

Reliable RPC

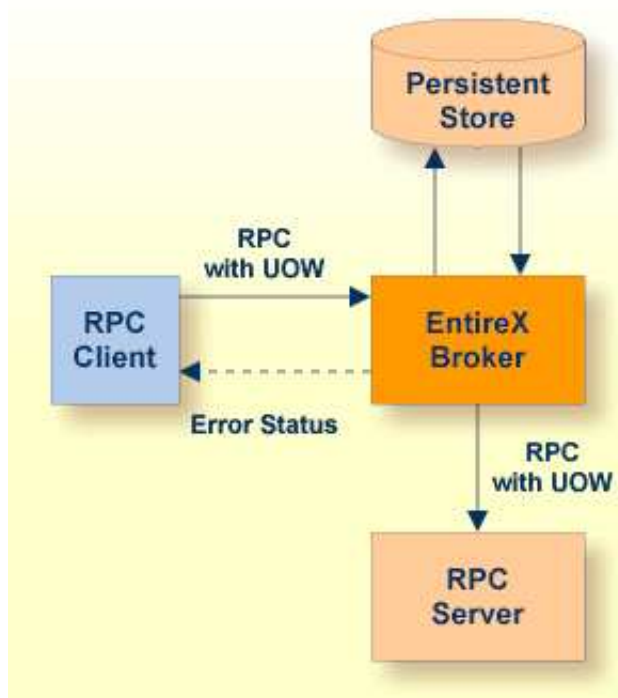
- General Information
 - Reliable RPC on the Natural RPC Client Side
 - Reliable RPC on the Natural RPC Server Side
 - Viewing the Status of Reliable RPC Messages
-

General Information

In the architecture of modern e-business applications (SOA), loosely coupled systems are becoming more and more important. Reliable messaging is one important technology for this type of system.

Reliable RPC is the Natural RPC implementation of a reliable messaging system. It combines the Natural RPC technology and persistence, which is implemented by means of units of work (UOWs) that are offered by the EntireX Broker. Reliable RPC is characterized by following features:

- The Natural RPC client executes a `CALLNAT` statement without waiting for a reply from the server (the RPC message is sent in asynchronous mode).
- An RPC server needs not be active at the time the `CALLNAT` is executed.
- The reliable RPC message is stored in the Broker's persistent store until an RPC server is available.
- The Natural RPC server executes the reliable RPC by calling the requested subprogram but does not send a reply to the RPC client.
- A Natural RPC client may ask the status of the sent reliable RPC messages.
- A Natural RPC client may send a reliable RPC message to an EntireX RPC server.
- A Natural RPC server may receive a reliable RPC message from an EntireX RPC client.



Reliable RPC on the Natural RPC Client Side

The Natural RPC client for a reliable RPC is configured in the same way as for a normal Natural RPC. The same Natural RPC client session can send standard RPC requests and reliable RPC messages.

To enable a Natural RPC client to use reliable RPC, the Natural RPC client must use application programming interface USR2071N for an explicit EntireX Broker logon. This implies that the RPC/NTRPC keyword subparameter ACIVERS must be set to 2 or higher.

Reliable RPC is used to send messages to a persistent EntireX Broker service. The messages are described by the PDA of the caller and may only contain output parameters. A parameter is defined as "output" in one of the following ways:

- If a Natural stub subprogram (interface object) is used:

In the `Attr` field on the Stub Generation screen of the `SYSRPC` utility, set the attribute of the parameter to `O` (output).

Note:

If your parameter definitions do not contain group structures, you must set the attribute `COMPAT` to `IDL` before generating the stub subprogram; see *Using the Stub Generation Function* in the `SYSRPC` Utility documentation.

- If no Natural stub subprogram is used:

Use the `AD=O` session parameter in the `CALLNAT` statement.

Note:

If you want to call an EntireX RPC server and if the corresponding IDL file contains group structures, then you must use a Natural stub subprogram, and the parameter definition for the Natural

stub subprogram must correspond to the group structure of the IDL file.

- If you generate a Natural stub subprogram from an IDL file, the attribute of the parameter is taken from the IDL file. In this case, the IDL file must only contain inbound (from the client’s point of view) parameters.

If you generate a Natural stub subprogram from an IDL file, the attribute of the parameter is taken from the IDL file. In this case, the IDL file must only contain inbound (from the client’s point of view) parameters.

Reliable RPC is enabled at runtime. The client has to set the one of two different modes before issuing a reliable RPC request:

- AUTO_COMMIT
- CLIENT_COMMIT

While AUTO_COMMIT commits each message implicitly after sending it, a series of RPC messages sent in a unit of work (UOW) can be committed or rolled back explicitly using CLIENT_COMMIT mode.

For this purpose, Natural provides the two application programming interfaces USR6304N and USR6305N. With interface USR6304N, the mode for reliable RPC is set. With interface USR6305N, a unit of work that has been created with CLIENT_COMMIT can be committed or rolled back.

▶ To make use of USR6304N

1. Copy the subprogram USR6304N from the library SYSEXT to the library SYSTEM or to the steplib library or to any application in the client environment.
2. Using the DEFINE DATA statement, specify the following parameters:


Parameter	I/O	Format	Description	
P-FUNC	I	A01	Function code; possible values are:	
			P (Put)	Set the mode for reliable RPC. The mode applies to all subsequent calls.
			G (Get)	Get the previously specified mode.
P-MODE	I/O	N01	Mode for reliable RPC:	
			0	No reliable RPC (standard RPC execution)
			1	Reliable RPC AUTO_COMMIT
			2	Reliable RPC CLIENT_COMMIT
P-RC	O	N04	Return code	
P-MESSAGE	O	A80	Message text	

3. In the calling program on the client side, specify the following statement:

```
CALLLNAT 'USR6304N' P-FUNC P-MODE P-RC P-MESSAGE
```

Note:

The mode `CLIENT_COMMIT` cannot be changed if reliable RPC messages have been sent but not yet committed or rolled back.

 **To make use of USR6305N**

1. Copy the subprogram `USR6305N` from the library `SYSEXT` to the library `SYSTEM` or to the `steplib` library or to any application in the client environment.
2. Using the `DEFINE DATA` statement, specify the following parameters:

Parameter	I/O	Format	Description	
P-FUNC	I	A08	Function code; possible values are:	
			COMMIT	Commit the sent reliable RPC messages. The messages are now available for an RPC server.
			ROLLBACK	Discard the already sent reliable RPC messages.
P-RC	O	N04	Return code	
P-MESSAGE	O	A80	Message text	

3. In the calling program on the client side, specify the following statement:

```
CALLLNAT 'USR6305N' P-FUNC P-MODE P-RC P-MESSAGE
```

Reliable RPC on the Natural RPC Server Side

The Natural RPC server for reliable RPC is configured in the same way as for normal Natural RPC. The same Natural RPC server session can process standard RPC requests and reliable RPC messages.

To enable the processing of reliable RPC messages, the `RPC/NTRPC` keyword subparameter `ACIVERS` must be set to 2 or higher.

Viewing the Status of Reliable RPC Messages

To view the status of sent reliable RPC messages, Natural provides the application programming interface `USR6306N`. With `USR6306N` you can get the status of all reliable RPC messages that you have previously sent under your user ID. `USR6306N` must not necessarily be called within the Natural session in which the reliable RPC messages have been sent. If `USR6306N` is used in a different Natural session, the application programming interface `USR2071N` must first be used to log on to the EntireX Broker with the same user ID that has been used to send the reliable RPC messages.

The reliable RPC messages are implemented by EntireX Broker unit of works (UOWs). The information about reliable RPC messages is therefore information about UOWs.

 **To make use of USR6306N**

1. Copy the subprogram USR6306N from the library SYSEXT to the library SYSTEM or to the steplib library or to any application in the client environment.
2. Using the DEFINE DATA statement, specify the following parameters:

Parameter	I/O	Format	Description
P-UOW-ID-IN	I	A16	ID for the UOW to be retrieved. Possible values are: UOWID of a UOW LAST: the UOW for the last reliable RPC message in the current Natural session ALL or blank: all UOWs for the logged on EntireX Broker user
P-USER-ID	O	A32	User ID of the user who has created the UOWs.
P-BROKER-ID	O	A32	Broker ID of the EntireX Broker that hosts the UOWs.
P-UOW-COUNT	O	I4	Number of UOWs in the P-UOW-INFO array.
P-UOW-INFO (/ 1 : *)			X-array with information about each UOW
P-UOW-ID	O	A32	ID of the UOW
P-UOW-STATUS	O	A10	Status of the UOW according to EntireX Broker The status depends on the processing state and is assigned by the EntireX Broker.
P-USERR-STATUS	O	A32	User information about the UOW. This is typically error information that has been set by the RPC server.
P-CREATE-TIME	O	A32	Creation time of the UOW according to EntireX Broker.
P-RC	O	N04	Return code
P-MESSAGE	O	A80	Message text

3. In the calling program on the client side, specify the following statement:

```
CALLNAT 'USR6306N' P-UOW-ID-IN P-USER-ID P-BROKER-ID P-UOW-COUNT P-UOW-INFO(*) P-RC P-MESSAGE
```