Prerequisites and Preparatory Information

This document provides an overview of the general prerequisites and and a short description of the facilities that are available in Natural for implementing a Natural Remote Procedure Call (RPC) environment.

The following topics are covered:

- Products Involved
- Natural Statements Involved
- Natural Utilities for Use with Natural RPC
- Application Programming Interfaces for Use with Natural RPC
- Software AG IDL to Natural Mapping

Products Involved

If the RPC environment is to be implemented on different plattforms, the corresponding current versions of Natural for Mainframes, Windows, UNIX or OpenVMS will be required. In addition, the following required or optional products, subproducts and facilities are available for use in a Natural RPC environment:

Product	Purpose
EntireX Communicator (EntireX Broker)	The EntireX Broker of the Software AG product EntireX Communicator usually establishes the middleware layer between the Natural client and a Natural server. It uses the ACI protocol and comprises the appropriate client/server stubs.
Entire Net-Work	This Software AG product is required if the transport method used by EntireX Broker is Entire Net-work. This is the preferred transport method.
TCP/IP	Required if the transport method used by EntireX Broker is TCP/IP. See also *Using TCP/IP as a Transport Method* in *Setting Up a Natural RPC Environment*.
EntireX Developer’s Kit	Included in the EntireX Communicator product, this kit is required for non-Natural programming language support.
Directory Services	A remote directory server (RDS) enables you to define directory definitions in one place so that its services can be used by all clients in an RPC environment. An RDS is required if the Location Transparency provided by EntireX Broker is used.
Natural Security	Optional. This Software AG add-on product is required on the server side to protect Natural RPC servers and services when security is active on the client and vice versa.
EntireX RPC	Optional. Natural RPC fully supports EntireX RPC for 3GL. EntireX RPC is included in the EntireX Communicator product.
EntireX Security	Optional. Natural RPC fully supports EntireX Security on the client and on the server side. EntireX Security is included in the EntireX Broker product.

For the supported Software AG product versions, refer to *Software AG Product Versions Required with Natural* in the current Natural *Release Notes* for Mainframes.

For information on other products that may be involved in a Natural RPC-based environment, see the corresponding product documentation.

Natural Statements Involved

The following Natural statements are used in the creation of a Natural RPC environment:

- `CALLNAT`

- `DEFINE DATA PARAMETER`
- `DEFINE DATA CONTEXT`
- `OPEN CONVERSATION`
- `CLOSE CONVERSATION`
- `PASSW`
- `STACK`

In the section *Restrictions and Limitations*, the paragraphs *Natural Statement Reactions* and *Notes on Natural Statements on the Server* provide information on what you should know about any deviating behavior of these statements when they are used in a Natural RPC environment.

Natural Utilities for Use with Natural RPC

The following Natural utilities are used in the creation and maintenance of a Natural RPC environment:

- `SYSRPC`
  This utility is used to maintain remote procedure call environments.
- `SYSEXT`

  This utility is used to locate and test Natural Application Programming Interfaces (APIs, see below) contained in the current system library `SYSEXT`.
- `SYSPARM`

  On mainframes, this utility is used for creating and maintaining a set of Natural profile parameters that is stored under a profile name.
- Configuration Utility

  Under Windows, UNIX and OpenVMS, this utility is used to modify global and local configuration files and to create or modify parameter files.

Application Programming Interfaces for Use with Natural RPC

The purpose of Natural Application Programming Interfaces (API) is to retrieve or modify information or use services that are not accessible by Natural statements.

The following Application Programming Interfaces available in the Natural library `SYSEXT` are intended for being used with the Natural RPC:

API	**Purpose**
`USR1071N`	Enables you to set user ID, password and ticket criteria for Natural RPC. See *Using Security*.

API	Purpose
`USR2007N`	Enables you to specify a default server address that is to be used each time a remote program cannot be addressed via the service directory. See *Specifying a Default Server Address within a Natural Session* in *Operating a Natural RPC Environment*.
`USR2032N`	Supports the commit for a `CLOSE CONVERSATION` statement on the client side. When called, this API will cause an implicit `END TRANSACTION` at the end of the individual conversation. See *Database Transactions* in *Introducing Natural RPC*, section *Conversational CALLNAT*.
`USR2035N`	Enables you to set the required SSL parameter string if the Secure Socket Layer (SSL) for the TCP/IP communication to the EntireX Broker is used. See *Using Secure Socket Layer* in *Operating a Natural RPC Environment*.
`USR2071N`	Has to be called in case of non-Natural Security clients for specifying logon data which are then passed to the server. See *Using Natural RPC with Natural Security* in *Using Security*.
`USR2072N`	Enables you to specify a password which is used for the `LOGON` in conjunction with profile parameter `SRVUSER`. See *Server Side* in *Using Security*, section *Using Natural RPC with EntireX Security*.
`USR2073N`	Enables you to ping or terminate an RPC server from within your application. See *Using Application Programming Interface USR2073N* in *Terminating a Natural RPC Server*.
`USR2074N`	Enables you to change the Natural Security password on the RPC server via a Natural RPC service request. See *Change Password* in *Using Security*.
`USR2075N`	Enables you to terminate an EntireX Broker Service from within your application. See *Using Application Programming Interface USR2075N* in *Terminating an EntireX Broker Service*.
`USR4008N`	Enables you (on the client side) to specifiy an alternate name of a library to which the server will logon. See *Using the Logon Option* in *Operating a Natural RPC Environment*.
`USR4009N`	Enables you to set/get parameters for EntireX in a Natural RPC client or server environment. See *Setting/Getting Parameters for EntireX* in *Operating a Natural RPC Environment*.

API	Purpose
`USR4010N`	Enables you (on the client side) to retrieve the runtime settings of a server. See *Retrieving the Runtime Settings of a Server* in *Operating a Natural RPC Environment*.
`USR4371N`	Enables you (on the client side) to set the user ID and ETID for Natural RPC servers which were configured with Impersonation = A (automatic logon).
`USR6304N`	Enables you to set or get the reliable state for the reliable Natural RPC. See *Reliable RPC on the Natural RPC Client Side* in *Reliable RPC*.
`USR6305N`	Enables you to commit or rollback reliable RPC message(s). This API is required if the reliable RPC state has been set to "client commit" See *Reliable RPC on the Natural RPC Client Side* in *Reliable RPC*.
`USR6306N`	Enables you to retrieve the status of all reliable RPC messages of the user who is currently logged on to the EntireX Broker. See *Reliable RPC on the Natural RPC Server Side* in *Reliable RPC*.

Note that the RPC-specific APIs accept all values also in mixed mode. An uppercase translation will take place only if the example program `USR`*`nnnn`*`P` (source object) is used to invoke the corresponding subprogram `USR`*`nnnn`*`N`. *Exception:* All `USR`*`nnnn`*`P` programs that deal with passwords provide an option to enter the passwords in mixed case mode.

For an explanation of the Natural object types that are typically provided for each API, see the Natural utility `SYSEXT`.

Software AG IDL to Natural Mapping

This section 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 Software AG IDL File (in the *EntireX* documentation).

- Mapping Software AG IDL Data Types to Natural Data Formats
- Mapping Library Name and Alias
- Mapping Program Name and Alias
- Mapping Parameter Names
- Mapping Fixed and Unbounded Arrays
- Mapping Groups and Periodic Groups
- Mapping Structures

- Mapping the Direction Attributes IN, OUT, INOUT
- Mapping the ALIGNED Attribute
- Calling Servers as Procedures or Functions

Mapping Software AG 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] surround optional lexical entities.
- The informal term *number* (or in some cases *number.number*) is a sequence of numeric characters, for example 123.

Software AG IDL	Description	Natural Data Format	Note
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*number*[.*number*]	Unpacked decimal	N*number*[.*number*]	
NU*number*[.*number*]	Unpacked decimal unsigned	N*number*[.*number*]	
P*number*[.*number*]	Packed decimal	P*number*[.*number*]	
PU*number*[.*number*]	Packed decimal unsigned	P*number*[.*number*]	
T	Time	T	3,4
U*number*	Unicode	U*number*	
UV	Unicode variable length	U DYNAMIC	
UV*number*	Unicode variable length with maximum length	U DYNAMIC	

See also the hints and restrictions on the Software AG IDL Data Types (in the *EntireX* documentation) valid for all language bindings.

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, errors due to rounding 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 tenth 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 date)
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

Mapping Library Name and Alias

The 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 client to a server, the following steps are required for the client:

To send a library name other than `SYSTEM` from a client to a server

1. On the client, turn on the logon option.
2. 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 is limited to 8 characters.

Mapping Program Name and Alias

The program name is sent from a client to the server. Special characters are not replaced. The program alias is not sent to the server.

The generated Natural interface object (stub subprogram) has the same name.

In the RPC server, the IDL program name sent is used to locate the Natural subprogram.

The length of the program name is limited to 8 characters.

Mapping Parameter Names

The parameter names as given in the parameter-data-definition of the IDL file are replaced by artificial names in the generated Natural interface object (stub subprogram).

See *parameter-data-definition* in the section *Software AG IDL Grammar* in the *EntireX* 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 the *array-definition* (in the section *Software AG IDL Grammar* in the *EntireX* 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 the *array-definition* (in the section *Software AG IDL Grammar* in the *EntireX* documentation) for the syntax of unbounded arrays within the IDL file and refer to unbounded-array-index.

  Note:
  Natural variable arrays (Natural notation (../1:V)) can be used on the Natural RPC server side instead of Natural fixed arrays or X-arrays. An RPC client can pass either an IDL fixed array or IDL unbounded array to a Natural RPC server with such a Natural variable array. In the RPC server, the variable array cannot be resized; this means the number of array occurrences cannot be changed, and the Natural RPC server will always pass back the same number of occurrences.

Mapping Groups and Periodic Groups

Groups within the IDL file are mapped to Natural groups. See *group-parameter-definition* (in the section *Software AG IDL Grammar* in the *EntireX* 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* (in the section *Software AG IDL Grammar* in the *EntireX* 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 INOUT 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 `INOUT` 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 *attribute-list* (in the section *Software AG IDL Grammar* in the *EntireX* 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 `SYSTRPC` is `OUT` in IDL
- `OUT` in `SYSTRPC` 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 *attribute-list* (in the section *Software AG IDL Grammar* in the *EntireX* 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 server. 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.