ESL Interface

A meaning of object **ESL Interface** is to give a survey and visualization to the relations among ESL scripts in application (a parent of object of ESL Interface type is system object).
Relations among scripts are formed by the calling of **RPC** or **RPCX** procedure. This callings identifies the procedure name by text string in target script. The text string is not checked at saving of script.

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

Example:

IMPLEMENTATION I.MsgServer

The name of one or more object of ESL Interface type must follow the key word IMPLEMENTATION. Any action is not 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 key word IMPLEMENTATION (it means that ESL script **implements the interface**).
From other side, it is possible to call the procedures of ESL script which is implemented by some interface through the name of proper interface. This enables to formalize (also to make survey) the logical relations among ESL scripts that are generated by mutual calling of RPC procedures. The meaning of interface is in dictation of procedures which the particular ESL script must implement - it implements them in fact.

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

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

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 interface procedure starts by key word 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 side of server (**E.MSGServer**), communication side (scheme S.MSGClient or other object) is not important, but important is implementation of required interface. That is why the client can be represented by another scheme (or objects of Event type) in real application. Important is to implement interface **I.MsgClient**.

Related pages:

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