CALL - Remote Procedure Call

CALL action - remote procedure call

CALL action for remote procedure call allows to call procedures, which are implemented in:

- the script of an object of Event type that must be Server event type,
- the script of an active picture (or sub picture),
- JAPI process.

Remote procedure call may be executed:

- **synchronously** - **CALL** action will wait to terminate the remote procedure execution and modified input-output parameters will be updated. If **CALL** action is noted down as an expression (_i := CALL ...), it is able to detect an unhandled error (exception) that occurred within the frame of the called procedure. The synchronous calling can cause deadlock. A dialog window with error message contains complete sequence of callings (CALL actions), that caused the deadlock.
- **asynchronously** - **CALL** action generates only request to execute a remote procedure (not waiting for its termination). The possibility to check the procedure execution for this type of calling.
  Following procedures can be called asynchronously:
  - RPC procedure in particular script of the particular process,
  - RPC procedure in scripts (objects of Event type (named as "Server event") or Picture) that are active in all running processes (**BROADCAST**) of D2000 HI or D2000 EventHandler,
  - JAPI process.

The declaration of an remote procedure must begin with the key word RPC.

Declaration - synchronous call

```
[_ret :=] CALL [objIdent] ProcName [(paramIdent1 [,paramIdent2]...)]
[SYNC] [ON procIdent [INSTANCE instanceExpr]] [PRTY exprIntPrty]
```

Declaration - asynchronous call

```
 CALL [objIdent] ProcName [(paramIdent1 [,paramIdent2]...)]  ASYNC [ON
procIdent [INSTANCE instanceExpr]] [PRTY exprIntPrty]
```

Declaration - asynchronous call BROADCAST

```
 CALL [objIdent] ProcName [(paramIdent1 [,paramIdent2]...)]  ASYNC ON ALL
[PRTY prtyIdent]
```

Declaration - synchronous call Client and Server configuration

```
[_ret :=] CALL [] ProcName [(paramIdent1 [,paramIdent2]...)] [SYNC] [ON
procIdent [INSTANCE instanceExpr]] [PRTY exprIntPrty]
```

Parameters

Procname	in	Procedure name (must meet the rules for object name).
paramIdent1, paramIdent2, ..., paramIdentN	in	Value identifier for the first (second, third, .., N) parameter. Number of parameters must be identical with the number of parameters of the procedure called.

objIdent	in	Object identifier (picture or system server event) that contains the script with the given procedure. Local variable, whose value is identical with the value of a *RefId* type variable linked to a graphic object of *Picture* type (subpicture procedure call). When calling the RPC from Local Script to Remote script (HIS Server), the identifier *objIdent* is not used. When calling JAPI process, the reference to object must be empty (see the example).
procIdent	in	Process identifier, where the object *objIdent* is located (e.g. SELF.EVH, SELF.HIP or WS_BC.HIP). If the CALL action is used within an active picture, then predefined variable _FROM_HIP contains identifier of HI process in which this picture is opened.
instanceExpr	in	Optional expression of *Int* type. It addresses a specific instance during the procedure call.
prtyIdent	in	Identifier of *Int* type. It defines the priority of executing RPC procedure. It can be assigned a value from the range: <Integer'Range> (-2147483648 to 2147483647). If it is not set, the default value is 0. The procedure with higher priority (higher number) is processed as first.

Description

CALL action will execute calling of the RPC (RPCX) procedure with the name *ProcName*. The procedure name is followed by a list of comma separated parameters.

The number (an types) of parameters must be equal to the number of parameters of the called procedure (if the number is not equal, there is generated the exception **_ERR_INV_NUM_PARAMS***).

If some of the parameters is specified as an input-output one in the procedure declaration, the corresponding parameter, during the procedure call, must not be a constant (if an error occurs, there will be generated the exception **_ERR_SET_CONST**).

procIdent is the reference to an object of Process type, where is the object *procIdent* opened in. For object of Event type it is the process D2000 EventHandler, which is its parent or process on which the called Event is opened (OPENEVENT action). For an object of **Picture** type it is the process D2000 HI, where is this picture opened in.
If the required object is not opened in the given process, the script generates the error **_ERR_OBJECT_NOT_FOUND***.

The parameter **INSTANCE** determines the instance number of the object (picture or event).

If a local variable of *RefId* type is used for the identifier *objIdent*, the parameters **INSTANCE** and **ON** are not admissible (their value are given by the calling context).

If RPC procedure is called between Local and Remote part of "Client and Server Event" configuration (HIS Server), *objIdent* is not used.

```
CALL [] ProcName [(paramIdent1 [,paramIdent2]...)] ...
```

The errors signed by the symbol * are generated in the called script. Called script (**CALL** action) ma detect this error merely during synchronous call and during the notation of an action with assignment. If an error occurs, its code will be assigned to the variable **_ret** that must be **INT** type.

Possible types of parameters, see the action PROCEDURE.

The calling of **BROADCAST** type is obligatory asynchronous. The process name (parameter *procIdent*) is key word **ALL**. By performing of this calling, the system distribute automatically a demand on procedure performing to all running processes of D2000 HI or D2000 Event Handler. These processes search all instances (or basic objects) of the scripts that are identified by parameter *objIdent* and generate the demands on procedure *ProcName* performing with relevant parameters.
The key word **INSTANCE** is disabled in this type of calling.

Since the D2000 V8.00.008 R9 the CALL action contains the new feature - optimization of the formal parameter transmission. This feature ensures the formal parameter of RPC procedure is a "link" to the real parameter. This causes the increasing of the speed of RPC procedure calling.
The conditions:

- the calling is realized in the same process *.EVH,
- the calling of procedure is synchronous,
- the parameter must be of *Record* type,
- the parameter must be *IN/OUT*.

The key word **PRTY** enables to set the priority of executing RPC procedure. The priority is defined by the parameter *prtyIdent* after **PRTY**.

RPC procedures are also intended for transfer data containers and a handle to database connections.

When calling JAPI process, these rules apply:

- *objIdent* is an empty object (see the example),
- the key word INSTANCE cannot be used,
- RPC procedure, which is an implementation of ESL interface, cannot be called,
- the process identifier (*procIdent*) must be of IC_HOBJ_EXPR (expression of HOBJ type) type,
- the receiving of calls on JAPI process must be implemented as listener, which is registered by a method D2Session::setRPCListener.

Example

Synchronous call of the remote procedure of the system script:

```
INT _i
 TEXT _personName
 _personName := "Peter"
 _i := CALL [E.Work] AddPerson(_personName) SYNC ON SELF.EVH

IF _i # _ERR_NO_ERROR THEN

  ; error when calling the remote procedure

ENDIF
```

Asynchronous call of the remote procedure of the active picture:

```
TEXT _msg

_msg := " ... "

CALL [S.Picture] SendMessage(_msg) ASYNC  ON _FROM_HIP
```

Asynchronous call of the remote procedure of the active picture:

```
TEXT _msg

_msg := " ... "

CALL [_subPicture] SendMessage(_msg) ASYNC
```

Synchronous call of the remote procedure, which is implemented by the process:

```
 TEXT _msg
 INT _hbj
 _msg := " ... "
 _hbj := ..... the value, which is obtained, for example when calling RPC
procedure from JAPI client, where one of the parameters is HOBJ of the
appropriate JAPI process.
 CALL [(0)] SendMessage(_msg) SYNC ON (_hbj)
```

Asynchronous call BROADCAST:

```
CALL [E.BROADCAST_RECEIVER] Broadcast(1) ASYNC ON ALL
```

Synchronous calling the remote procedure with priority:

```
INT _i
 INT _prty
 TEXT _personName

  _personName := "Peter"
  _prty := 100
 _i := CALL [E.Work] AddPerson(_personName) SYNC ON SELF.EVH PRTY _prty
 IF _i # _ERR_NO_ERROR THEN
 ; error when calling the RPC procedure
 ENDIF
```

Synchronous calling the remote procedure from Local script to Remote one:

```
 INT _i
 TEXT _personName
 _personName := "Peter"
 _i := CALL [] AddPerson(_personName) SYNC ON SELF.HIP
 IF _i # _ERR_NO_ERROR THEN
 ; error when calling the RPC procedure
 ENDIF
```

Note

Called remote procedure must be finished by the action **RETURN**, or **END ProcedureName**. In opposite case, it is not able to update possible input-output parameters. For a synchronous call, the return value is adjusted to the error **_ERR_MISSING_RETURN**.

Related pages:

Script actions
CALL action - local procedure call
Transfer of handle to database connection between the running ESL scripts
Data container transfer between running ESL scripts