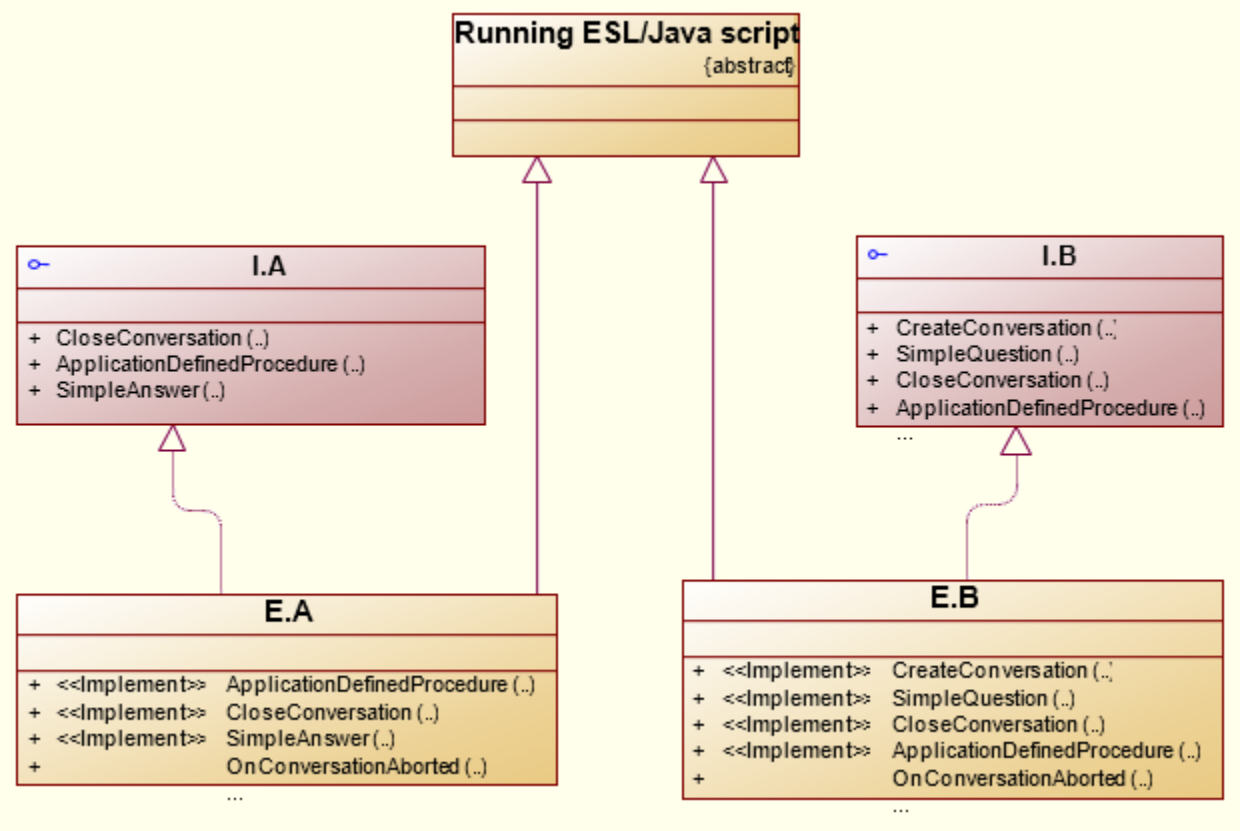
Using conversation

Using conversations

This article describes the use of conversations between ESL scripts.
We assume that there are instances of two scripts of *Server* type **E.A** and **E.B** (also opened picture can be a server script). These scripts run on two event handlers A.EVH and B.EVH with the instance 0 (each running script is uniquely addressable by a scheme *process;event;instance of event*).

In order to handle the conversations, there are created the objects of *ESL Interface* type I.A and I.B, which implement the events E.A and E.B (usage of objects of *ESL Interface* type is not a condition).

The situation is illustrated in the figure below:



Configuration of interfaces:

```
; this procedure uses the existing conversation
RPC PROCEDURE [_hTC, TC_C] ApplicationDefinedProcedure(IN INT _parameters)

; closing the conversation
RPC PROCEDURE [_hTC, TC_E] SimpleAnswer(IN INT _parameters)

; closing the conversation
RPC PROCEDURE [_hTC, TC_E] CloseConversation(IN INT _parameters)
```

ESL script, which implements I.A interface should contain the implementation of the ERROR procedure.

```
; called by an application server on transaction abort
RPC PROCEDURE [_hTC,  ERROR] OnConversationAborted
```

```
; procedure for creating the existing conversation (CALL TC_B)
RPC PROCEDURE [_hTC, TC_B] CreateConversation(IN INT _parameters)

; one question (more answers) (CALL TC_BE)
RPC PROCEDURE [_hTC, TC_BE] SimpleQuestion(IN INT _parameters)

; procedure for closing the conversation (CALL TC_E)
RPC PROCEDURE [_hTC, TC_E] CloseConversation(IN INT _parameters)

; procedure that uses the existing conversation
RPC PROCEDURE [_hTC, TC_C] ApplicationDefinedProcedure(IN INT _parameters)
```

ESL script which implements I.B interface, should contain the implementation of the ERROR procedure.

```
; called by an application server on conversation abort
RPC PROCEDURE [_hTC,  ERROR] OnConversationAborted
```

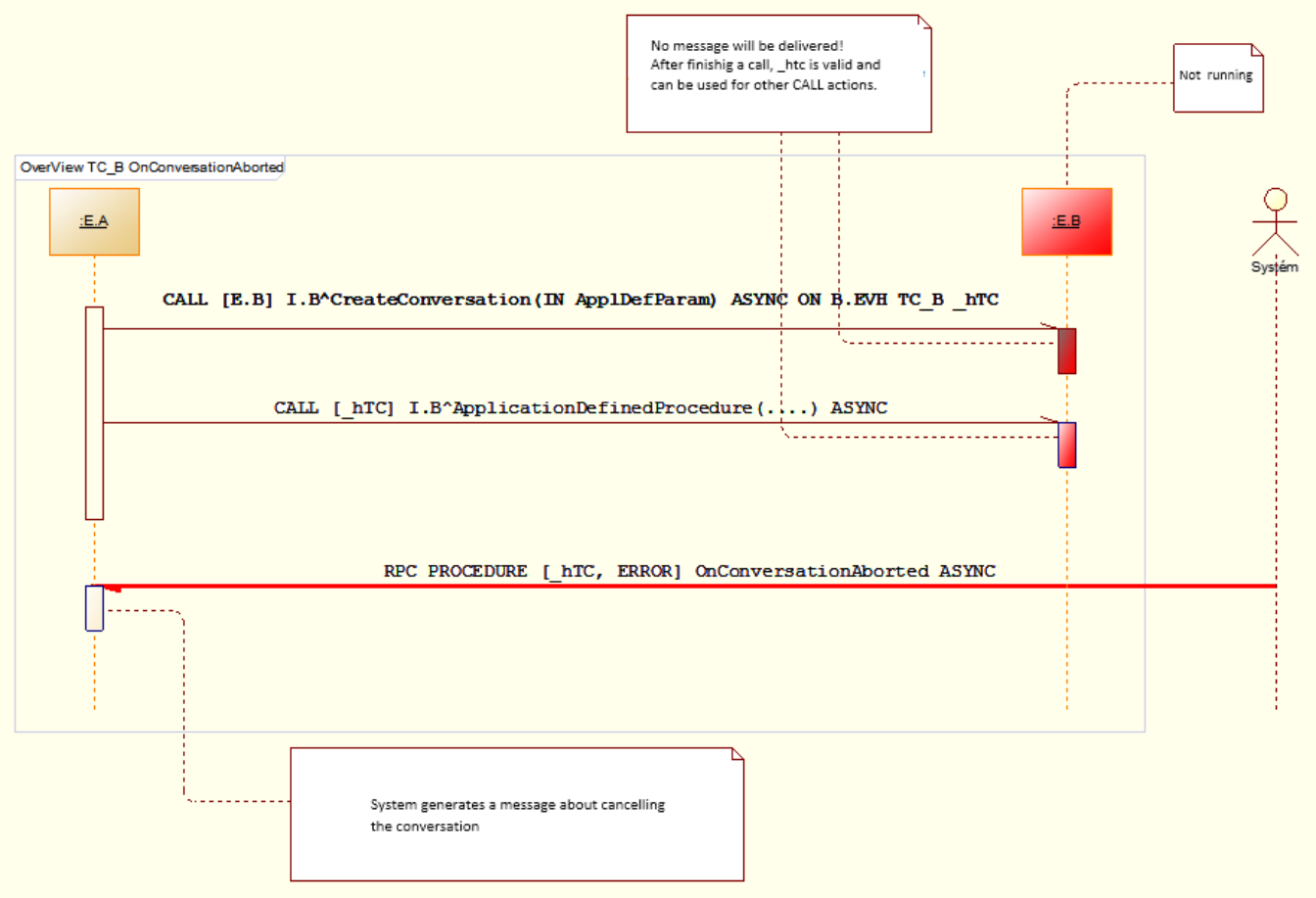
Creating conversation (TC_B)

It is supposed that the conversation, which is created by calling **CALL ... TC_B**, will exist longer, or is necessary for changing a more complex sequence of messages. The conversation is created the way that some script initiates it in relation to other scripts. In the example below, the conversation is initiated by E.A in relation to E.B:

```
CALL [E.B] I.B^CreateTransaction(IN ApplDefParam) ON B.EVH TC_B _hTC
```

_hTC is a declared variable of *INT* type. This action ensures creating the conversation, which will be identified by _hTC, and asynchronous sending of request for executing the RPC procedure *I.B^CreateTransaction* to E.B script. You should realize that if CALL action is executed successfully, the conversation will be created even though the addressed script E.EV;E.B;0 is, for some reason, inactive (impossible to execute the called procedure). This condition is considered to be a conversation error and the system ensures calling the procedure *[_hTrans, ERROR]* with appropriate _hTC.

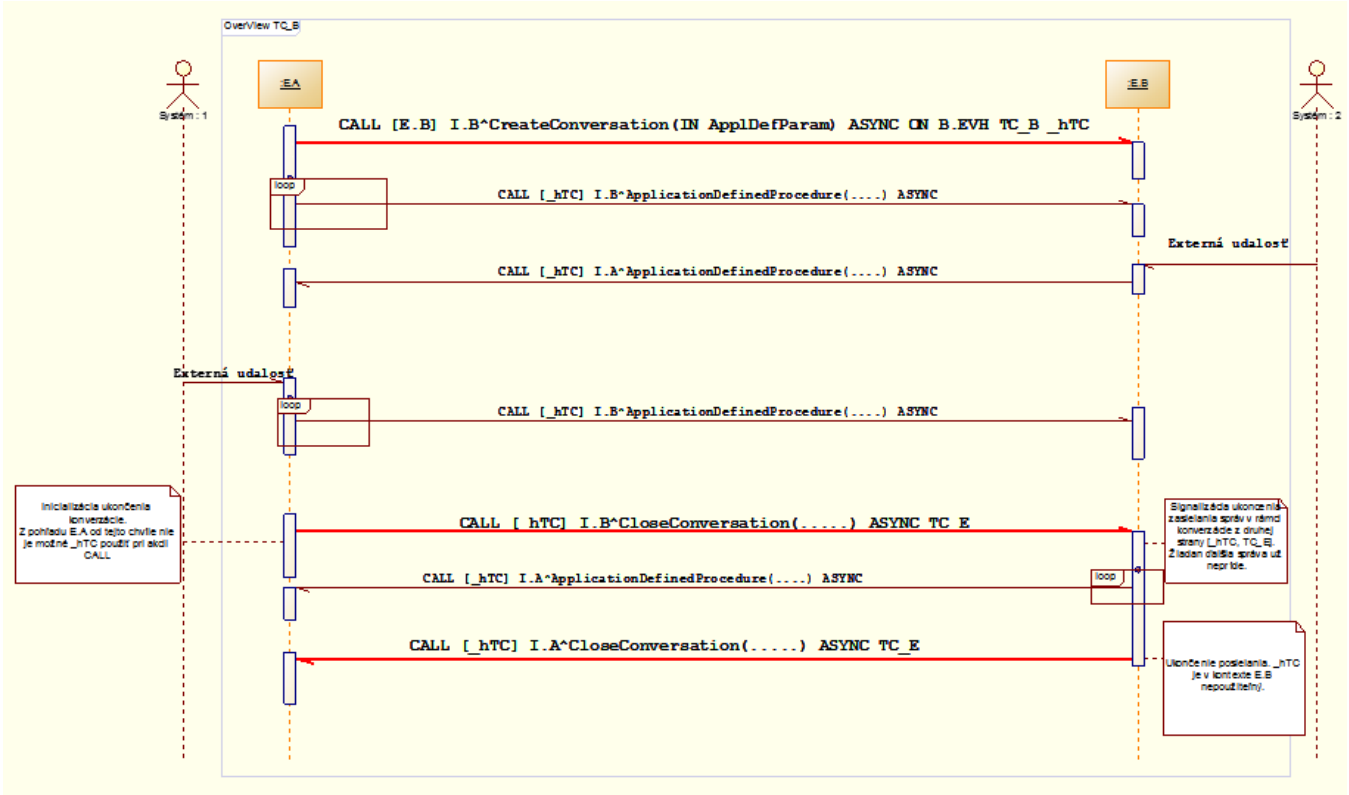
```
RPC PROCEDURE [_hTC, ERROR] OnConversationAborted
```



In case of successful call of procedure *I.B^CreateConversation*, the conversation is considered to be created and both scripts are, in relation to it, **equivalent** and they may use it for changing the messages as **asynchronous** calling the RPC procedures.

The conversation is closed by calling the procedure of another entity (as both scripts are equivalent, it is not important which entity initiates the closing) by CALL action with a keyword **TC_E**. In the example above, the action is called by E.A script. After calling the action, the identifier of conversation cannot be used for other callings of procedures (output is closed), but it can identify the conversation when processing the messages that may be still received (**RPC PROCEDURE [_hTC]**). The conversation will be completely closed after confirming this request from E.B script (E.B script calls CALL [_hTC] I. A^CloseTransaction(….) ASYNC **T_E**).

Changing the messages is described in the figure below:



Red lines represent the calls that manage the status of the conversation.

The system automatically generates new status of conversation in the events such as:

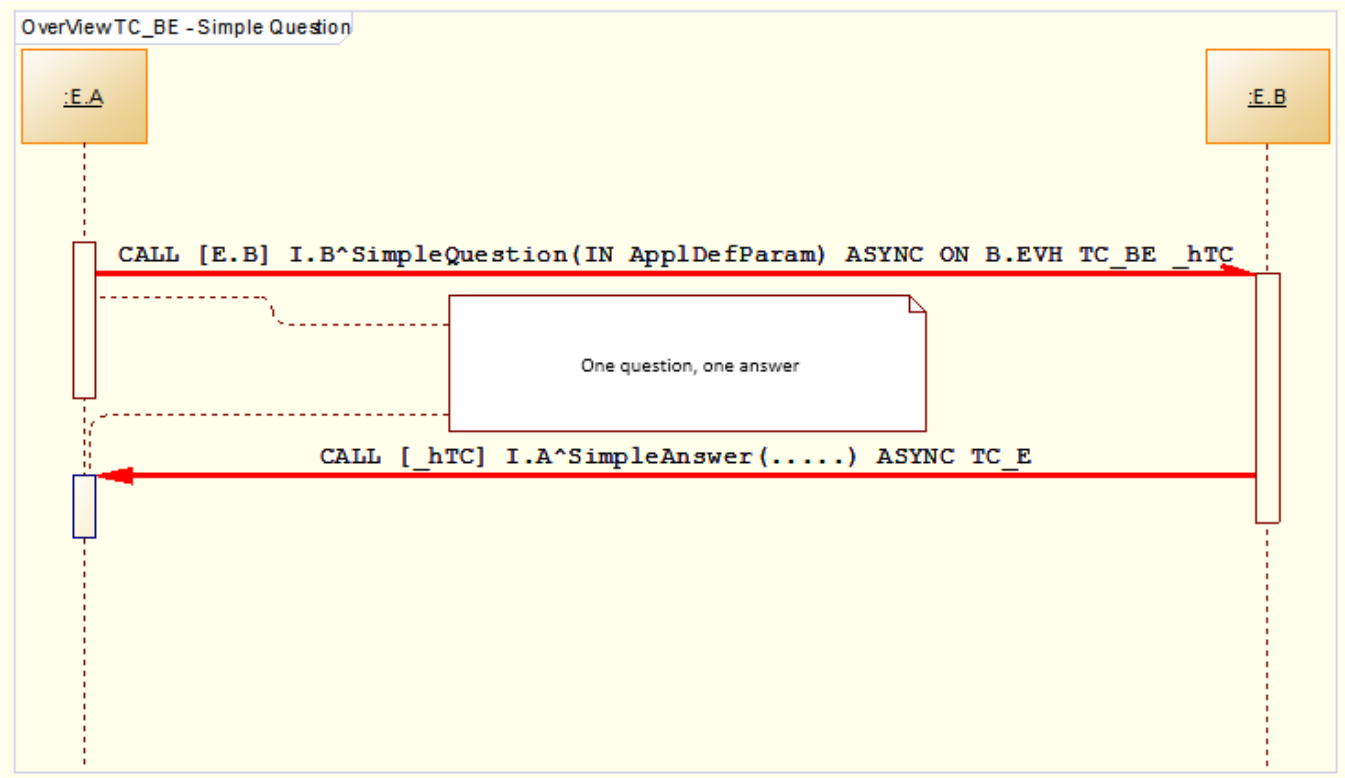
- stop or crash of process (A.EVH or B.EVH),
- stop the script that participates in conversation (uloženie z CNF, GRE, ...),
- ...

The changes of status, generated by the system, are sent to the participant of conversation as the call of **RPC PROCEDURE [_hTC, ERROR] ....**

Creating and simultaneous closing conversation - Question (TC_BE)

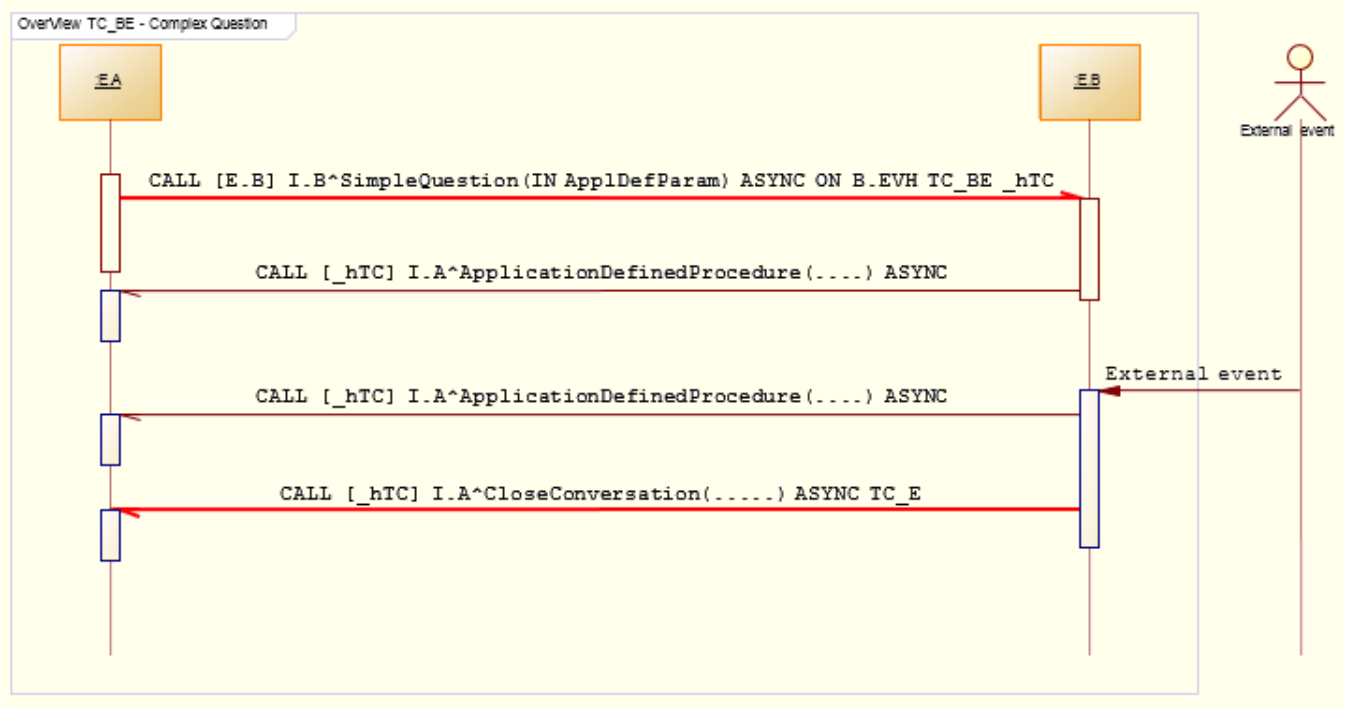
Opening the conversation of **TC_BE** type is important if an exchange of messages is in the format of one question and one or more responses.

The first situation:



If some error occurs when delivering the question (calls of RPC procedure), E.A script is informed about it by calling **RPC PROCEDURE [_hTC, ERROR]**.

The second situation:



Closing conversation (TC_E))

Each participating entity must close the conversation (if it was not closed by calling **RPC PROCEDURE [_hTC, ERROR]**).

Related pages:

Application-defined conversations
E.MAIL_SERVER example