1 Update Site Management

Besides the product .zips, TextGrid distributes TextGridLab components via *update sites* (aka *p2 repositories*). The update sites contain all tools – those that are packed with the product .zips and those that are only installable into an existing TextGridLab.

1.1 Users' View

Usually, end users will not deal with update sites individually but instead use the Marketplace and the built-in update facility. However, using the update sites directly is also possible, e.g., to install nightly versions of tools that are not referenced in the marketplace. *Help Install new software* in the Lab leads to the install software dialogue, one of the update site URLs below can be copied to the *Work with* field in order to have the dialog list the software available from the respective site. End user documentation for that is in the Installing and Removing Software chapter of the User's Manual.

There are three update sites for our tools:

1. http://www.textgridlab.org/updates/stable offers tools we consider stable – currently, these are those tools that are part of the packaged TextGridLab 2.0 release.
2. http://www.textgridlab.org/updates/beta offers tools in beta status.
3. http://www.textgridlab.org/updates/nightly points to the update sites generated by the Integration Build, i.e. here are the bleeding-edge, untested and probably broken versions of our tools.
   Additionally, while preparing new TextGridLab releases we use a separate update site for those builds that will lead up to the release, as opposed to the nightly site for the ongoing update:
4. http://dev.digital-humanities.de/updates/prerelease is the site for the release preparation builds.

1.2 Administrators' / Releng View

There are two kinds of update sites / p2 repositories: *Simple* and *Composite* Repositories.

Simple repositories directly contain artifacts and their metadata, i.e. there are *plugins* and *features* subdirectories with the actually downloadable files and (content|artifact).(xml|jar) files with the metadata. Those repositories are produced by the respective *eclipse-repository* modules of the TextGridLab Modular Build.

Composite repositories are simply lists of links to child repositories. A composite repository contains a compositeContents.xml and a compositeArtifacts.xml file (or their jar pendants) that lists the URIs of all child repositories. The children can themselves be either composite or child repositories.

Composite repositories can be *mirrored* into aggregate repositories – i.e. a tool creates a simple repository that contains everthing that is also available in the composite repository. Advantage is that when a tool (e.g. the TextGridLab's update check) reviews a composite repository, it needs to additionally load all the referenced children as well after parsing the composite repo, each with an individual request, while the contents of an aggregate repository can be retrieved in one go.

1.2.1 … in TextGrid

The three repositories mentioned above are composite repositories that live on textgridlab.org in subdirectories of /var/www/updates. The *stable* and *beta* repositories' contents also lives on that server, in respective subdirectories.

1.2.1.1 Policies

stable versions are released by the TextGrid community after some testing.

beta version releases are managed by the corresponding tool manager who will create a SVN tag of the reasonably tested version s/he wishes to release. The tool manager will then point someone with write permission for textgridlab.org (e.g., Thorsten Vitt) who will checkout and build the tag:

```
svn co https://develop.sub.uni-goettingen.de/repos/textgrid/tags/sometool_beta_1.0.1
cd sometool_beta_1.0.1
mvn3 package
```

The package zip is then copied to textgridlab.org and deployed as outlined below.

We'll sign the stuff when we have a signing key

1.2.1.2 Deploying a new (beta / stable) tool

1. Create a new directory with the tool name below /var/www/updates/beta or stable
2. Unzip the repository zip there
3. Edit *both* the compositeContent.xml and the compositeArtifacts.xml. If there are only jar files, need to unpack them first using `jar xf compositeContent.jar` . Example files:

compositeArtifacts.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<?compositeArtifactRepository version='1.0.0'?>
<repository name='TextGridLab Beta'
    type='org.eclipse.equinox.internal.p2.artifact.repository.CompositeArtifactRepository' version='1.0.0'>
  <children size='8'>
    <child location="glosses"/>
    <child location="linguistics"/>
    <child location="ttle"/>
    <child location="collatex"/>
    <child location="noteeditor"/>
    <child location="sadepublish"/>
    <child location="digilib"/>
    <child location="base-extras"/>
  </children>
</repository>
```

compositeContent.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<?compositeMetadataRepository version='1.0.0'?>
<repository name='TextGridLab Beta'
    type='org.eclipse.equinox.internal.p2.metadata.repository.CompositeMetadataRepository' version='1.0.0'>
  <children size='8'>
    <child location="glosses"/>
    <child location="linguistics"/>
    <child location="ttle"/>
    <child location="collatex"/>
    <child location="noteeditor"/>
    <child location="sadepublish"/>
    <child location="digilib"/>
    <child location="base-extras"/>
  </children>
</repository>
```

Note that you need to also edit the `size` attribute of the `children` element!

1.2.1.3 Deploying a new version of a beta / stable tool

We currently host all released versions in our repository. We currently don't modify the repositories generated by the Tycho build but rather aggregate versions using composite repositories.

Deploying the second version of a beta tool

1. cd to the tool's repository.
2. Create a directory for the first released version, e.g., `1.0`.
3. Move the files from the first release there.
4. In the tool directory, createa new directory for the second released version, e.g., `1.1`
5. Unzip the repository of the second version there.
6. Create `compositeContent.xml` and `compositeArtifact.xml` in the tool directory that point to 1.0 and 1.1, respectively.
7. (Optionally) run the optimization script, see below
8. Test.

An example is the TTLE site.

Further versions

… are obviously easier, just unzip them to a new version folder and adapt the composite*.xml files accordingly.

1.2.1.4 Optimizing the repositories

There's a shell script, `optimize-p2-repositories`, that optimizes the repository structure by jarring up XML files and generating p2.index files for composite repositories – this reduces the number of requests from clients.

There's an experimental maven plugin that can generate a human-readable index.html for a repository.

1.2.2 Converting composite to simple repositories

It is possible to create a simple p2 repository that contains everything that is also contained in a different repository that may be composite. This can be useful to reduce the number of checks each client must do – usually, a client follows every link to a composite child repository, with this technique we can simply create a simple repository out of a nested composite structure, and the client must only load the contents for this repository.

There is a plugin in Tycho Extras that does that. Here is a pom.xml that can be used to mirror the stable repo: Download it and run `mvn install`, the repository will be created in `target/repository`.

1.3 Updating end users' update site configuration

It is possible to deploy changes to the update site configuration to end users via updated artifacts by using Touchpoint Instructions in a `p2.inf` file. In TextGridLab, we use the p2.inf file in the base feature for that. The changes are applied when an update to this feature is installed, and their implications are also the default for new installations of the lab. To read this,

- `addRepository` adds an update site to the configuration. You must issue this once for each type (`type:0` = metadata, `type:1` = artifact repository), and you can specify via the `enabled` boolean parameter whether this repo should be enabled in the user's configuration or if the user must manually enable it (from the preferences / install / available software sites dialog).
- `removeRepository` can be used to remove obsolete repositories from the configuration.
- `addRepository` with `enabled:false` can be used to turn off a repository, but leave it configured so the user can easily re-enable it using a checkbox in the respective dialog.

1.3.1 Suggested configuration

TextGridLab Version	Update Site			
	Stable	**Beta**	**Prerelease**	**Nightly**
Nightly	disabled	enabled	disabled	enabled
Release Candidate	enabled	enabled	enabled	disabled
Release (= last build of the Relese Candidate!)	enabled	enabled	disabled	disabled