# NativeSPSessionInitiator

The <SessionInitiator> element is used to configure handlers that are responsible for initiating the process of authentication to the SP and establishing a session with it. This represents what single sign-on architecture refers to as an "SP-initiated" flow, where a browser starting at the application end needs to be referred to the IdP to login and then return with the appropriate information to login.

This is an advanced configuration feature. Most deployments can rely on the <SSO> shorthand element.

The specifics of the authentication request process vary by protocol, and the internal SessionInitiator API allows the SP to be extended with many request protocols. The handler is responsible for all of the details associated with making the request.

A special characteristic of this kind of handler is that it often runs implicitly as a result of the first access to a protected resource (e.g. when the `requireSession` content setting is in effect). For other kinds of use cases, such as the "passive/lazy session" feature that enables an application to defer the creation of a session, a simple (and extensible) protocol is implemented local to the SP software to enable applications to invoke these handlers using a standard HTTP redirect with a query string (see the NativeSPSessionCreationParameters topic).

The ability to configure multiple SessionInitiator handlers, and to combine them in chains, allows the deployer to control the selection of particular SSO protocols when more than one can be used, and to implement various approaches to IdPDiscovery (the process of figuring out which IdP to use in a federated environment).



## Types of SessionInitiators

Broadly speaking, there are two kinds of SessionInitiators: **protocol handlers** and **discovery handlers**.

A protocol handler requires that the name of an IdP (that is, its entityID) be supplied to the handler so that its metadata can be obtained to determine whether and where it supports a particular protocol. The `entityID` can be supplied in a number of ways, including via query string, a hardcoded <SessionInitiator> attribute, or via a content setting applied to the resource.

By contrast, a discovery handler requires that the IdP **not** be known, and is responsible for interacting with the browser in some way to determine the IdP to use. It can do this in any way it wants to, such as examining cookies, interacting with the user, or by redirecting the browser to some other server.

## Basic Configuration

An individual SessionInitiator can construct an authentication or discovery request for one of a variety of built-in protocols. One or more of these handlers can be placed into series by a "Chaining" SessionInitiator. The SP will attempt to initiate a session with each handler in the chain until one of them successfully returns something to the browser.

Typically, the deployer configures a chain in which a number of SSO protocols are listed, in order of preference, followed by a handler for some kind of "discovery" protocol that will take over if the IdP is not known or provided via a query string parameter. (In single-IdP scenarios, there may be no discovery handler.)

Although multiple handlers (or chains) can be defined, one <SessionInitiator> element is marked with `isDefault="true"`. Automatic session initiation via the `requireSession` setting uses this default SessionInitiator (unless told to use a different one using the `requireSessionWith` setting).

The default `shibboleth2.xml` file contains examples for common use cases.

## Advanced Configuration

## Common Attributes

- `type` (string)
  - Plugin type name.
- `Location` (relative path)
  - The location of the `SessionInitiator` (when combined with the base `handlerURL`). This is the location to redirect to when manually initiating a session using the query string protocol.
- `id` (string)
  - Optional, identifies a `SessionInitiator` so that it can be referenced by the `requireSessionWith` content setting.
- `isDefault` (boolean)
  - If true, establishes the default `SessionInitiator` used implicitly for content protected with the `requireSession` content setting. If none are labeled, the first is implicitly the default.
- `entityID` (URI)
  - If set, establishes an assumed IdP to use for authentication, if none is passed explicitly with a query string parameter or overridden via content settings.
- `relayState` (string)
  - Controls how information associated with the session request, primarily the original resource accessed, is preserved for the completion of the authentication process. Overrides the like-named attribute in the `<Sessions>` element.
- `acsIndex` (string)
  - This matches the `index` of the `<md:AssertionConsumerService>` element to use for the return message from the IdP. Prior to version 2.3, this property is named `defaultACSIndex`. Starting with version 2.4, this setting is optional and best avoided, in favor of letting the software automatically select the first compatible endpoint.
- `entityIDParam` (string)
  - Optional, advanced setting for overriding the name of the query string parameter used to override the IdP to use. Normally `"entityID"` and `"providerId"` are the parameter names supported. This is provided for supporting unusual application requirements.
- `target` (URL) (Version 2.4 and Above)
  - Allows the resources to return to after SSO to be "locked" to a specific value, even when running as a result of active protection of other resources. In other words, this value overrides the actual resource location when SSO redirection is automatic, including initial access and after a timeout.
- `signing` (see NativeSPSigningEncryption) (Version 2.6 and Above)
  - Controls outbound signing of XML messages subject to applicability to the protocol involved.
- `encryption` (see NativeSPSigningEncryption) (Version 2.6 and Above)
  - Controls outbound encryption of XML messages and content subject to applicability to the protocol involved.
- `externalInput` (boolean) (default is "true") (Version 2.6 and Above)
  - Allows handlers to disallow the use of externally supplied parameters / input to drive them. The specific settings this influences will vary by handler, and by default the full range of settings supported can be supplied from outside the SP, typically using query string parameters or form submission. For particularly sensitive or important options, this setting can be used to block that support. This primarily applies to the `"SAML2"` handler but may be honored by any handler as it deems appropriate.

---

## SAML2 SessionInitiator (Protocol Handler)

Indicated by `type="SAML2"`, supports SAML 2.0 authentication requests. As a protocol handler, an entityID must be specified/known, which is then used to check for metadata with an `<md:IDPSSODescriptor>` role supporting SAML 2.0. The absence of either causes a warning to be logged and the handler otherwise ignores the request.

### Attributes

- `template` (local pathname)
  - An HTML template used during transmission of the `<samlp:AuthnRequest>` message.
- `outgoingBindings` (space-delimited list of URIs)
  - List of SAML binding identifiers that determines the order of preferred `<md:SingleSignOnService>` bindings to use for the request. If this setting is used, failing to list a binding will prevent the use of an IdP that only supports the omitted binding.
- `acsByIndex` (boolean) (default is true up to Version 2.1, false after)
  - If true, the location of the assertion consumer service to return the assertion to is passed by reference (using an index), rather than passing an explicit URL and binding. Because of the difficulty of ensuring consistent indexing between local configuration and metadata, this is not an advisable feature.
- `postArtifact` (boolean) (default is false)
  - If true, the SAML artifact binding is implemented using a form POST rather then a redirect.
- `isPassive` (boolean) (default is false)

- If true, causes the <samlp:AuthnRequest>'s IsPassive attribute to be "true". Can be overridden by content setting or query string parameter.

- forceAuthn (boolean) (default is false)
  - If true, causes the <samlp:AuthnRequest>'s ForceAuthn attribute to be "true". Can be overridden by content setting or query string parameter. This asks for forced reauthentication by the IdP (bypassing SSO).

- authnContextClassRef (URI)
  - If set, inserts a <samlp:RequestedAuthnContext> element containing the class reference into the <samlp:AuthnRequest>. As of V2.5, this can be a whitespace-delimited list of classes to request. Can be overridden by content setting or query string parameter. As of V2.6, this can also be configured on a per-IdP basis via a RelyingParty setting (only applies if a more general value is not supplied).

- authnContextComparison ("exact", "minimum", "maximum", "better") (default is "exact")
  - If set, inserts a <samlp:RequestedAuthnContext> element containing the comparison operator into the <samlp:AuthnRequest>. Can be overridden by content setting or query string parameter. Ignored unless an authnContextClassRef value is set. As of V2.6, this can also be configured on a per-IdP basis via a RelyingParty setting (only applies if a more general value is not supplied).

- ECP (boolean) (default is false)
  - If set, enables Enhanced Client/Proxy profile support, causing the SP to recognize the headers sent by an ECP-enabled client and respond with an ECP request instead of a redirect. Note that when this occurs, the IdP need not be known for a request to be generated, unlike in the normal case.

- requestDelegation (boolean) (default is false) (Version 2.2 and Above)
  - If set, causes the request to carry a <saml:Conditions> element that includes a <saml:AudienceRestriction> identifying the IdP as a desired relying party for the resulting assertion. This convention is associated with support for delegation, in which the SP can authenticate itself with the assertion as the user in the course of subsequent requests to the IdP.

- NameIDFormat (URI) (Version 2.3 and Above)
  - If set, causes the request to require the IdP to respond with a NameID identifier of the given format. If the IdP can not fulfill this requirement, it will return an error response (if correctly implemented). As of V2.6, this can also be configured on a per-IdP basis via a RelyingParty setting (only applies if a more general value is not supplied).

- SPNameQualifier (URI) (Version 2.3 and Above)
  - If set, causes the authentication request to carry a saml:NameIDPolicy with an SPNameQualifier containing the provided value. If the receiving IdP can not fulfill this requirement, it will return an error response (if correctly implemented). As of V2.6, this can also be configured on a per-IdP basis via a RelyingParty setting (only applies if a more general value is not supplied).

### Child Elements

- <samlp:AuthnRequest> (optional)
  - If present, the XML is used as a template for the request issued. When the configuration file is validated during initial setup, some of the required (but meaningless) attributes on this element are required. This per-request information, such as IssueInstant and ID, is replaced/reset at runtime. Useful for supplying advanced request content that cannot be configured in a simpler way.

**Example of an Embedded AuthnRequest Template**

```
<SessionInitiator type="SAML2">
    <samlp:AuthnRequest xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol" ID="foo" Version="2.0"
IssueInstant="2012-01-01T00:00:00Z">
        <samlp:RequestedAuthnContext Comparison="exact" xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
            <saml:AuthnContextClassRef>https://federation.org/ac/type1</saml:AuthnContextClassRef>
            <saml:AuthnContextClassRef>https://federation.org/ac/type2</saml:AuthnContextClassRef>
        </samlp:RequestedAuthnContext>
    </samlp:AuthnRequest>
</SessionInitiator>
```

---

## Shib1 SessionInitiator (Protocol Handler)

Indicated by type="Shib1", supports Shibboleth 1.x authentication requests, an extension of the SAML 1.1 standard. As a protocol handler, an entityID must be specified/known, which is then used to check for metadata with an <md:IDPSSODescriptor> role supporting Shibboleth 1.x. The absence of either causes a warning to be logged and the handler otherwise ignores the request.

A "supporting" IdP's role element has a protocolSupportEnumeration attribute containing the value "urn:mace:shibboleth:1.0", with an accompanying <md:SingleSignOnService> with a Binding of "urn:mace:shibboleth:1.0:profiles:AuthnRequest".

---

## ADFS SessionInitiator (Protocol Handler)

⚠ The ADFS handler is only available if the adsfs.so extension library is loaded by the SP.

Indicated by `type="ADFS"`, supports Microsoft ADFS authentication requests, a subset of the WS-Federation passive requester profile. As a protocol handler, an `entityID` must be specified/known, which is then used to check for metadata with an `<md:IDPSSODescriptor>` role supporting ADFS. The absence of either causes a warning to be logged and the handler otherwise ignores the request.

A "supporting" IdP's role element has a `protocolSupportEnumeration` attribute containing the value `"http://schemas.xmlsoap.org/ws/2003/07/secext"`, with an accompanying `<md:SingleSignOnService>` with a `Binding` of `"http://schemas.xmlsoap.org/ws/2003/07/secext"`.

---

## SAMLDS SessionInitiator (Discovery Handler)

Indicated by `type="SAMLDS"`, refers the browser to a **discovery service** (DS) supporting the proposed SAML 2.0 IdP Discovery Protocol. As a discovery handler, no `entityID` can be known (or the handler will silently ignore the request, since discovery would serve no purpose).

The design of this protocol results in a redirect back to the SP with a single `entityID` selected, if one is chosen by the user. The handler implementation causes this redirect to be returned to itself in a manner that allows the handler to be configured in a "chain" with one or more protocol handlers. Used alone, the SAMLDS handler will simply display an error when the result is returned because it cannot itself cause an authentication request to be generated.

### Attributes

- `URL` (absolute URL)
  - The `URL` of a compliant discovery service.
- `isPassive` (boolean) (default is false)
  - If true, causes the request to the DS to carry an `IsPassive` parameter. Can be overridden by content setting or query string parameter.
- `discoveryPolicy` (string) (Version 2.5 and Above)
  - If set, causes the request to the DS to carry a `policy` parameter with the specified value. Can be overridden by content setting or query string parameter.

---

## WAYF SessionInitiator (Discovery Handler)

Indicated by `type="WAYF"`, refers the browser to a Shibboleth WAYF service. As a discovery handler, no `entityID` can be known (or the handler will silently ignore the request, since discovery would serve no purpose).

The design of this protocol results in the browser leaving the SP until a successful authentication response is returned. This is a legacy technique that limits the ability of an SP to effectively support newer protocols and more advanced features.

### Attributes

- `URL` (absolute URL)
  - The `URL` of a WAYF service.

---

## Cookie SessionInitiator (Discovery Handler)

Indicated by `type="Cookie"`, checks for a cookie maintained as part of the SP's IdP history feature and uses it to obtain the `entityID` to use for later `SessionInitiator` handlers in a chain. This handler doesn't actually cause a response to the browser, but it generally runs first in a chain, and allows the `entityID` to be set before other handlers run. As a discovery handler, no entityID can be known (or the handler will silently ignore the request, since discovery would serve no purpose).

The SP's IdP history can be enabled via the `idpHistory` attribute on the <Sessions> element.

### Attributes

- `followMultiple` (boolean) (default is false)
  - If true, a cookie containing more than one IdP will still be used to derive an `entityID` (with the last/most recent entityID used). If false, only a cookie with a single entityID in it will be followed.

---

## Form SessionInitiator (Discovery Handler)

Indicated by `type="Form"`, displays an HTML template containing a form to prompt the user for the `entityID` to use. As a discovery handler, no entityID can be known (or the handler will silently ignore the request, since discovery would serve no purpose).

This is a simple substitute for referring the user to another site, which is generally incapable of addressing scenarios involving multiple sets of unrelated IdPs. This handler can be combined with the `Transform SessionInitiator` to enable the user's input to be turned from something simpler into an `entityID`.

### Attributes

- `template` (local pathname)

- Path to the HTML template to display.

---

## Chaining SessionInitiator

Identified by `type="Chaining"`, wraps a sequence of `SessionInitiator` handlers so that they run in series. The series ends when a handler indicates that a response to the browser was returned. If no response is sent, an error results.

Options specified via attributes on the surrounding element will apply to all the embedded handlers (if not overridden inside them).

### Child Elements

- `<SessionInitiator>` (one or more)
  - Embedded plugins to instantiate.

---

## Transform SessionInitiator

Identified by `type="Transform"`, transforms an `entityID` according to a set of permutations until IdP metadata can be found. No specific protocol support is assumed; the first `entityID` for which a valid `<md:IDPSSODescriptor>` can be found terminates the handler's activity.

This handler doesn't actually cause a response to the browser, but it generally runs first in a chain, and allows the entityID to be manipulated before other handlers run. It serves a variety of purposes, from transforming user input into an entity to acting as a kind of "redirect" mechanism that turns one entityID into another.

### Attributes

- `alwaysRun` (boolean) (default is false)
  - If false, the initial entityID value is looked up, and if metadata is found, the handler exits. Set to true to perform at least one transform on even valid entityID values.

### Child Elements

- `<Subst>` (zero or more)
  - Simple transform whose element content consists of a string containing the substring `"$entityID"`, into which the current entityID value is substituted. If the element contains a `force` attribute set to `"true"`, the transform always takes effect. Otherwise the transform is only applied if metadata is found (which terminates the handler).

- `<Regex>` (zero or more)
  - Complex transform containing a `match` attribute containing a regular expression against which the current entityID value is applied, and whose element content contains a replacement expression to run based on the results of the match. Only numeric/positional group references (e.g. $1) are supported. If the element contains a `force` attribute set to `"true"`, the transform always takes effect. Otherwise the transform is only applied if metadata is found (which terminates the handler).

### Example

The example tries a sequence of transforms that allows any of the following to be turned into an InCommon IdP name (currently a URN containing a domain name):

- the domain name itself (e.g. osu.edu)
- an email address from the domain (e.g. foo@osu.edu)
- a subdomain of the domain (e.g. law.osu.edu)

```
<SessionInitiator type="Transform">
    <Subst>urn:mace:incommon:$entityID</Subst>
    <Regex match=".+@(.+)">urn:mace:incommon:$1</Regex>
    <Regex match="^[^.]+\.(.+)">urn:mace:incommon:$1</Regex>
</SessionInitiator>
```