User Identification in RPC Method

Often it is necessary to return from the RPC method only those data to which the user calling the RPC method has authorization. Sending a userName of the current user as a parameter of the RPC method is not a sufficient security measure because the SmartWeb architecture cannot prevent the logged in user to call the allowed RPC method with any parameters. This means that the user could identify through the userName parameter as someone else. That is why it is necessary to identify the calling user in particular ways described below.

The Smart Web supports authentication through D2000 users or through applicatively defined users in which case the SmartWeb server realizes the authentication through calling defined authentication RPC method.

Identification of D2000 User in RPC Method

In the case that the Smart Web application realizes the authentication through D2000 users, it is possible to call the GetUserObjName function in any RPC method that will return the userName of currently logged in user.

Identification of Applicatively Defined User in RPC Method

In the case of applicatively defined users, the process is more complicated and requires creating of a data container through the CNT_CREATE method where the key to container values will be the HOBJ of the logged in session and the value to the key userName. The container will be filled with calling CNT_INSERT in the so-called logon method which will be called by SmartWeb in every successful login. Acquiring a user name is then realized through calling CNT_GETITEM in the RPC method. A sample implementation of this functionality is presented below. RPC methods that need to identify a logged in user only have to call the method `getCurrentUserName`.

```
INT _CNT_sessions ; container of all active sessions of all users

; method is registered in Smart Web application as "logOn" method,
; which is called after every successful authentication
RPC PROCEDURE logOn(IN TEXT _UserName, BOOL _Ok)
  INT _hobj_session
  BOOL _bFound
  _Ok := @TRUE

  _hobj_session := %GetRPCCallerProcess()
  ; control of session existence, cannot exist, opposite state is error
  CNT_FIND _CNT_sessions, _hobj_session, _UserName, _bFound
  IF _bFound THEN
    LOGEX "ERR: same SESSION of user already exists: " + %IToStr(_hobj_session) PRIORITY _LOG_PRTY_ERROR
    _Ok := @FALSE
    RETURN
  ENDIF
  ; registration of calling method onSessionClosed in "session change",
  ; meaning either standard logout or any other session ending
  _Ok := %OpenRefToObject(_hobj_session, @TRUE)
  IF !_Ok THEN
    LOGEX "ERR: unknown SESSION for opening: " + %IToStr(_hobj_session) PRIORITY _LOG_PRTY_ERROR
    RETURN
  ENDIF
  ON CHANGE (_hobj_session) GOTO onSessionClosed
  ; inserting of mapping "session HOBJ -> user name" inot container
  CNT_INSERT _CNT_sessions, _hobj_session, _UserName
END logOn

; ended user session (by logout or by other ending of user session)
PROCEDURE onSessionClosed(IN INT _procValue, IN ALIAS _hobj, IN INT _row, _col)
  BOOL _bFound
  INT _hobj_session
  TEXT _userName
  _hobj_session := _hobj\HBJ

  CNT_FIND _CNT_sessions, _hobj_session, _userName, _bFound
  IF ! _bFound THEN
    LOGEX "ERR: unknown SESSION was ended " + %IToStr(_hobj_session) PRIORITY _LOG_PRTY_ERROR
    RETURN
  ENDIF
  CNT_DELETE _CNT_sessions, _hobj_session
  %CloseRefToObject(_hobj_session)
  ON CHANGE (_hobj_session) GOTO onSessionClosed NONE
END onSessionClosed

; RPC method returns name of currently logged in user
RPC PROCEDURE getCurrentUserName(TEXT _userName)
  INT _hobj_session
  BOOL _bFound
  _hobj_session := %GetRPCCallerProcess()
  CNT_FIND _CNT_sessions, _hobj_session, _userName, _bFound
  IF ! _bFound THEN
    LOGEX "ERR: unknown SESSION was ended " + %IToStr(_hobj_session) PRIORITY _LOG_PRTY_ERROR
    RETURN
  ENDIF
END getCurrentUserName

BEGIN
  CNT_CREATE _CNT_sessions
END
```