
✓	REDEFINE	2010-10-08 10:36:19
☐	SQUARE	2010-10-07 10:12:24

Total: 3

Extraction Settings

✓ Redesign the interfaces (set IN/OUT/INOUT, select REDEFINEs, suppress parameters etc.)

☐ Replace special characters in parameter names with underscore

Target (Eclipse Workspace)

Container: \Natural Projects

File Name: EXAMPLE

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 *Redesigning the Extracted Interface*. 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 *Software AG IDL File* in the IDL Editor documentation and optional CVM file allows you to customize the IDL (alias names, in/out/inout modifiers) for subsequent optimized client generation steps. See ***CVM File***.

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**.

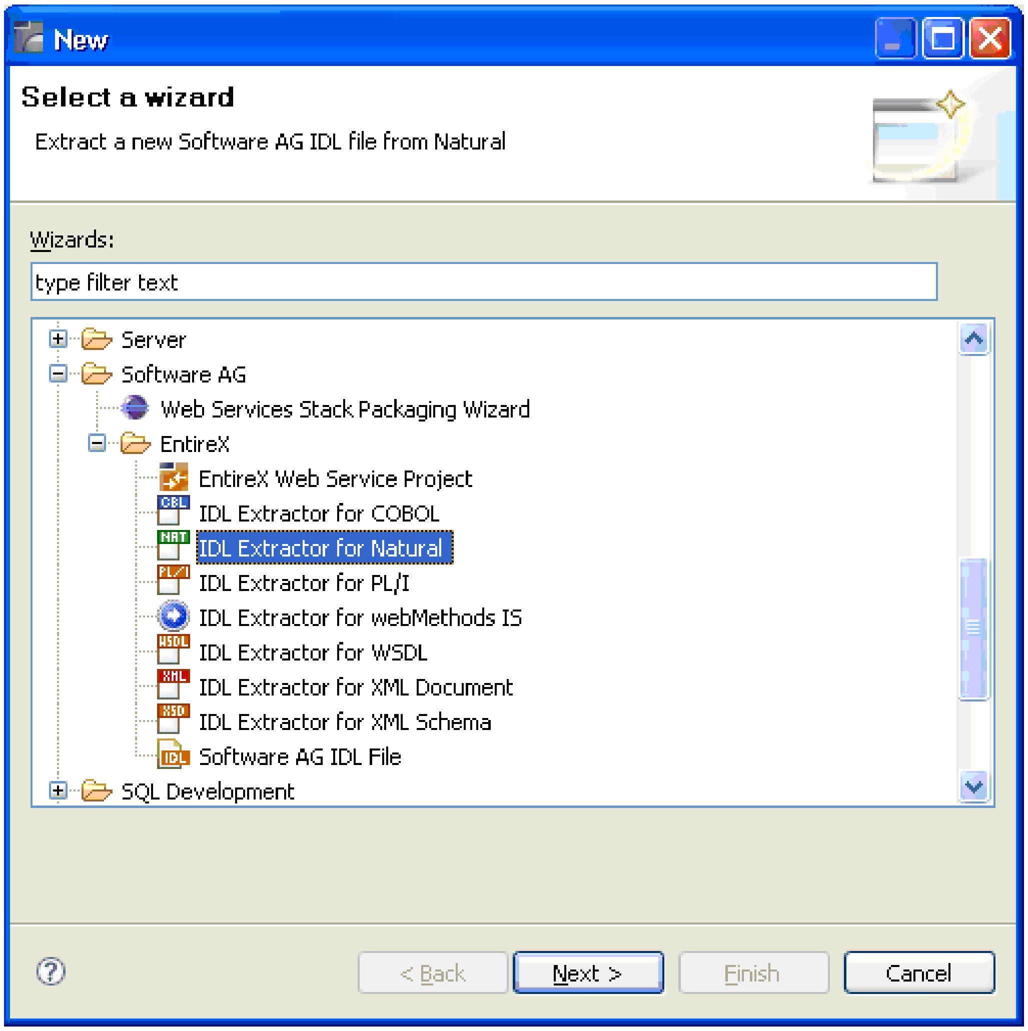
Choose **Finish** to start extraction with a default mapping. For more information see ***Extraction Result***.

Extracting Software AG IDL File from a New Natural RPC Environment

This section covers the following topics:

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

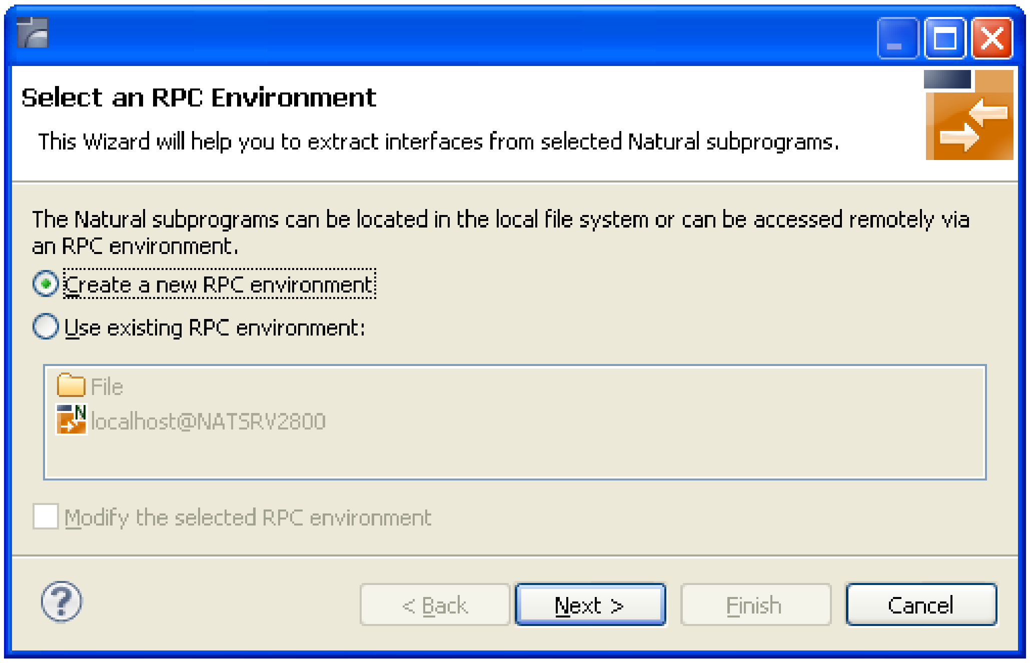
Step 1: Start the IDL Extractor for Natural Wizard


New
Select a wizard
Extract a new Software AG IDL file from Natural
Wizards:
type filter text
Server
Software AG
Web Services Stack Packaging Wizard
EntireX
EntireX Web Service Project
IDL Extractor for COBOL
IDL Extractor for Natural
IDL Extractor for PL/I
IDL Extractor for webMethods IS
IDL Extractor for WSDL
IDL Extractor for XML Document
IDL Extractor for XML Schema
Software AG IDL File
SQL Development
< Back
Next >
Finish
Cancel

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*.



Select an RPC Environment

This Wizard will help you to extract interfaces from selected Natural subprograms.

The Natural subprograms can be located in the local file system or can be accessed remotely via an RPC environment.

[✓ Create a new RPC environment]

Use existing RPC environment:

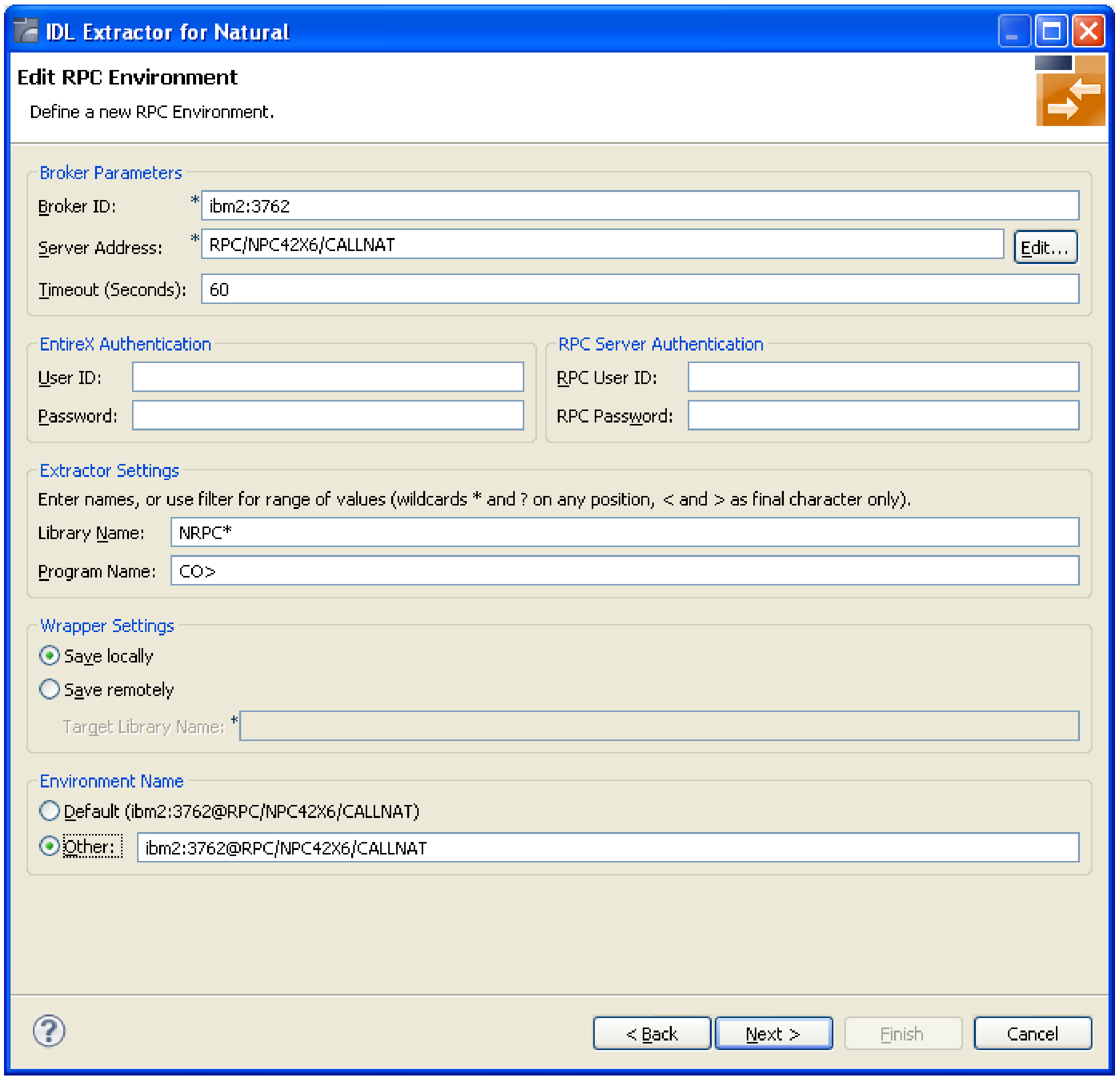
- File
- localhost@NATSRV2800

Modify the selected RPC environment

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



IDL Extractor for Natural

Edit RPC Environment

Define a new RPC Environment.

Broker Parameters

Broker ID: *	ibm2:3762
Server Address: *	RPC/NPC42X6/CALLNAT
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:	NRPC*
Program Name:	CO>

Wrapper Settings

[✓ Save locally]

Save remotely

Target Library Name: *

Environment Name

Default (ibm2:3762@RPC/NPC42X6/CALLNAT)

[✓ Other:] ibm2:3762@RPC/NPC42X6/CALLNAT

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 asterisk "*" (any position) to list libraries or program names matching any sequence of characters.
- wildcard question mark "?" (any position) to list libraries or program names matching any single character.
- wildcard greater than ">" (final character only) to list libraries or program names after.
- wildcard lower than "<" (final character only) to list libraries or program names before.
- no wildcard to extract from the given library or program. If the library or program does not exist, an error message will be displayed.

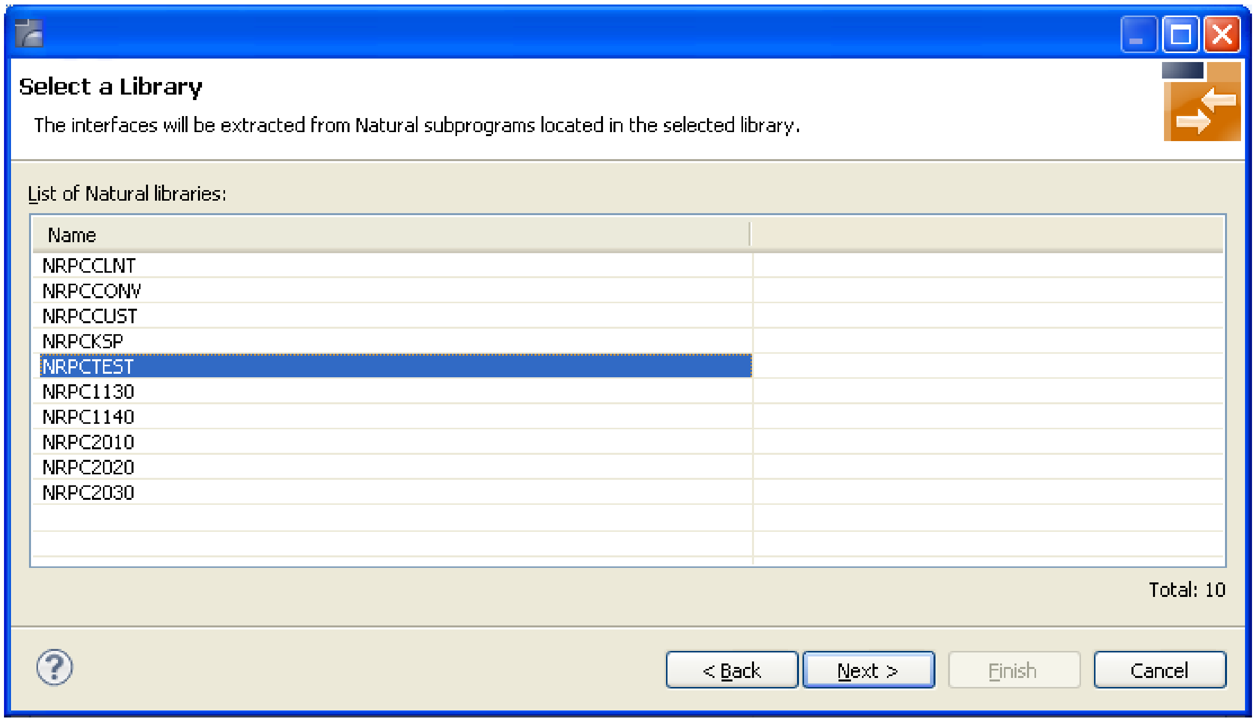
Any Natural RPC Server can be used. Only Natural libraries that reside in the FUSER system file of the Natural RPC Server can be accessed. Special configuration is required only for operating system IBM i; see *Natural RPC Server Configuration for the IDL Extractor for Natural* in the IBM i administration documentation.

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*.



Select a Library

The interfaces will be extracted from Natural subprograms located in the selected library.

List of Natural libraries:

Name	
NRPCCLNT	
NRPCCONV	
NRPCCUST	
NRPCKSP	
NRPCTEST	
NRPC1130	
NRPC1140	
NRPC2010	
NRPC2020	
NRPC2030	

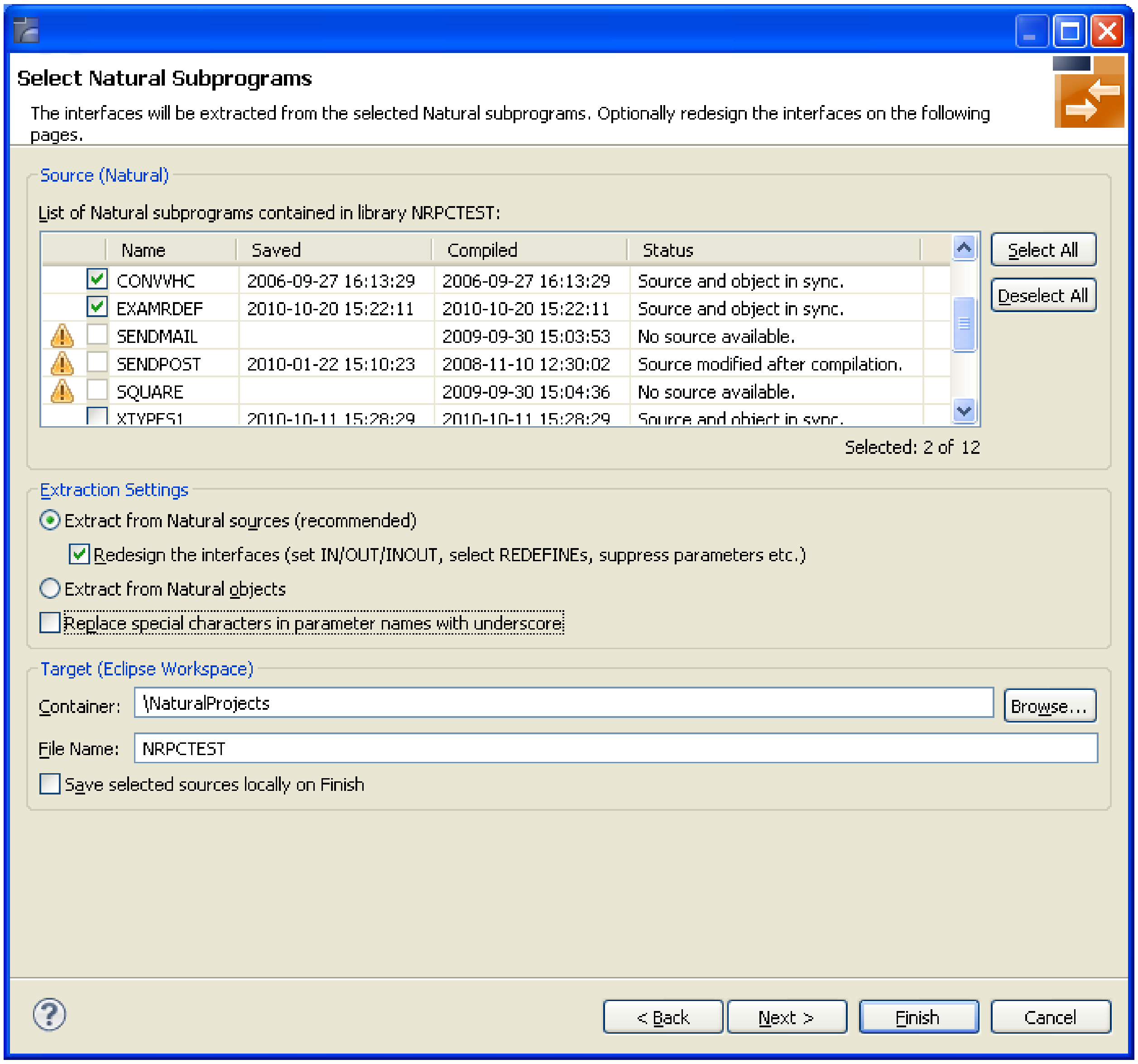
Total: 10

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*.



Select Natural Subprograms

The interfaces will be extracted from the selected Natural subprograms. Optionally redesign the interfaces on the following pages.

Source (Natural)

List of Natural subprograms contained in library NRPCTEST:

		Name	Saved	Compiled	Status
	✓	CONVVHC	2006-09-27 16:13:29	2006-09-27 16:13:29	Source and object in sync.
	✓	EXAMRDEF	2010-10-20 15:22:11	2010-10-20 15:22:11	Source and object in sync.
⚠	☐	SENDMAIL		2009-09-30 15:03:53	No source available.
⚠	☐	SENDPOST	2010-01-22 15:10:23	2008-11-10 12:30:02	Source modified after compilation.
⚠	☐	SQUARE		2009-09-30 15:04:36	No source available.
	☐	XTYPES1	2010-10-11 15:28:29	2010-10-11 15:28:29	Source and object in sync.

Selected: 2 of 12

Extraction Settings

[✓ Extract from Natural sources (recommended)]

[✓ Redesign the interfaces (set IN/OUT/INOUT, select REDEFINEs, suppress parameters etc.)]

Extract from Natural objects

Replace special characters in parameter names with underscore

Target (Eclipse Workspace)

Container: \NaturalProjects

File Name: NRPCTEST

Save selected sources locally on Finish

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 *Redesigning the Extracted Interface*. 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*.

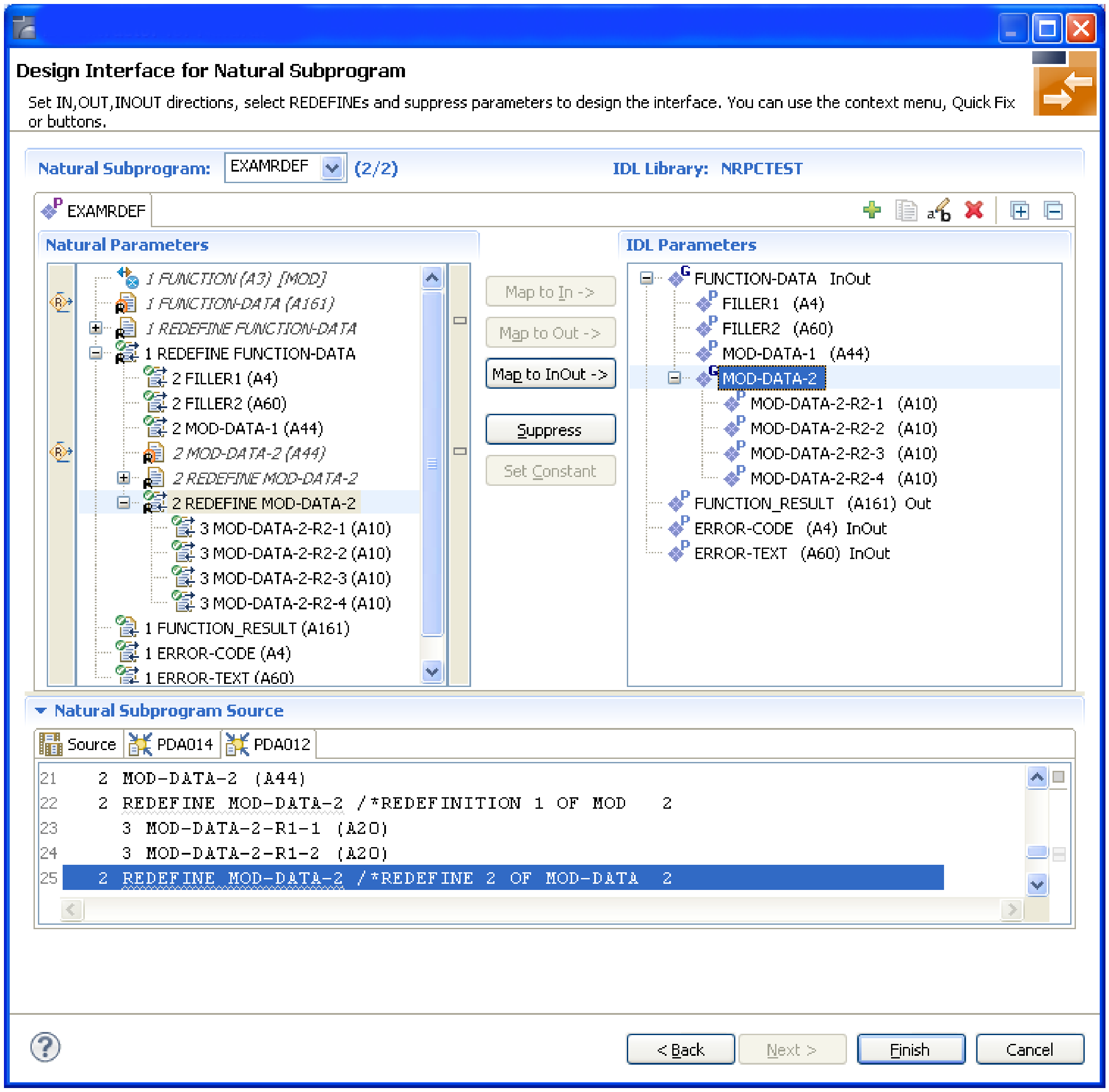
Choose **Next** to continue. If multiple Natural subprograms have been selected in the Natural subprogram selection step, redesign the next interface. You can see the current and total number of the subprogram you are extracting from in the title (n/m).

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*



Design Interface for Natural Subprogram

Set IN,OUT,INOUT directions, select REDEFINEs and suppress parameters to design the interface. You can use the context menu, Quick Fix or buttons.

Natural Subprogram: EXAMRDEF (2/2) IDL Library: NRPCTEST

EXAMRDEF

Natural Parameters

```
1 FUNCTION (A3) [MOD]
1 FUNCTION-DATA (A161)
1 REDEFINE FUNCTION-DATA
1 REDEFINE FUNCTION-DATA
  2 FILLER1 (A4)
  2 FILLER2 (A60)
  2 MOD-DATA-1 (A44)
  2 MOD-DATA-2 (A44)
  2 REDEFINE MOD-DATA-2
  2 REDEFINE MOD-DATA-2
    3 MOD-DATA-2-R2-1 (A10)
    3 MOD-DATA-2-R2-2 (A10)
    3 MOD-DATA-2-R2-3 (A10)
    3 MOD-DATA-2-R2-4 (A10)
1 FUNCTION_RESULT (A161)
1 ERROR-CODE (A4)
1 ERROR-TEXT (A60)
```

IDL Parameters

```
FUNCTION-DATA  InOut
  FILLER1  (A4)
  FILLER2  (A60)
  MOD-DATA-1  (A44)
  MOD-DATA-2
    MOD-DATA-2-R2-1  (A10)
    MOD-DATA-2-R2-2  (A10)
    MOD-DATA-2-R2-3  (A10)
    MOD-DATA-2-R2-4  (A10)
FUNCTION_RESULT  (A161)  Out
ERROR-CODE  (A4)  InOut
ERROR-TEXT  (A60)  InOut
```

Natural Subprogram Source

```
  2 MOD-DATA-2 (A44)
  2 REDEFINE MOD-DATA-2 /*REDEFINITION 1 OF MOD   2
    3 MOD-DATA-2-R1-1 (A20)
    3 MOD-DATA-2-R1-2 (A20)
  2 REDEFINE MOD-DATA-2 /*REDEFINE 2 OF MOD-DATA  2
```

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	Creates a new interface (IDL program) based on the original parameters of the Natural subprogram.
📄	Duplicate	Creates 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	Removes the current interface (active tab). At least one interface must exist.
⊞	Expand All	Expands 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).

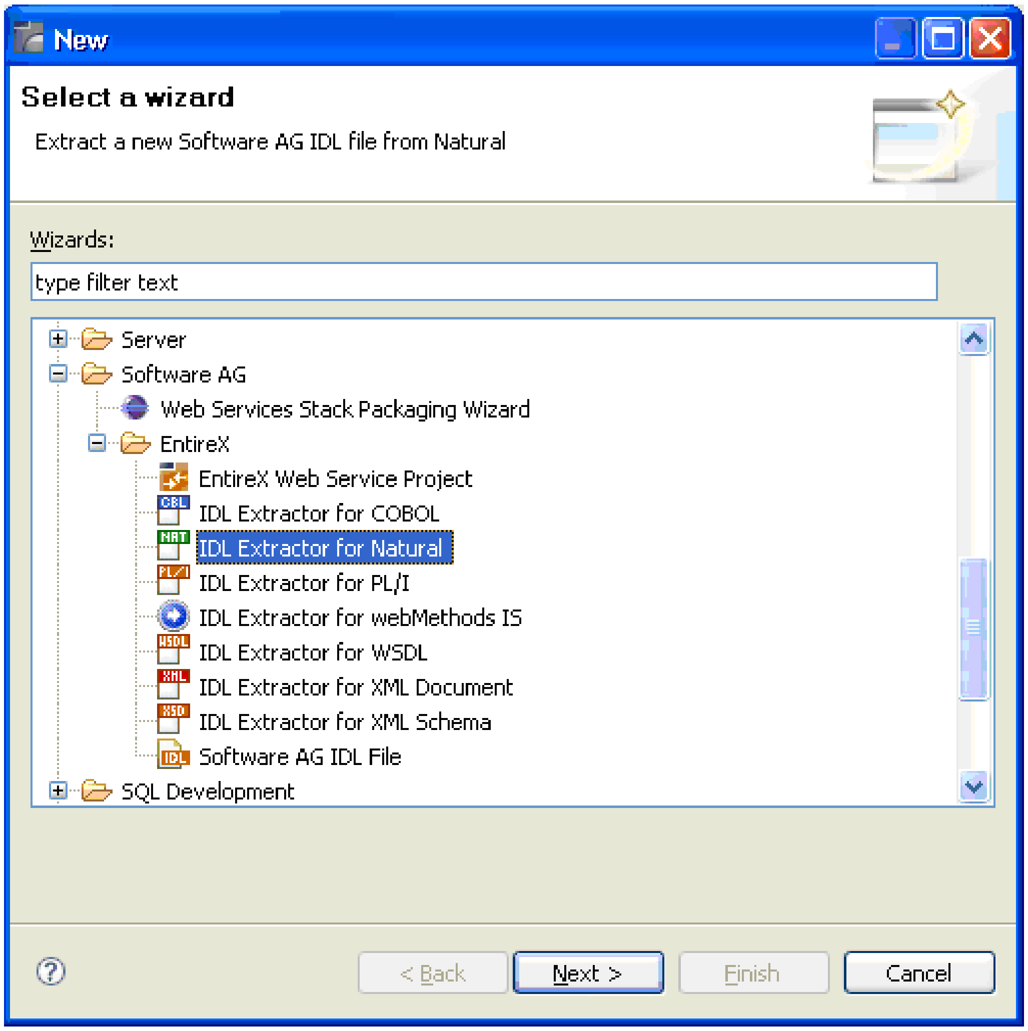
Choose **Next** to continue. If multiple Natural subprograms have been selected in the Natural subprogram selection step, redesign the next interface. You can see the current and total number of the subprogram you are extracting from in the title (n/m).

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 2: Select an RPC Environment

Step 1: Start the IDL Extractor for Natural Wizard



New

Select a wizard

Extract a new Software AG IDL file from Natural

Wizards:

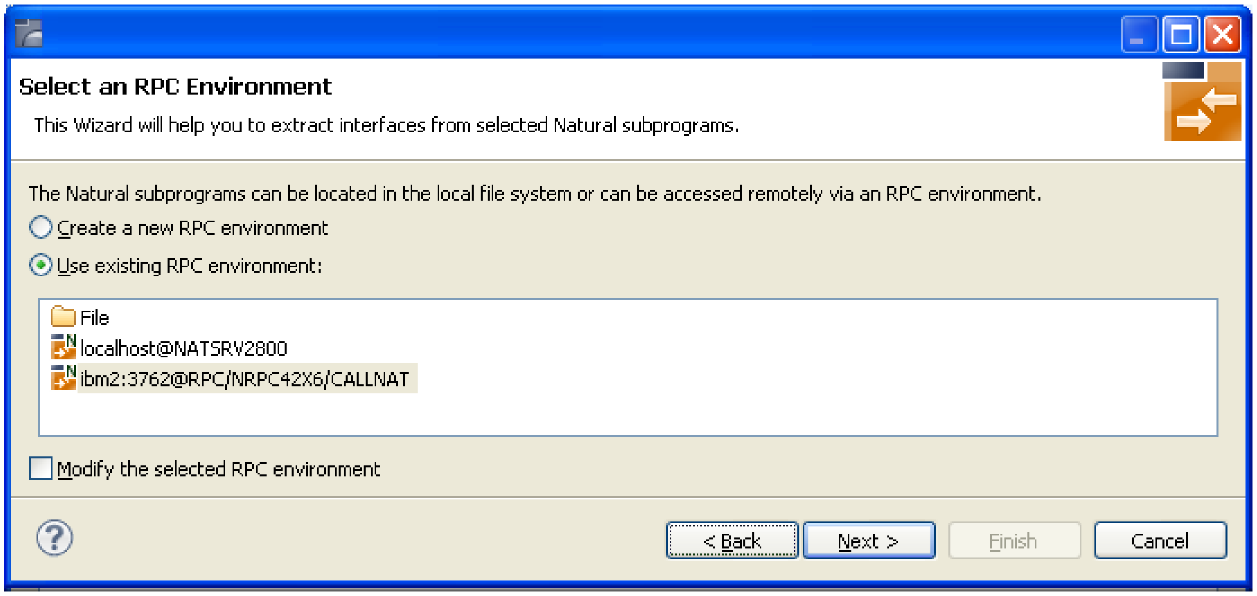
type filter text

- Server
- Software AG
  - Web Services Stack Packaging Wizard
  - EntireX
    - EntireX Web Service Project
    - IDL Extractor for COBOL
    - IDL Extractor for Natural
    - IDL Extractor for PL/I
    - IDL Extractor for webMethods IS
    - IDL Extractor for WSDL
    - IDL Extractor for XML Document
    - IDL Extractor for XML Schema
    - Software AG IDL File
- SQL Development

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 an RPC Environment

This Wizard will help you to extract interfaces from selected Natural subprograms.

The Natural subprograms can be located in the local file system or can be accessed remotely via an RPC environment.

( ) Create a new RPC environment

(●) Use existing RPC environment:

- File
- localhost@NATSRV2800
- ibm2:3762@RPC/NRPC42X6/CALLNAT

☐ Modify the selected RPC environment

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 plugins are installed.

Extraction Result

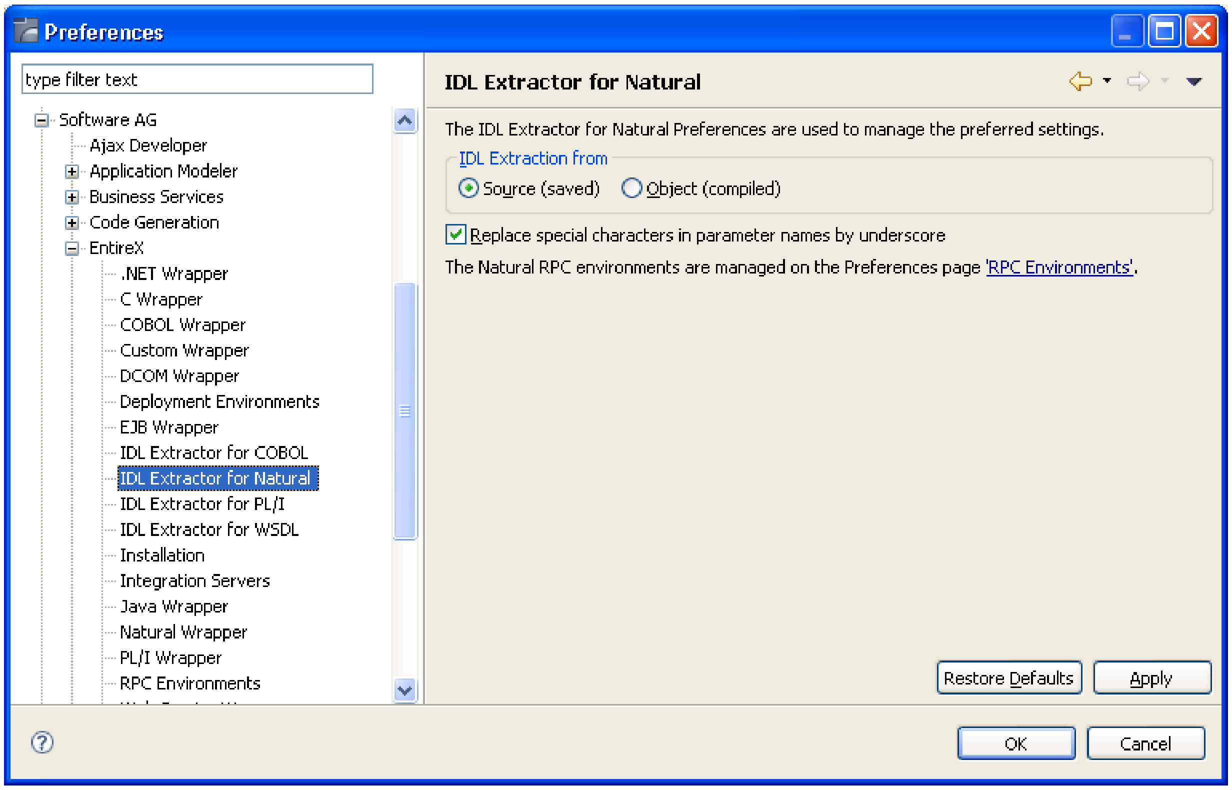
When the wizzard has finished successfully, you can see two additional files in your Eclipse project:

- **.idl file**
  The Software AG IDL file that describes the RPC interface of the Natural server component that implements the service's business logic together with an optional client-side server-mapping file (CVM). The IDL file is opened with the *IDL Editor*. See *IDL File*.
- **.cvm file (optional)**
  This 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). A CVM file always has to be kept in the same folder as its related IDL file. See ***CVM File***.

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.



IDL Extractor for Natural

The IDL Extractor for Natural Preferences are used to manage the preferred settings.

IDL Extraction from

[✓ Source (saved)] Object (compiled)

[✓] Replace special characters in parameter names by underscore

The Natural RPC environments are managed on the Preferences page 'RPC Environments'.

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***.

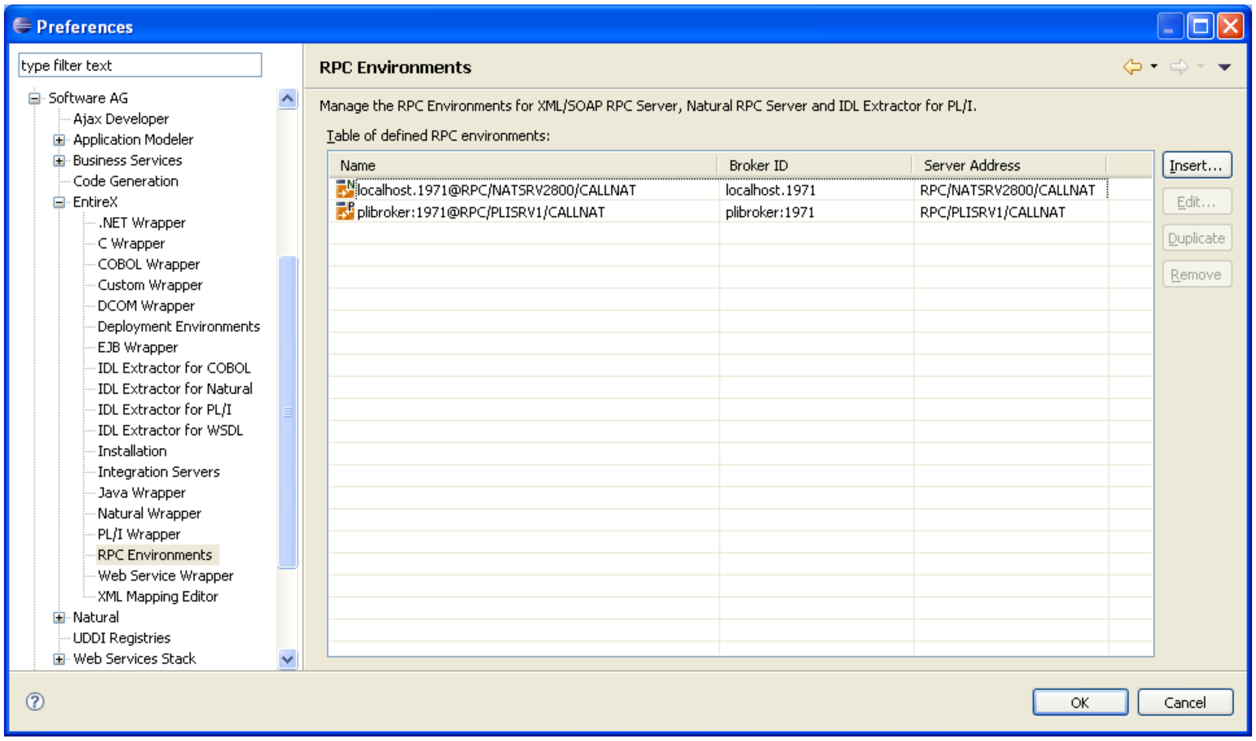
3 RPC Environment Manager

The RPC environment is managed on the RPC environment preference page. The RPC environments can be created, edited and removed. There are several types of RPC environment: Natural, PL/I and XML/SOAP. The RPC environment type will be used to prepare the selection lists of the following wizards:

- Natural RPC Server
- IDL Extractor for PL/I
- XML/SOAP RPC Server

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

Using these wizards, you can add new RPC environments of the respective type. To manage these RPC environments, open the **Preferences** page.



Manage the RPC Environments for XML/SOAP RPC Server, Natural RPC Server and IDL Extractor for PL/I.

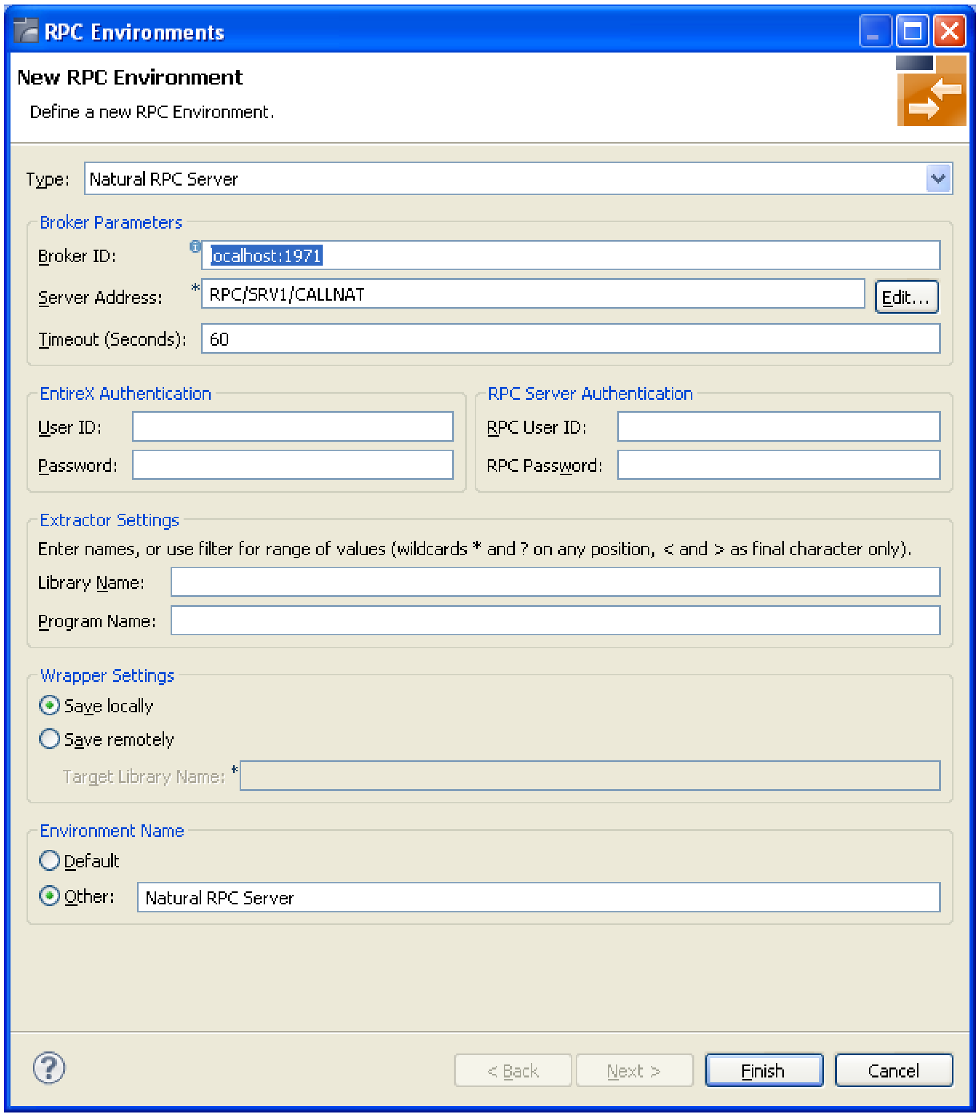
Table of defined RPC environments:

Name	Broker ID	Server Address
localhost.1971@RPC/NATSRV2800/CALLNAT	localhost.1971	RPC/NATSRV2800/CALLNAT
plibroker:1971@RPC/PLISRV1/CALLNAT	plibroker:1971	RPC/PLISRV1/CALLNAT

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, choose **Insert...**.



Type: Natural RPC Server

Broker Parameters

Broker ID: localhost:1971

Server Address: * RPC/SRV1/CALLNAT

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:

Program Name:

Wrapper Settings

[✓ Save locally]

Save remotely

Target Library Name: *

Environment Name

Default

[✓ Other:] Natural RPC Server

Choose the **Type** and 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 **Extraction Settings** are used for the IDL Extractors (Natural and PL/I) only. Use them to specify the name of the **Dataset/Library** and the **Member/Program** name.

The **Wrapper Settings** are used for Natural Wrapper only, and you can specify the operation type and target library name (not available for **Save locally**).

4 RPC Environment Monitor

The RPC Environment Monitor is part of the EntireX Workbench. 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 **Preference** page. See *RPC Environment Manager*.

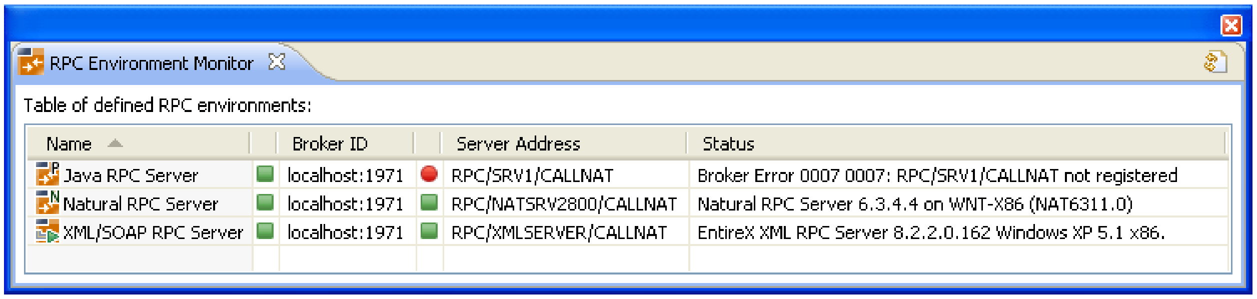


Table of defined RPC environments:

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).

5 Using the IDL Extractor for Natural in Command-line Mode

Command-line Options

See *Using the EntireX Workbench in 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 Workbench starter as described under *Using the EntireX Workbench in 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 progam 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.

6 Natural to IDL Mapping

This chapter describes how Natural data types are mapped to Software AG IDL files by the Software AG IDL Extractor for Natural and covers the following topics:

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.

The following metasymbols and informal terms are used for the IDL in the table below.

- The metasymbols "[" and "]" surround optional lexical entities
- The informal terms *n* and *m* are sequences of numeric characters, for example 123.

Natural Data Type	Software AG IDL Data Type	Description
`A`*number*	`A`*n*	Alphanumeric
`A DYNAMIC`	`AV`*n*	Alphanumeric variable length
`B`*number*	`B`*number*	Binary
`B DYNAMIC`	`BV`	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
`N`*number*`[.`*number*`]`	`N`*number*`[.`*number*`]`	Unpacked decimal
`P`*number*`[.`*number*`]`	`P`*number*`[.`*number*`]`	Packed decimal
`T`	`T`	Time
`Unumber`	`Unumber`	Unicode
`U DYNAMIC`	`UV`	Unicode variable length

Redesigning the Extracted Interface

The IDL Extractor for Natural allows you to design the interface to your Natural subprogram (`CALLNAT`). This includes

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

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 (`$`, `#`, `&`, `@`, `/`) in Natural parameter names to be replaced by underscores. 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 ***An Example for 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 ***An Example for Extracting Multiple 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

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   (I4/10)   /* shortcut for (A5/1:10)
1 #GROUP1   (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 occurences can be resized, i.e. 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. During runtime, it is also required to specify the number of occurences 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 20050031 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 technolgy, 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 repetion (`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, i.e. 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, i.e. 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 ***An Example for Set Constant***.

EntireX and Natural RPC make sure the constant value is passed to the Natural server during runtime. No data is transferred between the RPC client and the RPC server.

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 ***An Example for 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 during runtime. For more information, see ***Step 6: Redesign the Interface for Natural Subprograms (Optional)***.

7 Handling CVM Files

A client-side server mapping file (CVM) enables the RPC server to correctly support special Natural syntax such as `REDEFINE` and other special situations. If one of these elements is used, the EntireX Workbench automatically extracts a CVM file in addition to the IDL (interface definition language), or a CVM file is generated by the Natural Wrapper for a server skeleton. The CVM file is used at runtime to marshal and unmarshal the RPC data stream.

CVM Files in the EntireX Workbench

In the *EntireX Workbench* a CVM file has to relate to an appropriate IDL file; always keep the IDL file and the CVM file together in the same folder.

- If there is a CVM file and a corresponding IDL file, at least one of the IDL programs in the corresponding IDL file requires server-mapping information to correctly call the target server. For those IDL programs, there is a CVM entry (line) in the Workbench CVM file.
- If there is an IDL file but no corresponding CVM file, there is no IDL program that requires server mapping information.

Source Control of CVM Files

Because CVM entries within a CVM file contain text data only, a Workbench CVM file is text-based (although it is not intended for human consumption). Therefore, you can include it in your source control management together with the IDL file and the Natural source(s) as a triplet that should always be kept in sync.

Compare CVM Files

For CVM files in the *EntireX Workbench* format, you can use a third-party file/text compare tool to check if two files are identical.

The CVM entries (corresponding to lines in a Workbench CVM file) contain a creation timestamp at offset 276 (decimal) in the format *`YYYYMMDDHHIISST`*. The precision is 1/10 of a second.

When is a CVM File Required?

For the IDL Extractor for Natural

Natural Syntax	IDL Extractor for Natural	CVM Reqired	More Information

For the Natural Wrapper

Natural Wrapper	CVM Required	More Information
IDL program name is not a valid Natural name and is therefore adapted, or the Natural program name is customized	yes	*Step 2: Customize Natural Server Names* under *Using the Natural Wrapper for the Server Side within NaturalONE*

8 Examples for Redesigning the Extracted Interfaces

This chapter gives examples on how to redesign the extracted interface.

An Example for 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` under *Client and Server Examples for Natural* in the Natural Wrapper documentation.

```
****************************************************************
*
*  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 *An Example for 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".

An Example for Extracting Multiple Interfaces

Multiple functions are often implemented in a single Natural subprogram, for example in ***An Example for 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 wizzard (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`.

An Example for Extracting Natural REDEFINES

`REDEFINE`s 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 *An Example for 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`