jUDDI Committer Notes

1. Updating the jUDDI Web Site

You'll need the appropriate version of forrest as well as the appropriate version of ant. This document assumes you generally understand how to use these tools.

The site source is located under our source tree at ws-juddi/site. To build the site, simply invoke forrest in ws-juddi/site. The built sources will be located at ws-juddi/build/site. You should get a clean build, and no broken links. There aren't any known issues with the site.

You must test your builds locally by running forrest on your local box. You can do this by invoking 'forrest run' in ws-juddi/site.

The commit process is a bit complicated. The live site is hosted under the ws-site source tree, -not- the ws-juddi src tree. The ws-juddi/build/site directory hierarchy maps to the ws-site/targets/juddi directory hierarchy. Once you've built the site copy the latter hierarchy to the former, and then commit the changes. The live site is updated about every 6 hours from cvs.

You must -also- commit the changes you've made under ws-juddi/site to persist the changes for the next committer to be able to reproduce your modifications.

To fully clean any site build you've created, you can safely delete everything under ws-juddi/site/build.

2. Release Process

Release Manager

One committer will be elected or hopefully volunteer to assemble the binary releases and label the source tree.

Digitally Signing Releases

Apache policy requires binary releases be digitaly signed. The Apache process has not been formalized, but a general discussion about creating digital signatures and signing releases is available at http://nagoya.apache.org/wiki/apachewiki.cgi?SigningReleases. This covers some basics about using GnuPG to create key pairs and sign releases. Our goal here is to

discuss jUDDI signing requirements, and provide some useful examples to release managers, not discuss digital signatures or encryption technology. Our discussion uses GnuPG, but any compliant software could be used. The examples below come from the GnuPG manual. This discussion is not a subsitute for reading that manual.

Creating a key pair is pretty simple using gpg. Simply invoke gpg and take all the defaults when prompted. You will have to provide a passphrase. Be sure to remember the passphrase since you'll need it anytime you use the key pair. The passphrase should itself be sufficiently secure; it shouldn't simply be a word in a dictionary, should include a mix of digits and alphanumeric characters, etc.

```
gpg --gen-key
```

You should also generate a revocation certificate. This allows you to declare the key pair invalid publically, if you ever lose your private key, or it becomes compromised.

```
gpg --output revoke.as --gen-revoke mykey
```

The release manager is responsible for signing the binaries. The release manager must have a public key listed in the 'KEYS' file at the root of our source tree. The release manager must create a detached signature for each binary. This detached signature must be posted along with our binaries, and allow our users to verify the binary's integrity.

```
gpg --output jUDDI.tar.gzip.asc --detach-sig jUDDI.tar.gzip
```

All jUDDI committers are encouraged to create public/ private key pairs and place the public half into our 'KEYS' file at the root of our source tree. jUDDI committers are also encouraged to verify one another's keys and sign them, to help create a web of trust. Verifying a signature and a binary guarantees (in any real sense) the binary was assembled by the person that signed it. However, it does not prove the person signing it can be trusted. A web of trust can be created by signing one another's keys. This allows users and developers to 'trust' the person who created the document-signature pair to provide a secure, safe binary.