Generating Single Stubs with Parameter Specification

The **Stub Generation** function provides the option to generate single stub subprograms (interface objects) online by using a stub generation screen where you either type in the parameter definitions required or read them in from an existing subprogram.

This section covers the following topics:

- Using the Stub Generation Function
- Specifying Parameters
- Examples of Stub Generation

Using the Stub Generation Function

The stub subprograms (interface objects) are generated into the current Natural library in the current system file. Therefore, we strongly recommend that you log on to the application library (or one of its steplibs) used by the client at execution time of the remote `CALLNAT`.

Important:
The stub generation function overwrites any data contained in the source work area. When you invoke the stub generation function, a corresponding message will warn you not to delete any existing data unintentionally: choose PF12 to cancel the function or choose ENTER to confirm the action and overwrite the contents of the source work area.

To generate a single stub subprogram

1. Before you invoke the SYSRPC utility, log on to the library into which you want to generate the stub subprogram.
2. In the **Code** field of the **Client Maintenance** menu, enter the following command:

   ```
   SG
   ```

3. Choose ENTER.

   The **Generate Client Stub Routine** window appears.

4. In the **Program Name** field, enter the name of the stub subprogram to be generated.

   The name of the stub subprogram must be identical to the name of the remote `CALLNAT` program. The **Library** field is preset to the current library and cannot be changed.

   DBID, FNR are non-modifiable fields that display the database ID (DBID), the file number (FNR) and the type of Natural file (`FNAT` = system, `FUSER` = user) for the current library.

In the **Compression** field, enter compression type `0`, `1` or `2` (default is `1`); see *Using Compression* described in *Operating a Natural RPC Environment* in the *Natural Remote Procedure Call (RPC)* documentation.

5. Choose ENTER.

   - If the name entered in the **Program Name** field corresponds to the name of an object that already exists in the assigned library, a window appears with an appropriate message:

     Enter an `N` (No) and choose ENTER if you if you want to cancel the operation. You will return to the **Client Maintenance** menu.

     Or:
     Enter a `Y` (Yes) and choose ENTER if you want to continue with the stub generation.

     If the specified name is identical to a cataloged object of the type subprogram, the parameter definitions of the respective subprogram are displayed on the **Stub Generation** screen.

     If the specified name is identical to a stub subprogram for which also a source object exists, all field attributes (see also *Specifying Parameters*) from a previous stub generation are retained. Otherwise, all field attributes are set to `M` (modifiable).

   - If the name entered in the **Program Name** field, does *not* correspond to the name of an object that already exists in the assigned library, an empty **Stub Generation** screen is displayed.

6. On the **Stub Generation** screen, add or modify the parameters to be used in the stub subprogram as described in *Specifying Parameters*.

   The commands provided on the **Stub Generation** screen correspond to the commands described in *Commands and PF Keys* in the section *Service Directory Maintenance*.

   Exceptions:

Attribute	Values
EXPIRATION	Not applicable to stub generation.
COMPAT	IDL | NONE | void
	IDL Stub generation according to IDL requirements. **NONE** Stub generation according to Natural requirements. **void** Show COMPAT setting. **Note:** See also: *Special Considerations for Reliable RPC* and *Special Considerations for Calling EntireX RPC Servers* .
LIMIT	32000 | 1GB | void
	32000 Sets the upper size limit to 32000 bytes. **1GB** Sets the upper size limit to 1 GB. **void** removes a size limit set with `LIMIT 32000` or `LIMIT 1GB`

7. Choose ENTER to generate the stub subprogram and to exit. The stub subprogram is generated in the assigned library.

   The **SYSRPC - Information** window appears which indicates the size the stub subprogram requires for sending data from the client to the server or vice versa. The size includes internal RPC information used for the stub subprogram. The indication of the size helps you configure the middleware layer used; for example, the Broker attribute file when EntireX Broker is used.

   The following message appears in the **SYSRPC - Information** window when you generate a stub subprogram from the example subprogram TESTS5 (see *Example 1* below):

```
Stub TESTS5 is generated in library SAGTEST (99,49).
   It requires:
        Send length: 2249 bytes
     Receive length: 2221 bytes
```

   If dynamic parameters, X-arrays or X-group arrays are used, this message only indicates the minimum length requirements. The actual length requirements can only be determined during program execution and may be different from call to call. If the `Send length` or the `Receive length` exceeds the Entire Net-Work limit of 32000 bytes, a window appears with a corresponding warning:

   Enter a `Y` (Yes) to continue, or an `N` (No) to cancel the generation.

   If you enter a `Y`, this setting is kept for the entire SYSRPC session, that is, you can continue generating stub subprograms without receiving further warnings.

   If the total data (without internal RPC information) sent or received exceeds the limit of 1073739357 bytes (which is 1 GB minus 2467 bytes of internal RPC information), SYSRPC stops processing and issues a corresponding error message. This error message displays the subtotal of the data in bytes that could be transferred at the field up to which the subtotal was calculated. The corresponding field is then marked. In this case, reduce the amount of data and then continue generating the stub

subprogram.

If the stub subprogram was generated in the Natural system library SYSRPC, you must move the stub subprogram to the application library or steplib using the Natural transfer utility SYSMAIN or the Object Handler. Note that you may have to recatalog the source of the stub subprogram in the target environment.

Specifying Parameters

In the input fields provided on the **Stub Generation** screen, you can enter the parameter definitions that are used in the stub subprogram (interface object). You can specify a maximum of 5000 parameters. Unless indicated in the table below, input in the fields is mandatory.

Field	Description
Level	The level of the field. A level can be a number in the range from `01` (highest level) to `99` (lowest level). The leading `0` is optional. See also *Defining Groups* and *Example 2* for an example of a group definition.
Attr	The attribute of the parameter: `M` (modifiable - INOUT), `O` (output - OUT) or `I` (input - IN). Parameters assigned a level number of `2` or greater are considered to be a part of a group. Parameters within a group must have the same attribute as the immediately preceding group that is assigned one level higher. For nested groups, this is the attribute of the group with the highest level. For an example of a group definition, see *Example 2*. If a stub subprogram has been generated from a subprogram, the attribute is `M` by default and may need modification. If a stub subprogram has been generated from another stub subprogram, the attribute values specified for the original stub subprogram are retained. The generated stub subprogram contains a comment that indicates the attribute specified for the parameter: `IN`, `OUT` or `INOUT`.
Type	A Natural data format such as `N` (numeric) and `G` (group), or `K` (Kanji). Natural data formats `C` (attribute control) and Handle are not allowed. For a description of Natural data formats, see *Format and Length of User-Defined Variables* and *Special Formats* in the section *User-Defined Variables* in the *Programming Guide*.

Field	Description
Length	The length of the parameter or `DYNAMIC`. This field does not apply to the following Natural data formats: `D` (date), `G` (group), `L` (logical) and `T` (time). The Natural data format `A` is restricted to 1073739357 bytes, Natural data format `B` is restricted to 536869678 bytes. `DYNAMIC` indicates a dynamic parameter and applies to the Natural data formats `A` and `B`.
Prec	Only applies to Natural data formats `N` (numeric) and `P` (packed). Optional. The precision of the parameter, that is, the number of digits after the decimal point.
Dimension ½/3	Only applies to arrays. Optional. The first, second and third dimension of the parameter. An X-array or an X-group array is specified by entering an asterisk (*) for a dimension. See also *Defining X-Arrays and X-Group Arrays*.

The section below contains information on:

- Defining Groups
- Defining X-Arrays and X-Group Arrays
- Special Considerations for Reliable RPC
- Special Considerations for Calling EntireX RPC Servers

Defining Groups

You only need to define a group structure for a client Natural object that calls a non-Natural object located on an EntireX RPC server. The group structure must correspond to the IDL definition in EntireX (see *Special Considerations for Calling EntireX RPC Servers*). A group structure is not required for a client Natural object that calls a subprogram located on a Natural RPC server.

Group arrays and X-group arrays passed from a client Natural object to a stub subprogram must be contiguous. Therefore, we 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, the stub subprogram and the server program.

Warning:
Any group definitions in a subprogram will be ignored when a stub subprogram is generated from this subprogram. In this case, you must define the group again on the Stub Generation screen and adapt the dimension of the group elements accordingly. (Dimensions defined within a group are propagated to the parameter definitions at a lower level.) If you generate a stub subprogram from another stub subprogram that contains a group, the group definitions will be retained.

See also *Example 2* for an example of a group definition.

Defining X-Arrays and X-Group Arrays

If any dimension of a parameter is extensible, all other dimensions of the parameter are also extensible. If you define extensible and fixed dimensions for a parameter in a subprogram, the stub generation function issues a warning and automatically changes the fixed dimension to an extensible dimension as demonstrated in *Example 3*. In a group structure, you can define either an extensible or a fixed dimension for each level. There is no automatic change of a fixed dimension to an extensible dimension between levels.

Natural RPC only supports extensible upper bounds. All X-arrays and X-group arrays in the generated `DEFINE DATA PARAMETER` area of the stub subprogram are therefore defined as `(1:*)`.

Warning:
If you generate a stub subprogram from a subprogram that contains an X-array or X-group array with an extensible lower bound, the extensible lower bound will be converted to an extensible upper bound.

For an example of a group with an extensible dimension, see *Example 3*.

Special Considerations for Reliable RPC

If you want to use reliable RPC and your parameter definitions do not contain group structures, you must set `COMPAT IDL` before generating the stub subprogram.

Special Considerations for Calling EntireX RPC Servers

The attribute definitions on the **Stub Generation** screen depend on the perspective of the client. Conversely, the parameter direction in the IDL definition depends on the perspective of the server. This means:

- `OUT` on the **Stub Generation** screen corresponds to `IN` in the IDL definition.
- `IN` on the **Stub Generation** screen corresponds to `OUT` in the IDL definition.

If you want to call an EntireX RPC server and the parameter definitions on the **Stub Generation** screen contain group structures, group structure and attribute definitions on the **Stub Generation** screen must correspond to the group structure and parameter direction in the IDL definition.

If you want to call an EntireX RPC server and the corresponding IDL file does not contain group structures, it is recommended to set `COMPAT IDL` before generating the stub subprogram. In this case, the attribute definitions on the **Stub Generation** screen must correspond to the parameter direction in the IDL definition.

Examples of Stub Generation

This section provides examples of Natural subprograms and the stub subprograms (interface objects) generated from them.

The parameter definitions indicated below are extracted from example subprograms, which are supplied in the Natural system library SYSRPC.

Example 1

The following `DEFINE DATA PARAMETER` area (example subprogram TESTS5) shows four modifiable parameters and the corresponding parameter definitions on the **Stub Generation** screen:

```
DEFINE DATA
PARAMETER
  01 #IDENTIFIER  (A10)
  01 #N-OF-ID     (I4)
  01 #FREQ        (P5.2)
  01 #A100        (A100/5,4)
```

Stub Generation								
	Level	Attr	Type	Length	Prec	Dimension 1	Dimension 2	Dimension 3
1	01	M	A	10				
2	01	M	I	4				
3	01	M	P	5	2			
4	01	M	A	100		5	4	

Example 2

The following `DEFINE DATA PARAMETER` area (example subprogram TESTS6) shows a nested group structure and the corresponding parameter definitions on the **Stub Generation** screen:

```
DEFINE DATA
PARAMETER
 01 GROUP-1(10)
   02 A (A20)
   02 B (A20)
   02 GROUP-2(20)
     03 C (A10/5)
     03 D (A10)
 01 LINE (A) DYNAMIC
```

Stub Generation								
	Level	Attr	Type	Length	Prec	Dimension 1	Dimension 2	Dimension 3
1	01	M	G			10		
2	02	M	A	20				
3	02	M	A	20				
4	02	M	G			20		
5	03	M	A	10		5		
6	03	M	A	10				
7	01	M	A	DYNAMIC				

Example 3

The following `DEFINE DATA PARAMETER` area (example subprogram TESTS7) shows a nested group structure with extensible dimensions and the corresponding parameter definitions on the **Stub Generation** screen.

```
DEFINE DATA
PARAMETER
 01 GROUP-1(10)
   02 A (A20)
   02 B (A20)
   02 GROUP-2(0:*)
     03 C (A10/5)
     03 D (A10)
 01 LINE (A) DYNAMIC
```

Stub Generation								
	Level	Attr	Type	Length	Prec	Dimension 1	Dimension 2	Dimension 3
1	01	M	G			10		
2	02	M	A	20				
3	02	M	A	20				
4	02	M	G			*		
5	03	M	A	10		5		
6	03	M	A	10				
7	01	M	A	DYNAMIC				