Roundtable White Paper
Managing Customizations

Prerequisites

1. Read the entire Roundtable User's Guide with special attention to workspaces, workspace sources, products, workspace modules, product modules, and object variants.
2. Read the white paper "Schema made easy" if you plan on customizing schema.

Object Variants Described

Customizations

Customizations typically consist of quick bug fixes or feature enhancements for a particular user. These are tracked in Roundtable through custom product modules and custom workspaces (see "object variants" in the Roundtable User's Guide). Objects are "flagged" as variants when they are assigned to a custom product module.

For example

You divide your application into two workspace modules, "code" and "dict" (probably because you had a subdirectory called "code" before Roundtable - it is sometimes easier to think of workspace modules as pointers to directories). Your define a core product called "coreprod" with two product modules, "corecode" and "coredict". These two product modules are mapped to the "code" and "dict" workspace modules (see the Roundtable User's Guide for information on setting up products and modules). You then define a second product to track customizations for "customer1" called "cust1prod" with two product modules, "cust1code" and "cust1dict". You also map these two product modules to the "code" and "dict" workspace modules.

You create two development workspaces: COREDEVEL for working on your core baseline, CUST1DEVEL for working on customizations for "customer1". Your COREDEVEL workspace would have the "coreprod" product and product modules sourced as primary. Your CUST1DEVEL workspace would have the "cust1prod" product and product modules sourced as primary. You would also source the "coreprod" product and modules from COREDEVEL. This is done so that you can import the core base-line "coreprod" from COREDEVEL. All versioning will be done using the primary "cust1prod" product modules (see below).

Creating the object variant

When you want to create a customization in the CUST1DEVEL workspace, you "*move*" the object from the "corecode" product module to the "cust1code" product module (I recommend adding security so that developers cannot check out objects still in the "corecode" product module). You move an object using the same steps as creating a new object - when Roundtable finds the same object already in a different product module, it will ask you if you are trying to move the object. The move process simply changes which product module is assigned to the object and checks it out with version numbers unique to that object in that module.

If the core product module and the custom product module are mapped to the same workspace module (as in our example where "corecode" and "cust1code" are both mapped to the "code" workspace module), the physical object does not actually move. Roundtable now knows the object is a variant since it is in another module. When you change the object in COREDEVEL and import into CUST1DEVEL from

COREDEVEL, Roundtable will not overwrite the object variant. You no longer have to use separate directories for variants.

Moving new changes/fixes forward into custom workspaces

The import list

After creating many changes in COREDEVEL, you typically mark a baseline by creating a release. You then import that baseline into Test or QA areas (see the Roundtable user's Guide form more information about releases and imports). You also want to move this baseline into your custom development workspaces, either straight from COREDEVEL or from one of the test or production workspace further down the chain. You do NOT want Roundtable to overwrite an object you customized with an object changed in the core baseline.

When you build the import list, Roundtable automatically excludes objects that are in different product modules between the two workspaces. If you highlight a customized object in the list, you can see in the details that the object is in (using our example) the "corecode" product module in the source workspace and in the "cust1code" product module in the target workspace. If you toggle the object to INCLUDE, the object from the core baseline will be imported (the object is then back in the "corecode" product module).

Release Report

A release report is a good tool for finding changes in the baseline that need to be merged into custom objects. By default, the release report shows all objects changed between two releases. In Roundtable V9.1C and above, there is an option on the release report to also show if the object has a variant in another workspace, so that you know every change that must be merged into a variant.

You can also modify the release report in earlier version of Roundtable to automatically show all object variants. The release report is provided in source form as <rtb install directory>/rtb/w/rtb_0405.w. Add the following code just before the comment, " /* --- Find next object to process --- */", in the "do_report" procedure:

```
/* GW find object variants. */
   DEFINE BUFFER B2rtb_object FOR rtb.rtb_object.

   FOR EACH B2rtb_object WHERE B2rtb_object.wspace-id <> Brtb_hist.wspace-id
                           AND B2rtb_object.obj-type  = Brtb_hist.obj-type:U
                           AND B2rtb_object.object    = Brtb_hist.object
                           NO-LOCK:

      IF B2rtb_object.pmod <> Brtb_hist.pmod THEN DO:
        PUT "  OBJECT VARIANT: workspace- "
            B2rtb_object.wspace-id
            "pmod - "
            B2rtb_object.pmod
            SKIP.
      END.
    END. /* FOR EACH B2RTB_OBJECT */

   /* --- Find next object to process --- */
```

This customization to the release report will show you every workspace that contains a custom version of the object (just like the V9.1C report).

Merging customizations back into the core baseline

PCODE

After you have identified the objects that have been customized, you need to determine if the object variant can be brought directly back into the core base-line or if the customizations must be merged into the core object.

Typically, if the object has been changed in the core baseline since your customization, you must merge the customizations to the core object. To do this in our example, you would check out (version) the object in COREDEVEL and add the changes (use the differencing tools in Roundtable to compare the core object to the object variant and find the differences). You then import the new object version into CUST1DEVEL (or simply assign the object version if you don't want to bother creating a release and importing). The object will be "excluded" in the import list by default since Roundtable sees it as a custom variant. Toggle the object to "include" and RTB will remove the variant and assign (bring over) the new object version that is in the core product module.

For example, you have V020000 of "program.p" in the "corecode" module in both COREDEVEL and CUST1DEVEL. You then move the object to "cust1code" in the CUST1DEVEL workspace creating V010000. Later, you check out "program.p" in COREDEVEL creating V020001 in the "corecode" product module. There are now changes made in both the custom workspace and to core object. The change in the core object is not in the customization. To merge the customization into the base-line, you would version "program.p" in COREDEVEL to V020100 and modify it to include the changes from V010000 over in "cust1code". You would then import V020100 forward into CUST1DEVEL, eliminating the custom version from the custom baseline.

If the object has not changed in the core baseline, you could use the same method above. It may be easier to "move" the object in CUSE1DEVEL from the "cust1code" product module to the "corecode" product module, then check it in (the move checks out the object). You can then either reverse the source from CUST1DEVEL to COREDEVEL and import it back to COREDEVEL, or simply assign the new object version in the COREDEVEL workspace (the object is already in the "corecode" module).

Schema objects

You use the same method to merge schema objects back into the base-line. You must remember the Roundtable rules of schema. Table objects are assigned to the workspace and are assigned to databases. Field objects are assigned to the workspace and to tables.

A "version" of a database has a certain set of tables assigned to it. If V010000 of the "sports" database has the "customer" (V010000) and "invoice"(V010000) tables assigned to it, then you check out sports (V020000) and add the "order" (V010000) table, V020000 of sports now has three tables assigned to it.

If you then check out the "customer" table creating V010001 of the table, the sports database version DID NOT CHANGE. It still has the same "set" of tables assigned to it; "customer", "invoice", and "order". It is still at V020000.

Lets now lets apply this to our example case. You have Sports V010100 in the coredict product module with the "customer" and "invoice" tables assigned to it . You move it to the "cust1dict" product module in the CUST1DEVEL workspace creating V010000 (unique version numbers for this object in this module). You then add the "order" table to it and check it in.

Back in COREDEVEL, you check out "sports" in the "coredict" module creating V010200. You then add the "sales-rep" table and check it in. You have now both customized and modified the database. To bring the order table into the core baseline, you cannot simply move it nor can you move just the table

object to the "coredict" product module. The table object must be moved from "cust1dict" to "coredict", and the "sports" database must be checked out so that the table can be assigned to it.

You would check out the "sports" database in the COREDEVEL workspace (currently assigned the "coredict" product module) creating V010300. You would move the "order" table object from the "cust1dict" product module to the "coredict" product module (remember that "cust1dict" is not sourced in COREDEVEL). You can then assign the "order" table to the "sports" database (see the Roundtable User's Guide for directions on expanding databases and assigning tables).

The exact same information is true for the relationship of field objects to table objects.