Configuration

Note that a lot of advanced use cases will require you to make use of the Java API documentation, which can be found here for later use:

- https://shibboleth.net/cgi-bin/java-support.cgi
- https://shibboleth.net/cgi-bin/spring-extensions.cgi
- https://shibboleth.net/cgi-bin/java-opensaml.cgi
- https://shibboleth.net/cgi-bin/java-idp.cgi

The above contains all of the API (and in many cases implementation class) documentation for all of the code provided by the project, but does not include numerous third party APIs (e.g. Spring itself).

When you see a syntax like this:

> *name* [3.2]

That signifies a feature, setting, property, etc. that is only available in the corresponding minor version or later of the software. You will encounter features that are marked with unreleased versions of the software, which is necessary in order for us to avoid having to document all the new features at the end of a release cycle.

Configuration Overviews and References

Topics exist for each general configuration area to go into detail on how to do various things and to provide a definitive reference on configuration settings, beans, properties, etc. Before digging into details, you should take a look at the layout summary below to get a general idea of where things live and what not to change.

Each of the detailed pages makes note of the files involved in that topic and notes the equivalent "legacy" V2 files where applicable to give upgraders a point of reference.

If you're coming into this cold, you really need to review these topics first, just to get the lay of the land, and because the core "language" for many of the configuration files is Spring, and because debugging your changes will usually require some logging familiarity.

- Spring Configuration Background
- Configuration File Summary
- Logging

If you don't know Spring XML configuration syntax, that's not unusual, but doing a little bit of reading on it will be essential, if not right away then almost immediately after you try and actually configure something.

To configure a new IdP from scratch, you will need to address these areas first:

- Metadata
- Authentication
- Attribute Resolver
- Attribute Filter

If you need to dig into more advanced SAML configuration needs, or need to interoperate with commercial cloud services, you will usually need to tackle these areas:

- SAML NameID Generation
- Profiles and Per-RelyingParty Behavior
    - Unsolicited (IdP-initiated) SAML

Native CAS support is discussed in:

- CAS Protocol

Advanced configuration topics:

- Activation Conditions
- Security
- Subject Canonicalization
- Profile Intercepts
    - Consent
- SAML NameID Consumption

- Logout Configuration [3.2]

Finally, configuration related to "productionalizing" an IdP:

- Services and Reloadable Configuration
  - HttpClient Configuration
- Error Handling
- Storage Configuration
- Access Control
- Instrumentation [3.3]
- Administrative Configuration [3.3]

Installation Layout

The following summary will guide you in understanding the installed software layout and how to locate important files.

The most important thing to note is the separation of configuration files into user- and system-level files. The contents of the **system** directory and its subdirectories are meant to be left unmodified, and they are created as read-only files to emphasize this. This is to ensure that backward-compatible upgrades can be accomplished safely without reapplying local changes, and so that internal configuration changes required by newer versions can be applied automatically.

There are a number of interdependencies between the Spring configuration files in various locations and in **system** that are like a contract between the user-modifiable configuration and the system configuration. In most cases, these dependencies can be identified via the use of Spring bean names that contain the prefix "shibboleth." When in doubt, don't remove a bean name that contains such a prefix, or comment it out (unless it starts out commented).

Directory	Explanation
bin	Contains command line tools, and any Java libraries needed during installation. During upgrades any additional files you add will be preserved, so you can store your own command line scripts here.
conf	The main configuration tree. During any installation (first time or upgrades), files are *never* replaced in this directory. New files required by the IdP version being installed will be populated if and only if they do not exist.
credentials	Contains your keys, certificates, and keystores, as well as credentials on which you rely such as for metadata signature validation. Files in this directory should generally be readable only by the user account the IdP will run under (certificates aren't secret, but it's easiest to just lock down everything). During any installation (first time or upgrades), files are *never* replaced in this directory. In unusual cases, new files may be created if they do not exist.
dist	Contains the original/default versions of the contents of the **conf**, **flows**, **messages**, and **view** directories. This folder is *always* deleted and re-created from the distribution on every install. This directory can be used as a reference against any locally modified copies of these files.
doc	Contains documentation, licenses, and the like. This folder is *always* deleted and re-created from the distribution on every install.
edit-webapp	This directory is created on initial install and thereafter not touched. You may place any local configuration you wish to include in your packed warfile. During warfile creation, the contents of this tree are copied over top of the **webapp** directory, from which the war file is then built. Thus, it is an overlay tool for your local modifications and extensions.
flows	Contains any user-editable Spring Web Flow definitions. During any installation (first time or upgrades), files are *never* replaced in this directory. New files required by the IdP version being installed will be populated if and only if they do not exist.
logs	Contains the IdP diagnostic and audit logs by default.
messages	Contains internationalized message properties used in various UI templates. As of V3.3, this contains only new or overriden message properties or post-install translations, with all of the default messages and translations moved to the system tree. During any installation (first time or upgrades), files are *never* replaced in this directory. New files required by the IdP version being installed will be populated if and only if they do not exist.
metadata	A storage location for SAML metadata used by the IdP (see MetadataConfiguration). During initial installation, some representative SAML metadata for the IdP is generated based on the installation inputs and placed in this directory in a file named *idp-metadata.xml*. Note that the IdP does **not** need to load its own metadata, a change from V2. Also note that the metadata is generated as a one-time operation during installation. It does not result from an in-depth analysis of the IdP configuration and does not change when the configuration changes.

system	Contains read-only internal system configuration that should not be modified. The contents of this directory tree will be deleted and re-created by an install or upgrade.
views	Contains Velocity page templates displayed to users of the IdP. While JSP views (and the V2 taglibs) are generally supported, most of the default webflow views provided are now Velocity templates that can be maintained outside the warfile and changed at runtime. During any installation (first time or upgrades), files are *never* replaced in this directory. New files required by the IdP version being installed will be populated if and only if they do not exist.
war	Contains the packed IdP warfile for container deployment. The warfile can be rebuilt at any time by running the *build.sh* or *build.bat* script in the **bin** directory. It wll prompt you to verify the installation directory (which in theory allows for multiple installations).

Backup Directories

Except on Windows, the installation process always preserves old files in a directory called '**old-[date][timestamp]**'. This can be helpful for reverting upgrades (but note that the contents of **conf**, **flows**, **messages**, and **views** are never replaced, though new files may be added).

On all platforms, a pre-upgraded V2 configuration is preserved in folders with the suffix ".v2" appended to the name (**conf.v2**, **war.v2**).

Windows-Only

On Windows, if Jetty has been installed there will be extra directories created.

Directory	Explanation
jetty-base	Contains the Jetty configuration. The only file which may be edited and which is guaranteed to survive upgrades is *start.d\idp.ini* If you need to edit anything else in this directory, you should deploy your own container.
static 3.1	Contains and data you want to serve statically from the Jetty installation (`https://yourHost/`). Examples might be favicon.ico
C:\Program Files\Shibboleth\Jetty or **C:\Program Files (x86)\Shibboleth\Jetty**	Contains the Jetty installation. **None** of its contents should be edited; it is always deleted and recreated during an upgrade. The file *JETTY_VERSION.TXT* contains the precise Jetty version. The log files for the jetty instance are locate in this directory. The can be of interest to debug IdP initialization issues.
C:\Program Files\Shibboleth\ProcRun or **C:\Program Files (x86)\Shibboleth\ProcRun**	Contains the executables that allow the IdP to run as a user mode system service. None of its contents should be edited; it is always deleted and recreated during an upgrade. The executable *shibd_idpw.exe* can control the configuration of the user mode system service, but any configuration is not guaranteed to survive an upgrade. The log file for the procrun service are located in this directory. This records any output from the process until such time as the jetty logging occurs. It is unusual that this has any data of interest.

Configuration Changes (and Non-Changes) from V2

You'll see a lot of new files in **conf** as well as a few familiar ones (summary here). There are three older files that are designed to work fully, or almost-fully, from V2:

- attribute-resolver.xml
- attribute-filter.xml
- relying-party.xml

These are, not surprisingly, the files containing most of the frequently modified configuration in older versions. The attribute configuration files and the metadata portions of relying-party.xml are still considered the supported configuration formats for these features, while the rest of relying-party.xml is either ignored, or is deprecated.

There are caveats to this compatibility, which are discussed in the subtopics discussing these particular files. Apart from the noted issues there, any failures to load or operate as expected with any older V2 configuration files should be considered a high-priority bug and reported. We can't test all the possible options out there, but any regressions will be treated as important issues to correct.

The other files are essentially new configuration, or in a few cases are refactored subsets of the original relying-party.xml configuration, which is discussed in that subtopic.

The most noteworthy change is that authentication is configured very differently in V3. There is no handler.xml file any longer, but there are substantial overlaps in the common cases of the UsernamePassword or RemoteUser login handlers, and there's a similar feature to the External login handler. In particular, the hardest aspects of configuring those handlers should translate more or less directly to this version.