ODE Code Extraction and Build

You'll first export the full ODE code into a backing build (that you create with mkbb), using the RTC 3.0 Eclipse client (or with similar scm.exe commands).

First create the backing build:

```
mkbb –dir c:\ode\builds mybb
```

Now create a new RTC repository workspace that flows to the ODE5.0 stream. Do NOT load the contents anywhere during this creation process.

Now you will need to follow these steps to extract the files from RTC (WARNING: this includes steps for extracting the 3rd party source code, which may require legal approval in your department…if you wish, you may ignore the "src" tree related extract steps from the ode.3rdparty component and instead extract the "obj" tree in a similar fashion AFTER you have built ODE, which will replace the built stub regex.obj and gregex.obj files with the full versions…note these are only used on Windows systems, and were compiled with MSVC++ 6.0). For all of these load commands, you should be using your Team Artifacts view and doing a "Show Repository Files" for the ode.3rdparty and ode.base components. All extracts (loads) will be done from individual folders and files, not from the full components.

1. Show the ode.base files, select the "src" folder, right-click, and select "Load As…". Expand the Advanced Options section and select the radio button "Load the selected folders but do not create Eclipse projects", and change the Sandbox path to be "c:\ode\builds\mybb", or whatever you created above. You should notice the "Local Path" field will automatically change to "c:\ode\builds\mybb\src". Click "Finish" and agree to overwrite/replace the src dir. Once the files are loaded, you need to immediately unload the component from your workspace view, but ONLY select to Disconnect, NOT delete the loaded files. This will remove the RTC metadata from the file system and allow you to load more data into the same directory area.
2. Show the ode.3rdparty files, drill down to export/classes. Extract the "classes" dir as in step 1 above (Load As…), where the Sandbox path will be "c:\ode\builds\mybb\export". Again, unload/disconnect immediately after the files are loaded.

3. Drill down to obj/x86_nt_5 (if building on Windows). Extract the "odeclw.dll" file to Sandbox "c:\ode\builds\mybb\obj\x86_nt_5". Unload/disconnect. **Repeat for the file "odecplw.dll".** These are the MSVC++ 6.0 C/C++ libraries (renamed to more easily coexist with different default versions on the same system). You may wish to build your own or change the linker commands in our build process to just use your default runtime DLLs.
4. Drill down to src/bin/jikes. Extract the "jikes" dir to Sandbox "c:\ode\builds\mybb\src\bin". Unload/disconnect.
5. Drill down to src/include/lib/portable/native (if building on Windows). Extract the "gregex.h" file to Sandbox "c:\ode\builds\mybb\src\include\lib\portable\native". Unload/disconnect.
6. Drill down to src/lib/portable/native (if building on Windows). Extract the "gregex.c" file to Sandbox "c:\ode\builds\mybb\src\lib\portable\native". Unload/disconnect.

Note: you will also notice cutime.c / cutime.h in the 3rdparty lib/portable/native src tree. These are only of historical interest, as they were used on VMS (whose native C library didn't have a utime function).

That should populate the mkbb backing build with all of the ODE source code and required jars. Note that this tutorial is for *building* ODE only, not making changes. In order to do a build, changes have to be made to the source code after extraction (with the prebld.pl script described below), but these changes should NOT be committed back to the repository. This is one reason you should always disconnect after the load (the other reason is because you can't load different repository paths to the same directory tree).

If you need to make bug fixes or enhancements to ODE, it is preferable to use Eclipse projects to do so (and export to a sandbox as needed to build and test).

After you extract the ODE source code, you must run a script to process macros that are used in some files (like src/Makeconf), which will replace the macros with actual values. These macros control the desired Release and Build numbers (which can be whatever you wish, but 5.0 is appropriate for Release and the date/time format "YYYYMMDD.hhmm" is what has been traditionally used for Build, so e.g. 20041004.0559). The Perl script to run is src/scripts/bld/prebld.pl (run it with -? to show the usage text).

So, for example, cd to src/scripts/bld and then run this:

```
perl prebld.pl /home/username/builds/mybb 5.0 20041004.0559
```

FYI, this script uses the VerRel.lst file (also in the scripts/bld dir) to determine which files to process. You may edit that file if you wish to add or remove files to process.

As mentioned above, once you have run prebld.pl, **DO NOT CHECK IN THE CHANGED FILES** to RTC (i.e., the files listed in VerRel.lst)! The files with macros in them should remain with those macros in RTC so they don't become hardcoded as a specific release/build. The process you should follow is to run prebld.pl ONLY for the backing build (which should be treated as a read-only sandbox), and only make changes to files in a sandbox (on which you do NOT run prebld.pl).

You also need to unjar the two 3rdpary class libraries that were extracted to export/classes (after which you will have subdirectories called “org” and “javax” in export/classes).

You should define an environment variable TOOLSBASE to the path where your ODE tools (that you’ll use for building) are installed, and you must include a trailing slash. So for example, on Windows, you would do this:

```
SET TOOLSBASE=C:\ODE\BIN\
```

Then you can use workon and mk (or build by itself) to build the source code as you would any other project.

Building the Java source code (the packaging class library) requires defining BUILDJAVA (and optionally USE_JAVA_RESPFILE if you expect long command lines to cause problems on your machine, which is often the case on Windows). The values of these variables are unimportant (as long as they are defined to something).

Further problems/questions can be discussed in the developers forum for ODE on Community Source.