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 a 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 can detect an unhandled error (exception) that occurred within the frame of the called procedure. The synchronous call can cause deadlock. A dialog window with an error message contains a complete sequence of calls (CALL actions), that caused the deadlock.
- **asynchronously** - **CALL** action generates only a request to execute a remote procedure (not waiting for its termination). This type of call doesn't provide the way to check the success of procedure execution.
- Following procedures can be called asynchronously:
  - RPC procedure in a 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 a remote procedure must begin with the keyword 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	• An object identifier (picture or system server event) that contains the script with the given procedure. • A local variable, whose value is identical to 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 the 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 an identifier of the 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 the 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 the call of the RPC (RPCX) procedure with the name *ProcName*. The procedure name is followed by a list of comma-separated parameters.

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

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

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

If a local variable of *RefId* type is used for the *objIdent* identifier, the **INSTANCE** and **ON** parameters 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) can detect this error only during synchronous call and during the notation of action with an assignment. If an error occurs, its code will be assigned to the variable **_ret** that must be of **INT** type.

For possible types of parameters, see the action PROCEDURE.

The call of the **BROADCAST** type is always asynchronous. The process name (the *procIdent* parameter) is a keyword **ALL**. By performing this call, the system distributes automatically a request to execute the procedure 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 the *objIdent* parameter and generate the requests to execute the *ProcName* procedure with the specified parameters.
The keyword **INSTANCE** is disabled in this type of call.

Since the D2000 V8.00.008 R9, the CALL action contains a 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 call.
The conditions are:

- The call is realized in the same process *.EVH,
- the call of the procedure is synchronous,

- the parameter must be of *Record* type,
- the parameter must be *IN/OUT*.

The keyword **PRTY** enables setting the priority of executing the RPC procedure. The priority is defined by the *prtyIdent* parameter after **PRTY** keyword.

RPC procedures are also support transfer of data containers and handles to database connections.

When calling JAPI process procedures, these rules apply:

- *objIdent* is an empty object (see the example),
- the INSTANCE keyword 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 in a JAPI process must be implemented as listener, which is registered by a D2Session::setRPCListener method.

Example

Synchronous call of a remote procedure of a 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 a remote procedure of an active picture:

```
TEXT _msg

_msg := " ... "

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

Asynchronous call of a remote procedure of an active picture:

```
TEXT _msg

_msg := " ... "

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

Synchronous call of a remote procedure, which is implemented by a JAPI 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 BROADCAST call:

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

Synchronous call of a 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 call of a 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 ended by a **RETURN**, or **END ProcedureName** actions. Otherwise, it is not able to update possible input-output parameters. For a synchronous call, the return value is set to the **_ERR_MISSING_RETURN** error.

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