Adding a new package or OS to the RM

Add A Package

To add a new top level package, (i.e. like ScienceTools, GlastRelease, etc) all that needs to be done is to add a new entry into the ***package*** database. This will cause the specified packageName to appear in the various tools such as the Release Manager web pages and the RMViewer program and allow the name to be used in the various RM programs such as *deletedBuild* and *triggerBuild*.

The following command,

```
INSERT INTO package VALUES ('<packageID>','<packageName>');
```

should be sufficient to create the new entry where <packageID> is the value for the new package ID number (the next one should be 6), and <packageName> is the name you want the package to have, i.e. ScienceTools, GlastRelease, etc, in the RM

However, all this does is create the name and ID for the package that the RM will use. You can't actually create any build packages until you set up a build package configuration (see below)

Add An Operating System

Adding a new operation system is similar to adding a package. You simply create a new entry for the new operating system in the ***os*** table. Like adding an entry to the package table, all this really does is create the names and ID for the operating system to be used by the various RM systems. To actually enable a build in the new OS, you need to create a build package configuration (see below).

The following command,

```
INSERT INTO os VALUES ('<osId>','<osName>','<osType>','<nickname>','true');
```

should be sufficient to create the new entry. See the ***os*** database table description for the details of the values of the various parameters.

(jrb) **Note:** I would use instead something like

Add new OS

```
INSERT INTO os (osName, osType, nickname, active) values ('<osName>', '<osType>', '<nickname>', 'true');
```

The first column, osId, is auto-increment, so it's best to let the system set it to the next available integer.

Add A Build Package Configuration

In order to actually create a build using a new OS or a new package, you have to first create a build package configuration in the ***settings*** database table. A build package configuration corresponds to a unique combination of **packageId**, **osId**, **variantId**, and **versionTypeId**. For each variation in any one of those ID values, you need a separate set of ***settings*** table entries. If you are just adding an OS to an existing package, you have to create about 19 entries per build package in the table. If you are adding a new package, there are a few additional entries, which apply across OSes and variants (but are unique the the different versionTypes) , that you also need to add.

New Package Only Settings

The following table lists the settings table entries (see the Release Manager Settings page for details of the various parameters) that need to be added to the ***settings*** database table for a new package. It lists the setting value name, which versionTypes (Release, ReleaseCandidate, and Integration) they are needed for, and a sample value for Integration (i.e. LATEST) builds. You can check the values of these settings in the database itself for more examples.

Name	Release	Release Candidate	Integration	Sample Value (Integration)
automaticTrigger	⊕	⊕	⊕	true
checkTime	⊕	⊕	⊕	3600000
cvsPackage	⊕	⊕	⊕	ScienceTools-scons
doxygenLocation	⊕	⊕	⊕	/nfs/farm/g/glast/u35/doxygen/ScienceTools/LATEST-1-${VERSION}
excludeClean	⊕	⊕	⊕	bin,data,exe,include,lib,python,site_scons,xml,sconsTools
initialVersion	⭐	★	★	(any version number)
keepNumTags	⭐	★	★	20
maxHasSource	⭐	⭐	⭐	2

maxKeep	⭐	⭐	★	20
submitterOs	⊕	⊕	⊕	1
submitterVariant	⊕	⊕	⊕	1
subPackageTagFormat	⊕	⊕	⊕	${PACKAGE}-\d{2}-\d{2}-\d{2}$
versionFormat	⊕	⊕	⊕	ScienceTools-LATEST-1-\d+
versionPrepend	⊕	⊕	⊕	ScienceTools-LATEST-1-

⊕ = required setting

★ = optional setting but usually present (see the Release Manager Settings page for setting details and use)

⭐ = optional setting but not usually present

General Build Package Settings

The following table shows all the entries that you need for each build package along with some of the optional ones commonly used. Sample values are provided from the ScienceTools Debug LATEST Mountain Lions builds (the most recent package added). Again descriptions of what all the settings control can be found on the Release Manager Settings page.

Name	Required	Sample Value	Comments
automaticTrigger	★	true	if not included it is assumed to be false
buildLocation	⊕	/data/RM/ReleaseManagerBuild/mountainlion-x86_64-64bit-gcc44/Debug/ScienceTools/LATEST-1-${VERSION}	
compileTime	⊕	10800000	
cvsCommand	⊕	cvs	
cvsRoot	⊕	:ext:centaurusa.slac.stanford.edu:/nfs/slac/g/glast/ground/cvs	
cvsRsh	⊕	ssh	
develReleaseLocation	⊕	${BUILDLOCATION}/ScienceTools-LATEST- 1-${VERSION}-devel.tar.gz	
distributionLocation	⊕	/nfs/farm/g/glast/u35/ReleaseManagerBuild/mountainlion-x86_64-64bit-gcc44/Debug/ScienceTools	
distributionPath	⭐	/data/RM/ReleaseManagerBuild/mountainlion-x86_64-64bit-gcc44/Debug/ScienceTools	only needed on Mac and Windows
externalsLocation	⊕	/data/RM/externals/mountainlion-x86_64-64bit-gcc44	
extInstallerLocation	⊕	/nfs/farm/g/glast/u35/externals/mountainlion-x86_64-64bit-gcc44/${EXTNAME}-${EXTVERSION}.tar.gz	
extInstallerPath	⭐	/data/RM/externals/mountainlion-x86_64-64bit-gcc44/${EXTNAME}-${EXTVERSION}.tar.gz	only needed on Mac
initialVersion	★	4035	Should be included on all new OS additions or the RM will try to build all old versions
NotifyOwners	⭐	true	only included for builds you want e-mails to go out on. If not included it is assumed to be false.
sconsLocation	⊕	/data/RM/tools/SCons/2.1.0/bin/scons	
sconsOptions	⊕	--rm,-k,--with-GLAST-EXT=${GLAST_EXT},--compile-debug,--variant=mountainlion-x86_64-64bit-gcc44-Debug,--with-cc=/opt/local/bin/gcc,--with-cxx=/opt/local/bin/g++,all	
sourceReleaseLocation	⊕	${BUILDLOCATION}/ScienceTools-LATEST- 1-${VERSION}-source.tar.gz	
testTime	⊕	1800000	
userReleaseLocation	⊕	${BUILDLOCATION}/ScienceTools-LATEST- 1-${VERSION}-user.tar.gz	
workflowStart	⊕	Launch	This has a value of 'Checkout' for the NFS based builds

⊕ = required setting

★ = optional setting but usually present (see the Release Manager Settings page for setting details and use)

★ = optional setting but not usually present

Note: Many of these settings are the same across all the build packages and could probably have been made generic and only had to have been set once per package+versionType combination but this is the way Navid set it up.

Example Scripts

 There are example scripts that have been used to add these settings to the database for various packages and OSes that you can look at to see samples for other OSs or you can look up the values in the database. The sample scripts can be found in the *sql-scripts* directory under the home directory of the **gl astrm** account on the central Unix machines at SLAC.