Using Event Hooks and API Roundtable TSMS

Roundtable Development Team

TUGBOAT
SOFTWARE

Table of Contents

Table of Contents **1**

1 Introduction **2**

2 Intercepting Roundtable Events **2**

2.1 The Event Procedure **2**

2.2 Example – Enforcing Update Notes **3**

3 Using the Roundtable API **4**

3.1 The API Procedure **4**

3.2 Example – Creating a Task **4**

3.2.1 Initializing the API 4

3.2.2 Making the API Call 5

3.2.3 Shutting Down the API 5

3.2.4 The Entire Procedure 5

4 Conclusion **6**

1 Introduction

Roundtable TSMS delivers Software Configuration Management benefits to your OpenEdge development environment. The Roundtable event hook extensions and application programming interface allow you to take those benefits one step further by automating processes as well as enhancing or overriding default behavior. Since Roundtable is written in the OpenEdge ABL, all event hook and API coding is also done using the ABL!

The Roundtable event hook extensions allow you to customize behavior without having to modify Roundtable source code. All event hooks are implemented through a single event procedure. Each time an event occurs in Roundtable, such as checking-in an Object or creating a Task, Roundtable publishes both before and after context of this event to the event procedure. This allows you to build single point of intercept for all event hooks.

The Roundtable API is also designed for simplicity and portability by providing a single entry point for all API calls. This methodology insulates your custom applications from changes made to Roundtable core architecture and ensures that your customizations will continue to work with upgrades and service packs.

This white paper provides a head start on using the Roundtable GUI client or server event handler and API in Roundtable TSMS 9.1D and greater.

2 Intercepting Roundtable Events

The Roundtable event handler extension makes use of the ABL PUBLISH and SUBSCRIBE statements.

2.1 The Event Procedure

Roundtable publishes before and after context to the Roundtable event handler procedure. This procedure is deployed as source and is located in *<rtb install dir>/rtb_events.p*. Every time a supported event occurs, the following parameters are published in active OpenEdge session to the event handler procedure.

```
DEFINE INPUT  PARAMETER Pevent   AS CHARACTER.
DEFINE INPUT  PARAMETER Pcontext AS CHARACTER.
DEFINE INPUT  PARAMETER Pother   AS CHARACTER.
DEFINE OUTPUT PARAMETER Pok      AS LOGICAL INIT YES.
```

Parameter	Description
p_event	The name of event.
p_context	The context in which the hook was called, such as the ROWID of the record being acted upon.
p_other	Additional context information that may be needed by programs intercepting the hook, such as the Workspace currently selected during the intercepted event.
p_ok	Setting to FALSE will allow you to cancel default behavior from the “before” event hooks.

Please see the Roundtable TSMS User Guide or the comments at the top of *rtb_events.p* for a complete description of published events and the data passed in each parameter.

2.2 Example – Enforcing Update Notes

A common example of enhancing default behavior would be to disallow the check-in of an Object version that does not have update notes. This simple customization ensures that developers document changes that they made to an Object version.

```
PROCEDURE evRtbUserEvent :
/*------------------------------------------------------------------------------
  Purpose:     Roundtable Event Handler
  Parameters:  Pevent
               Pcontext
               Pother
               Pok
  Notes:       RETURN a value from this procedure to return an error message
               to the caller.
 ------------------------------------------------------------------------------*/
  DEFINE INPUT  PARAMETER Pevent   AS CHARACTER.
  DEFINE INPUT  PARAMETER Pcontext AS CHARACTER.
  DEFINE INPUT  PARAMETER Pother   AS CHARACTER.
  DEFINE OUTPUT PARAMETER Pok      AS LOGICAL INIT YES.

  DEFINE VARIABLE cError AS CHARACTER NO-UNDO INITIAL "".

  IF Pevent = "checkinObjectBefore" THEN DO:

    DEFINE VARIABLE hVer    AS HANDLE NO-UNDO.
    DEFINE VARIABLE hObj    AS HANDLE NO-UNDO.
    DEFINE VARIABLE hVerBuf AS HANDLE NO-UNDO.
    DEFINE VARIABLE hObjBuf AS HANDLE NO-UNDO.

    /*
     Use Roundtable proxy procedures to get the Object and Version data
    */
    RUN rtb/proxy/p/rtbGetObjectByRowid.p
     (INPUT Pcontext, OUTPUT TABLE-HANDLE hObj).

    hObjBuf = hObj:DEFAULT-BUFFER-HANDLE.
    hObjBuf:FIND-FIRST.

    RUN rtb/proxy/p/rtbGetVersionByRowid.p
     (INPUT hObjBuf::objVersionRowid, OUTPUT TABLE-HANDLE hVer).

    hVerBuf = hVer:DEFAULT-BUFFER-HANDLE.
    hVerBuf:FIND-FIRST.

    /*
     If not update notes are present, set the error message
    */
    IF hVerBuf::upd-notes = "" THEN DO:
      cError = "Object version update notes cannot be blank!".
      pOk = FALSE.
    END.

    DELETE OBJECT hVer NO-ERROR.
    DELETE OBJECT hObj NO-ERROR.

  END.

  RETURN cError.

END PROCEDURE.
```

Since the rtb_object ROWID is passed via Pcontext, it could have been possible to look up in the rtb_object and rtb_ver records directly. However, direct access of the Roundtable repository database is discouraged unless absolutely necessary.

3 Using the Roundtable API

The Roundtable API provides access to common Roundtable functionality through a single procedure. The API can be run either stand-alone or from within an active Roundtable session.

3.1 The API Procedure

The Roundtable API procedure is deployed as source and is located in *<rtb install dir>/rtb/p/rtb_api.p*. Complete details on using the API can be found in the definitions section of the API procedure.

3.2 Example – Creating a Task

3.2.1 Initializing the API

In its most basic form, initializing the API is just a matter running the API procedure persistently and obtaining its handle. In this example, it is assumed that we are running the API as a stand-alone process therefore there are additional initialization steps needed to set the environment.

```
/*
 Roundtable add task API example
*/
&SCOPED-DEFINE RTB "c:/work/demo/rtbdemo102b"

DEFINE VARIABLE cError     AS CHARACTER NO-UNDO.
DEFINE VARIABLE cDlcPath   AS CHARACTER NO-UNDO.
DEFINE VARIABLE hApi       AS HANDLE    NO-UNDO.
DEFINE VARIABLE iSid       AS INTEGER   NO-UNDO.
DEFINE VARIABLE rTaskRecid AS RECID     NO-UNDO.

IF NOT CONNECTED("rtb") THEN
  CONNECT -db rtb.db -1.

ASSIGN
  cDlcPath = PROPATH /* Preserve the default PROPATH */
  PROPATH  = {&RTB} + "," + cDlcPath.

RUN rtb/p/rtb_api.p PERSISTENT SET hApi.

RUN login IN hApi
  (INPUT "sysop",    /* User */
   INPUT "password", /* Password */
   OUTPUT iSid,      /* Session-Id */
   OUTPUT cError).

RUN set paths IN hApi
  (INPUT {&RTB},     /* RTB Install Path */
   INPUT cDlcPath,   /* Default PROPATH */
   OUTPUT cError).
```

3.2.2 Making the API Call

Once the API procedure is running, it is now just a matter of calling the appropriate routine in hApi with the appropriate parameters. The following example creates a new Task in the Roundtable repository.

```
RUN add_task IN hApi
  (INPUT "devel",            /* Task Workspace */
   INPUT "sysop",            /* Task User */
   INPUT "sysop",            /* Task Manager */
   INPUT "Task summary",     /* Task Summary */
   INPUT "",                 /* User Ref */
   INPUT "Task description", /* Task Description */
   INPUT "Central",          /* Share Status */
   INPUT "",                 /* Task Directory */
   OUTPUT rTaskRecid,        /* rtb_task RECID */
   OUTPUT cError).
```

3.2.3 Shutting Down the API

When you are done, you should log out of the Roundtable session and delete the API procedure that is running persistently.

```
RUN logout IN hApi
  (INPUT iSid,     /* Session-Id */
   OUTPUT cError).

DELETE PROCEDURE hApi NO-ERROR.
```

3.2.4 The Entire Procedure

Below is the entire add_task API example. Please note that this is not a production quality example. It would be up to you to add necessary error checking and other optimizations. For this white paper, I wanted to keep the code as simple as possible.

```
/*
 Roundtable add_task API example
*/
&SCOPED-DEFINE RTB "c:/work/demo/rtbdemo102b"

DEFINE VARIABLE cError     AS CHARACTER NO-UNDO.
DEFINE VARIABLE cDlcPath   AS CHARACTER NO-UNDO.
DEFINE VARIABLE hApi       AS HANDLE    NO-UNDO.
DEFINE VARIABLE iSid       AS INTEGER   NO-UNDO.
DEFINE VARIABLE rTaskRecid AS RECID     NO-UNDO.

IF NOT CONNECTED("rtb") THEN
  CONNECT -db rtb.db -1.

ASSIGN
  cDlcPath = PROPATH /* Preserve the default PROPATH */
  PROPATH  = {&RTB} + "," + cDlcPath.

RUN rtb/p/rtb_api.p PERSISTENT SET hApi.

RUN login IN hApi
  (INPUT "sysop",    /* User */
   INPUT "password", /* Password */
   OUTPUT iSid,      /* Session-Id */
   OUTPUT cError).

RUN set paths IN hApi
  (INPUT {&RTB},     /* RTB Install Path */
   INPUT cDlcPath,   /* Default PROPATH */
   OUTPUT cError).
```

```
RUN add_task IN hApi
  (INPUT "devel",            /* Task Workspace */
   INPUT "sysop",            /* Task User */
   INPUT "sysop",            /* Task Manager */
   INPUT "Task summary",     /* Task Summary */
   INPUT "",                 /* User Ref */
   INPUT "Task description", /* Task Description */
   INPUT "Central",          /* Share Status */
   INPUT "",                 /* Task Directory */
   OUTPUT rTaskRecid,        /* rtb task RECID */
   OUTPUT cError).

RUN logout IN hApi
  (INPUT iSid,     /* Session-Id */
   OUTPUT cError).

DELETE PROCEDURE hApi NO-ERROR.

RETURN.
```

4 Conclusion

Using the Roundtable TSMS event handler or API is fairly straight-forward and simple. By combining your OpenEdge ABL expertise with your Roundtable knowledge, you will soon be on your way to creating useful and time-saving extensions and customizations. Good luck!