IdP Build Documentation

IdP

The IdP is built as a master directory including a master set of configuration files and a master WAR file. To modify the configuration, fiddle with /opt /shibboleth-idp/conf and copy changes when ready into /srv/salt/dev/opt/shibboleth-idp/conf. To replace the WAR file, rerun the build script at /opt /shibboleth-identityprovider-ver.si.on/install.sh. Don't overwrite the config. Copy the new WAR file to /srv/salt/dev/opt/shibboleth-idp/war.

We simplified this installation by creating a repository in Salt to handle all of the IdP installation. Salt then manages the IdP directories on all of the Jetty Servers, meaning that any change to the local copy on Salt will cause the minion to have the same change when the state is propigated.

Another interesting note is that IdP is configuring its portion of the Jetty xml file. This means that for each state enforcement, Jetty re-sets its configuration files to default, and then IdP re-writes its changes each time. There are many ways to solve this, however right now we are working on differentiating an installation and an update or enforcement state change through appropriate "unless" messages.

Attribute Release

The IdP has a vague and general access release policy that will send a handful of predetermined attributes to every SP that is part of the CommIT project. However, no attribute release may take place without the explicit informed consent of the user, enforced by uApprove.

If an SP indicates a desire to receive *fewer* attributes(for regulatory or other reasons), the IdP can directly accommodate that with an overriding DENY rule for that entityID with those attributes enumerated.

Logging

Logs for both Tomcat and the IdP are piped to a central rsyslog server to reduce the storage needs of each node and to improve potential incident response.

I've modified the IdP's log rotation behavior to use a fixed window appender and gzip logs by default:

http://logback.qos.ch/manual/appenders.html

This should also cap the IdP's logs at less than 675MB(gzipped) per node, which I think is pretty generous. I'll crop it down further if necessary. I'll look into something similar for Tomcat once that transition is done. I don't know of any system services that involve perpetual logs, so that might be the extent of it on the IdP VM's.

```
<appender name="IDP_ACCESS" class="ch.qos.logback.core.rolling.RollingFileAppender">
    <File>/opt/shibboleth-idp/logs/idp-access.log</File>

    <rollingPolicy class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
        <FileNamePattern>/opt/shibboleth-idp/logs/idp-access-%d{yyyy-MM-dd}-%i.log.gz</FileNamePattern>
        <minIndex>1</minIndex>
        <maxIndex>9</maxIndex>
        <timeBasedFileNamingAndTriggeringPolicy
            class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP">
            <!-- or whenever the file size reaches 25MB -->
            <maxFileSize>25MB</maxFileSize>
        </timeBasedFileNamingAndTriggeringPolicy>
    </rollingPolicy>

    <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
        <charset>UTF-8</charset>
        <Pattern>%msg%n</Pattern>
    </encoder>
</appender>
```