webMethods EntireX

EntireX Natural Wrapper

Version 10.5

October 2019

WEBMETHODS

This document applies to webMethods EntireX Version 10.5 and all subsequent releases.

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

Document ID: EXX-EEXXNATWRAPPER-105-20220422

Table of Contents

1 About this Documentation

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
`Monospace font`	Identifies service names and locations in the format *`folder.subfolder.service`*, APIs, Java classes, methods, properties.
`Italic`	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.softwareag.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 Natural Wrapper

The Natural Wrapper is part of the Designer. It supports users to develop Natural client applications that access RPC-based server components and to create Natural RPC server skeletons, which you can use as a basis for writing Natural RPC servers that can be accessed by RPC clients.

The Natural Wrapper starts with a Software AG IDL file that describes the interfaces of the RPC server or RPC client components. During wrapping the IDL program names are mapped to suitable Natural names, which you can customize. This means that IDL program and IDL library names can be longer than eight characters and can even contain characters not allowed in Natural names.

For Natural client applications it generates the following:

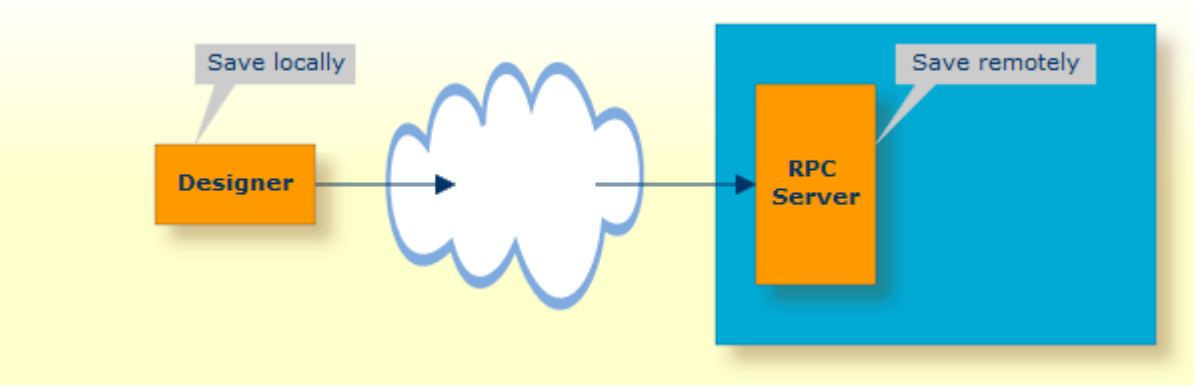
- Natural client interface objects (which are Natural subprograms; file extension .NSN) that can be used in Natural client applications to access a remote RPC component. If there is a related client-side mapping file (Natural | COBOL), this is also used (internally).
- Parameter data areas (PDA; one for each client interface object; file extension .NSA) defining the interface to the Natural client interface objects. It is recommended to use the PDA in your Natural client application.
- Sample Natural program (one for each interface object; file extension .NSP) that demonstrate how to call the client interface object. See ***Sample Generation Result for the Client Side*** for more information.
- In addition, a helper local data area (LDA), NATRPCL, can be generated to support correct local compilation in a NaturalONE project in Designer. See ***Step 1: Specify the RPC Environment***.

For Natural server applications it generates the following:

- Natural RPC server skeletons (which are Natural subprograms; file extension .NSN).
- Parameter data areas (PDA; one for each server skeleton; file extension .NSA) defining the interface to the RPC server.
- If the server names (automatically generated or customized) differ from the IDL program names, a client-side mapping file is required. See *Server Mapping Files for Natural*. It is generated during

generation of the RPC Server and has to be used in subsequent steps (wrapped into RPC client components).

To generate these source files, you need a Natural RPC environment:

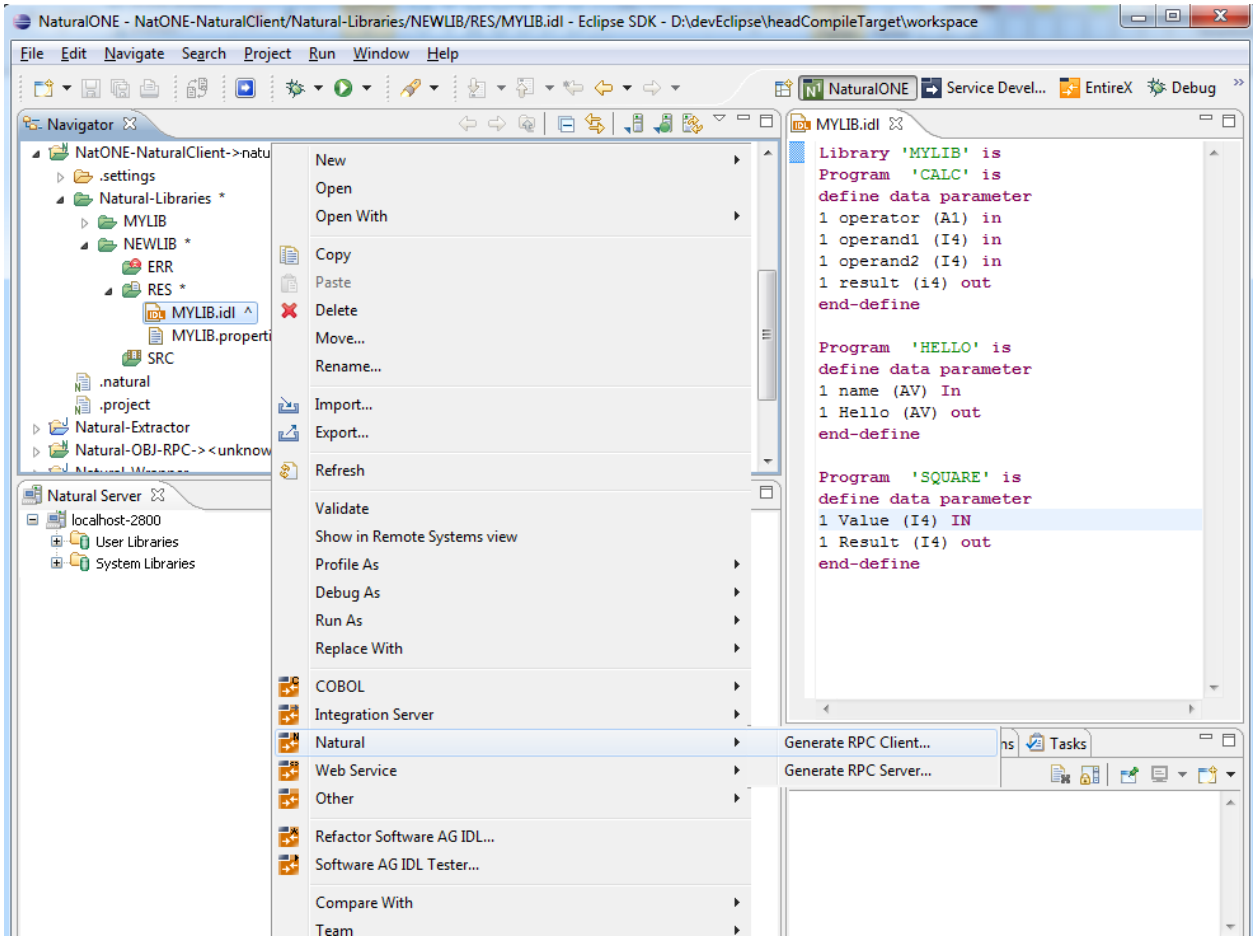


The IDL file is sent to the Natural RPC server where generation takes place. Optionally you can save the generated source files remotely in a Natural target library of the Natural RPC server or locally in the Designer. See *RPC Environment Manager for Natural RPC Server*.

3 Using the Natural Wrapper

Starting the Natural Wrapper

Start the Natural Wrapper from the context menu of an IDL file: **Natural > Generate RPC Client**.

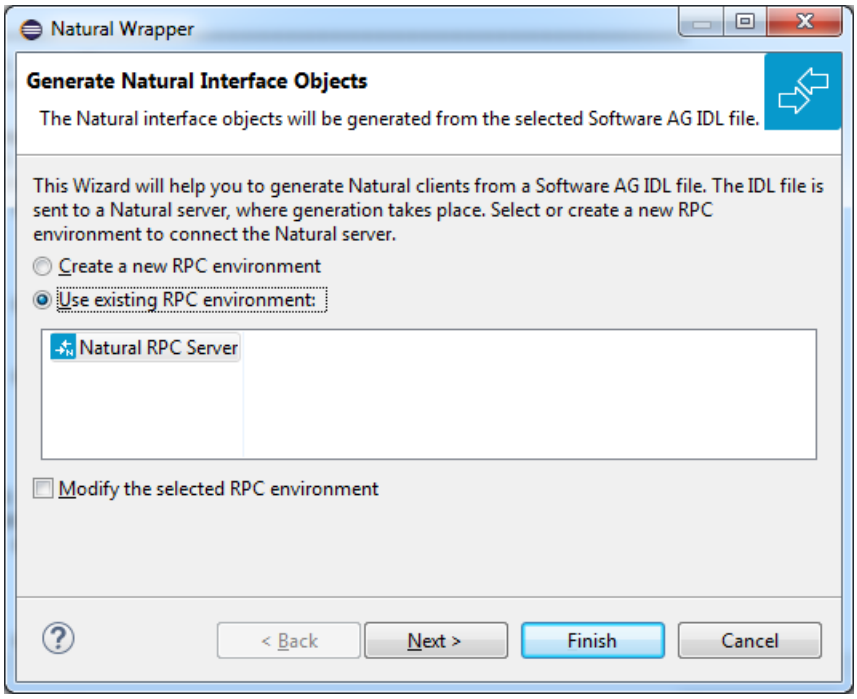


Using the Natural Wrapper for the Client Side

Using the Natural Wrapper for the Client Side within NaturalONE

Step 1: Specify the RPC Environment

The first wizard page prompts you to select an existing RPC environment or define a new one. When running with NaturalONE, a default RPC environment is available. This is typically named "localhost:1971@NATSRV2800" and refers to a running EntireX broker and Natural RPC server on the local machine.



Check **Use existing RPC environment**, select the default RPC environment for NaturalONE and press **Next**. Continue with *Step 2: Customize Natural Client Names*.

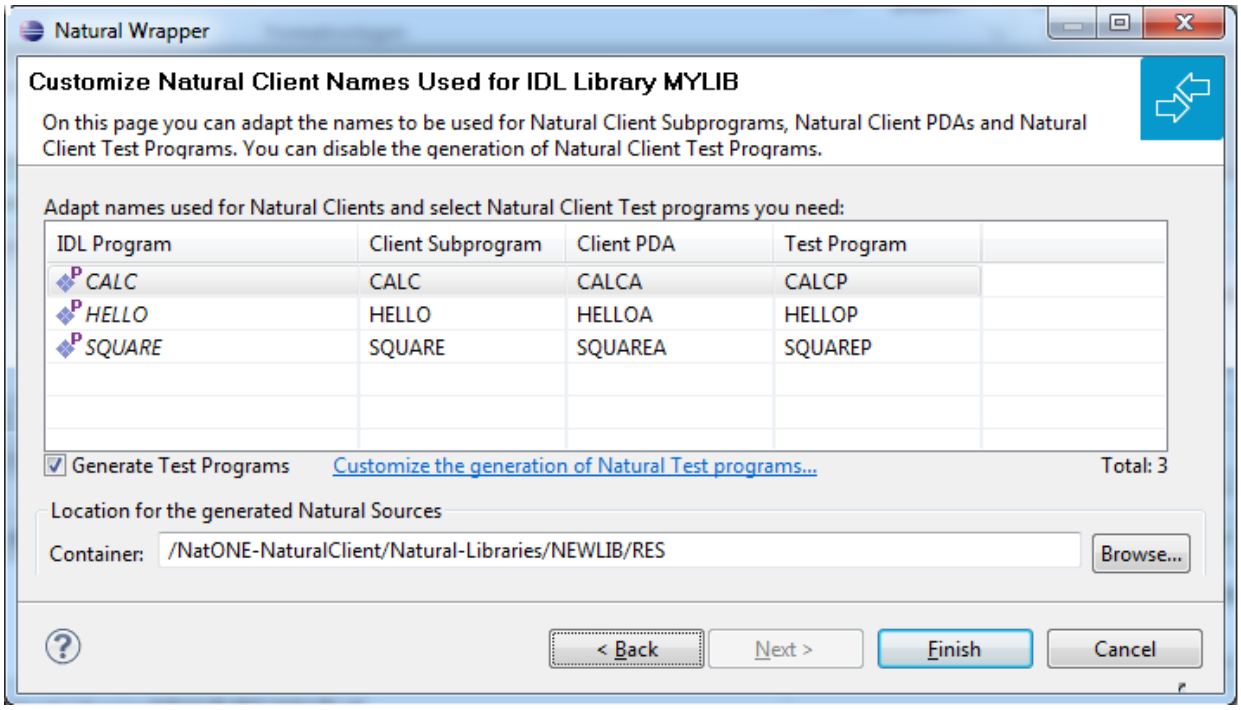
Changing the default RPC environment for NaturalONE is not required and not recommended, so it is best to clear **Modify the selected RPC environment**. More information on changing an

RPC environment can be found in *Step 2: Edit the RPC Environment (Optional)* under *Using the Natural Wrapper for the Client Side with a Remote RPC Environment*.

To use a different Natural RPC server for generation, create a new RPC environment. See *Step 1: Specify the RPC Environment* under *Using the Natural Wrapper for the Client Side with a Remote RPC Environment*.

Step 2: Customize Natural Client Names

On this page, adapt the names for the Natural client interface objects (subprograms (NSNs) with their parameter data areas (PDAs)), names of the client test programs and specify the location to which all Natural sources are to be written. The generation of client test programs can be disabled.



Note: If your IDL file contains more than one IDL library, the additional column **IDL Library** is displayed.

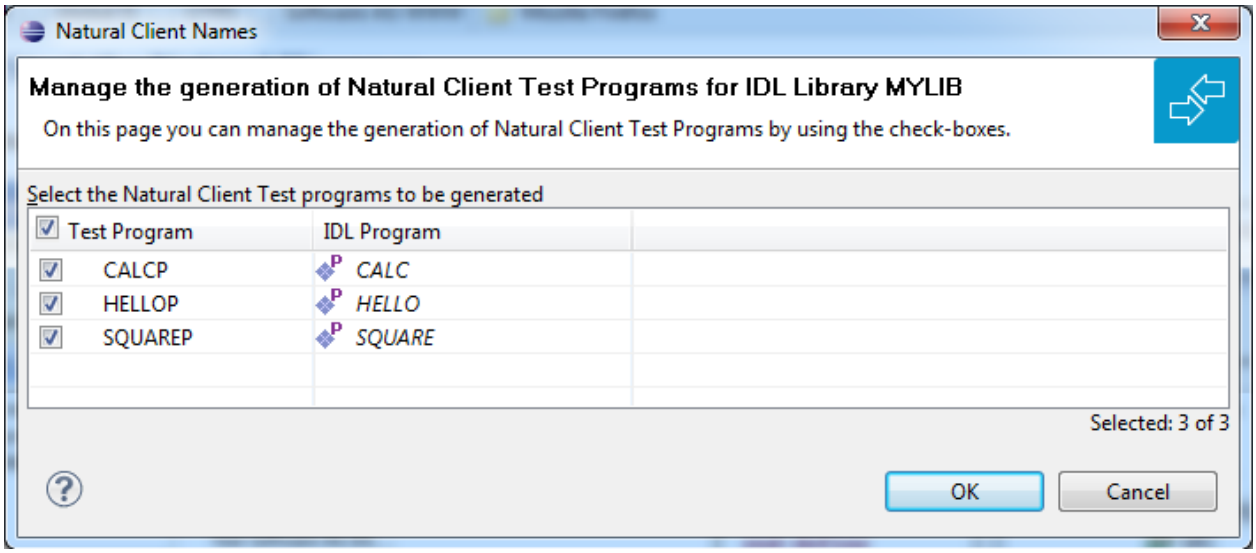
Check **Generate Test Programs** if client test programs for client interface objects are to be generated. Clear this box if no test programs are to be generated. Use the link **Customize the generation of Natural Test programs** to specify for each IDL program whether a test program is generated or not. See *Generate Test Programs* for more information.

For Natural ONE, the **Location for the generated Natural sources** should be in a Natural project. If the target container is a source folder of a Natural project, the Natural builder will automatically compile and build the source.

Press **Next** to start generation.

Generate Test Programs

On this page, specify for each IDL program whether a client test program is generated or not.

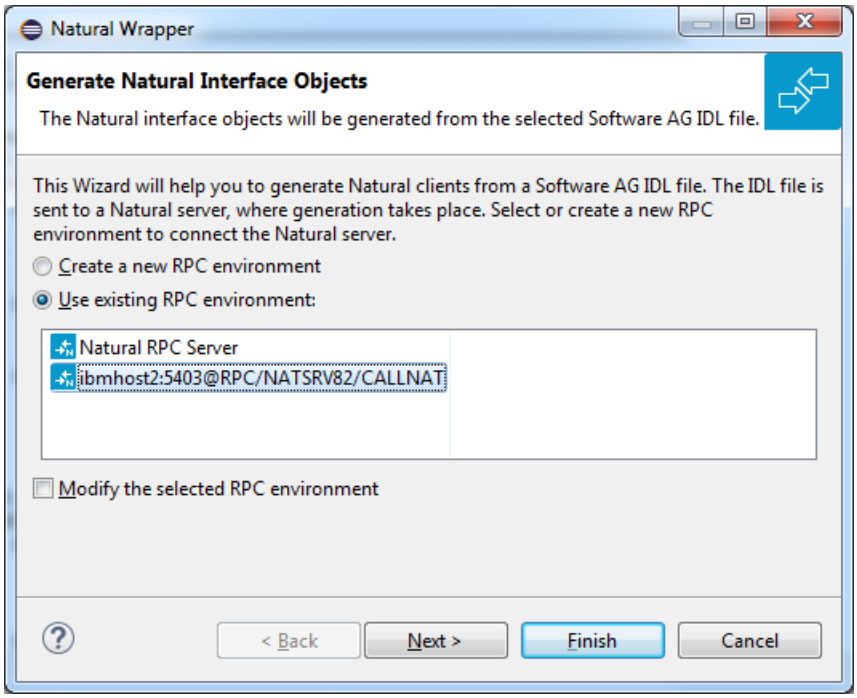


Check the IDL programs for which a client test program is to be generated. If a box is not checked, no client test program is generated for that IDL program. Press **OK** to save the settings.

Using the Natural Wrapper for the Client Side with a Remote RPC Environment

Step 1: Specify the RPC Environment

The first wizard page prompts you to use an existing RPC environment or create a new one.



To use an existing RPC environment

1 Check **Use existing RPC environment**, select the desired RPC environment and press **Next**. If your previously defined filter (see *Step 2: Edit the RPC Environment (Optional)*) matches more than one Natural library, continue with *Step 3: Select a Library (Optional)*. If the filter matches exactly one Natural library, continue with *Step 4a: Customize Natural Client Names and Save Remotely (Optional)*.

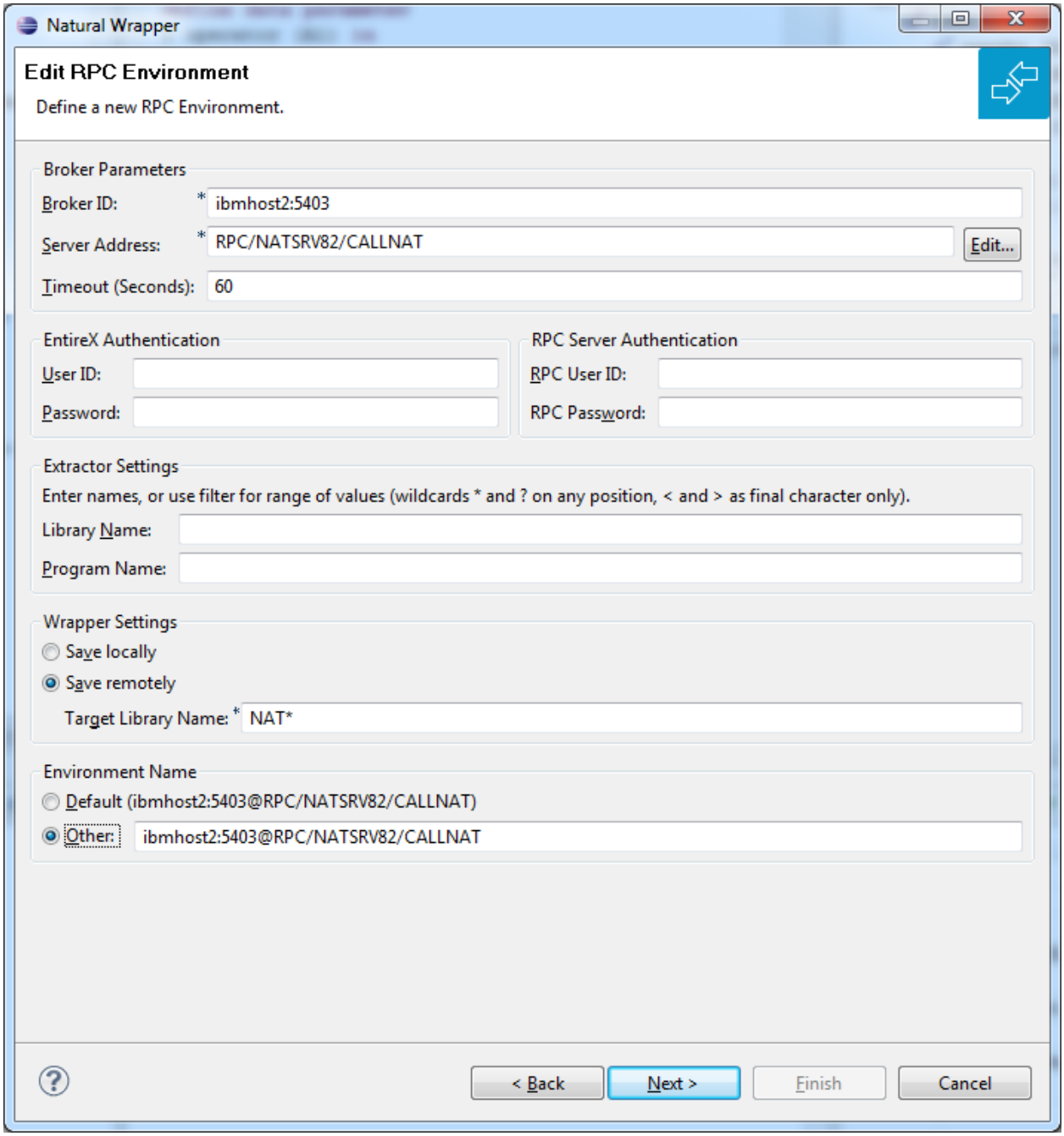
2 Check **Modify the selected RPC environment** if you want to change settings prior to generation. Press **Next** and continue with *Step 2: Edit the RPC Environment (Optional)*.

To create a new RPC environment

- Check **Create a new RPC environment** and press **Next**. Continue with *Step 2: Edit the RPC Environment (Optional)*.

Step 2: Edit the RPC Environment (Optional)

On this page, connection and authentication settings for the EntireX Broker and Natural RPC server are managed, and you define if the Natural objects are saved remotely on the Natural RPC server or locally within Software AG Designer.



Define a new RPC environment or modify an existing one 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 is operating 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.

You can save the generated Natural sources locally or remotely:

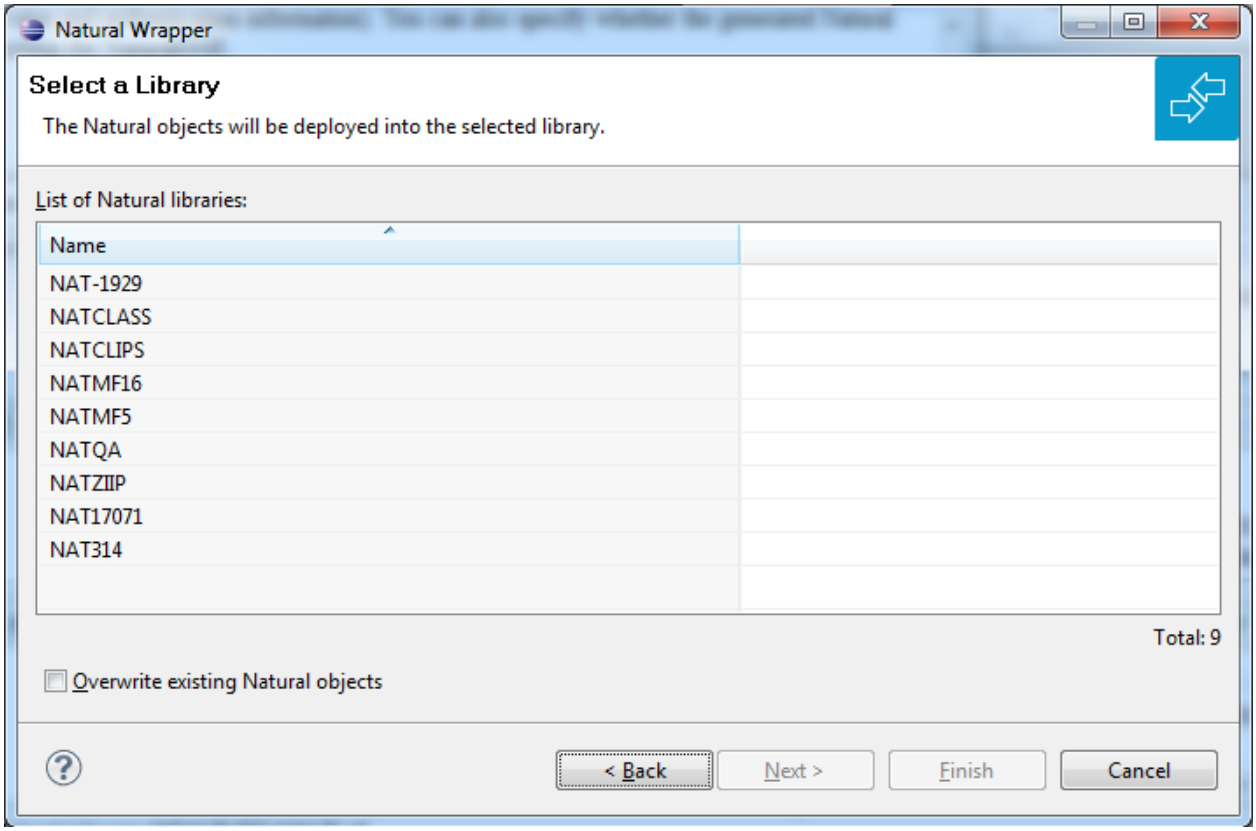
- **Locally**
  Sources are saved locally in the Software AG Designer. This setting is preferred if you develop with NaturalONE and will additionally create a helper LDA, NATRPCL for correct local compilation in a NaturalONE project in Software AG Designer. If you select this option, continue with *Step 4b: Customize Natural Client Names and Save Locally (Optional)* and save locally.
- **Remotely**
  Sources are saved remotely on the Natural RPC server. If you save the Natural sources remotely, you have to specify a target Natural library. Enter either an exact name or use a filter for range of values. The following wildcards are available:
  - asterisk "*" (any position) to list names matching any sequence of characters
  - question mark "?" (any position) to list names matching any single character
  - greater than ">" (final character only) to list names after
  - lower than "<" (final character only) to list names before

  Only Natural libraries that reside in the FUSER system file of the Natural RPC server can be specified. The Natural library specified is not created from the Designer, it must exist in the FUSER system file (at least one Natural object stored here).

  Press **Next**. If your filter matches more than one Natural library, continue with *Step 3: Select a Library (Optional)*. If the filter matches exactly one Natural library, continue with *Step 4a: Customize Natural Client Names and Save Remotely (Optional)* and save remotely.

Step 3: Select a Library (Optional)

All Natural libraries that reside in the FUSER system file of the Natural RPC server and that match the filter defined in *Step 2: Edit the RPC Environment (Optional)* are listed here. This step is skipped if exactly one Natural library matches the filter specification defined in *Step 2: Edit the RPC Environment (Optional)*. In this case, continue with *Step 4a: Customize Natural Client Names and Save Remotely (Optional)*.

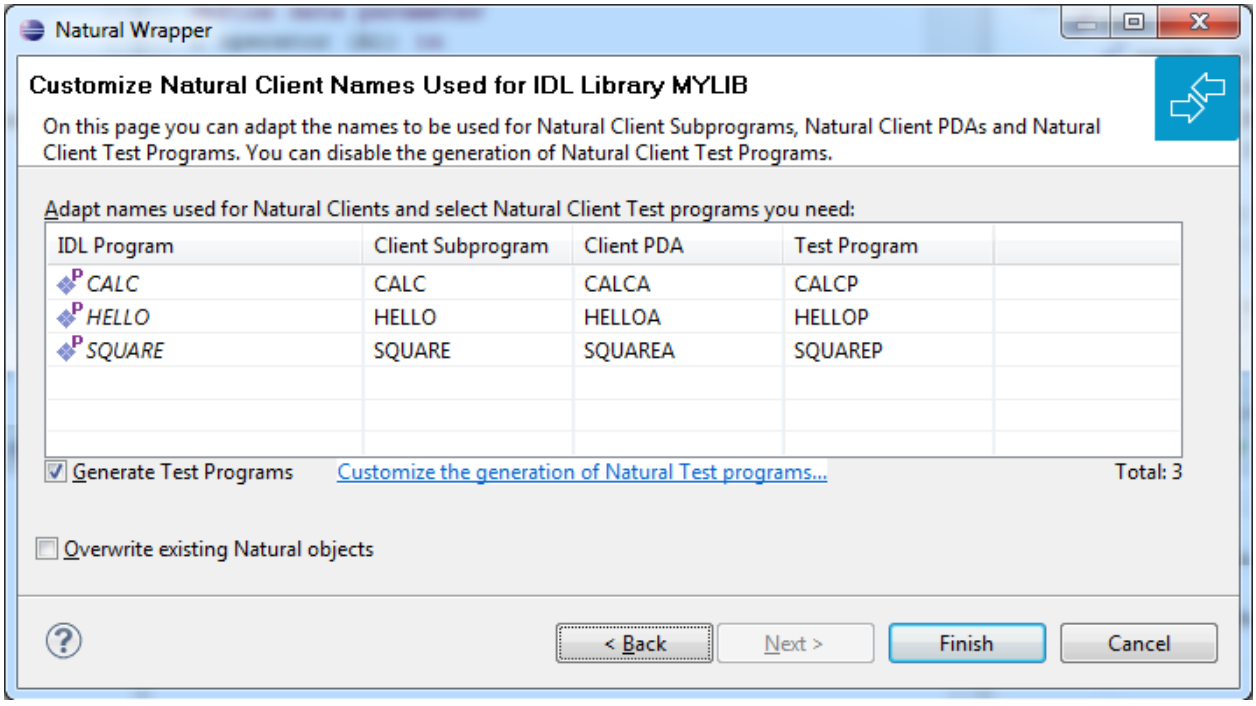


Select the Natural library where you want to save the generated Natural sources and press **Next**. Continue with *Step 4a: Customize Natural Client Names and Save Remotely (Optional)*.

Press **Finish** to skip all preceding steps and start generation immediately. If the generated Natural objects already exist from a previous generation, check **Overwrite existing Natural objects**.

Step 4a: Customize Natural Client Names and Save Remotely (Optional)

On this page, adapt the names for the Natural client interface objects (subprograms (NSNs) with their parameter data areas (PDAs)), and the names of the client test programs. The generation of client test programs can be disabled.



Note: If your IDL file contains more than one IDL library, the additional column **IDL Library** is displayed.

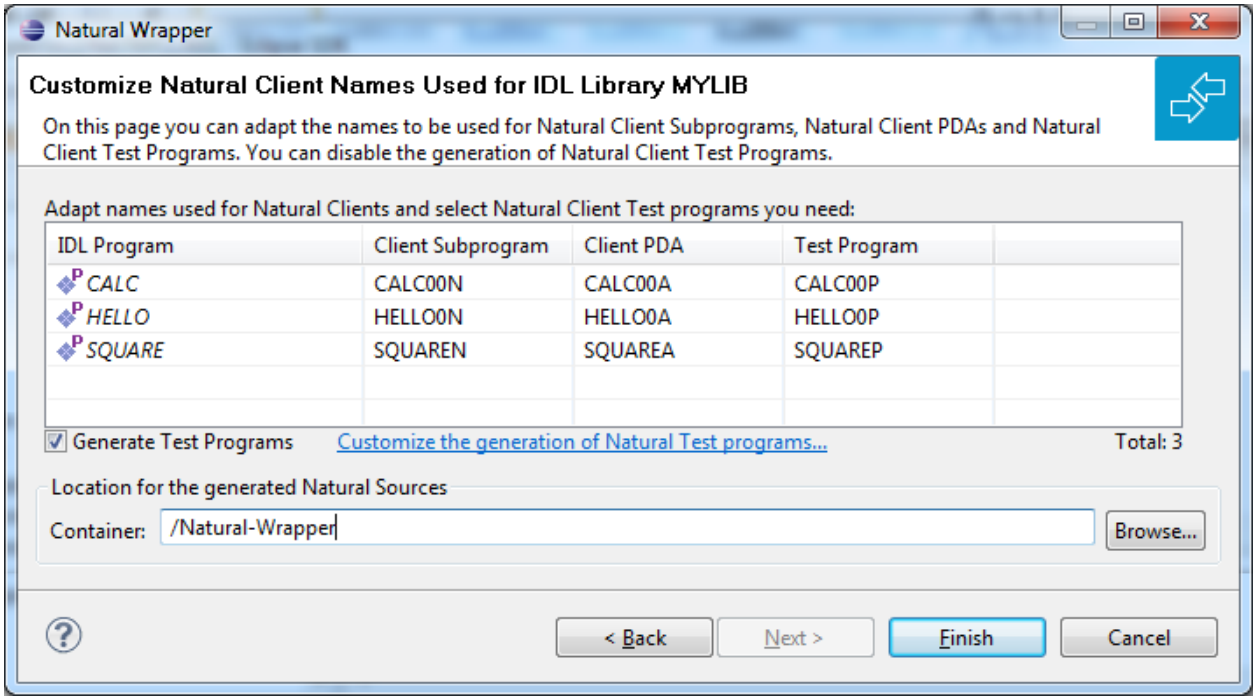
Check **Generate Test Programs** if client test programs for client interface objects are to be generated. Clear this box if no test programs are to be generated. Use the link **Customize the generation of Natural Test programs** to specify for each IDL program whether a test program is generated or not. See *Generate Test Programs* for more information.

If the generated Natural objects already exist from a previous generation, check **Overwrite existing Natural objects**.

Press **Next** to start generation.

Step 4b: Customize Natural Client Names and Save Locally (Optional)

On this page, adapt the names for the Natural client interface objects (subprograms (NSNs) with their parameter data areas (PDAs)), names of the client test programs, and specify the location to which all Natural sources are to be written. The generation of client test programs can be disabled.



Note: If your IDL file contains more than one IDL library, the additional column **IDL Library** is displayed.

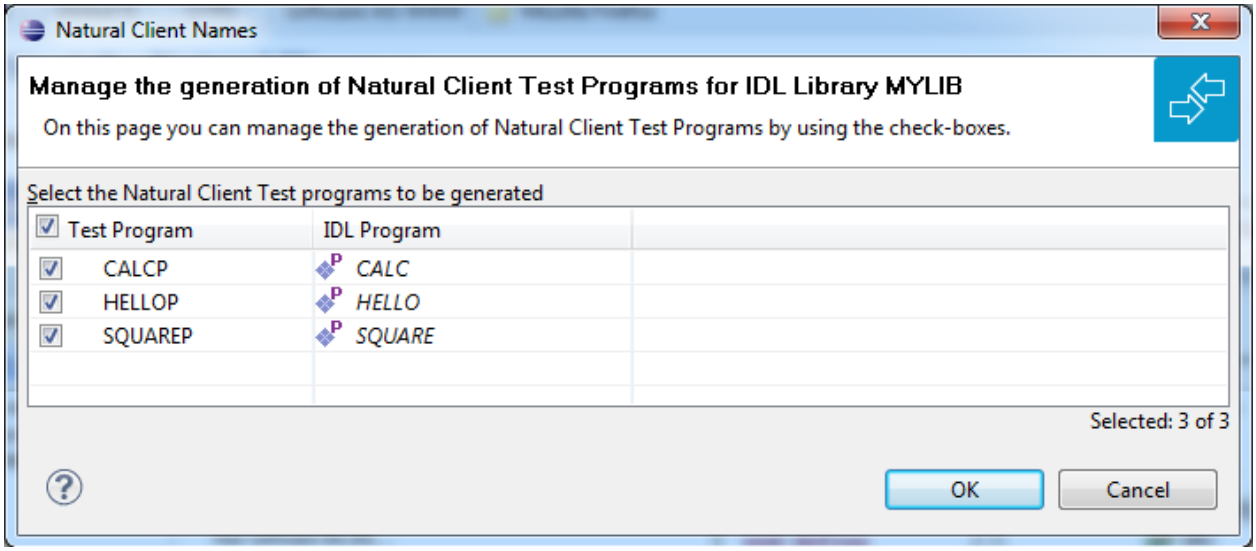
Check **Generate Test Programs** if client test programs for client interface objects are to be generated. Clear this box if no test programs are to be generated. Use the link **Customize the generation of Natural Test programs** to specify for each IDL program whether a test program is generated or not. See *Generate Test Programs* for more information.

Specify the **Location for the generated Natural sources**. Select a container (folder or project) in Software AG Designer.

Press **Next** to start generation. Continue with *Sample Generation Result for the Client Side*.

Generate Test Programs

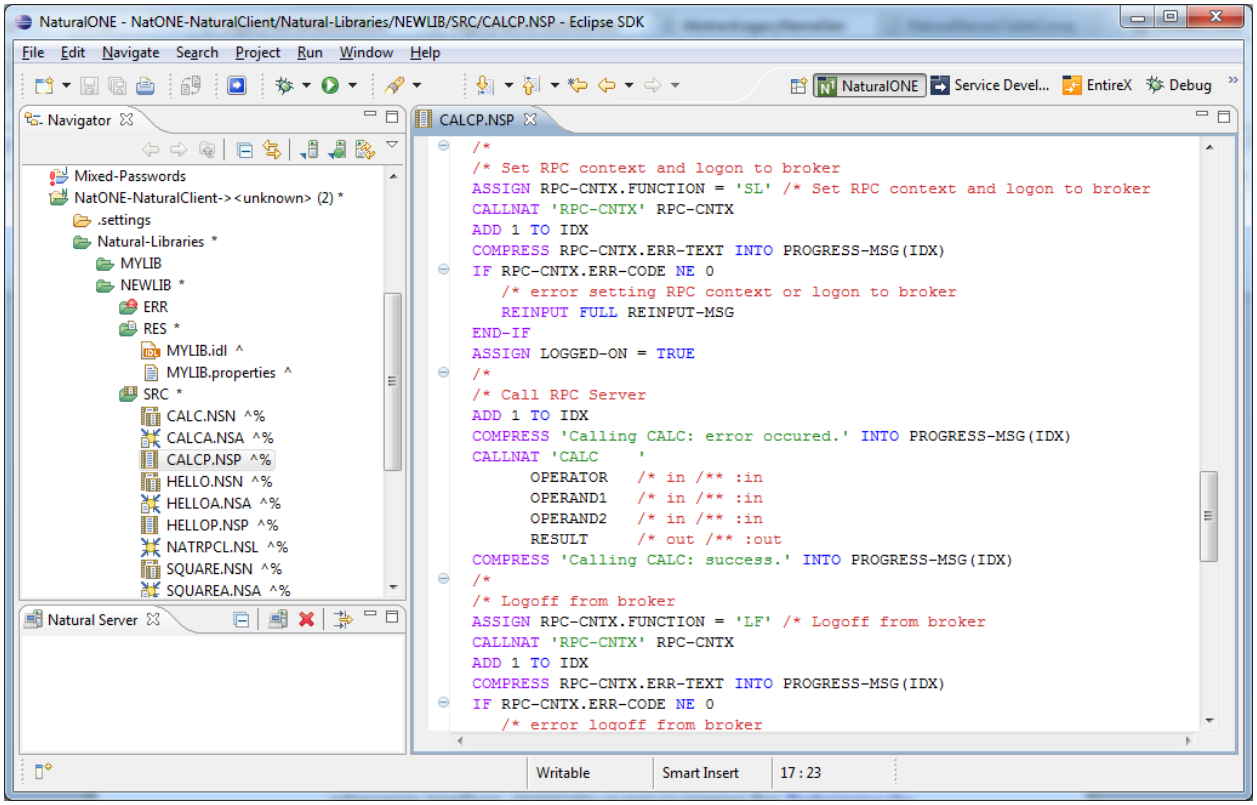
On this page, specify for each IDL program whether a client test program is generated or not.



Check the IDL programs for which a client test program is to be generated. If a box is not checked, no client test program is generated for that IDL program. Press **OK** to save the settings.

Sample Generation Result for the Client Side

The following screen shows the successful generation of Natural client interface objects (subprograms (NSNn) with their parameter data areas (PDAs (extension NSA)), client test program (NSPs) and the common data area NATRPCL.NSL). The **Wrapper Settings** included **Save locally** in a NaturalONE environment.



Using the Generated Sample Natural Program

Before you use the generated sample Natural program (extension .NSP), make sure the Natural library SYSEXT is defined as a steplib for the Natural environment. Use one of the following methods, depending on your environment:

- for NaturalONE, use the context menu of the NaturalONE project and choose **Properties > Natural > Environment, add SYSEXT**
- for Natural Security environments, use SYSSEC
- for any other Natural environment, use NATPARM

The generated sample Natural program (extension .NSP) demonstrates standard RPC usage.

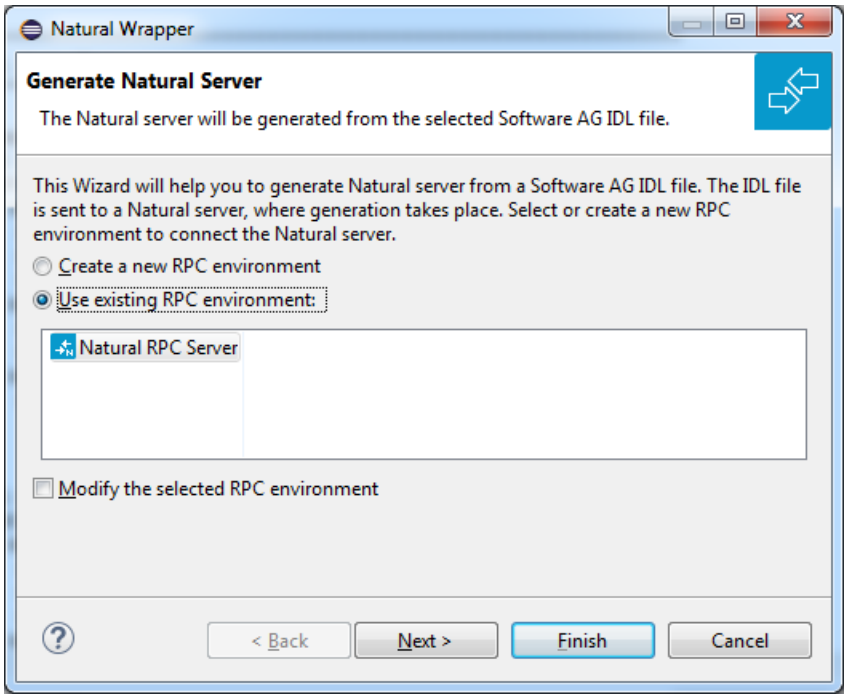
See also the readme files of the *Client and Server Examples for Natural* for more information on working with Natural RPC clients.

Using the Natural Wrapper for the Server Side

Using the Natural Wrapper for the Server Side within NaturalONE

Step 1: Specify the RPC Environment

The first wizard page prompts you to select an existing RPC environment or define a new one. When running with NaturalONE, a default RPC environment is available. This is typically named "localhost:1971@NATSRV2800" and refers to a running EntireX broker and Natural RPC server on the local machine.



Check **Use existing RPC environment**, select the default RPC environment for NaturalONE and press **Next**. Continue with *Step 2: Customize Natural Server Names*.

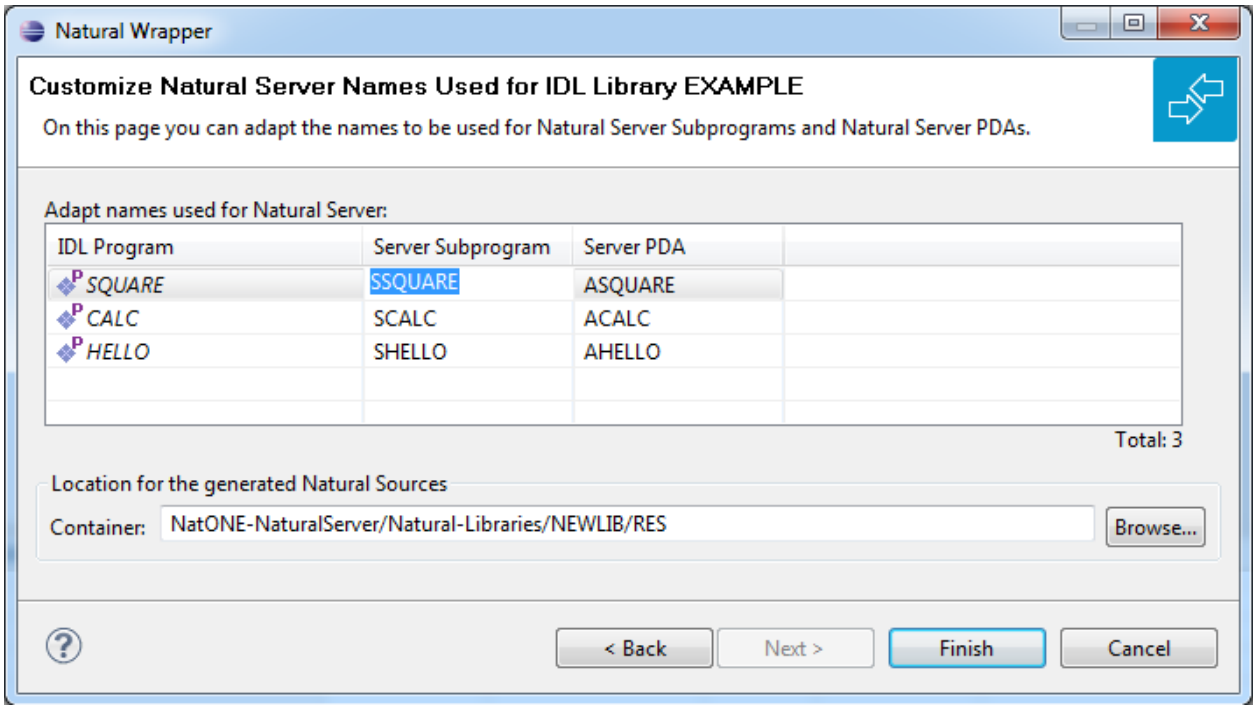
Changing the default RPC environment for NaturalONE is not required and not recommended, so it is best to clear **Modify the selected RPC environment**. More information on changing an

RPC environment can be found in *Step 2: Edit the RPC Environment (Optional)* under *Using the Natural Wrapper for the Server Side with a Remote RPC Environment*.

To use a different Natural RPC server for generation, create a new RPC environment. See *Step 1: Specify the RPC Environment* under *Using the Natural Wrapper for the Server Side with a Remote RPC Environment*.

Step 2: Customize Natural Server Names

On this page, adapt the names for the Natural RPC server skeletons (subprograms (NSNs) with their parameter data areas (PDAs)) and specify the location to which all Natural objects are to be written.



Note: If your IDL file contains more than one IDL library, the additional column **IDL Library** is displayed.

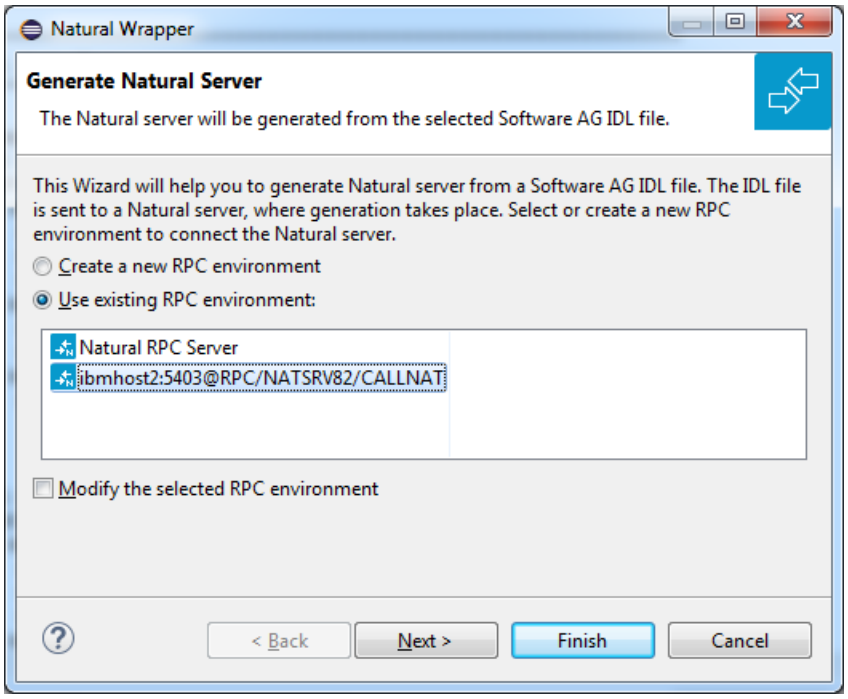
For NaturalONE, the location for the generated Natural sources should be in a Natural project. If the target container is a source folder of a Natural project, the Natural builder will automatically compile and build the source.

Press **Next** to start generation. Continue with *Sample Generation Result for the Server Side*.

Using the Natural Wrapper for the Server Side with a Remote RPC Environment

Step 1: Specify the RPC Environment

The first wizard page prompts you to use an existing RPC environment or create a new one.



To use an existing RPC environment

1 Check **Use existing RPC environment**, select the desired RPC environment and press **Next**. If your previously defined filter (see *Step 2: Edit the RPC Environment (Optional)*) matches more than one Natural library, continue with *Step 3: Select a Library (Optional)*. If the filter matches exactly one Natural library, continue with *Step 4a: Customize Natural Server Names and Save Remotely (Optional)*.

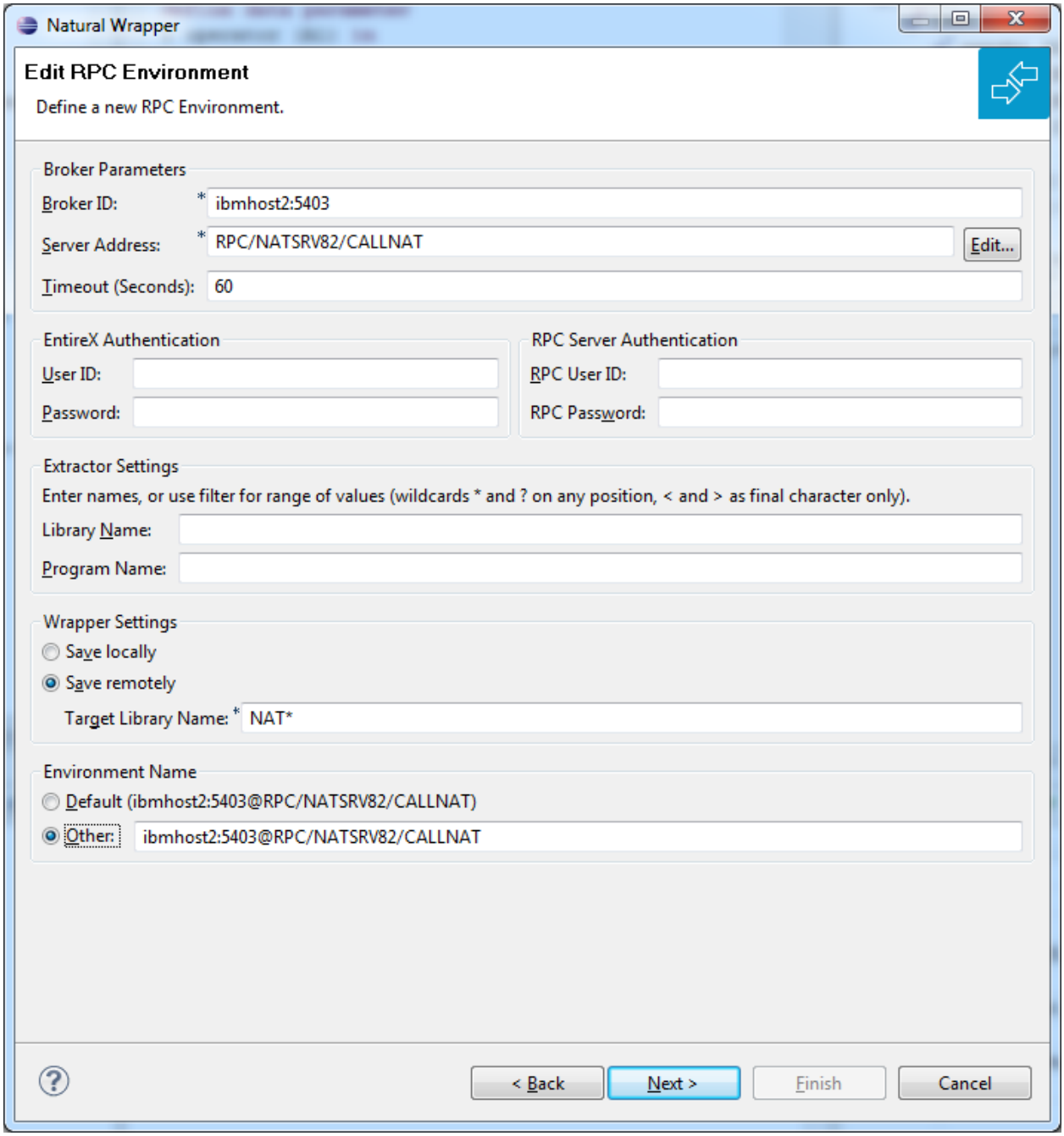
2 Check **Modify the selected RPC environment** if you want to change settings prior to generation. Press **Next** and continue with *Step 2: Edit the RPC Environment (Optional)*.

To create a new RPC environment

- Check **Create a new RPC environment** and press **Next**. Continue with *Step 2: Edit the RPC Environment (Optional)*.

Step 2: Edit the RPC Environment (Optional)

On this page, connection and authentication settings for the EntireX Broker and Natural RPC server are managed, and you define whether the Natural sources are saved remotely on the Natural RPC server or locally within Software AG Designer.



Define a new RPC environment or modify an existing one 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 is operating 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.

You can save the generated Natural sources locally or remotely:

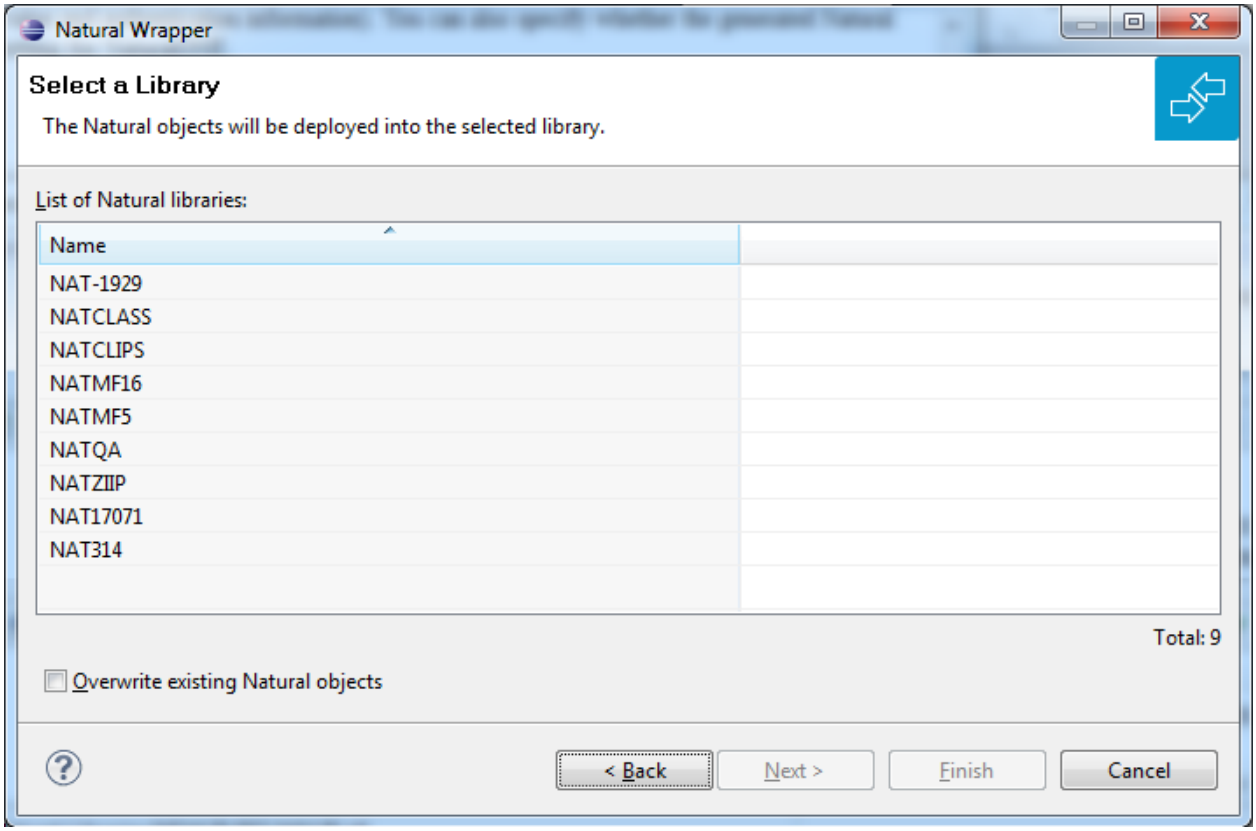
- **Locally**
  Sources are saved locally in the Software AG Designer. This setting is preferred if you develop with NaturalONE. If you select this option, continue with *Step 4b: Customize Natural Server Names and Save Locally (Optional)* and save locally.
- **Remotely**
  Sources are saved remotely on the Natural RPC server. If you save the Natural sources remotely, you have to specify a target Natural library. Enter either an exact name or use a filter for range of values. The following wildcards are available:
  - asterisk "*" (any position) to list names matching any sequence of characters
  - question mark "?" (any position) to list names matching any single character
  - greater than ">" (final character only) to list names after
  - lower than "<" (final character only) to list names before

  Only Natural libraries that reside in the FUSER system file of the Natural RPC server can be specified. The Natural library specified is not created from the Designer, it must exist in the FUSER system file (at least one Natural object stored here).

  Press **Next**. If your filter matches more than one Natural library, continue with *Step 3: Select a Library (Optional)*. If the filter matches exactly one Natural library, continue with *Step 4a: Customize Natural Server Names and Save Remotely (Optional)* and save remotely.

Step 3: Select a Library (Optional)

All Natural libraries that reside in the FUSER system file of the Natural RPC server and that match the filter defined in *Step 2: Edit the RPC Environment (Optional)* are listed here. This step is skipped if exactly one Natural library matches the filter specification defined in *Step 2: Edit the RPC Environment (Optional)*. In this case, continue with *Step 4a: Customize Natural Server Names and Save Remotely (Optional)*.

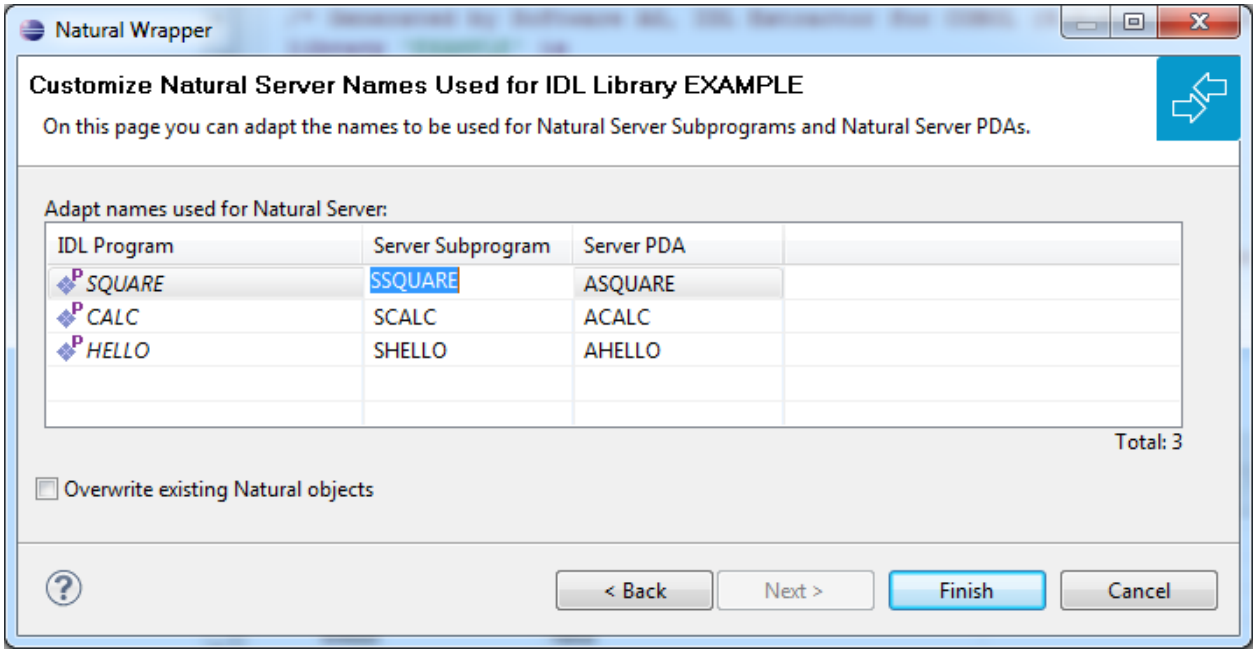


Select the Natural library where you want to save the generated Natural sources and press **Next**. Continue with *Step 4a: Customize Natural Server Names and Save Remotely (Optional)*.

The **Finish** is enabled if no related XMM file exists. If you press **Finish** and the generated Natural objects already exist from a previous generation, check **Overwrite existing Natural objects**.

Step 4a: Customize Natural Server Names and Save Remotely (Optional)

On this page, adapt the names for the Natural RPC server skeletons (subprograms (NSNs) with their parameter data areas (PDAs)).



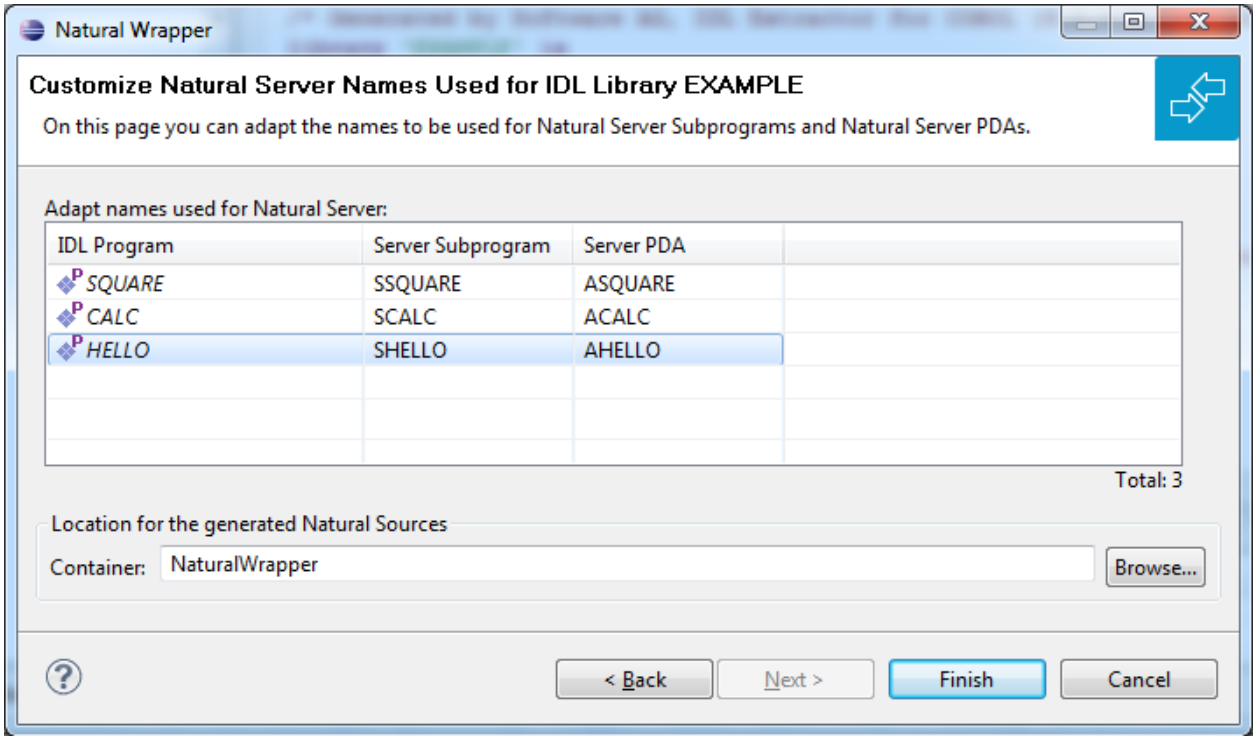
Note: If your IDL file contains more than one IDL library, the additional column **IDL Library** is displayed.

If the generated Natural objects already exist from a previous generation, check **Overwrite existing Natural objects**.

Press **Next** to start generation.

Step 4b: Customize Natural Server Names and Save Locally (Optional)

On this page, adapt the names for the Natural RPC server skeletons (subprograms (NSNs) with their parameter data areas (PDAs)) and specify the location to which all Natural sources are to be written.



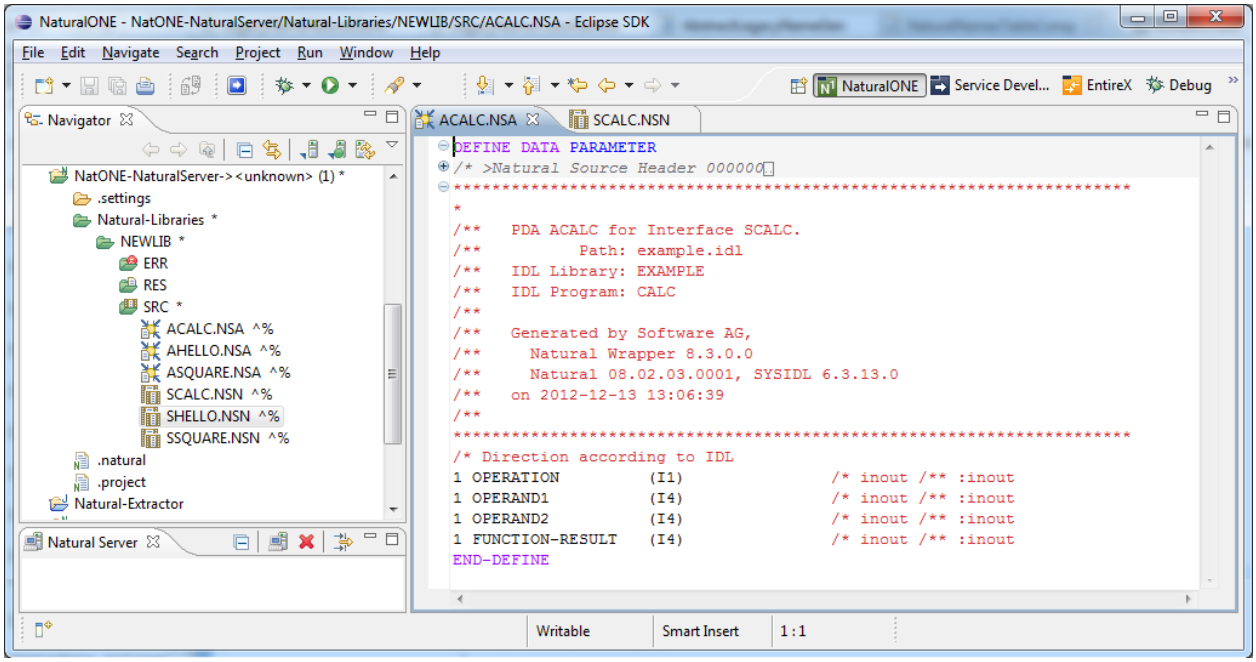
Note: If your IDL file contains more than one IDL library, the additional column **IDL Library** is displayed.

Specify the **Location for the generated Natural sources**. Select a container (folder or project) in Software AG Designer.

Press **Next** to start generation. Continue with *Sample Generation Result for the Server Side*.

Sample Generation Result for the Server Side

The following screen shows the successful generation of the Natural RPC server skeletons (subprograms (NSNs) with their parameter data areas (PDAs (extension NSA))). The **Wrapper Settings** included **Save locally in a NaturalONE environment**.



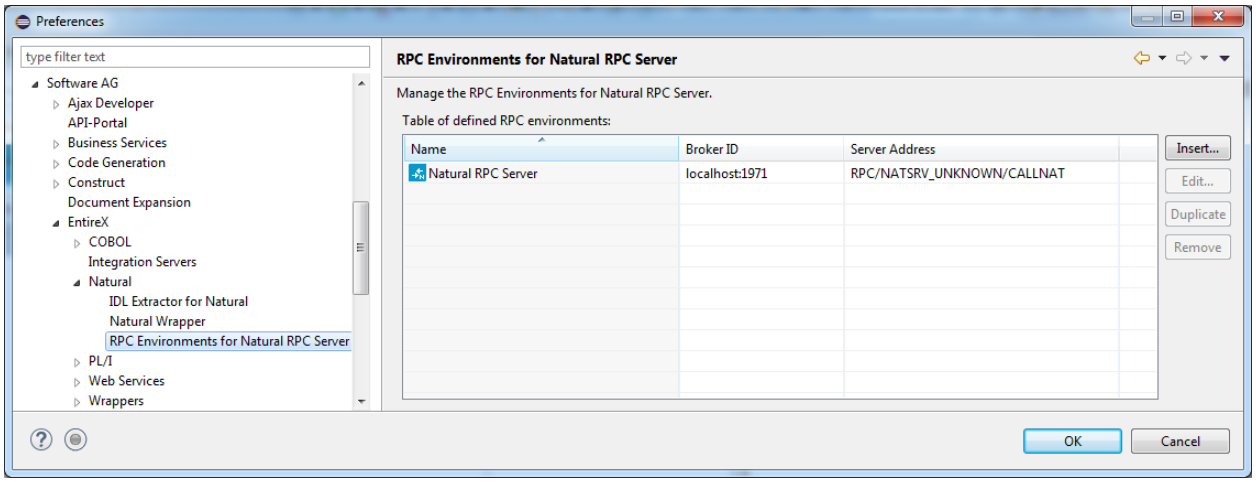
See also the readme files of the *Client and Server Examples for Natural* for more information on working with Natural RPC servers.

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 for Natural RPC Server* 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

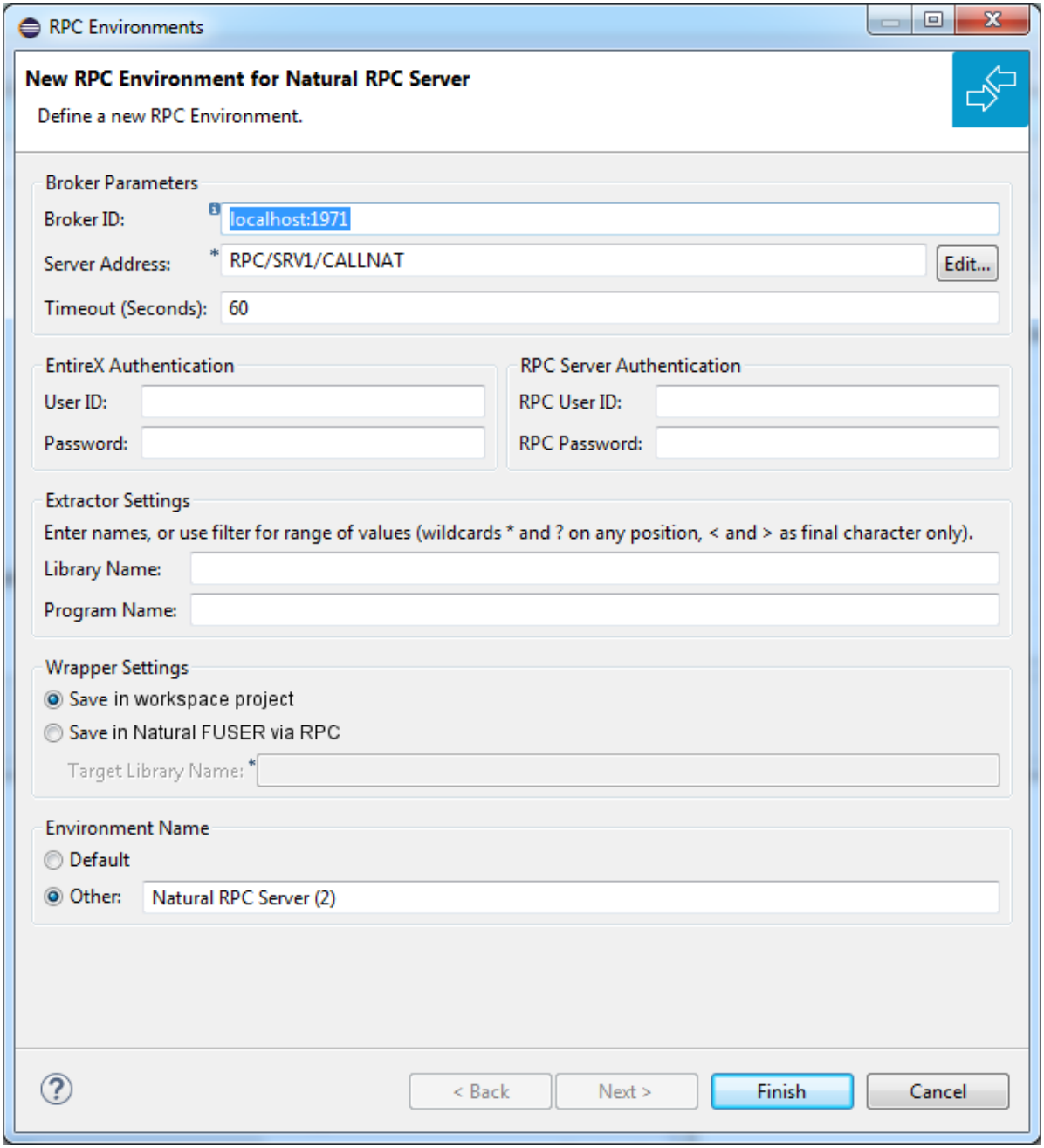
- Select the table row and press **Edit...**. If multiple entries are selected, the first entry is used.

To remove an RPC environment

- Slect 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.

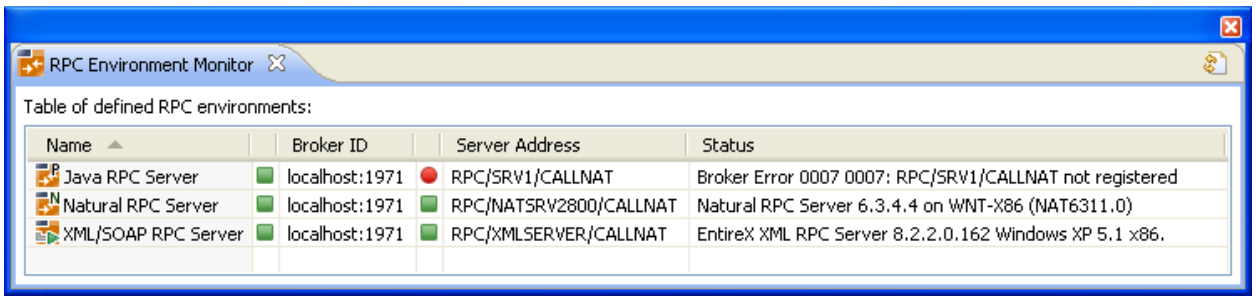
To open the RPC Environment Monitor from the EntireX perspective

- 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 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 Natural Wrapper in Command-line Mode

Command-line Options

This section provides the command-line options for the following tasks:

- Generating a Natural RPC Client from an IDL File
- Generating a Natural RPC Server from an IDL File

See *Using EntireX in the Designer Command-line Mode* for the general command-line syntax.

Generating a Natural RPC Client from an IDL File

To generate a Natural RPC client from the specified IDL file, use the following command with options in table below:

```
-natural:client
```

Option	Description
`-brokerpassword`	Password used for broker authentication.
`-brokeruser`	User used for broker authentication.
`-environment`	Name of the environment or an RPC server description.
`-help`	Display this usage message.
`-operationtype`	The operation type. Valid values are: `SAVE` Generate Natural interface objects remotely on the server side `GET` Generate Natural interface objects locally using Software AG Designer
`-overwrite`	Overwrite existing Natural interface objects on the server side (`SAVE` command only).
`-targetlibrary`	The target library for the Natural interface objects on the server side (`SAVE` command only).
`-rpcpassword`	Password used for RPC server authentication.
`-rpcuser`	User used for RPC server authentication.

Generating a Natural RPC Server from an IDL File

To generate a Natural RPC server from the specified IDL file, use the following command with options in table below:

```
-natural:server
```

Option	Description
`-brokerpassword`	Password used for broker authentication.
`-brokeruser`	User used for broker authentication.
`-environment`	Name of the environment or an RPC server description.
`-help`	Display this usage message.
`-operationtype`	The operation type. Valid values are: `SAVE` Generate Natural interface objects remotely on the server side. `GET` Generate Natural interface objects locally using Software AG Designer.
`-overwrite`	Overwrite existing Natural interface objects on the server side (SAVE command only).
`-targetlibrary`	The target library for the Natural interface objects on the server side (SAVE command only).
`-rpcpassword`	Password used for RPC server authentication.
`-rpcuser`	User used for RPC server authentication.

Example: Generating an RPC Client

```
<workbench> -natural:client /Demo/Example.idl -environment localhost:1971@SRV1 ↩
-operationtype SAVE -targetlibrary MYLIB
```

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

The name of the IDL file includes the project name. In the example, the project *Demo* is used. If the IDL file name describes a file inside the Eclipse workspace, the name is case-sensitive.

If the first part of the IDL file name is not a project name in the current workspace, the IDL file name is used as a relative (based on the IDL file) or absolute file name in the file system. Thus, the IDL files do not need to be part of an Eclipse project.

Example: Generating an RPC Server

```
<workbench> -natural:server /Demo/Example.idl -environment localhost:1971@SRV1 ↩
-operationtype SAVE -targetlibrary MYLIB
```

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

Caution: Take care not to overwrite an existing server implementation with a server skeleton. We recommend you move your server implementation to a different folder.

Further Examples

Example 1

```
<workbench> -natural:client /Demo/example.idl -environment localhost:1971@SRV1 ↩
-operationtype GET
```

Uses the IDL file */Demo/example.idl* and generates the Natural source files in parallel to the IDL file of the project */Demo*. Output to standard output:

```
Using workspace file:/C:/myWorkspace/.
Processing IDL file C:/myWorkspace/Demo/example.idl to get the Natural interface ↩
objects via RPC environment localhost:1971@SRV1
Store Natural Source file C:\myWorkspace\Demo/CALC.NSN
Exit value: 0
```

Example 2

```
<workbench> -natural:client /Demo/example.idl -environment localhost:1971@SRV1 ↩
-operationtype SAVE -targetlibrary TEST
```

Uses the IDL file */Demo/example.idl* and generates the Natural source files on the server side into the library TEST. Output to standard output:

```
Using workspace file:/C:/myWorkspace/.
Processing IDL file C:/myWorkspace/Demo/example.idl to stow the Natural interface ↩
objects via RPC environment localhost:1971@SRV1
Exit value: 0
```

7 Software AG IDL to Natural Mapping

This chapter describes the specific mapping of Software AG IDL data types, groups, arrays and structures to the Natural programming language. Please note also the remarks and hints on the IDL data types valid for all language bindings found in the IDL file. See *Software AG IDL File* in the IDL Editor documentation.

Mapping IDL Data Types to Natural Data Formats

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

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

Software AG IDL	Description	Natural Data Format	Notes
A*number*	Alphanumeric	A*number*	
AV	Alphanumeric variable length	A DYNAMIC	
AV*number*	Alphanumeric variable length with maximum length	A DYNAMIC	
B*number*	Binary	B*number*	
BV	Binary variable length	B DYNAMIC	
BV*number*	Binary variable length with maximum length	B DYNAMIC	
D	Date	D	3,5
F4	Floating point (small)	F4	2
F8	Floating point (large)	F8	2
I1	Integer (small)	I1	
I2	Integer (medium)	I2	
I4	Integer (large)	I4	
K*number*	Kanji	A*number*	1
KV	Kanji variable length	A DYNAMIC	1
KV*number*	Kanji variable length with maximum length	A DYNAMIC	1
L	Logical	L	
N*number1*[.*number2*]	Unpacked decimal	N*number1*[.*number2*]	6
NU*number1*[.*number2*]	Unpacked decimal unsigned	N*number1*[.*number2*]	6
P*number1*[.*number2*]	Packed decimal	P*number1*[.*number2*]	6
PU*number1*[.*number2*]	Packed decimal unsigned	P*number1*[.*number2*]	6
T	Time	T	4,5
U*number*	Unicode	U*number*	
UV	Unicode variable length	U DYNAMIC	

Software AG IDL	Description	Natural Data Format	Notes
UV*number*	Unicode variable length with maximum length	U DYNAMIC	

See also the hints and restrictions valid for all language bindings under *IDL Data Types*.

Notes

1. Data type K is an RPC-specific data format that is not part of the Natural language.
2. When floating-point data types are used, rounding errors can occur, so that the values of senders and receivers might differ slightly. This is especially true if client and server use different representations for floating point data (IEEE, HFP).
3. Count of days AD (anno domini, after the birth of Christ). The valid range is from 1.1.0001 up to 28.11.2737. Mapping of the number to the date in the complete range from 1.1.0001 on, follows the Julian and Gregorian calendar, taking into consideration the following rules:

   1. Years that are evenly divisible by 4 are leap years.
   2. Years that are evenly divisible by 100 are not leap years unless rule 3, below, is true.
   3. Years that are evenly divisible by 400 are leap years.
   4. Before the year 1582 AD, rule 1 from the Julian calendar is used. After the year 1582 AD, rules 1, 2 and 3 of the Gregorian calendar are used.

   See the following table for the relation of the packed number to a real date:

Date / Range of Dates	Value / Range of Values
1.1.0000	0 (special value - no date)
undefined dates	1 - 364 (do not use)
1.1.0001	365
1.1.1970	719527 (start of C-time functions)
28.11.2737	999999 (maximum date)

4. Count of tenths of seconds AD (anno domini, after the birth of Christ). The valid range is from 1.1.0001 00:00:00.0 up to 16.11.3168 9:46:39 plus 0.9 seconds. See the following table for the relation of the packed number to a real time:

Time / Range of Times	Value / Range of Values
1.1.0000 00:00:00.0	0 (special value - no time)
undefined times	1 - 315359999
1.1.0001 00:00:00.0	315360000
1.1.1970 00:00:00.0	621671328000 (start of C-time functions)

5. The relation between the packed number of a Date and Time data type is as follows:

   tenths of a second per day = 24*60*60*10 = 864000

   number of time = number of date * 864000

315360000	= 365	* 864000 1.1.0001 00:00:00.0
621671328000	= 719527	* 864000 1.1.1970 00:00:00.0
number of date	= number of time	/ 864000
365	= 315360000	/ 864000 1.1.0001
719527	= 621671328000	/ 864000 1.1.1970

6. For Natural, the total number of digits (*number1*+*number2*) is restricted to 29.

   If you connect two endpoints, the total number of digits used must be lower or equal than the maxima of both endpoints. For the supported total number of digits for endpoints, see the notes under data types N, NU, P and PU in section *Mapping Software AG IDL Data Types* in the respective Wrapper or language-specific documentation.

Mapping Library Name and Alias

Client Side

If you are using client interface objects (recommended) generated for the client side (see ***Using the Natural Wrapper for the Client Side***), the IDL library name as specified in the IDL file (there is no 8 character limitation) is sent from a Natural client to the server. Special characters are not replaced. The IDL library alias is neither sent to the server nor used for other purposes on the Natural client side.

If you are using instead so-called stubs generated with SYSRPC (not recommended) or stubless Natural RPC (also not recommended), the IDL library name as specified in the IDL file is not supported by Natural. By default, a Natural client sends the library name SYSTEM to the server. To send a library name other than SYSTEM from a Natural client to a server, the following steps are required for the client:

- Turn on the logon option.
- Call application programming interface USR4008N to specify the name of the library, otherwise the name of the current library is sent. The length of the library name sent is limited to 8 characters.

Server Side

A Natural RPC Server considers the library name (up to 8 characters) received from an RPC client only if the Natural Logon option is set by the sending RPC client.

- If the RPC client *does not* set its Natural Logon option, the Library name is ignored by the Natural RPC server and the target RPC server (Natural subprogram, extension .NSN) is searched in the startup Natural library and defined steplibs of the Natural RPC server.
- If the RPC client *does* set its Natural Logon option, the target RPC server (Natural subprogram, extension .NSN) must reside in the Natural library, specified as IDL library name in the IDL file.

EntireX RPC client components provide a possibility to set the Natural Logon option. See *Natural Logon or Changing the Library Name* and the documentation of the EntireX RPC client component.

Mapping Program Name and Alias

Client Side

If you are using client interface objects (recommended) generated with the Natural Wrapper for the client side (see ***Using the Natural Wrapper for the Client Side***), the IDL program name as specified in the IDL file (there is no 8-character limitation) is sent from a Natural client to the server. Special characters are not replaced. The IDL program alias is not sent to the server, but it is used to derive the suggestion for the source file names of the client interface objects (NSN, PDA, PGM) instead using the IDL program names. See ***Step 2: Customize Natural Client Names***.

If you are using instead so-called stubs generated with SYSRPC (not recommended) or stubless Natural RPC (also not recommended) the IDL program name as specified in the IDL must match the name in your CALLNAT statement. In this case the IDL program is limited to 8 characters. Example:

```
CALLNAT 'MYSRV' P1 P2 P3
```

Server Side

If you are using a server mapping file (see *Server Mapping Files for Natural*), the target RPC server (Natural subprogram, extension .NSN) is located with the help of this file. This has the following advantages:

- IDL program names do not have to match the target RPC server (Natural subprogram, extension .NSN) names.

- Target RPC server names (Natural subprogram, extension .NSN) can be customized during wrapping (see ***Step 2: Customize Natural Server Names***) or during extraction (see *Step 6: Redesign the Interface for Natural Subprograms (Optional)*).
- IDL program names are not limited to 8 characters.

The server mapping file is generated either during wrapping (see ***Using the Natural Wrapper for the Server Side***) or during extraction (see IDL Extractor for Natural). It is wrapped into the RPC client components and the relevant information is sent from a client to the server. Therefore it is important to generate or extract the target Natural RPC (Natural subprogram, extension .NSN) server first, before creating any RPC client component.

If you are *not* using a server mapping file, the target RPC server (Natural subprogram, extension .NSN) must match the IDL program name. In this case the length of the IDL program name is limited to 8 characters.

Mapping Parameter Names

The parameter names as given in the IDL file are replaced by artificial names in the generated Natural interface object (stub subprogram). See `parameter-data-definition` under *Software AG IDL Grammar* in the IDL Editor documentation.

Mapping Fixed and Unbounded Arrays

- Fixed arrays within the IDL file are mapped to fixed Natural arrays. The lower bound is set to 1 and the upper bound is set to the upper bound given in the IDL file.

  See `array-definition` under *Software AG IDL Grammar* in the IDL Editor documentation for the syntax on how to describe fixed arrays within the IDL file and refer to `fixed-bound-array-index`.
- Unbounded arrays within the IDL file are mapped to Natural X-arrays. The lower bound is always fixed and set to 1.

  See `array-definition` under *Software AG IDL Grammar* in the IDL Editor documentation for the syntax of unbounded arrays within the IDL file and refer to `unbounded-array-index`.

Mapping Groups and Periodic Groups

Groups within the IDL file are mapped to Natural groups. See the `group-parameter-definition` under *Software AG IDL Grammar* in the IDL Editor documentation for the syntax on how to describe groups within the IDL file.

Mapping Structures

Structures within the IDL file are mapped to Natural groups. See `structure-definition` under *Software AG IDL Grammar* in the IDL Editor documentation for the syntax on how to describe structures within the IDL file.

Mapping the Direction Attributes In, Out, InOut

The IDL syntax allows you to define parameters as `IN` parameters, `OUT` parameters, or `IN OUT` parameters (which is the default if nothing is specified). This direction specification is reflected by Natural as follows:

- Parameters with the `OUT` attribute are sent from the RPC client to the RPC server. They are always provided with the call by reference method.
- Parameters with the `IN` attribute are sent from the RPC server to the RPC client. They are always provided with the call by reference method.
- Parameters with the `IN OUT` attribute are sent from the RPC client to the RPC server and then back to the RPC client.
- Only the direction information of the top-level fields (level 1) is relevant. Group fields always inherit the specification from their parent. A different specification is ignored.

See the `attribute-list` under *Software AG IDL Grammar* in the IDL Editor documentation for the syntax on how to describe attributes within the IDL file and refer to `direction` attribute.

Note: If you define an interface object layout in the Natural application SYSRPC, the meaning of the direction attributes `IN` and `OUT` are reversed compared to the IDL:

- `IN` in SYSRPC is `OUT` in IDL
- `OUT` in SYSRPC is `IN` in IDL

Mapping the ALIGNED Attribute

The `ALIGNED` attribute is not relevant for the programming language Natural. However, a Natural client can send the `ALIGNED` attribute to an RPC server where it might be needed. To do this you need a Natural interface object (stub subprogram) that has been generated from an IDL file.

See the `attribute-list` under *Software AG IDL Grammar* in the IDL Editor documentation for the syntax of attributes in the IDL file and refer to the `aligned` attribute.

Calling Servers as Procedures or Functions

The IDL syntax allows definitions of procedures only. It does not have the concept of a function. A function is a procedure which, in addition to the parameters, returns a value. Procedures and functions are transparent between clients and servers. This means a client using a function can call a server implemented as a procedure, and vice versa.

Client and Server Side

The Natural RPC does not support functions.

8 Writing Applications with the Natural Wrapper

Writing RPC Clients for RPC-ACI Bridge in Natural

Prerequisites

The parameters of the IDL program must contain an `A` field with a length of at least 254 bytes or of unlimited length. You need to have dynamic data types in your IDL. You must use `COMPR=2` (send buffer completion).

Writing a Natural Client

To write a Natural client

1 Use `CALLNAT` for the Natural RPC client.

2 Provide a context for EntireX as described under ***Interface `RPC-CNTX` for the Natural RPC Client Programmer*** below. See the Natural documentation for details.

The RPC-ACI Bridge reports errors on the ACI side or the Broker for ACI with Natural error 998 (Internal error details). The details contain the error number followed by a description. Use Natural subroutine `USR2030N` to retrieve error class, error code and description. The error class for the RPC-ACI Bridge is 1018. See *Message Class 1018 - EntireX RPC-ACI Bridge*.

For error handling, use for example (the declarations of the variables are omitted):

```
ON ERROR
  IF *ERROR-NR = 998 THEN
    CALLNAT 'USR2030N' ERR-PARM(*) OCC RESPONSE
    IF OCC > 0 THEN
      MOVE SUBSTRING(ERR-PARM(1), 1, 8) to #ERR-CODE
      MOVE SUBSTRING(ERR-PARM(1), 10) to #ERR-DETAIL
      WRITE #ERR-CODE
      WRITE #ERR-DETAIL (AL=79)
      ESCAPE ROUTINE
    END-IF
  END-IF
END-ERROR
```

Interface RPC-CNTX for the Natural RPC Client Programmer

API RPC-CNTX is used for providing a context for RPC client applications. RPC-CNTX combines the functionality of several APIs and is available in library SYSTEM. There is no need for extra preparations such as setting a STEPLIB or copying APIs from SYSEXT to user libraries.

RPC-CNTX makes the following interfaces from SYSEXT obsolete: USR1071N, (USR4371N), USR6304N, USR2007N, USR4008, USR4009, USR2071N.

For the usage of RPC-CNTX refer to the test programs generated by the Natural Wrapper, see *Sample Generation Result for the Client Side*.

For further information refer to the Natural RPC documentation.

Returning Application Errors from an RPC Server to an RPC Client

Application error codes enable the RPC server to return customer-invented errors back to the RPC client in a standardized way without defining an error code field in the IDL file.

USR4012N from Natural library SYSEXT may be used to enforce Natural error NAT1999 on the Natural RPC server and to pass back the provided error text to the RPC client.

Example

```
/* Application Error
COMPRESS "ERROR: " #ERR-TEXT INTO #LOG-TEXT
/* Send back Application Error to Client
CALLNAT 'USR4012N' USING #ERR-TEXT
```

For more information:

- Logon to the Natural library SYSEXT within your Natural installation, enter command `MENU` and select `USR4012N`.
- Refer to the **http://documentation.softwareag.com** > *Natural Product Line*.
- See the *Basic RPC Server Examples - `CALC`, `SQUARE`*.

Interfaces (APIs) Available in SYSEXT

The Natural library SYSEXT provides APIs for RPC programming. Log on to the library SYSEXT and enter `MENU`. To list only RPC-related APIs, enter the keyword `RPC`. To make these APIs available, make the necessary STEPLIP settings or copy the APIs from SYSEXT to user libraries. For more information, refer to the API documentation provided by SYSEXT.

Interface	Comment
USR1071N	Set user ID and password for RPC
USR2007N	Set/get RPC default server information.
USR2015N	EBCDIC or ASCII translation table for Natural RPC.
USR2032N	Support of commit for CLOSE CONVERSATION.
USR2035N	Support of SSL.
USR2071N	Support of EntireX Security on client side.
USR2072N	Support of EntireX Security on server side.
USR2073N	Ping or terminate an RPC server.
USR2074N	Set new password for NSC user in RPC context.
USR2075N	Terminate EntireX Broker service.
USR4008N	Set library for RPC execution.
USR4009N	Set parameters for EntireX.
USR4010N	Retrieve runtime settings of server.
USR4012N	Support of application error.
USR4371N	Set user ID and ETID for RPC.
USR6304N	Set/get reliable state for RPC execution.
USR6305N	Commit/rollback reliable RPC message(s).
USR6306N	Status of UOWs of current EntireX Broker user.
etc.	

Using SSL/TLS

RPC client applications can use Secure Sockets Layer/Transport Layer Security (SSL/TLS) as the transport medium. The term “SSL” in this section refers to both SSL and TLS. RPC-based clients are always SSL clients. The SSL server can be either the EntireX Broker or Direct RPC in webMethods Integration Server (IS inbound). For an introduction see *SSL/TLS and Certificates with EntireX* in the Platform-independent Administration documentation. This section describes using SSL with the Natural Wrapper on z/OS and z/VSE.

z/OS

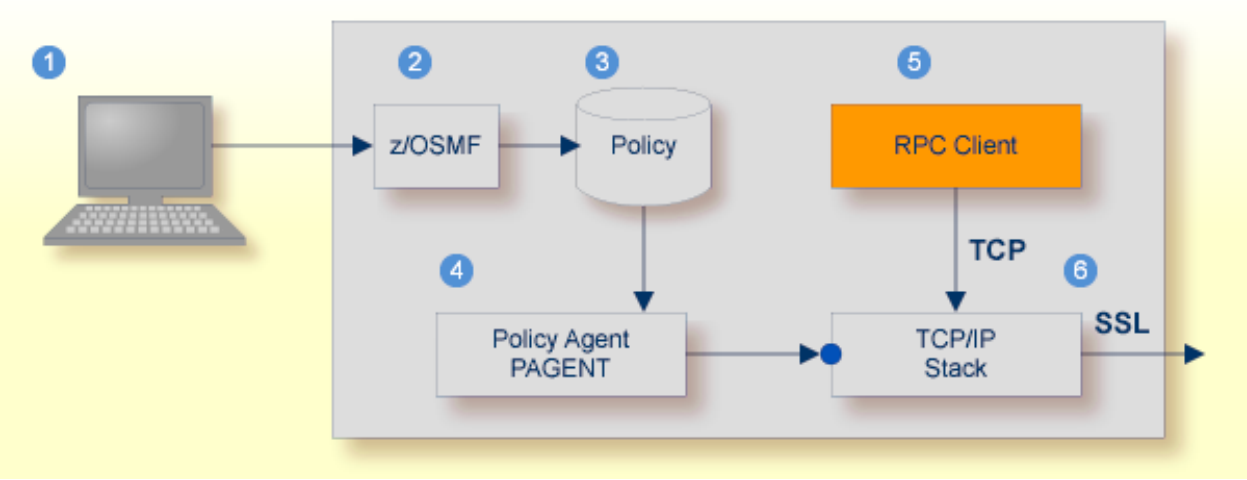
SSL delivered on a z/OS mainframe will typically use the Resource Access Control Facility (RACF) as the certificate authority (CA). Certificates managed by RACF can only be accessed through the RACF keyring container. A keyring is a collection of certificates that identify a networking trust relationship (also called a trust policy). In an SSL client/server network environment, entities identify themselves using digital certificates called through a keyring. Server applications on z/OS that wish to establish network connections to other entities can use keyrings and their certificate contents to determine the trustworthiness of the client or peer entity. Note that certificates can belong to more than one keyring, and you can assign different users to the same keyring. Because of the way RACF internally references certificates, they must be uniquely identifiable by owner and label, and also unique by serial number plus data set name (DSN).

For establishing an SSL connection on z/OS, IBM's Application Transparent Transport Layer Security (AT-TLS) can be used, where the establishment of the SSL connection is pushed down the stack into the TCP layer.

With the Natural Wrapper you can use IBM's Application Transparent Transport Layer Security (AT-TLS), where the establishment of the SSL connection is pushed down the stack into the TCP layer.

Using IBM's Application Transparent Transport Layer Security (AT-TLS)

Configure the AT-TLS rules for the policy agent (PAGENT) 4 using an appropriate client 1 and the z/OS Management Facility (z/OSMF) 2. Together with SSL parameters (to provide certificates stored in z/OS as RACF keyrings) define AT-TLS rules, for example by using the application 5 job name and remote TCP port number. If the rules match, the TCP connection is turned into an SSL connection 6. Refer to your IBM documentation for more information, for example the IBM Redbook *Communications Server for z/OS VxRy TCP/IP Implementation Volume 4: Security and Policy-Based Networking*.



1. Client to interact with z/OS Management Facility (z/OSMF).
2. AT-TLS rules are defined with z/OSMF policy management.
3. Policy Repository with AT-TLS rules stored as z/OS files.
4. Policy Agent, MVS task PAGENT, provides AT-TLS rules through a policy enforcement point (PEP) to TCP/IP stack.
5. Application using TCP connection.
6. If AT-TLS rules match, the TCP connection is turned into an SSL connection.

Notes:

1. The client 1 may vary per operating system, for example a Web browser for z/OS 2.1.

2. z/OSMF (2) includes other administration and management tasks in addition to policy management.
3. Policy Management (3) includes other rules, such as IP filtering, network address translation etc.

To set up SSL with AT-TLS

1 To operate with SSL, certificates need to be provided and maintained. Depending on the platform, Software AG provides default certificates, but we strongly recommend that you create your own. See *SSL/TLS Sample Certificates Delivered with EntireX* in the EntireX Security documentation.

2 Set up the RPC component for a TCP/IP connection. On mainframe platforms, use *Transport-method-style Broker ID*. Example:

```
ETB024:1699:TCP
```

3 Configure AT-TLS to turn the TCP/IP connection to an SSL connection, using a client to interact with the z/OS Management Facility (z/OSMF). The outcome of this configuration is a Policy Repository with AT-TLS rules stored as z/OS files. This file is the configuration file for the Policy Agent, MVS task `PAGENT`.

4 Make sure the SSL server to which the RPC component connects is prepared for SSL connections as well. The SSL server can be EntireX Broker, Broker SSL Agent, or Direct RPC in webMethods Integration Server (IS inbound). See:

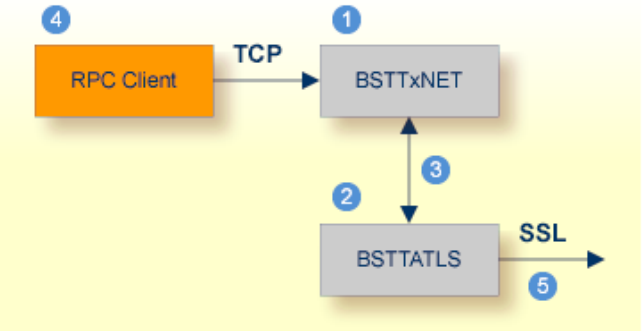
- *Running Broker with SSL/TLS Transport* in the platform-specific Administration documentation
- Broker SSL Agent in the UNIX and Windows Administration documentation
- *Support for SSL/TLS* in the EntireX Adapter documentation (for Direct RPC)

z/VSE

Establishing an SSL connection on z/VSE requires BSI's Automatic Transport Layer Security (ATLS). This facility is similar to z/OS Application Transparent - Transport Layer Security (AT-TLS). ATLS is supported by the BSI stack only.

Using BSI's Automatic Transport Layer Security (ATLS)

Together with SSL parameters (to provide certificates), define ATLS rules for socket interception in the ATLS daemon startup job `BSTTATLS` (2). If the rules match, the socket connection is turned into an SSL connection (5). Refer to your IBM documentation for further information. For an overview, refer to the IBM Redbook *Enhanced Networking on IBM z/VSE*; for a more detailed description, refer to *BSI SSL Installation, Programming and User's Guide*.



1. BSI TCP/IP Stack, either BSTTINET (IPv4) or BSTT6NET (IPv6).
2. ATLS rules are defined manually. See Sample ATLS Daemon Configuration below.
3. BSTTATLS is associated with a TCP/IP stack.
4. Application using TCP connection.
5. BSTTATLS intercepts outbound TCP connection and converts it to SSL connection. For inbound, SSL connections can also be intercepted and converted to TCP connections.

To set up SSL with ATLS

1. To operate with SSL, certificates need to be provided and maintained. Depending on the platform, Software AG provides default certificates, but we strongly recommend that you create your own. See *SSL/TLS Sample Certificates Delivered with EntireX* in the EntireX Security documentation.
2. Set up the RPC component for a TCP/IP connection. On mainframe platforms, use *Transport-method-style Broker ID*. Example:

```
ETB024:1699:TCP
```

3. Configure ATLS to turn the TCP/IP connection to an SSL connection, see above.
4. Make sure the SSL server to which the RPC component connects is prepared for SSL connections as well. The SSL server can be EntireX Broker, Broker SSL Agent, or Direct RPC in webMethods Integration Server (IS inbound). See:

- *Running Broker with SSL/TLS Transport* in the platform-specific Administration documentation
- Broker SSL Agent in the UNIX and Windows Administration documentation
- *Support for SSL/TLS* in the EntireX Adapter documentation (for Direct RPC)

Sample ATLS Daemon Configuration

```
* Converting inbound EntireX Broker connection
* Converts listen port 1971 to SSL listen port 1972
OPTION SERVER
ATTLS 1971 AS 2071 SSL
*
* Converting outbound client connection
* Converts connect to 192.168.2.100:1972:TCP to 192.168.2.100:2072:SSL
OPTION CLIENT
ATTLS 1972 TO 192.168.2.100 AS 2072 SSL
```

Note: We recommend setting `SETPARM` value `SUBTASK` to a value greater than 0 in the ATLS daemon startup job (valid values 0-16, default=0). For example:

```
// SETPARM SUBTASK=8
```

See also *BSI SSL Installation, Programming and User's Guide*.

9 Client and Server Examples for Natural

This chapter describes the examples provided for Natural. All examples here can be found in the EntireX *examples/RPC* directory under UNIX and Windows.

Basic RPC Client Examples - CALC, SQUARE

Name	Type	Description	Notes
CALCCLT.NSP	Natural Program	A client application calling the remote procedure (RPC service) CALC, with associated example.idl.	1, 2
SQRECLT.NSP	Natural Program	A client application calling the remote procedure (RPC service) SQUARE, with associated example.idl.	1, 2

Notes:

1. Requires a Natural version supported by Software AG.
2. Application uses a client interface object built with the Natural Wrapper.

For more information see the readme file in EntireX directory *examples/RPC/NaturalClient* under UNIX or Windows.

Basic RPC Server Examples - CALC, SQUARE

Name	Type	Description	Notes
CALC.NSN	Natural Subprogram (CALLNAT)	A server application providing the remote procedure CALC (RPC service), with associated example.idl.	1
SQUARE.NSN	Natural Subprogram (CALLNAT)	A server application providing the remote procedure SQUARE (RPC service), with associated example.idl.	1

Notes:

1. Requires a Natural version supported by Software AG and a Natural RPC Server.

For more information, see the readme file in EntireX directory *examples/RPC/NaturalServer* under UNIX or Windows.