pucm plugin

Deprecated: This plugin has been removed from the Jenkins Update Centre

Please use the ClearCase UCM Plugin instead.
Archived versions of this plugin remain available for download.

Plugin Information
No information for the plugin 'PUCM' is available. It may have been removed from distribution.

A Praqmatic integration to ClearCase UCM - This plugin is renamed and replaced by the 'ClearCase UCM Plugin' - have a look at that instead!

What can Praqmatic UCM do?

- Monitor changes in your ClearCase UCM VOBs and polls for new baselines on a given stream, in a given component with a given promotion level.
- Establish a snapshot view which your build- and post-build steps can use.
- Promote or reject the baseline given the build result, success, unstable or failure.
- Set the description of the individual job, listing the baseline built, the result and the new promotion level.
- Recommend the baseline if it is promoted.
- Tag the baseline with the result of each individual job, so that history and statistics are persisted, not only in Jenkins, but also in ClearCase itself.
- Supports concurrent builds - without finding the same baseline twice even when previously started jobs are still in progress (currently only supported with one iterator per slave).
- ClearCase MultiSite is fully supported, Slaves and masters can be in different ClearCase regions and even at completely different Sites.
- Advanced features can deliver baselines from one stream to another and baseline the deliver including "official" four-level version numbers.
- ...and even a handful more of nice features.

Prerequisites:

- PUCM is designed to follow what we consider UCM best practice. Be inspired here:UCM Best Practice
- PUCM assumes that a default view storage location is available in your region (it probably is, and if it's not you can easily create one using 'cleartool mkstgloc').
- If you want to use the 'tag' function in the post-build section, you have to set up a hyperlink type named 'tag' in your VOB (details below).
- If you want to use the build number features in the advanced section you will have to create the buildnumber.* attributes (details below).
- ClearCase Client must be installed on master and all slaves.

Limitations

- The plugin is only compliant with ClearCase Windows Clients.
- PUCM supports concurrent builds. But currently you can only have one build at the time *per node*. Set # of executors to 1 in your node configuration if you want to use concurrent builds with PUCM.

Run Jenkins service under a valid ClearCase account

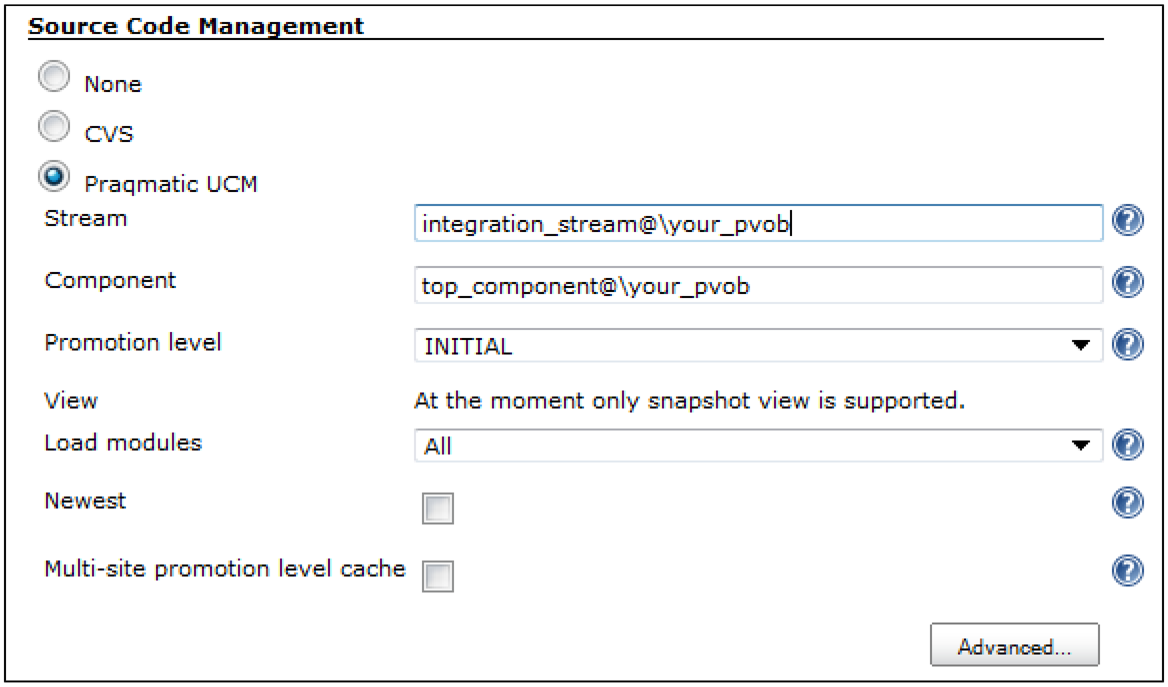
Jenkins needs to be authenticated by ClearCase, so it's important that you run the Jenkins service under an account that has the sufficient access to ClearCase. PUCM fully supports that a slave can be in a different ClearCase region or even at a completely different ClearCase MultiSite than the master. If you utilize this feature, it's required that the slave is running Jenkins under an account which has the sufficient credential on the *remote* site

MultiSite - tag your nodes

If you have slaves in different MultiSites than the master, you can tie the jobs that monitors stream that has mastership on foreign sites to slaves that belongs to those sites. A simple strategy for this is that you add a tag to all you slaves telling which MultiSite they belong to and then you tie the jobs to those labeled slaves.

The Source Code Management section setup

When you install the PUCM plugin it will become available as an SCM option in the Source Code Management section. When you build (or poll) PUCM will look for baselines with the given *Promotion level* on the *Stream* and within the *Component* you have given.


Source Code Management

None

CVS

[✓ Praqmatic UCM]

Stream: integration_stream@\your_pvob

Component: top_component@\your_pvob

Promotion level: INITIAL

View: At the moment only snapshot view is supported.

Load modules: All

Newest: ☐

Multi-site promotion level cache: ☐

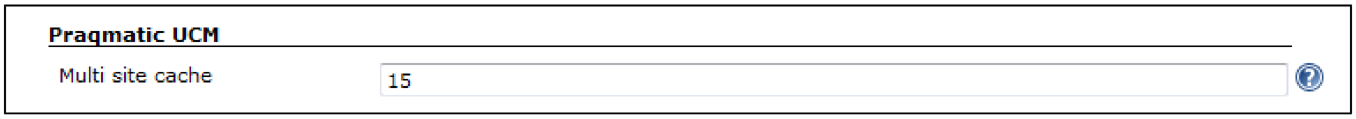
- **Stream** - The stream that PUCM shall monitor, you need to specify the fully qualified name, as shown in the example above.
- **Component** - The component in which to look for baselines. Currently PUCM supports only one component. To accommodate that, you should setup a rootless component and get into the habit of making *all* baselines composites on this component (Be inspired here:UCM Best Practice).
- **Promotion level** - The promotion level that qualifies the baseline for build. If your product is small and your build and validation is fast, you may only need one job that runs on INITIAL. But sometimes you will have so much validation to do on each baseline, that some level of *devide and concur* is needed. In that case you can set up one job that looks for INITIAL to do the initial validation (e.g. compile and link). Then another job can be setup that looks for promotion level BUILT which perhaps runs the unit tests and do the coverage metrics. And then again, after that, you could have a third job running which looks for the promotion level TESTED which could deliver the baseline to stable stream on which a new baseline is created, containing a neat four-level version number.
- **Load modules** - to save you the trouble of messing with load rules in the GUI of the PUCM plugin we have provided you with the option simply to load all modules contained in the UCM project which the found baselines belongs to or only the modifyable ones.
- **Newest** - If more than one baseline matches the query, then PUCM will default build the *next* baseline, but if you would rather just jump directly to the *Newest* and skip the intermediate ones you simply check Newest.
- **MultiSite promotion level cache** - In the internal Jenkins architecture it's by design the master that is responsible for the polling of the SCM and therefore a potential problem could occur when you have slaves on a different MultiSite than the master; When the master finds a baseline and tells the slave to build it, the slave will promote or reject the baseline, but the change of the promotion level is done on the remote site and it is not instantly replicated to the master. If the poll frequency is shorter than the MultiSite synchronization frequency, then the same baseline is found again by the master - only this time it shouldn't find it since the promotion level really doesn't match, the master just thinks so due to a synchronization delay between sites. The MultiSite cache is implemented to solves this issue. What happens is that the slave is instantly reporting the new baseline state to the master ahead of the MultiSite synchronization, and the master is then using that cached information rather than the real-time information local to the master. The validity of the cached information is set in the global Jenkins settings and is default set to 15 minutes. If an entry in the cache falls for this time limit it's purged. Wether or not the master should use real time information of the MultiSite cache is set on each individual job with this checkbox . You shouldn't enable it at all if you don't use MultiSite.

Advanced SCM setup

Hidden underneath the "advanced" button is an additional setting:

- **Build Project** - The plugin creates a separate read/only development stream and snapshot view for every job on every slave that executes the job. The stream and view is created the first time the job is executed on the slave and then reused by successive job executions. If you just go with the defaults settings the streams will be create as child streams to the integration stream in the project holding the baseline that is being built, but despite the reuse of these Jenkins related streams, there is still a tendency that you end up with a lot of extra streams on behalf of PUCM. You can remove tem if you want to, but the jobs will then just create them again next time it runs. In order to keep your ClearCase Project Explorer tidy we've made the option to place them in an auxiliary project made specifically for this purpose (in general, development streams *must* be in the same UCM project as the baselines you rebase them against , but read/only streams are considered special cases and they can live safely be in a separate project). Here's the deal: If you make a project in ClearCase named 'hudson', PUCM will put the auxiliary streams there. If you prefer another name for this special PUCM-related project, you can specify it in the Build project box in the 'Advanced' section

PUCM general settings


Praqmatic UCM

Multi site cache: 15

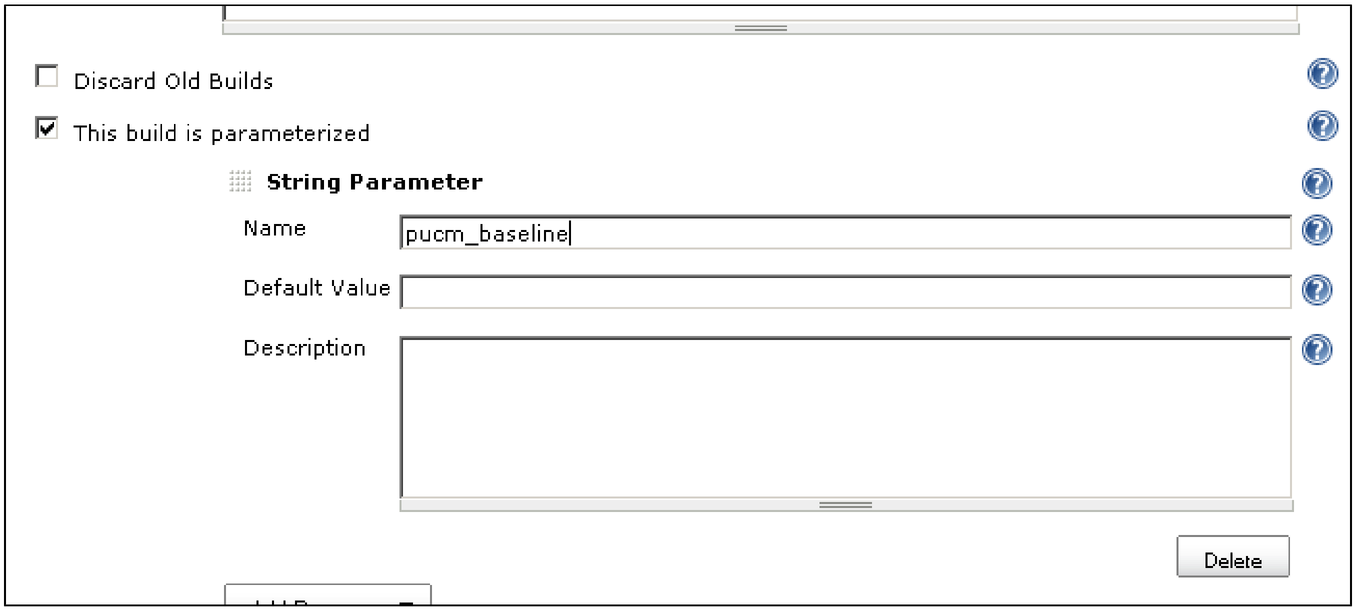
- **MultiSite cache** - The number of minutes that the entries in the MultiSite cache is valid before they are automatically purged. Your should adjust this to reflect the synchronization scheme between sites and then add a few extra minutes to apply the packages in your incoming bays.

"Nothing to do"

PUCM differs from most other Jenkins SCM plugins in the way it behaves when you push the "Build now" button. Normally an SCM plugin will build *somethi ng* in that situation. Even if the latest commit has already been build, most other SCM plugins will typically just build the latest commit again. But PUCM is different, because it will always match the criteria you have specified (stream, component and promotion level), and there might not always be a baseline that satisfies your requirements - in that case the build will not be built the status "not built".

Build a specific baseline with a parameterized build

If you want a job that builds a specific baseline from ClearCase (e.g. a *rebuild*), then you can force PUCM to override the Stream, Component and Promotion level settings all together and build a specific named baseline. The way you do this, is simply to make the job *parameterized* (check 'This build is parameterized' in the project options) and set up a String Parameter called **pucm_baseline**:


☐ Discard Old Builds

☑ This build is parameterized

String Parameter

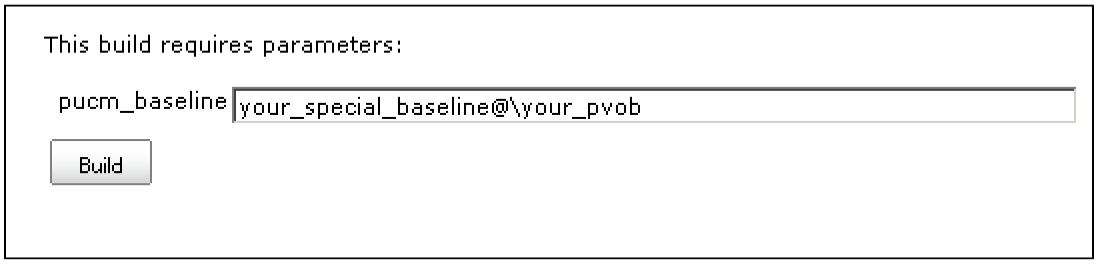
Name: pucm_baseline

Default Value:

Description:

You still need to choose the Praqmatic UCM plugin in the Source Control Management section, but when you have an environment variable with the magic name 'pucm_baseline' you don't need to set any of the component, stream or promotion level parameters.

When you build the job, you just need to fill in a fully qualified baseline:


This build requires parameters:

pucm_baseline: your_special_baseline@\your_pvob

Build Section

During the job execution the baseline being build is stored in the environment variable "cc_baseline".

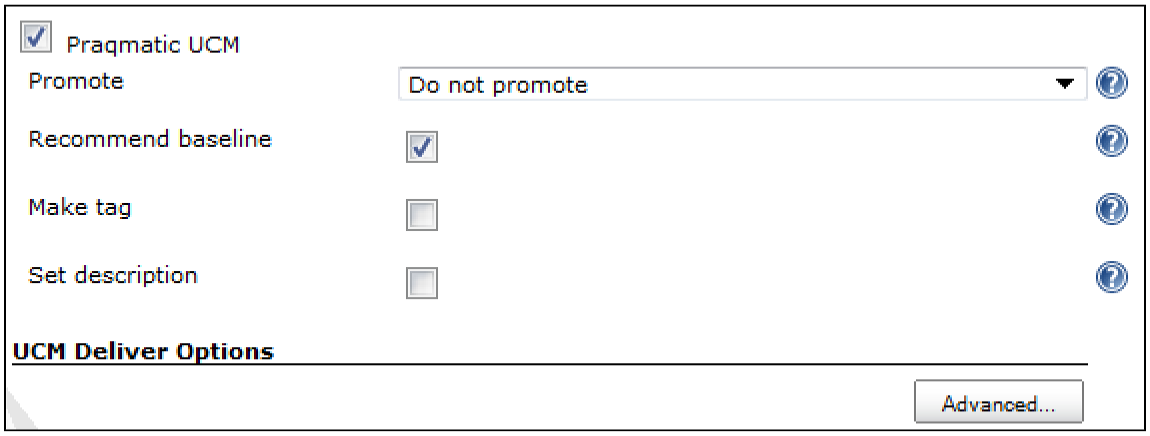
The view root of the job is located in a directory named *view* in the jenkins workspace and can directly be referred to as:

```
#perl style:
${workspace}/view

#Windows shell style:
%WORKSPACE%/view
```

Post Build Actions

The plugin also offers your some post build actions. These post-build actions will only be invoked if you've selected PUCM as your Source Code management in the pre-build section.


[✓] Praqmatic UCM

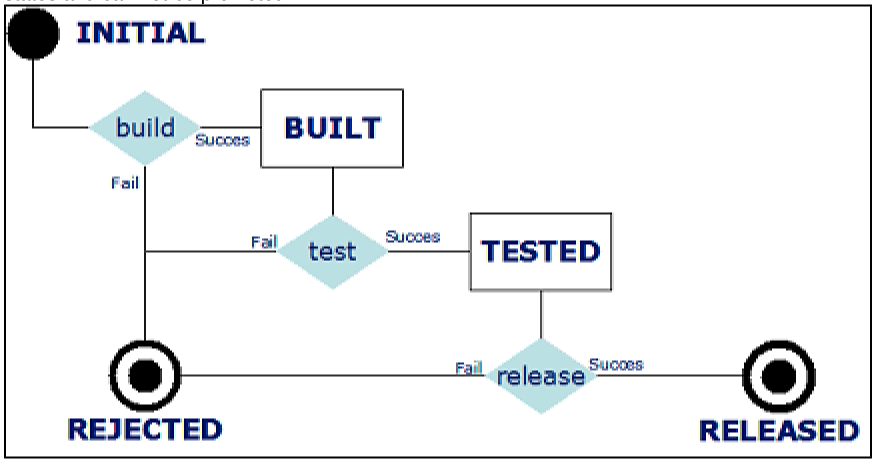
Promote: Do not promote

Recommend baseline: [✓]

Make tag: [ ]

Set description: [ ]

UCM Deliver Options

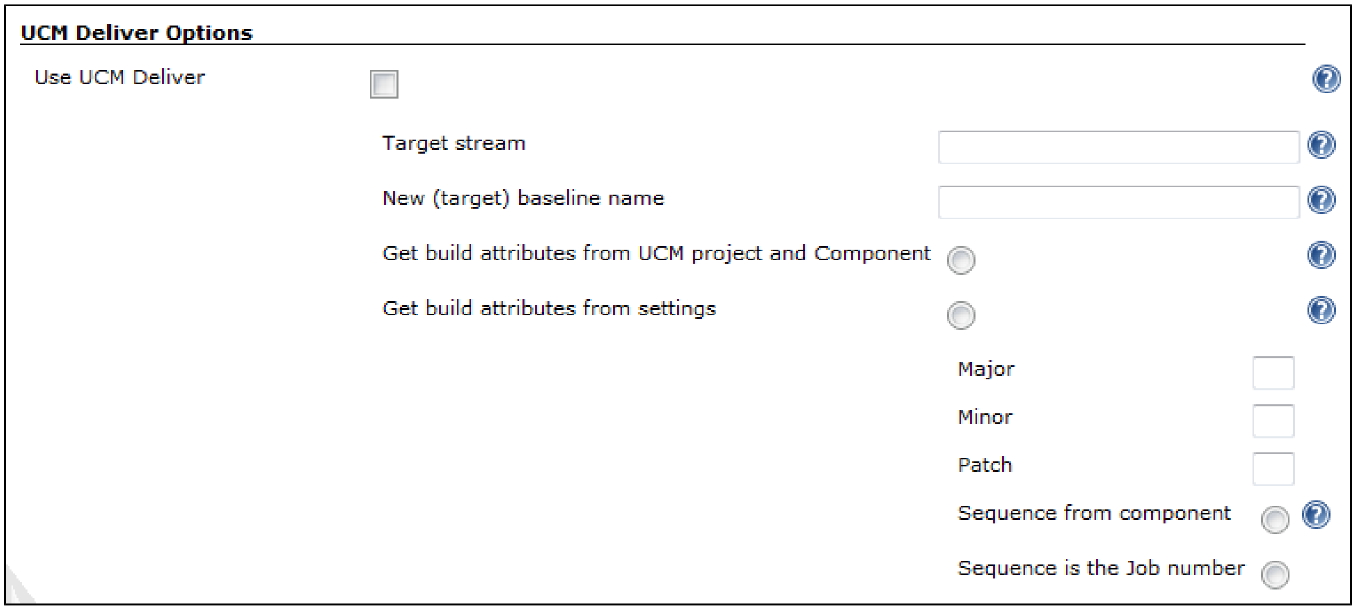
- **Promote** - When the Promote option is in effect the baseline is guaranteed to have a different promotion level once the job execution is done. You should always use this setting when you have configured your job to poll the SCM, if you don't the poller will find the same baseline over and over again (both boring and unnecessary). PUCM uses the default promotion levels defined by UCM: INITIAL, BUILT, TESTED, RELEASED and REJECTED. If you have redefined your promotion levels to something else then PUCM will not be compliant.
  If a build fails it becomes REJECTED. If it succeeds then it's promoted to the next level, both RELEASED and REJECTED are considered end-states and can not be promoted.
  

  The promote drop-down has three possible options:
  - *Do not promote* - Does not promote or reject the baseline, the promotion level is the same after the execution as it was before (don't use this setting if you have configured your job to poll the SCM).
  - *Promote stable builds* - Stable (blue) builds are promoted. Unstable (yellow) and failed (red) builds are rejected.
  - *Promote unstable builds* - Both stable (blue) builds as unstable (yellow) builds are promoted, only failed (red) builds are rejected.
- **Recommend baseline** - when this option is checked, the baseline willl be recommended on the stream if the job was promoted but not if it was rejected.
- **Make tag** - option will make a tag on your baseline in ClearCase with information about the build status (it will make a hlink of hltype:tag using the -totext part of the hyperlink to describe the status). This way the job status is persisted, not only on the Jenkins server, but also in ClearCase itself. Remember, that if you want to use this option, you need a hyperlink type called tag as described in the prerequisites section.
- **Set description** - Will summarize the job result in the build description with the name of the baseline that was build, the new promotion level, and, if selected, whether it was recommended or not.

A job will be set to unstable if the any of the options fails to be executed.

Advanced Post Build Actions

PUCM is capable of deliver the baseline to another stream and create a baseline on that stream. The delivery will only occur on promoted job executions. It's possible to setup a job which has an empty build and only delivers baselines.


UCM Deliver Options

Use UCM Deliver ☐

Target stream

New (target) baseline name

Get build attributes from UCM project and Component ○

Get build attributes from settings ○

Major

Minor

Patch

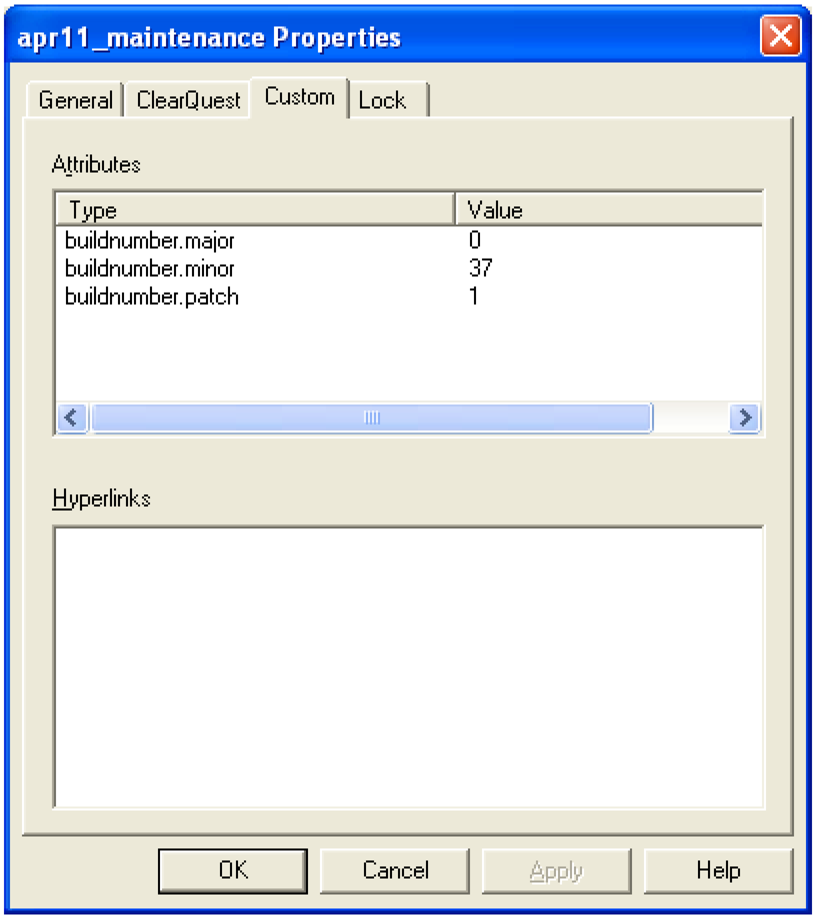
Sequence from component ○

Sequence is the Job number ○

UCM deliver - Check this option if you want to use any of the advanced deliver options. You don't necessarily have to specify more details, the deliver can run on default values (described in detail below).

- **Target stream** - defines the stream that will receive the deliver. It must be specified using fully qualified notation (e.g thestream@\pvob). If you don't specify a stream, the default target of the stream holding the baseline is used. This field can not be left empty if this stream does not have a default target setup in ClearCase.
- **New (target) baseline name** - defines the name of the baseline that will be created on the target stream. You should avoid white space characters and you should only specify that base name (don't use the fully qualified notation in this setting). If this field is left empty no baseline is created.
- **Get build attributes from UCM project and Component** - If *New (target) baseline name* above is not empty and this option is checked, then the base name will automatically be suffixed with a four-level build number (major_minor_patch_sequence). The three first numbers will be taken from attributes attached to the UCM project which the target stream belongs to. The attributes that are looked for are:

```
buildnumber.major
buildnumber.minor
buildnumber.patch
```

- 

Type	Value
buildnumber.major	0
buildnumber.minor	37
buildnumber.patch	1

The last level, the sequence, is fetched and automatically incremented from another attribute named:

```
buildnumber.sequence
```

which is attached to the component, in which the baseline is found. This attribute should be instantiated with either an integer (or nul) and hereafter it is incremented automatically.
If you specify this option and anyone of these four attributes isn't found as described then the job will become unstable. When using this setting, it's required that the account under which the Jenkins service is running has write access to the buildnumber.sequence attribute attached to the component.

- **Get attributes from settings** - As described above, the basename will be suffixed with a four-level version number, but in this case the values aren't taken from from the four attributes described above, but from the settings below
  - **Major** - the major build number (1st level)
  - **Minor** - the minor build number (2nd level)
  - **Patch** - the patch build number (3rd level)
  - **Sequence from component** - as described above, the 4th level will be taken and automatically incremented from the buildnumber. sequence attribute attached to the component.
  - **Sequence is the job number** - if you check this option the job number of the current Jenkins job is used as the 4th level.

The job will be set to unstable if the delivery is unsuccessful.

Installing Version Attributes

If you choose to use any of the options that use some of our predefined types, then you must create these types as global types in the PVOB(s).
We have created the attributes like this:

```
 cleartool mkhltype -global -c "Used to support the PUCM plugin, automatically attached to
baselines"                                    tag@\Cool_PVOB
 cleartool mkattype -global -c "Used to support the PUCM plugin, attach to UCM projects"        -vtype integer -
default 0 buildnumber.major@\Cool_PVOB
 cleartool mkattype -global -c "Used to support the PUCM plugin, attach to UCM projects"        -vtype integer -
default 0 buildnumber.minor@\Cool_PVOB
 cleartool mkattype -global -c "Used to support the PUCM plugin, attach to UCM projects"        -vtype integer -
default 0 buildnumber.patch@\Cool_PVOB
 cleartool mkattype -global -c "Used to support the PUCM plugin, attach to Top-level component" -vtype integer -
default 0 buildnumber.sequence@\Cool_PVOB
```

If you put them in PVOBs (recommended) you must use the -global switch as above. If you are in a MultiSite environment consider using the -shared switch as well. In general when you create types (any) in a PVOB you should also consider using the -acquire switch too.

To assign the buildnumber.* attributes correct you can either use the ClearCase Project Explorer or use the cleartool CLI, we went:

```
cleartool mkattr buildnumber.major 0   project:PUCM@\Cool_PVOB
cleartool mkattr buildnumber.minor 3   project:PUCM@\Cool_PVOB
cleartool mkattr buildnumber.patch 20  project:PUCM@\Cool_PVOB
cleartool mkattr buildnumber.sequence  component:_System@\Cool_PVOB
```

You can set the major, minor, patch and sequence to your your preferences. Notice, that if the attributes has default values, you don't have to specify a value.

Change log

Version 0.3.22(2011-5-5)

- Fixed a backwards compatibility issue, regarding global configuration

Version 0.3.21(2011-4-13)

- Fixed deliver issues
- Trimmed configuration UI
- Nothing to do results in "not build" not "failed"

Issues

type	key	summary
⚠ Can't show details. Ask your admin to whitelist this Jira URL. View these issues in Jira		

Issues regarding PUCM are tracked at jenkins-ci.org please report *any* issue or feature request you may have there.

Feed-back in general is also welcome, please send us a mail at coolers@praqma.net.