UpgradingFromV2

General documentation on upgrading from Version 2 to 3 follows.

- **Before You Begin**
  - The Old System
  - The New Environment
- Windows Installer Upgrade
- Non-Windows / Standard Upgrade
- Next Steps
  - Authentication
  - Name Identifier Handling
  - Initial Testing
- Post-Upgrade Tasks for metadata-providers.xml
- Post-Upgrade Tasks for relying-party.xml
- Additional Post-Upgrade Tasks

**For InCommon participants**
If you are an InCommon participant, please read this first! Upgrading to Shibboleth IdP V3

**For SWAMID members**
If you manage a Identity Provider within SWAMID, please read this first! Uppgradera Shibboleth IdP från version 2 till version 3

**Before You Begin**

It is not safe in general to upgrade a production installation directly. You must have a development environment in which to test the upgraded software or you will experience extensive downtime and probably end up reverting the upgrade several times while trying to get back to a working system. While extensive compatibility is provided for attribute, metadata, and relying-party configuration, authentication typically must be manually configured and you will need to test to ensure your particular configuration isn't using advanced features that may strain the compatibility provided.

**The Old System**

There are some steps to take to “clean up” your V2 configuration to prevent inadvertent changes in behavior after you upgrade, principally related to *Name Identifier* handling. You need to ensure that any integrations your old system is supporting through the use of Name Identifiers are configured correctly or you will be likely to observe changes in this area. Specifically, you need to ensure that the selection of the appropriate Name Identifier format is being done appropriately. There are only three appropriate ways to do this with V2, as documented:

- the SP selects the format itself at runtime (quite rare)
- the SP’s metadata contains one or more `<NameIDFormat>` elements indicating what it needs
- the IdP configures a precedence of formats to use with an SP based on a relying party override using the `nameIDFormatPrecedence` attribute

Those are the only three mechanisms that are supported, and they are supported in identical fashion in V3, ensuring compatibility in format selection.

The problem is that many deployers did not understand how to do this correctly, and there are a lot of legacy systems around that are selecting formats in an inappropriate "ad hoc" fashion, usually by suppressing the release of IdP attributes associated with non-desired formats using an attribute filter policy (i.e., in layman’s terms, a policy saying to deny the release of one or more attributes for a particular SP). It is very rare for a deny rule to be needed in either V2 or V3, so such a filter rule tends to be a red flag that the system may have been setup incorrectly with respect to Name Identifier format selection.

After upgrading such a system, SPs that were receiving a Name Identifier in particular formats may end up receiving a transient format. This is a clear sign that the original V2 configuration was not correct. Cleaning this up ahead of time will prevent this problem from occurring.

This problem is particularly common in cases in which the "urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified" Format was used. The only proper way to configure use of that Format in either V2 and V3 involves the use of the `nameIDFormatPrecedence` attribute in the relying-party.xml file, so if you don't see this in your configuration, an upgraded system will not produce that type of identifier when you expect it to.

**The New Environment**

Refer to the SystemRequirements page for details on supported software platforms. There are differences, so your existing system may well not support the new version.


The SecurityAndNetworking page, much of which may be new for you, includes material describing some newer features and an explanation of the full range of keys used by a "standard" V3 installation. As noted below, the upgrade process does not generate new keypairs in order to limit the changes introduced, but it's useful to understand the difference between an upgraded system and a new installation.
As when upgrading any software, you should take a good backup of your system, especially the old IdP's installation and configuration directories. The upgrade process will not destroy any information intentionally, but a backup makes restoring the old version easier.

The upgrade process will suggest or generate the following information for you:

- the IdP's `entityID` (which you absolutely must override to match your existing configuration; do not change a system's `entityID` for any reason short of your organization selling its old domain name)
- a secret key and key version file for securing cookies and other data produced by the IdP for its own use (this is a special Java keystore of type "JCEKS")
- a default set of IdP configuration files based on this information, modified to include your original configuration where possible

Windows Installer Upgrade

The Windows Installer will recognize and upgrade the V2 "Quick Install". It should automatically locate the V2 configuration (defaulted to `C:\Program Files\Internet2\Shib2Idp`) and perform an upgrade. It will also configure the included Jetty container with the SAML key and certificate already in place. You will need to re-specify the Active Directory location and credentials when performing the upgrade.

Non-Windows / Standard Upgrade

Performing a V2 to V3 upgrade is similar to a standard V3 installation. When specifying an existing installation directory, the installer will recognize an existing V2 configuration and prompt for less information, and additional information is output.

When an upgrade is detected:

- The V2 configuration files are copied to a new directory, `conf.v2`
- The packed V2 warfile is copied to a new directory, `war.v2`
- The directories `logs` and `metadata` are left untouched (note that leaving the latter in place means that new example metadata for the IdP is not generated).
- Specific legacy files that are generally compatible with V3 (`attribute-resolver.xml`, `attribute-filter.xml`, `relying-party.xml`) are pre-populated into the new config directory; these make up the bulk of your older configuration, save for authentication.
- The old `relying-party.xml` file is also copied to `metadata-providers.xml`
- The rest of the configuration is populated as for a new installation.
- Finally, `services.properties` is altered to enable a legacy relying party configuration.

Next Steps

Authentication

After the upgrade you will need to configure authentication and make any adjustments required for advanced use (see Configuration), but authentication and UI considerations are typically the main thing you'll need to deal with to get things going and get back to a working system. While there are strong parallels provided to what came with V2 (JAAS-based form login and RemoteUser-based authentication), there are key differences that have be accounted for in many cases:

- The vt-ladp JAAS module provided with V2 has been replaced with a newer implementation based on the successor project called ldaptive. While the newer module is generally more capable (and has the advantage that it’s fully supported), there can be differences in configuration.
- The login form customization features in V2 were based on JSP, while Velocity templates are the strongly recommended approach in V3. Even if JSP is used, there will be inevitable changes required. In most cases, it’s best to leave the default form in place during initial testing and save the UI migration for a later step.

Name Identifier Handling

If you're experiencing changes to the Name Identifier format particular services are receiving, refer to the warning in the first section above, under "The Old System". The change is caused by a mistake in your older configuration that has been carried forward and is now being exposed. You may want to go back and make corrections to the original system and then redo the upgrade (or at least manually apply the same corrections to the new system).

Initial Testing

After starting the new version for the first time, watch for any errors in the log, and make a note of any warnings; most of those will be due to the use of a legacy relying-party configuration, but some will also note use of deprecated features or syntax. You can ignore deprecation issues until after you have a stable system.

If you encounter errors, you may be using features that are no longer supported and are encouraged to ask for help on the support list, but this should be rare.

You can test that the IdP is at least running successfully in the container with the status command line utility (`idp.home/bin/status.sh` or `idp.home\bin\status.bat`).

Once you have a basically functional IdP, you'll want to focus on testing attribute resolution and filtering policy to verify that the output is consistent with V2. The upgraded command line utility, named `aacli` as before, is much faster because it operates as a localhost-only web service rather than loading an embedded copy of the IdP on every invocation, and it now also provides an indication of what kind of `<NameID>` content will appear for particular SPs, often crucial for commercial services using dumbed-down SAML software that you need to remain compatible with.
Post-Upgrade Tasks for metadata-providers.xml

No changes will normally be required to reach a working metadata-providers.xml configuration. However, there will be a number of warnings in idp-process.log resulting from ignored configuration elements and you should identify those to remove the warnings before proceeding.

- Inside the outer level <RelyingPartyGroup>, retain the <AnonymousRelyingParty> and <DefaultRelyingParty> elements, as these are required by the legacy document schema, though they are not relevant to your metadata configuration.
- You can remove any <ProfileConfiguration> elements within <DefaultRelyingParty>, as they will also be ignored anyway.
- Remove the <security:Credential> with id="IdPCredential" if present; this is not related to metadata verification.
- In V2, the relying-party.xml file (which is copied during the upgrade into the V3 metadata-providers.xml) typically included a <MetadataProvider id="IdPM"> element which loaded the IdP's own metadata. In V3, this is no longer required, so we recommend that you remove this <MetadataProvider> element from the configuration.
- Remove the text between the <!-- DO NOT EDIT BELOW THIS POINT --> comment and the closing </RelyingPartyGroup> tag. This will include a number of occurrences of <TrustEngine> and <SecurityPolicy> which are ignored as part of the upgraded configuration.
- You should retain any custom <TrustEngine> configurations used by your metadata provider configurations.

After you have a simplified, working, configuration, we recommend that you migrate it entirely to the new configuration style at your leisure. An example of the new configuration style for metadata-providers.xml can be found in dist/conf/metadata-providers.xml.dist.

Post-Upgrade Tasks for relying-party.xml

Check your configuration for any uses of the defaultAuthenticationMethod attribute on <RelyingParty>, <AnonymousRelyingParty> or <DefaultRelyingParty> elements. In V2 configurations, the value urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport was sometimes used to indicate user/password authentication in general. In a V3 configuration, this will mean that only that authentication method will be available by default, and this may cause SAML 1 username/password authentication to become unavailable. The typical error seen in the idp-process.log file in this case will be:

Profile Action SelectAuthenticationFlow: None of the potential authentication flows can satisfy the request

In a V3 configuration, the available authentication flows are configured elsewhere, and if that has been done then the simplest corrective action will often be to remove all occurrences of the defaultAuthenticationMethod attribute from your configuration.

No other changes will normally be required to reach a working relying-party.xml configuration. However, there will be a number of warnings in idp-process.log resulting from ignored configuration elements and you should identify those to remove the warnings before proceeding. You should normally be able to remove everything from relying-party.xml other than a <RelyingPartyGroup> element containing:

- an <AnonymousRelyingParty> element,
- a <DefaultRelyingParty> element,
- any <security:Credential> elements referred to as signing credentials by the other elements.

In particular, you can normally remove:

- all <MetadataProvider> elements (metadata providers are now configured in metadata-providers.xml),
- any <security:TrustEngine> elements referred to by the metadata providers,
- everything from the <!-- DO NOT EDIT BELOW THIS POINT --> comment to the final closing </RelyingPartyGroup> tag (you can safely ignore the stern warning in the comment there, none of that security configuration is used any longer)

After you have a simplified, working, configuration, we recommend that you migrate it entirely to the new configuration style at your leisure. An example of the new configuration style for relying-party.xml can be found in dist/conf/relying-party.xml.dist . In many cases, the following steps will be sufficient to migrate:

- Copy dist/conf/relying-party.xml.dist to conf/relying-party.xml.
- Change services.properties to replace the definition of the idp.service.relyingparty.resources property with the (commented out) default. Replace this text:

```
# Set to shibboleth.LegacyRelyingPartyResolverResources with legacy V2 relying-party.xml
idp.service.relyingparty.resources = shibboleth.LegacyRelyingPartyResolverResources
```

- The revised definition should be:

```
# Set to shibboleth.LegacyRelyingPartyResolverResources with legacy V2 relying-party.xml
idp.service.relyingparty.resources = shibboleth.LegacyRelyingPartyResolverResources
```

- Create new-style relying party overrides based on your existing overrides to reproduce your current behavior. The RelyingPartyConfiguration topic describes how the new syntax and machinery works. Initially, copying over new overrides one for one will be simplest; later on you may find you can consolidate them using new features to select the relevant SPs (TBD, need to provide some help for this).
- Copy dist/conf/credentials.xml.dist to conf/credentials.xml if the latter file does not already exist.
• Make sure that your signing key and certificate files in the credentials directory have names matching the idp.signing.key and idp.signing.cert properties in idp.properties. If they do not match (V2 IdPs frequently used the filenames idp.key and idp.crt while the default names in V3 are idp-signing.key and idp-signing.crt), then you can either change the settings in idp.properties or rename the files in the credentials directory, your choice.

• As an upgraded V2 IdP will not have credentials suitable for decrypting content sent to the IdP, for now just edit conf/credentials.xml to comment out the encryption credential, as follows:

```xml
<!--
   <bean class="net.shibboleth.idp.profile.spring.factory.BasicX509CredentialFactoryBean"
     p:privateKeyResource="${idp.encryption.key}"
     p:certificateResource="${idp.encryption.cert}"
     p:entityId-ref="entityID" />
-->
```

• If you wish to enable decryption functionality in the IdP, you must create a keypair, uncomment this bean again and update the metadata you give to relying parties to include a new <KeyDescriptor use="encryption"> element. Note that while it's possible to reuse your signing keypair for this purpose, this is not advisable and since you're starting fresh anyway, it's simplest to just do the right thing.

Moving to the new configuration style will by default enable the new user consent functionality of the IdP, and you may wish to configure that separately.

User consent functionality can not be enabled while still using a legacy relying-party.xml; you must migrate to the new configuration style for this functionality to be available. Many other new features also require this conversion.

Additional Post-Upgrade Tasks

The default configuration of the V3 IdP, unlike V2, relies on client-side (cookie) storage of state information encrypted by a "DataSealer" component as described in SecretKeyManagement. An initial data sealer keystore and secret key is generated as part of the V2 to V3 upgrade process, but you will need to arrange for the seckeygen utility to be executed periodically in order to limit the time during which each secret key is in use, for example by a daily cron job running a script similar to the example given in SecretKeyManagement.

If you rely on the status page in V2, you may have modified the old web.xml file with a list of addresses to allow. Those should be copied over to the bean named shibboleth.IPRangeAccessControl in conf/access-control.xml to achieve the same result (and you won't need to rebuild or restart the IdP to change that list).