Restrictions and Limitations

This document informs you about some restrictions and limitations that you should observe when you are using the Natural Remote Procedure Call (RPC) facility.

User Context Transfer

Excepting the user identification, no user context is transferred to the server session, for example:

- all client session parameters remain unchanged and do not affect the execution on the server side;
- open transactions on the client side cannot be closed by the server and vice versa;
- client report handling and work file processing cannot be continued on the server side and vice versa;
- the handling of the Natural stack cannot be continued either.

System Variable Transfer

No system variables except `*USER` can be transferred from the client to the server side.

Application-Independent Variables

In an RPC server, application-independent variables (AIVs) are not deallocated implicitly, but stay active across RPC requests, because different clients may have access to the same variables on the RPC server. This means they must be deallocated explicitly using the `RELEASE VARIABLES` statement.

If you want to release AIVs at the end of an RPC request, you may use the Natural RPC user exit NATRPC03 for this purpose.

Parameter Handling in Error Situations

Parameter handling in error situations is different:

- If an error occurs during local execution, all parameter modifications performed so far are in effect, because parameters are passed via "call by reference".
- If an error occurs during remote execution, however, all parameters remain unchanged.

Variable Arrays in Subprograms

If the parameter data area of the subprogram contains a variable number of occurrences (1:V notation), you should not use a stub to call this subprogram. As a stub only supports array definitions with a fixed number of occurrences, you cannot vary the number of occurrences from call to call.

If you have to use a stub subprogram (interface object), for example, because you want to call an EntireX RPC server with the same program, you should use an X-array on the Natural client side. With X-arrays, it is possible to vary the number of occurrences from call to call even when using a stub subprogram. In this case, the X-array on the client side is passed to the (fixed) variable array on the server side. The variable array is fixed because the server program may receive a varying number of occurrences from call to call but cannot change the number of occurrences.

X-Arrays

X-arrays are supported in the parameter list of a remote CALLNAT statement execution. The server may increase or decrease the number of occurrences.

Restrictions

- In case of a multidimensional array, all dimensions of the array must be extensible.
- The lower bound must not be extensible, that is, only extensible upper bounds are allowed.
- If you want to use an X-group array that contains an array with constant bounds or a group array that contains an X-array, you must use a stub subprogram. When generating the stub subprogram, you must define the group structure in the Stub Generation screen.

  Examples:

```
01 X-Group-Array (/1:*)
02 Array (A10/1:10)
*
01 Group-Array (/1:10)
02 X-Array (A10/1:*)
```

Groups and Stub Subprograms

If group arrays or X-group arrays are present in the parameter list of a remote CALLNAT statement execution and a stub subprogram is used, the following restrictions exist.

Restrictions

- You must not use the AD=O or AD=A session parameter settings (attribute definition) in the CALLNAT statement.
- Group arrays and X-group arrays passed from a client Natural object to a stub subprogram must be contiguous. We therefore strongly recommend that you always pass a complete array to the stub subprogram by using asterisk (*) notation for all dimensions. We also strongly recommend that you use identical data definitions in the client Natural program, in the stub subprogram, and in the server program.

Group Arrays on the RPC Server Side

The storage layout of group arrays in the DEFINE DATA PARAMETER area of subprograms on the RPC server side is not necessarily identical with respect to the syntax. Do not redefine fields within a group array or pass the group array to a 3GL program. If you need to do so, copy the group array to a group array with the same layout in the DEFINE DATA LOCAL area and use this local group array in the call to the 3GL program.

EntireX RPC Server

If you want to call an EntireX RPC server with a remote CALLNAT statement execution, it is strongly recommended to use a stub subprogram. A stub subprogram is required if the IDL (Interface Definition Language) definition of the subprogram you want to call on an EntireX RPC server contains a group structure. In this case, you must define the same group structure during the stub generation on the Stub Generation screen or generate the stub subprogram from the EntireX IDL file (Windows only).

Using VSAM

If you access a VSAM dataset in a subtasking environment or if you share a VSAM dataset across regions, you must consider the required share options. For example, you may have to set cross-region SHAREOPTIONS 4 instead of SHAREOPTIONS 2 to enforce buffer invalidation if records are inserted in a VSAM dataset in one address space and the same VSAM dataset is read in another address space. Otherwise, records recently inserted may not be found.

You should also consider to use record level sharing (RLS).

For furthur information, refer to the relevant VSAM documentation of IBM.

Natural Statement Reactions

Several Natural statements may react in a different way when used in a Natural RPC context, for example:

Statement	Description
OPEN CONVERSATION CLOSE CONVERSATION	If executed on a server, these statements do not affect the client session. When the server itself acts as a client for another server (as agent), these statements only affect the conversations on the second server.
PASSW	The password setting remains active on the server side only, also for subsequent executions by other users.
SET CONTROL SET GLOBALS SET KEY SET TIME SET WINDOW	No settings are returned to the caller.
STACK	All stack data are released after execution.
STOP TERMINATE	These statements *do not* stop the client session. For information on how to terminate a Natural RPC server, see *Terminating a Natural RPC Server*.

Notes on Natural Statements on the Server

The use of the following statements on an Natural RPC is theoretically possible, but not recommended, as it causes undesired effects:

Statement	Description
TERMINATE	Using this statement causes the server to be terminated, regardless of conversations that may still be open.
FETCH RUN STOP	Using these statements causes the CALLNAT context to be lost. Upon a FETCH, RUN or STOP statement, the server detects that it has lost its CALLNAT context and returns a corresponding Natural error message to the client; at that time, however, the statement has already been executed by the server. *Exception:* This does not apply to FETCH RETURN.
INPUT	Input values are unpredictable when the input data are read from a file (and not from the stack).