Back-channel SAML Protocols

> **Community Review in progress!**
>
> This document contains DRAFT material intended for discussion and comment by the InCommon participant community. Comments and questions should be sent to the InCommon participants mailing list (participants@incommon.org).

Back-channel SAML Protocols

Initial IdP deployment decisions may have costly long-range consequences. Once an IdP protocol endpoint is exposed in metadata, SPs will automatically start using it. When that happens, the IdP quickly becomes locked into a long-term "support contract" that may have little (or no) return on investment.

By definition, a *back-channel protocol* is a SAML protocol profiled for use with the synchronous SOAP binding (with one exception, as noted at the end of this topic). Note that some protocols (such as Attribute Query and Artifact Resolution) *require* the use of SOAP while other protocols (such as Single Logout) are profiled for use with front-channel bindings in addition to the back-channel SOAP binding.

Some characterizations may be helpful. A front-channel protocol flow is browser-facing whereas a back-channel protocol is server-to-server. Protocols paired with front-channel bindings usually run on the standard port 443 while back-channel protocols typically run on a nonstandard port such as 8443.

The various SAML protocols have significantly different *total cost of deployment* (TCD) depending on the binding. In general, the back-channel protocols have a much higher TCD since:

- back-channel protocols typically employ a transport-level security model
- back-channel protocols require a stateful IdP

These issues are discussed more fully in the following sections.

SAML Security Considerations

All browser-facing endpoints are protected with TLS. Additionally, SAML messages transmitted via a front channel are usually signed, and often encrypted, at the message level. The latter provides end-to-end security as the message transits a potentially hostile browser environment.

The security model is typically quite different on the back channel. Since messages are transmitted point-to-point, mutually authenticated TLS provides end-to-end security. Unlike browser-facing TLS, where the trust model is based on X.509 certificates signed by a trusted CA, the trust model on the back channel is based on explicit keys published in trusted metadata.

In summary, the overall security model on the front channel is based on XML Security and browser-facing TLS such that the TLS certificates are trusted by the browser. The security model on the back channel is based on TLS only where the TLS certificates are long-lived, self-signed certificates bound to trusted metadata.

This is one (important) reason why an IdP that supports both front and back-channel protocols is more difficult to deploy, manage, and maintain: such IdPs must support two completely different security models.

Artifact Resolution

An *artifact* is a reference to a SAML assertion. An IdP transmits an artifact on the front channel and responds to the SP's request to resolve the artifact on the back channel. The Artifact Resolution protocol has better confidentiality properties than front-channel SSO but that benefit must be balanced by the deployment costs of Artifact Resolution, which are significant.

If an IdP issues an artifact, it has to be prepared to resolve that artifact (into an actual assertion) at some later point in time. This requires the IdP to maintain state. Thus an IdP that supports Artifact Resolution is necessarily a stateful IdP, which is more difficult to scale in the long run.

> **Avoid publishing ArtifactResolutionService endpoints**
>
> Typical SSO use cases have low to moderate privacy requirements, so deployments can (and should) avoid Artifact Resolution if possible. New deployments, in particular, should **not** publish `ArtifactResolutionService` endpoints in metadata.

Attribute Query

Prior to SAML2, to help prevent leakage of PII, an IdP would push an authentication statement, leaving the SP to query for attributes on a back channel. Thus the IdP had to be able to map the subject in the query to the subject in the authentication statement, which required the IdP to maintain state. With the introduction of SAML2 (and XML Encryption), IdPs began pushing everything over a front channel so that back-channel Attribute Query became mostly unnecessary.

> **Do not publish a SAML2 AttributeService endpoint**
>
> An IdP that supports SAML2 Attribute Query is openly inviting redundant attribute queries and so a new IdP is advised **not** to publish a SAML2 `AttributeService` endpoint in metadata.

Some clever IdP implementations (such as the Shibboleth IdP) securely bind the authenticated subject to the opaque `NameID` asserted on the front channel. Thus an IdP that supports Attribute Query can remain stateless. AFAIK no such optimization is possible with Artifact Resolution, however, so an IdP that supports artifact should do so consciously and deliberately.

SAML1

SAML1 Web Browser SSO is a profile, not a protocol, but we mention it here because supporting SAML1 usually means supporting Attribute Query, at least as it’s deployed in the InCommon Federation (and other Federations with significant Shibboleth deployments).

Avoid publishing SAML1 endpoints

Don't support SAML1 Web Browser SSO unless you have to. In any case, do your part and put pressure on partner SPs to support SAML2 if necessary.

There are about a dozen SAML1-only IdPs and about six dozen SAML1-only SPs in the InCommon Federation. Those numbers are not growing. On the contrary, the proportion of entities that support SAML1-only is shrinking to the point where an IdP can easily avoid the SAML1 protocol with little (if any) consequence.

Moreover, the legacy SAML1-only SPs that remain in the InCommon Federation are likely to be non-Shibboleth SPs and we know that few non-Shibboleth implementations have ever supported Attribute Query in the first place. This means that you can sometimes get away with supporting SAML1 on the front channel only (if you must support it at all).

Single Logout

SAML Single Logout is by-and-large poorly understood and implemented, and has very limited deployment in the InCommon Federation. There may be hope for Single Logout if we specify the right combination of bindings at both the IdP and the SP. One approach being explored is for the SP to initiate SLO on the front channel, which prompts the IdP to contact relevant SPs via a back channel. Consequently:

- The IdP must maintain sufficient session state to support SLO in any case
- To support SLO on the back channel, the IdP needs a (self-signed) TLS certificate in metadata to authenticate to the SP’s SOAP endpoint
- The SP must deploy a SOAP endpoint (which today is not the case)

So SLO is an expensive protocol to support, for both the IdP and the SP. It remains to be seen if SLO finds wide support in the SAML community at large, but in any case, SLO shouldn’t drive the IdP deployer’s decision-making process since supporting SLO only makes sense today in the simplest of deployment scenarios.

ECP

One final consideration is ECP, which is an outlier. ECP is a SOAP-based protocol (like attribute query and artifact resolution) but ECP is in fact a front-channel protocol, and moreover, there is no widespread agreement on what the security model should be.