Java Modularity

This document describes automatic module naming from the following releases onwards:

- version 3.4.0 of the Shibboleth Identity Provider
- version 0.10.0 of the Shibboleth Metadata Aggregator

Java 9 introduced the Java Platform Module System (JPMS), allowing — amongst other advantages — better encapsulation of the internals of software components.

Current Shibboleth products are modularised in the sense of being broken into components presented as JARs, but were not designed for use with the JPMS:

- Although an API vs. implementation separation is maintained in the OpenSAML library and the Identity Provider product, this separation depends on convention rather than language-supplied mechanisms. Shibboleth JARs do not explicitly declare which of their packages are available to other modules.
- Shibboleth JARs do not declare their dependencies on other Java modules. Dependencies are instead expressed through the Maven `pom.xml` as being on other Maven artifacts.
- Shibboleth JARs are only tested on the class path, not the module path. As access rules are different when the module path is used, this may cause problems if a Shibboleth JAR is used as an "automatic module".

Compatibility with, and taking advantage of, the Java module system is a long-term rather than a short-term goal of the Shibboleth Project:

- Please report any modularity issues with Shibboleth using JIRA.
- Shibboleth JARs are provided with appropriate automatic module names to reduce future name stability issues.

Automatic Module Names

One of the transition mechanisms implemented by Java to assist with a migration towards fully modularised applications is to allow a conventional non-modular JAR to be placed on the application's module path. In this case it is treated as an *automatic module,* and given a computed list of exports and dependencies. It is also given a *module name* automatically derived from the JAR file name. This can result in a loss of global uniqueness in module names and lack of stability in module names both when JAR names change and when a JAR is ultimately converted to a "real" module.

To avoid these naming problems, it is regarded as best practice to provide non-modular JARs with a *reserved module name* through an entry in the JAR's manifest. This then replaces the default automatic module name and provides name stability for modular clients.

In general, Shibboleth modules are given reserved module names based on their root package name. The following sections describe the specific allocation for each current Shibboleth JAR.

OpenSAML

OpenSAML "impl" modules are named as sub-modules of the associated API modules, which in turn will be named after their root package.

Artifact	Module	Notes
opensaml-core	org.opensaml.core	R
opensaml-messaging-api	org.opensaml.messaging	R
opensaml-messaging-impl	org.opensaml.messaging.impl	S
opensaml-parent	n/a	1
opensaml-profile-api	org.opensaml.profile	R
opensaml-profile-impl	org.opensaml.profile.impl	S
opensaml-saml-api	org.opensaml.saml	R
opensaml-saml-impl	org.opensaml.saml.impl	S
opensaml-security-api	org.opensaml.security	R
opensaml-security-impl	org.opensaml.security.impl	S
opensaml-soap-api	org.opensaml.soap	R
opensaml-soap-impl	org.opensaml.soap.impl	S

opensaml-storage-api	org.opensaml.storage	R
opensaml-storage-impl	org.opensaml.storage.impl	S
opensaml-xacml-api	org.opensaml.xacml	R
opensaml-xacml-impl	org.opensaml.xacml.impl	S
opensaml-xacml-saml-api	org.opensaml.xacml.profile.saml	2, R
opensaml-xacml-saml-impl	org.opensaml.xacml.profile.saml.impl	2, S
opensaml-xmlsec-api	org.opensaml.xmlsec	R
opensaml-xmlsec-impl	org.opensaml.xlmsec.impl	S

Notes:

1. No artifact, so no module name required.
2. The API module root package is (unusually for an `-api` module) `org.opensaml.xacml.profile.saml` rather than `org.opensaml.xacml.saml`. The proposal here goes with the root package convention for the API module and the submodule convention for the `-impl` module.
3. R = Named according to the root package convention.
4. S = Named as a submodule.

Identity Provider

Identity Provider "impl" modules are named as sub-modules of the associated API modules, which in turn are named after their root package.

Artifact	Module	Notes
idp-admin-api	net.shibboleth.idp.admin	R
idp-admin-impl	net.shibboleth.idp.admin.impl	R, S
idp-attribute-api	net.shibboleth.idp.attribute	R
idp-attribute-filter-api	net.shibboleth.idp.attribute.filter	R
idp-attribute-filter-impl	net.shibboleth.idp.attribute.filter.impl	S
idp-attribute-filter-spring	net.shibboleth.idp.attribute.filter.spring	R
idp-attribute-impl	net.shibboleth.idp.attribute.impl	R, S
idp-attribute-resolver-api	net.shibboleth.attribute.resolver	R
idp-attribute-resolver-impl	net.shibboleth.attribute.resolver.impl	3, S
idp-attribute-resolver-spring	net.shibboleth.attribute.resolver.spring	R
idp-authn-api	net.shibboleth.idp.authn	R
idp-authn-impl	net.shibboleth.idp.authn.impl	S
idp-cas-api	net.shibboleth.idp.cas	R
idp-cas-impl	net.shibboleth.idp.cas.impl	S
idp-conf	net.shibboleth.idp.conf	7
idp-consent-api	net.shibboleth.idp.consent	4
idp-consent-impl	net.shibboleth.idp.consent.impl	4, S
idp-core	net.shibboleth.idp	R, 5
idp-distribution	n/a	1
idp-installer	net.shibboleth.idp.installer	R
idp-parent	n/a	1
idp-profile-api	net.shibboleth.idp.profile	R, 6
idp-profile-impl	net.shibboleth.idp.profile.impl	S, 6

idp-profile-spring	net.shibboleth.idp.profile.spring	R
idp-saml-api	net.shibboleth.idp.saml	R
idp-saml-impl	net.shibboleth.idp.saml.impl	S
idp-schema	net.shibboleth.idp.schema	2
idp-session-api	net.shibboleth.idp.session	R
idp-session-impl	net.shibboleth.idp.session.impl	S
idp-ui	net.shibboleth.idp.ui	R
idp-war	n/a	1

Notes:

1. No artifact, so no module name required.
2. This module will be problematic in its current form should we move to real use of the JPMS. `idp-schema` has resources under a `schema` path, which is to say as far as Java is concerned in a package called `schema`. While this is valid, it's not ideal long term for the usual reasons around uniqueness: only one module may *contain* a given package, even if the package is not exported.
3. `idp-attribute-resolver-impl` does not have a root package, but two. As well as `net.shibboleth` packages, it also includes a legacy package `edu.internet2.middleware.shibboleth.common.attribute.provider` (and two classes: `BasicAttribute` and `V2SAMLProfileRequestContext`) for backwards compatibility - see IDP-1457 - Remove Legacy (V2) scripting support for attribute resolver. OPEN . This module will be named ignoring the legacy package.
4. This module's actual root (and only) package is `net.shibboleth.idp.consent.context`. However, the corresponding `-impl` module's packages do not lie under this but under `net.shibboleth.idp.consent`. We therefore use the latter as the implicit root package for the module to provide consistency with other modules.
5. Note that this omits the `-core` part of the artifact name. Is this an issue?
6. `idp-profile-api` also includes `net.shibboleth.idp.relyingparty` so does not have a true root package. We should consider renaming this oddball package or moving it to a new module. Another one to have resolved if possible for V3.4.
7. The JAR is not used on the class path, so doesn't need to be assigned a module name. However, an implementation detail means that we want to assign one anyway, see JPAR-113.
8. R = Named according to the root package convention.
9. S = Named as a submodule.

Metadata Agggregator

`aggregator-parent`, `aggregator-bom` do not generate JARs, so don't need names.

`aggregator-pipeline` by root package: `net.shibboleth.metadata`

`aggregator-cli` by root package: `net.shibboleth.metadata.cli`

`aggregator-blacklists` is currently problematic due to a split package with `aggregator-pipeline`. Proposed resolution: `net.shibboleth.metadata.blacklists`

The split package issue (MDA-205) for `aggregator-blacklists` (`net.shibboleth.metadata.validate.x509`) will be resolved by moving the blacklist resources *up* to `net.shibboleth.metadata.blacklists` for v0.10 and using that as the module name as well.

`java-support` Library

By root package: `net.shibboleth.utilities.java.support`

`spring-extensions` Library

By root package: `net.shibboleth.ext.spring`