# InCommon Federation Software Guidelines

**Jump to:**



# Introduction

Imagine it's 1996 and you just bought a brand new PC. You want to connect it to this new "Internet" thing you've heard about, but it turns out that your PC doesn't have a modem or network card. Sure, it can be modified to have these things, but that's extra work. Ensuring that you are using the correct federating software in InCommon is a lot like shopping for a PC that has the features you need to connect globally. Not all are created the same.

The InCommon Federation, along with research and education (R&E) identity federations worldwide, rely on ***specific profiles*** of the Security Assertion Markup Language (SAML) to exchange trusted information in a secure way. This document provides guidance for choosing an appropriate SAML software product for use in the InCommon Federation.

InCommon does not *require* participants to use any particular SAML software implementation. Instead, we value proper conformance with standard protocols (SAML and its applicable profiles and extensions). In order to work in InCommon, your software of choice **MUST** include SAML 2.0 protocol support and **MU ST *fully support*** the SAML v2.0 Metadata Interoperability Profile. It **MUST** conform to the SAML V2.0 Implementation Profile for Federation Interoperability.

In addition, your software **MUST** have the ability to **refresh and verify metadata** at least daily. Ideally, it should support the SAML Metadata Query (MDQ) Protocol. Regular metadata refresh promotes interoperability, protects users against spoofing and phishing, and is a necessary precaution in the event of key compromise. Failure to refresh metadata exposes Federation users to unnecessary risk.

See Best practices when consuming InCommon metadata for more information about metadata refresh.

# Recommended Software

The software listed in this section meet the minimum requirements of the SAML V2.0 Implementation Profile for Federation Interoperability. They have been proven to work with InCommon and other global, multilateral research and education federations:

1. Shibboleth
2. simpleSAMLphp
3. IdentityPython (pySAML)
4. Apereo CAS

# Shibboleth

Shibboleth is a family of free and open source federated single sign-on software. Shibboleth is secure, privacy-preserving, and built from the ground up to work in R&E settings with has strong metadata management capabilities for multilateral federation.  Shibboleth is maintained by the Shibboleth Consortium.

Shibboleth's family of software includes a SAML Identity Provider (IdP) and a SAML Service Provider (SP). To learn more about Shibboleth:

- Shibboleth website: http://shibboleth.net/
- Shibboleth documentation: https://wiki.shibboleth.net/
- Shibboleth mailing lists: http://shibboleth.net/community/lists.html
- Configure Shibboleth to consume InCommon metadata.
- InCommon offers pre-configured, containerized, quick-deploy versions of Shibboleth through its Trusted Access Platform

## SimpleSAMLphp

SimpleSAMLphp is a native PHP SAML implementation. It is free and open source. SimpleSAMLphp is a popular choice for organizations with substantial PHP software deployments. It offers both a SAML IdP and SAML SP, and can act as a SAML proxy. UNINETT leads the open-source simpleSAMLphp project. To learn more about SimpleSAMLphp:

- SimpleSAMLphp website: http://simplesamlphp.org/
- SimpleSAMLphp documenation: http://simplesamlphp.org/docs/stable/
- SimpleSAMLphp documenation: http://simplesamlphp.org/lists

## Related content

- Signaling Encryption Method Support for a Service Provider
- Signing and Encryption Keys
- Introduction to Federation Manager
- Working with SAML metadata
- Tagging an entity with Research and Scholarship entity attribute
- Federation references

## Get help

Can't find what you are looking for?

help Ask the community

## IdentityPython (pySAML)

The IdentityPython is a set of open source projects that provide implementation of key federation and identity technologies including OpenID Connect, SAML, xmldsig, OAuth, JWT, proxying/token translation, etc – all implemented in Python:

- IdentityPython website: https://idpy.org/
- IdentityPython documentation: https://idpy.org/projects/
- IdentityPython mailing lists: https://idpy.org/contribute/

## Apereo CAS

CAS provides enterprise single-sign-on for the web:

- CAS website: https://www.apereo.org/projects/cas
- CAS mailing lists: https://www.apereo.org/projects/cas/cas-community-support-and-mailing-lists

# Using other software

> If you decide to adopt a SAML software implementation not listed above, make sure to double check their support for federation interoperability. In many cases, this software will incur additional deployment costs and/or introduce operational challenges when working in a ***multilateral*** federated environment. You may cause interoperability issues with your federation partners, or may not be able to use other software in InCommon or other federations, at all.

Consider the following when evaluating federating software:

**Identity Provider** - an identity provider's the primary responsibility is to authenticate users and issue accurate assertions about them. In most cases, deficiencies in federated SAML support will not negatively impact your ability to authenticate a user and issue assertions However, occasional disruptions in service are likely, especially if your deployment lacks appropriate metadata support. Without the ability to consume federation metadata, you will need to manually maintain service provider registrations and track operational changes made by Service Providers.

**Service Provider** - A Service Provider is principally concerned with securing its systems and maintaining a sufficient degree of protection of its resources. In addition to hindering operational change management activities, lacking federation metadata support can limit an SP's ability to deal with a compromised signing key from an IdP partner. The use of metadata in the Federation is tightly orchestrated to address various risks that aren't adequately addressed by manually configuring metadata in the fashion that many products do.

**Proxies** - These are implementations that can translate between SAML/SAML, SAML/OAuth, or other combinations of federating protocols. A proxy which is capable of working correctly in InCommon, such as Satosa from the Identity Python project or SimpleSAMLphp, can allow otherwise non-interoperable federating software to work in InCommon.

## Microsoft Active Directory Federation Services (AD FS)

Microsoft AD FS is a popular SSO solution in higher education. Modern versions of AD FS can be successfully configured to work in the federation, with some limitations.

**AD FS can be successfully deployed as an Identity Provider** if the deployer performs the following:

- **Deploy ADFS Toolkit (or equivalent) to refresh and verify metadata automatically** - AD FS cannot consume federation metadata without the help of the additional software.
- **Ensure that all SP partners follow InCommon recommendations outlined in certificates in metadata .** Specifically:
  - certificates should be self-signed (since AD FS will actually try to check any CRLs or OCSP endpoints contained in the certificate)
  - certificates should not be expired (since AD FS will not consume an `<md:EntityDescriptor>` element that contains an expired certificate)

The Shibboleth documentation wiki has a good description on Microsoft Interoperability

## Other Cloud-based, SaaS software

Cloud identity products typically do not consume InCommon or other multilateral federation metadata. If you are choosing one of them, you will need to add additional multilateral federation-aware proxies such as SATOSA, simpleSAMLphp, or a commercial proxy to provide these capabilities.

# Impact of metadata digital certificates in software selection

A well known interoperability issue in SAML implementations is that the different implementations have varying degrees of support for X.509 certificates in metadata. When selecting your SAML software, consider these limitations carefully.

In particular, to fully support key rollover, a implementation MUST support the following features:

1. An implementation MUST be able to consume and utilize two signing keys bound to a single role descriptor. There are two cases that MUST be supported:
    a. `<md:KeyDescriptor use="signing">` and `<md:KeyDescriptor use="signing">` in a single role descriptor
    b. `<md:KeyDescriptor use="signing">` and `<md:KeyDescriptor>` in a single role descriptor
2. If an implementation supports inbound encryption, it MUST be configurable with up to two decryption keys.
3. An implementation MAY support (i.e., consume and utilize) multiple encryption keys per role descriptor in metadata. In particular, the implementation MAY support two `<md:KeyDescriptor>` elements (with no `use` XML attribute) in a single role descriptor, but this is not strictly required since it can usually be avoided in practice.

Consult your software documentation to better understand its capabilities. Indeed, evaluate software capabilities with respect to certificate handling *before* you deploy, if at all possible.