Troubleshooting

Startup Issues

Slow startup time

There are two known reasons for experiencing unusually slow startup: using Tomcat in certain ways and a blocking PRNG. Taken in order:

If you're using Tomcat, this issue is discussed on the prep page along with some workarounds, primarily forcing it to expand the warfile, counter to the usual advice we give with Jetty.

With all containers, it's not uncommon for startup to be delayed sometimes several minutes because Java defaults to using **/dev/random** as a source of randomness, which is a blocking device. The most common workaround for this is to switch to **/dev/urandom** instead, which can be done by passing a system property to Java during startup (e.g. on Jetty via start.ini):

```
-Djava.security.egd=file:/dev/urandom
```

Traditionally this has been viewed as less safe. There are alternate views of this. The Shibboleth Project does not take an official position on this question.

Common Log Messages

Invalid XXXXService configuration

The XXXX above might be "NameIdentifierGeneration" or "RelyingPartyResolver" or "AttributeResolver", or any of the other service names. When you see this message while debugging something that's not working, you are missing the real problem and focusing on the aftermath.

The log is **not** just the last message you see; you have to look at all of it, particularly during startup or reloading a new configuration for a particular part of the system. The real error is that you've broken a part of the configuration in a particularly significant way, usually with a syntax error of some sort, and you need to look at the error that describes that problem to make any progress.

Sometimes that will be a fairly complex Spring message because of a problem creating Java objects, but occasionally it will be fairly obvious what the error is if the problem mentions an XML syntax error or a property name that turns out to be misspelled. Regardless, that's the real problem and you should ignore whatever comes after it until you resolve the original issue and get the configuration of the affected service at least loading again.

If you're really confused, try changing the relevant "failfast" property in the *services.properties* file to force the IdP to fail at startup. At least then you'll know where the real issue is.

Error: "unable to find resource 'status.vm'"

The first time the status page is accessed after starting the IdP, the following log message will be seen:

logs/idp-warn.log

```
ERROR [org.apache.velocity:96] - ResourceManager : unable to find resource 'status.vm' in any resource loader.
```

Although this is presented by the Velocity library as an `ERROR` level message, this should be ignored unless you have customised the status page yourself. In fact, the default status page is implemented as a JSP and not as a Velocity template, and the message merely indicates that the fallback from Velocity to JSP is occurring for the status view the first time it is invoked. This message is filtered in the logging configuration in newer IdP versions.

Error: "unable to find resource 'login.vm'"

If the default login UI is changed from a Velocity template to a JSP, the following error will be seen at least once:

logs/idp-warn.log

```
ERROR [org.apache.velocity:96] - ResourceManager : unable to find resource 'login.vm' in any resource loader.
```

Although this is presented by the Velocity library as an `ERROR` level message, this can be ignored. The message merely indicates that the fallback from Velocity to JSP is occurring. You may wish to filter it with your logging configuration.

Miscellaneous issues

IdP stops sending authentication statements after 2 weeks; restarting the IdP temporarily resolves the issue

IdPs that consume metadata using the `FileBackedHTTPMetadataProvider` regularly download metadata from an external metadata source. The IdP has no control over the size of that metadata file, and it may have sufficiently increased in size from one download to the next that the IdP cannot process the later file. Since the IdP stores the previous file, it continues to use that until the metadata it contains expires (perhaps a couple of weeks after the initial out-of-memory error) at which point the IdP stops recognising SPs. Re-starting the IdP temporarily resolves the issue because it allows the IdP to download new metadata until the next time it runs out of memory.

Symptoms include the idp-warn.log showing entries like this:

logs/idp-warn.log

```
2017-01-16 14:02:16,700 - DEBUG [org.opensaml.saml.metadata.resolver.impl.AbstractReloadingMetadataResolver:
380] - Metadata Resolver FileBackedHTTPMetadataResolver UKfederationMetadata: Unmarshalling metadata from
'http://metadata.ukfederation.org.uk/ukfederation-metadata.xml'
2017-01-16 14:02:22,323 - ERROR [net.shibboleth.utilities.java.support.service.AbstractReloadableService:188] -
Service 'shibboleth.MetadataResolverService': Unexpected error during initial load {} org.springframework.beans.
factory.BeanCreationException: Error creating bean with name 'ShibbolethMetadata': Cannot create inner bean
```

Please see the Installation pages for your particular deployment to determine how to set an appropriate Java heap size, and what that size should be.

Unhelpful logging information from exception traces.

The default logging provided in logback.xml is for a truncated information from exceptions. Particularly when debugging a development system it may be helpful to turn up the amount of information in the log. This page has mpore details but replacing this line in `logback.`

```
<Pattern>%date{ISO8601} - %level [%logger:%line] - %msg%n%ex{short}</Pattern>
```

```
<Pattern>%date{ISO8601} - %level [%logger:%line] - %msg%n%ex{full}</Pattern>
```

Can often provide a useful hint of where the fault has come from. For instance:

```
Caused by:
java.lang.IllegalArgumentException: name
at sun.misc.URLClassPath$Loader.findResource(Unknown Source)
at sun.misc.URLClassPath.findResource(Unknown Source)
at java.net.URLClassLoader$2.run(Unknown Source)
at java.net.URLClassLoader$2.run(Unknown Source)
at java.security.AccessController.doPrivileged(Native Method)
at java.net.URLClassLoader.findResource(Unknown Source)
at org.eclipse.jetty.webapp.WebAppClassLoader.getResource(WebAppClassLoader.java:402)
at java.net.URLClassLoader.getResourceAsStream(Unknown Source)
at org.springframework.core.io.ClassPathResource.getInputStream(ClassPathResource.java:166)
at net.shibboleth.idp.profile.spring.factory.BasicX509CredentialFactoryBean.getCertificates
(BasicX509CredentialFactoryBean.java:177)
```

Is more helpful when debugging than

```
nested exception is org.springframework.beans.factory.BeanCreationException: Error creating bean with name
'(inner bean)#6a98f353': Cannot create inner bean '(inner bean)#3b35798' of type [net.shibboleth.idp.profile.
spring.factory.BasicX509CredentialFactoryBean] while setting bean property 'trustCredential';
nested exception is org.springframework.beans.factory.BeanCreationException: Error creating bean with name
'(inner bean)#3b35798': Invocation of init method failed;
nested exception is java.lang.IllegalArgumentException: name
```