ESL Interface

The meaning of the **ESL Interface** object is to provide a way to clarify and tidy up the relations among ESL scripts in an application (a parent of the object of ESL Interface type is a system object ESLINTERFACES).
Relations among scripts are formed by the calling of the **RPC** or **RPCX** procedures. This calling identifies the procedure name by text string in the target script. The text string is not checked at saving of script.

The object ESL Interface represents a definition of one or more ESL procedures. It defines a group of RPC procedures. The name of the object ESL Interface can be used in the definition of ESL script after keyword IMPLEMENTATION.

Example:

IMPLEMENTATION I.MsgServer

The name of one or more objects of ESL Interface type must follow the keyword IMPLEMENTATION. No action is allowed before it (e.g. definition of procedure, variable or other action).
ESL script must implement all procedures of each ESL Interface which follows the keyword IMPLEMENTATION (it means that the ESL script **implements the interface**).
On the other side, it is possible to call the procedures of ESL script which is implemented by some interface through the name of the proper interface. This enables to formalize (also to clarify) the logical relations among ESL scripts that are generated by the mutual calling of RPC procedures. The meaning of interface is in the dictation of procedures that the particular ESL script must implement.

The following example describes the use of the ESL interface in the design of simple action - sending the messages among users. On the user side, there is a simple scheme **S.MSGClient** and on the server side, there is an object of **Event E.MSGServer** type.

E.MSGServer represents the server that offers some services. These services are assured by **I.MsgServer** ESL Interface which guarantees the existence (implementation) of all necessary procedures (declared in the interface). The interface **I.MsgServer** provides the **RegisterClient** procedure. Each client who wants to send the messages is checked in by this procedure and gets the unique identifier. As the primary need is receiving messages asynchronously by the (registered) client, **E.MSGServer** provides this action and implements the **SendMessage** procedure which calls the registered client internally. This client must implement the **I.MsgClient** interface which ensures the existence of **ReceivMessage** procedure.

Interfaces implemented by scripts

E.MSGServer - interface **I.MsgServer**:

```
 ;*********************************************************************************
 ; Object name: I.MsgServer
 ; Interface of MSG Server
 ; Each MSG Server must implement following procedures

 ; Obligatory client registration
 ; Registration assigns the unique _clientUID
 RPC PROCEDURE RegisterClient(IN TEXT _clientName, IN INT _clientHOBJ, IN INT _clientProcessHOBJ, INT
_clientUID)
 ; Displacing of the client
 RPC PROCEDURE UnRegistrateClient(INT _clientUID)

 ; Procedure will return the list of all registered clients
 RPC PROCEDURE  GetClientList(RECORD NOALIAS (SD.Public_ClientList) _clients)

 ; Procedure will send the message _msg to registered client _dstClientUID
 ; Return code _retCode
 ; 0 - OK
 ; 1 - client _dstClientUID has not been registered
 RPC PROCEDURE SendMessage(IN INT _dstClientUID, IN TEXT _msg, INT _retCode)
 ;*********************************************************************************
```

Minimal (nonfunctional) implementation in ESL script:

```
 ; Interface of MSG Server
 IMPLEMENTATION I.MsgServer

 ; Client registration
 IMPLEMENTATION RPC PROCEDURE I.MsgServer^RegisterClient(IN TEXT _clientName, IN INT _clientHOBJ, IN INT
_clientProcessHOBJ, INT _clientUID)
 END RegisterClient

 ; Displacing of the client
 IMPLEMENTATION RPC PROCEDURE I.MsgServer^UnRegistrateClient(INT _clientUID)
 END UnRegistrateClient

 ; Procedure will return the list of all registered clients
 IMPLEMENTATION RPC PROCEDURE I.MsgServer^GetClientList(RECORD NOALIAS (SD.Public_ClientList) _clients)
 END GetClientList

 ; Procedure will send the message _msg to registered client _dstClientUID
 ; Return code _retCode
 ;   0 - OK
 ;   1 - client _dstClientUID has not been registered
 IMPLEMENTATION RPC PROCEDURE I.MsgServer^SendMessage(IN INT _srcClientUID, _dstClientUID, IN TEXT  _msg, INT
_retCode)
 END SendMessage
```

The implementation of the interface procedure starts with the keyword IMPLEMENTATION. Its name consists of the interface name and a symbol '^'. ESL script must implement all prescribed procedures.

S.MSGClient - interface **I.MsgClient**:

```
;*************************************************************************************
 ; Object name: I.MsgClient
 ; Interface of MSG Client
 ; Each MSG Client must implement following procedures

 ; Receiving the message _msg from client _srcClientUID
 RPC PROCEDURE ReceivMessage(IN INT _srcClientUID, IN TEXT _msg)
 ;*************************************************************************************
```

From the server's point of view (**E.MSGServer**), it is not important who communicates with the server (S.MSGClient picture or another object), but important is the implementation of the required interface. That is why the client can be represented by another scheme (or objects of Event type) in the real application. Important is to implement **I.MsgClient** interface.

Related pages:

ESL Interface configuration
PROCEDURE action
IMPLEMENTATION action
Fill procedures in the ESL script editor