Roundtable White Paper
Double Compile

Summary

This white paper describes how to set up Roundtable to copy an object to a second machine and compile it before performing the standard workspace compilation. It was prompted by the request for a way to have RTB automatically perform a UNIX compile on an object every time it is compiled in an NT workspace. It requires the use of RTB V9.1B or higher.

The AppServer Interface API and standard RTB client hooks are utilized in this example. The AppServer Interface Tool is a utility designed to facilitate copying of objects to a directory on an AppServer machine. It requires the installation of the Progress AppServer. By copying one of RTB's directories into the AppServer's ProPath, RTB is able to use the AppServer to move code to the AppServer machine. The code is passed via a Progress Temp-Table, eliminating the need for OS-Copy or FTP. In this case, the AppServer session must be connected to the RTB repository. See the RTB User's Guide and the files in "<rtb install>/rtb/appsrvr" for more information on the RTB AppServer Interface.

The Double Compile is a three-part process. First, the OBJECT-COMPILE-BEFORE hook is intercepted in the RTB GUI client. If the object should be moved to and compiled on the UNIX machine, the AppServer Interface API is called from the hook to move the object. Third, a hook in the AppServer Interface Tool on the AppServer then is intercepted and used to compile the object in the AppServer Session.

First – Intercept the "OBJECT-COMPILE-BEFORE" Hook

This part is easy. Add the following block to your copy of "<rtb install>/rtb_evnt.p" and recompile it. Rtb_evnt.p is the standard RTB hook procedure. When the event passed to it is "OBJECT-COMPILE-BEFORE", we know to move and compile the object using the context accessible to the hook.

```
/*
 Double Compile Example
 Call rtb_msrv.p to move object with the AppServer Interface API and
 compile on the AppServer
*/
IF p_event = "OBJECT-COMPILE-BEFORE" THEN
  RUN myhacks/p/rtb_msrv.p (INPUT p_context).
```

Second – Check and Move the Object

Place the following "rtb_msrv.p" file in a directory called "myhacks/p" off of your Roundtable install directory (or anywhere in your ProPath). Change the value of Mappserver to match your environment. Mremote-path reflects the root directory for your application on the AppServer machine. The .p checks to see if the object needs to be processed. If so, it connects to the AppServer using the Progress AppServer Utils then calls the RTB AppServer Interface API to move the object.

```
/*
 FILE: rtb_msrv.p

 DESCRIPTION:
 Move file to Appserver.

 Invoke from the "OBJECT-COMPILE-BEFORE" hook:

 IF p_event = "OBJECT-COMPILE-BEFORE" THEN
   RUN myhacks/p/rtb_moveserver.p (INPUT p_context).

 This example connects to the AppServer from withing this .p then
 disconnects.  If this process is going to be run alot, it would be
 better to connect in the select workspace hook and leave the
 connection open until another workspace is selected.
*/

DEFINE INPUT PARAMETER Pobj-recid AS CHARACTER NO-UNDO.

DEFINE VARIABLE MAppHdl         AS HANDLE       NO-UNDO.
DEFINE VARIABLE Merror          AS CHARACTER    NO-UNDO.
DEFINE VARIABLE Mappserver      AS CHARACTER    NO-UNDO.
DEFINE VARIABLE Mremote-path    AS CHARACTER    NO-UNDO.

/* Change to match your AppService name */
ASSIGN Mappserver   = "unix"
       Mremote-path = "/work/appserver".

/* Progress AppServer Utils */
{adecomm/appserv.i}

/*
 Add code here to check if the object should be moved and compiled on
 the AppServer.  For example, you may move only objects in a specific
 module or of a specific subtype.  If the object should not be moved,
 simply leave the procedure.
*/

/* Connect to the AppServer */
RUN appServerConnect IN appSrvUtils
  (INPUT  Mappserver,  /* Application Service */
   INPUT  NO,          /* Security */
   INPUT  "",          /* Info */
   OUTPUT MAppHdl).    /* Handle to AppServer */

RUN rtb/appsrvr/rtb_assnd.p
  (INPUT Pobj-recid,
   INPUT MApphdl,
   INPUT Mremote-path,
   INPUT NO,
   OUTPUT Merror).

IF Merror <> "" THEN
  MESSAGE Merror VIEW-AS ALERT-BOX.

RUN appServerDisconnect IN appSrvUtils (INPUT Mappserver).
```

Third – Compile the file using the “OBJECT-WRITE” hook

The programs running on the AppServer also provides hooks. Add the following block to “rtb_asevnt.p” on the AppServer to intercept the move of an object then pass rtb_comp2.p the context so it can compile the file.

```
/* Compile object after it has been moved */
IF p_event = "object-write" THEN
  RUN rtb_comp2.p(INPUT p_context,
                  INPUT p_other).
```

Compiling the object is the trickiest part. You could simply execute the compile command against the file that was moved to the machine. This example makes sure that the RTB repository is connected before continuing. It then finds the object and ver records in the repository. Using the fetched information, it determines the workspace name so as to connect to the proper workspace databases (I recommend removing that part and connecting to the workspace databases in the AppServer = you would use separate AppServer partitions for each workspace). It then traps the compiler output to a file in case the program does not compile. If any errors are encountered in this process, they are sent to a .log file.

As is, these programs would return and the RTB GUI client would continue on and compile the workspace object. You would have to check the .log file to see how things went. It would be better to add a flag to the repository, mark it as false if the compile fails, then cancel the workspace compile in the “OBJECT-COMPILE-BEFORE” hook that originally started this process. That is why we used the “BEFORE” hook. When the workspace compile is cancelled, the object status remains as “compile required”.

```
/*
 FILE:
 rtb_comp2.p

 DESCRIPTION:
 Compile an object that has just been moved to the AppServer with the
 RTB AppServer Interface Tool.

 If the AppServer Interface Tool is being called through it's API from
 the before object compile hook, the "best" way to handle failed compiles
 would be to set a flag in the database that the compile failed then
 cancel the regular compile.  This example just writes to a "rtbcomp.log"
 file in the current directory.

 Make sure the agent that moves the code is also connected to the repository
 so that the lookups have tables to look-up.

 WARNING:
 When compiling an object RTB assumes the include files are current. If
 you are batch moving objects
*/

DEFINE INPUT PARAMETER Pobj-recid   AS CHARACTER    NO-UNDO.
DEFINE INPUT PARAMETER Pfile-name   AS CHARACTER    NO-UNDO FORMAT "x(60)".

DEFINE VARIABLE Merror          AS LOGICAL      NO-UNDO.
DEFINE VARIABLE Mwarning        AS LOGICAL      NO-UNDO.
DEFINE VARIABLE Merror-msg      AS CHARACTER    NO-UNDO.
DEFINE VARIABLE Merror-line     AS CHARACTER    NO-UNDO.
DEFINE BUFFER   Brtb_object     FOR rtb.rtb_object.
DEFINE BUFFER   Brtb_ver        FOR rtb.rtb_ver.
DEFINE STREAM   Slog.
```

```
/* Output file to report any issues */
OUTPUT STREAM Slog TO c:\work\rtbcomp.LOG APPEND.

/* Make sure the RTB database is connected */
IF NOT CONNECTED("rtb") THEN DO:
  PUT STREAM Slog TODAY STRING(TIME,"HH:MM:SS") SKIP
      "Logical DB RTB not connected"  SKIP.
  OUTPUT STREAM Slog CLOSE.
  LEAVE.
END.

/* Find rtb_ver record to figure out if the object is compilable */
FIND Brtb_object WHERE
     RECID(Brtb_object) = INTEGER(Pobj-recid) NO-LOCK NO-ERROR.
FIND Brtb_ver OF Brtb_object NO-LOCK NO-ERROR.

/* The rtb_ver record should ALWAYS be available from here */
IF NOT AVAILABLE Brtb_ver THEN DO:
  PUT STREAM Slog TODAY STRING(TIME,"HH:MM:SS") SKIP
      Pobj-recid     SKIP
      "REPO ERROR!: Could not find object in the repository"   SKIP.
  OUTPUT STREAM Slog CLOSE.
  LEAVE.
END.
ELSE DO:
  /* Don't compile unless the object is marked compilable */
  IF Brtb_ver.Compiles =FALSE THEN DO:
    OUTPUT STREAM Slog CLOSE.
    LEAVE.
  END.

  /*
   Database connection time!!!  It is easier just to put the
   connection information here rather than look up the parameters
   in the repo - besides, you may be managing the connections for
   the workspace databases manually (blanking out the connection
   params for the PDBASE objects in the workspace).

   It would be better just to have an Appserver partition dedicated
   to each workspace so that the workspace DB connections can be
   placed on the Broker startup line,
  */

  CASE Brtb_object.wspace-id:
  WHEN "devel" THEN DO:
    CONNECT -db c:\work\demo9\demodata\devel\sports.db -RO NO-ERROR.
  END.
  END CASE.

  /* Temporary output file for the compile error messages */
  OUTPUT TO rtberror.LOG NO-ECHO KEEP-MESSAGES.

  /* Compile the file */
  COMPILE VALUE(Pfile-name) SAVE.
  OUTPUT CLOSE.
```

```
  ASSIGN Merror   = COMPILER:ERROR
         Mwarning = COMPILER:WARNING.

  /* disconnect the workspace DBs --- */
  DISCONNECT sports.

  /*
   Put any error messages in log file - the best thing would be to set a
   flag in the repository so that the before compile hook can cancel the
   standard compile if it called the AppServer Interface API
  */

  IF Mwarning OR Merror THEN DO:
    /* Populate Merror-msg with the compiler errors */
    INPUT FROM rtberror.LOG.
    REPEAT:
      IMPORT UNFORMATTED Merror-line.
      ASSIGN Merror-msg = Merror-msg + Merror-line.
    END.
    INPUT CLOSE.

    /* output the compiler errors to the log file */
    PUT STREAM Slog TODAY STRING(TIME,"HH:MM:SS") SKIP
              Pfile-name                          SKIP.
    PUT STREAM Slog UNFORMATTED Merror-msg        SKIP.
    OUTPUT STREAM Slog CLOSE.
    LEAVE.
  END. /* if Mwarning or Merror */

END. /* else do (if not available Brtb_ver)*/
```