SystemRequirements

Supported Platforms and Versions

The following Java distributions are *fully supported*:

- Amazon Corretto 8 for Linux
- Amazon Corretto 8 for Windows
- Amazon Corretto 11 for Linux
- Amazon Corretto 11 for Windows
- Oracle Java 8 for Linux (from the Oracle Technology Network)
- Oracle Java 8 for Windows (from the Oracle Technology Network)
- Red Hat's OpenJDK 8 for Linux as supplied under Red Hat Enterprise Linux 7
- Red Hat's OpenJDK 11 for Linux as supplied under Red Hat Enterprise Linux 7

> In the case of Java > 8, do review LDAPonJava>8 as this bug is serious and usually will need to be accounted for by switching LDAP providers.

Note that Oracle's Java is no longer free for production use.

The following distributions are *partially supported:*

- Debian's OpenJDK 8 as supplied under Debian 9 "stretch"
- Debian's OpenJDK 11 as supplied under Debian 10 "buster"

Other Java distributions that are substantially identical or meant to be fully compatible with OpenJDK 8 or 11 are of course likely to work, but are officially regarded as unsupported to limit the range of environments we need to be able to reproduce problems under to a manageable set.

Other platform/version requirements for V3:

- The Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files are required, but are installed by default on all the supported Java distributions.
- A servlet container implementing Servlet API 3.0 is required. For example:
  - Tomcat 7 or later (but we strongly recommend 8, we have at least one open bug indicating there may be problems using 7)
  - Jetty 8 or later
- Only Tomcat 8+ and Jetty 9.2+ are officially supported by the project at this time. While older versions of Tomcat and Jetty are nominally suitable (see above), neither has been tested and have been obsoleted in any case.
- We also do not officially support any "packaged" containers provided by OS vendors. We do not test on these containers so we cannot assess what changes may have been made by the packaging process.
- The **recommended container implementation is Jetty** and all development and most testing time by the core project team is confined to the Jetty platform. At present, Jetty 9.2 is recommended for Java 7 use, Jetty 9.3 is recommended for Java 8, and Jetty 9.4 is recommended for Java 11.
- There are no specific requirements regarding Operating Systems, but in practice this is inherently limited by the Java distributions supported, as noted above.

Unusable Platforms and Versions

The following common configurations, and versions often in use with prior IdP versions, are specifically NOT usable with V3:

- Java version 7 or earlier, by virtue of them having been superseded by the Java 8 and Java 11 long term support releases
- Java without the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files
- Tomcat 6 or earlier. Note that RHEL 5's system-supplied Tomcat is Tomcat 5 and RHEL 6's system-supplied Tomcat is Tomcat 6. Deploying IdP V3 on these systems therefore requires the installation of an alternative application container or the use of RHEL 7, which supplies Tomcat 7 (but note the recommendation at the top to stick with Tomcat 8).
- Jetty 7 or earlier

User Agent Assumptions

There are no specific requirements regarding Browsers, but we test on only relatively recent, mainstream software, and certain features like HTML Local Storage assume standards-compliant software.

The HTML-based user interfaces make relatively minimal use of Javascript with the exception of the Duo and logout features, which include a dependency on JQuery (a version of which is included with the software).

The IdP requires the use of first-party cookies and is not designed to function without them.

The IdP does **not** require third-party cookies to be enabled and does **not** support the embedding of the supported user interfaces in frames hosted by a third party. While this may work, it is not guaranteed to work, and V3.4 and above ship with a configuration that explicitly blocks the use of frames via response headers. The blocking behavior is configurable and can be disabled, though this is not recommended for newer deployments.

Alternative JAXP Implementations

While we support only the Java distributions noted above, and the JAXP XML Parser implementation included with them, it is possible in principle to use alternatives. They will not in general be likely to work out of the box (or at least not safely) because our default configuration includes settings to secure the XML parser that are built into the Java reference implementation. We strongly recommend against use of an alternative parser, but there are hooks built into the software to allow for it.

V3.4 and Above

An alternative XML parser configuration can be established by defining a custom bean of type ParserPool, and providing its name via the **idp.xml.parserPool** property. This is typically done through reuse of the BasicParserPool class by copying the system-provided bean in *system/conf/global-system.xml* into a new bean in *conf/global.xml* and adjusting the default attributes and features to match the JAXP implementation in use. Any appropriate security measures and protections required are the deployer's responsibility, and there is no guarantee of interoperability.

V3.3 and Earlier

At minimum, you will need to change or remove the "SecurityManager" implementation specified in *system/conf/global-system.xml* and you will be forced to take responsibility for the result of that change, which could introduce vulnerabilities (typically denial of service vectors) into the software.

An alternative custom SecurityManager class, if one exists, can be established via the **idp.xml.securityManager** property.