PingFederate Integration Guide

Description

This document will provide our partners with an overview and understanding of ID.me and the implementation with PingFederate.

Prerequisites

- ☐ Established relationship with ID.me
- ☐ Understanding of the OIDC & SAML protocols
- ☐ Understanding PingFederate and external IdPs
- ☐ Access to the appropriate development environment and resources

Required Information

ID.me will provide:

OIDC:

- ☐ **IdP Client ID & Client Secret** – The Client ID & Client Secret is leveraged to point to your specific ID.me Organization & Application

SAML:

- ☐ **IdP metadata** – The metadata document describes the identity provider (IdP) to the relying party (RP) and includes the following elements
    - ☐ Refer to the SAML Metadata section below

Your organization will provide:

OIDC:

- ☐ **redirect_uri** – a custom redirect_uri value to facilitate where ID.me returns the authorization_code

SAML:

- ☐ **SP metadata** – is an XML document which contains information necessary for interaction with **SAML**-enabled identity or service providers. The document contains URLs of endpoints, information about supported bindings, identifiers and public keys.
    - ☐ Refer to the SAML Metadata section below

SAML Metadata

Once an account is created, SAML metadata (along with keys) must be exchanged to ensure proper configuration of the endpoints.

Sandbox IdP Metadata
Production IdP Metadata

Note: Preserving formatting and whitespace is important when importing any XML metadata.

The metadata document describes the IdP to a SP, including the following elements:

- [ ] The endpoint addresses for communication
- [ ] The X.509 certificates being used to sign and encrypt SAML assertions
- [ ] The SAML bindings supported by the service provider

SAML Bindings

The ID.me IdP SAML service supports HTTP POST and HTTP Redirect bindings.

Name Identifier

The ID.me IdP SAML service supports the following NameID formats:

urn:oasis:names:tc:SAML:2.0:nameid-format:persistent

urn:oasis:names:tc:SAML:2.0:nameid-format:transient

urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified

urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress

Best Practice

ID.me recommends nameid-format:persistent based on UUID as the UUID within ID.me is a unique identifier which will not change and the other NameID values can vary.

Authentication Context

The ID.me IdP SAML service supports invoking different authentication and verification policies on a per-application or per-request basis. The policy name is required to be passed along within the AuthnContext.

Sandbox Example:

Note

The following is an IdP initiated SSO example. ID.me strongly recommends an SP initiated SSO where the SP generates the AuthnRequest.

https://api.idmelabs.com/saml/SingleSignOnService?EntityID=<EntityID>&Binding=<binding>&AuthnContext=<policy-handle>

Where <policy-handle> would be replaced with the appropriate policy name which is provided by ID.me.

For more information about available policies and support for setting these up, please contact partnersupport@id.me.

SAML is a secure protocol, which supports encryption and message signing. In addition, the HTTP communication security between the SP and the IdP is ensured by using SSL (TLS v1.1 or higher).

XML Signature

All ID.me SAML messages are digitally signed. This includes all requests, assertions and metadata. The XML signature is contained within the element. The signature serves as proof that only the IdP could have signed the element, and also to guarantee the integrity of the assertion. ID.me signs messages using SHA256, SHA384 and SHA512 algorithms.

XML Encryption

ID.me requires all SAML assertions to be encrypted. This ensures the privacy of any confidential data contained within the response transmission. The encrypted assertion is contained within the element.
ID.me supports using AES-128, AES-192 and AES-256 as message encryption algorithms.

Resources and Reference Materials

ID.me Button and Brand Guidelines

ID.me offers several styles of buttons as well as guidelines for using brand assets.
https://developers.id.me/brand-assets

Swagger

On Swagger, you will find examples of ID.me code, errors and more:
https://app.swaggerhub.com/search?owner=ID.me
Multi-factor Authentication:
https://app.swaggerhub.com/apis/ID.me/multifactor/1.0.0
Digital Identity Verification:
https://app.swaggerhub.com/apis/ID.me/digital_identity/1.0.0

Steps to Integrate

Step 1: Create Developer Account

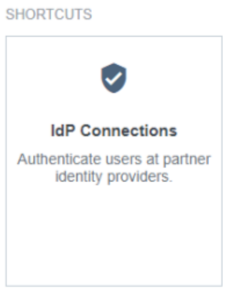
Sign up for a developer account at https://developers.id.me/. This will enable access to additional developer documentation.

Step 2: Download & Install of PingFederate

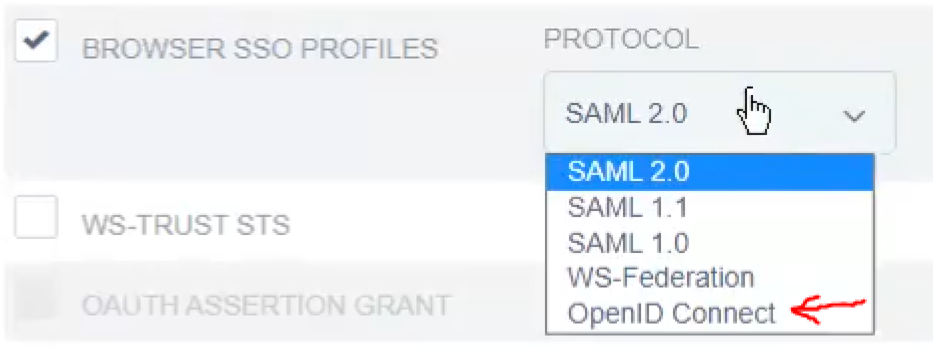
Please refer to the [Ping Identity documentation](#) on how to **download and install** PingFederate, including connecting to your **data store** of choice.

Step 3: Add an OIDC Identity Provider to PingFederate

- ☐ Log into your PingFederate Admin Portal: https://**<hostname>**:9999/pingfederate/app#/
- ☐ Navigate to **Authentication -> IdP Connections**

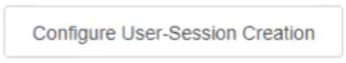


- ☐ Select **Creation Connection**
- ☐ Select **Browser SSO Profiles** and select **OpenID Connect**

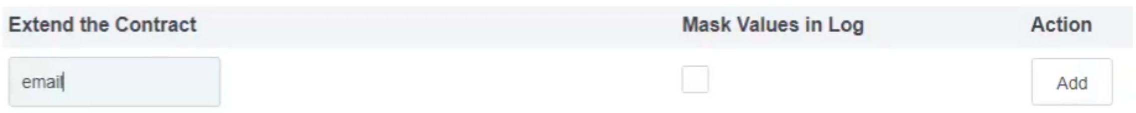


- ☐ Select **Next**
- ☐ Deselect **JIT Provisioning** *(we will do this as a later time)*
- ☐ Select **Next**
- ☐ Enter the **ID.me Issuer**

- ☐ **Sandbox**: https://api.idmelabs.com/oidc
  - ☐ **Production**: https://api.id.me/oidc
- ☐ Select the **Load Metadata** button
- ☐ Enter your desired Connection Name of choice
- ☐ Enter the **Client ID** & **Client Secret** provided by ID.me or find this by logging into your ID.me Developer Account
- ☐ Select **Next**
- ☐ Select **Configure Browser SSO**
- ☐ Select **IdP-initiated SSO**
- ☐ Select **Next**

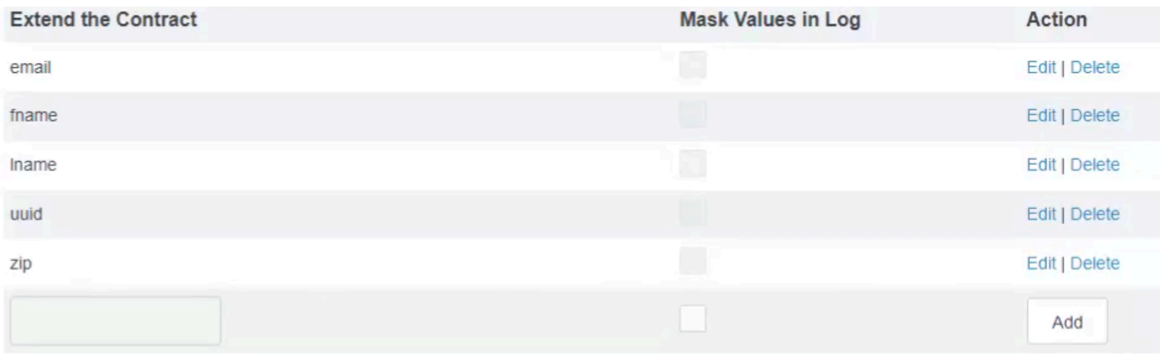


- ☐ Select **Configure User-Session Creation**
- ☐ Select **Account Mapping**
- ☐ Select **Next**
- ☐ Under **Extend the Contract,** enter **Email** and select **Add**



Extend the Contract	Mask Values in Log	Action
email	☐	Add

- ☐ Repeat the previous steps for the ID.me Attributes provided by your ID.me Solutions Consultant, such as: **fname, lname, zip, uuid**

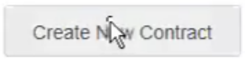


Extend the Contract	Mask Values in Log	Action
email	☐	Edit | Delete
fname	☐	Edit | Delete
lname	☐	Edit | Delete
uuid	☐	Edit | Delete
zip	☐	Edit | Delete
	☐	Add

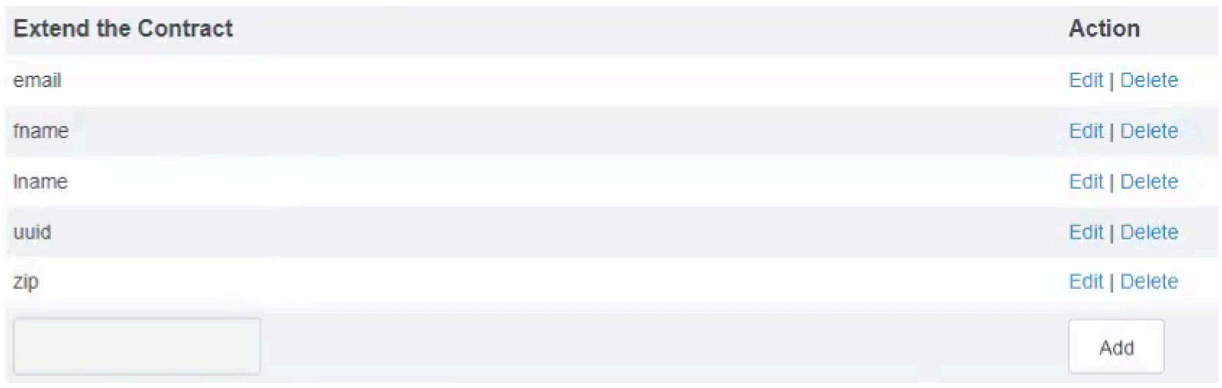
- ☐ Select **Next**

- ☐ Select **Map New Authentication Policy**

- ☐ Select **Manage Policy Contract**

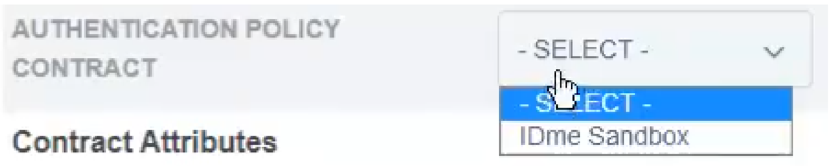


- ☐ Select **Create New Contract**
- ☐ Enter your desired **Contract Name**
- ☐ Under **Extend the Contract**, add the **same attributes** entered previously

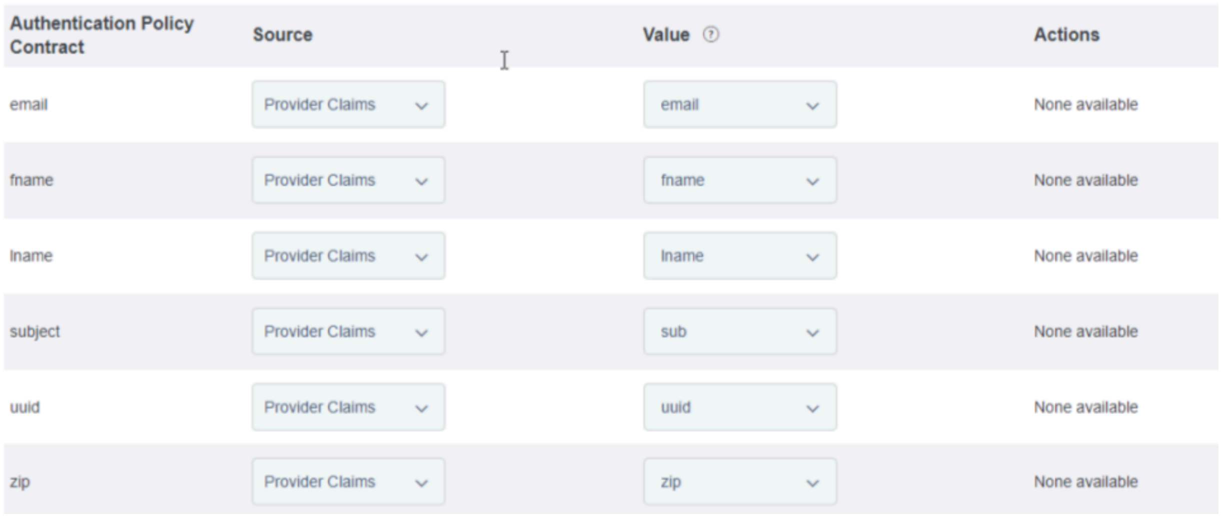


Extend the Contract	Action
email	Edit | Delete
fname	Edit | Delete
lname	Edit | Delete
uuid	Edit | Delete
zip	Edit | Delete
	Add

- ☐ Select **Next**
- ☐ Select **Save**
- ☐ Select **Done**
- ☐ Under **Authentication Policy Contract**, select your new Contract



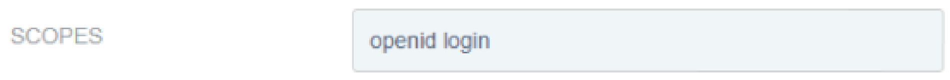
- ☐ Select **Next**
- ☐ Select **Use Only the Attributes Available in the SSO Assertion**
- ☐ Select **Next**
- ☐ Under **Source**, select **Provider Claims** for each value and map the values 1:1



Authentication Policy Contract	Source	Value	Actions
email	Provider Claims	email	None available
fname	Provider Claims	fname	None available
lname	Provider Claims	lname	None available
subject	Provider Claims	sub	None available
uuid	Provider Claims	uuid	None available
zip	Provider Claims	zip	None available

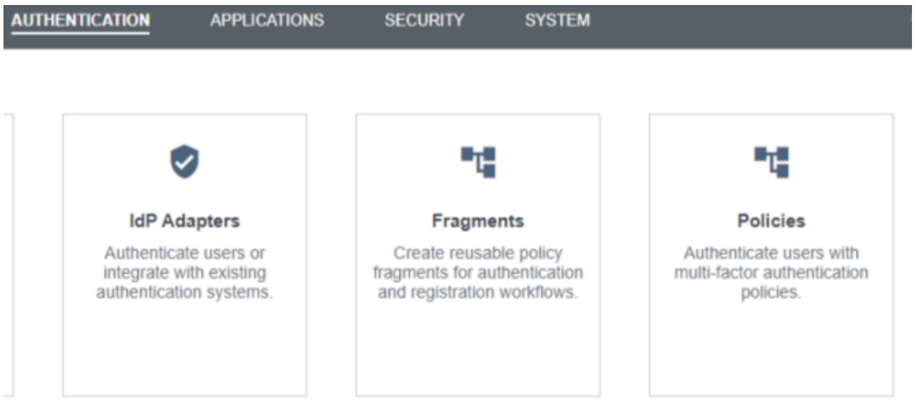
- ☐ Select **Next**
- ☐ <u>**Note**</u>**:** The Issuance Criteria can be leveraged to create conditional logic to determine whether or not to continue the SSO transaction. In this guide, we will not be configuring an Issuance Criteria
- ☐ Select **Next**

- ☐ Select **Done**
- ☐ Select **Next**
- ☐ Select **Done**
- ☐ Select **Configure User-Session Creation**
- ☐ Select **Done**
- ☐ Select **Next**
- ☐ Select **Configure Protocol**
- ☐ Under Scopes, enter **openid <scope>,** replacing <scope> with your ID.me policy

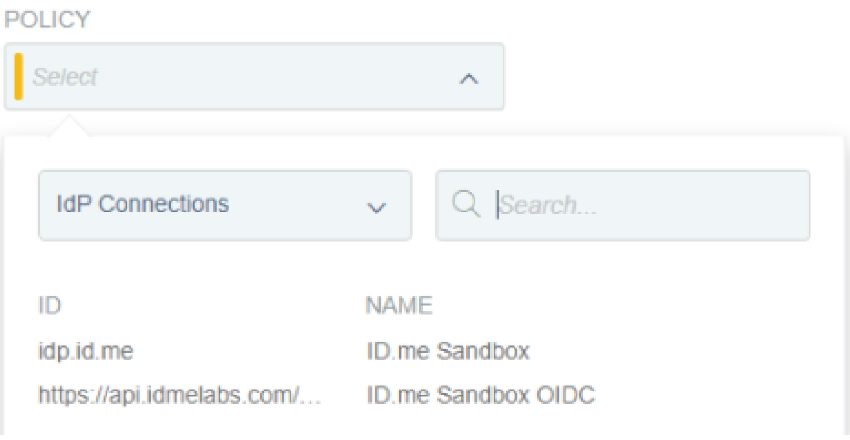


SCOPES openid login

- ☐ Under **Authorization Endpoint**, enter the appropriate ID.me Authorization Endpoint
    - ☐ **Sandbox**: https://api.idmelabs.com/oauth/authorize
    - ☐ **Production**: https://api.id.me/oauth/authorize
- ☐ Select **Code** for OpenID Connect Login Type
- ☐ Select **Basic** for the Authentication Scheme
- ☐ Under **Token Endpoint**, enter the appropriate ID.me Token Endpoint
    - ☐ **Sandbox**: https://api.idmelabs.com/oauth/token
    - ☐ **Production**: https://api.id.me/oauth/token
- ☐ Leave **UserInfo Endpoint** blank *(PingFederate currently does not support a JWT OIDC integration)*
- ☐ Under **JWKS URL**, enter the appropriate ID.me Token Endpoint
    - ☐ **Sandbox**: https://api.idmelabs.com/oidc/.well-known/jwks
    - ☐ **Production**: https://api.id.me/oidc/.well-known/jwks
- ☐ Select **Next**
- ☐ Select **Next**
- ☐ Select **Done**
- ☐ Select **Next**
- ☐ Select **Done**
- ☐ Select **Next**
- ☐ Select **Save**
- ☐ Navigate to **Authentication -> Policies**

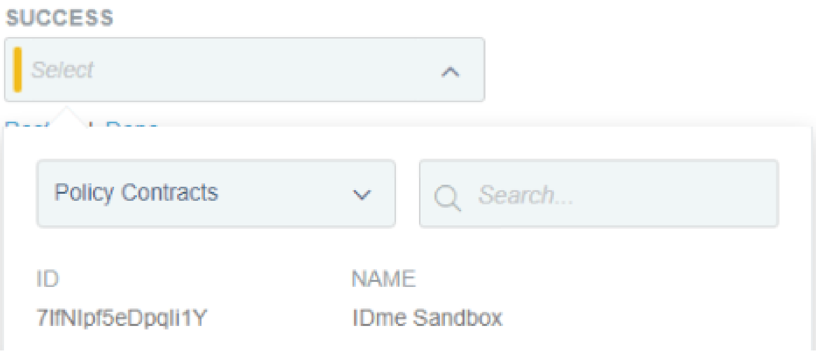

AUTHENTICATION
APPLICATIONS
SECURITY
SYSTEM
IdP Adapters
Authenticate users or integrate with existing authentication systems.
Fragments
Create reusable policy fragments for authentication and registration workflows.
Policies
Authenticate users with multi-factor authentication policies.

- ☐ Select **Add Policy**
- ☐ Enter a **Policy Name** of choice
- ☐ Under **Policy**, Select **IdP Connection** and Select your newly created IdP Connect



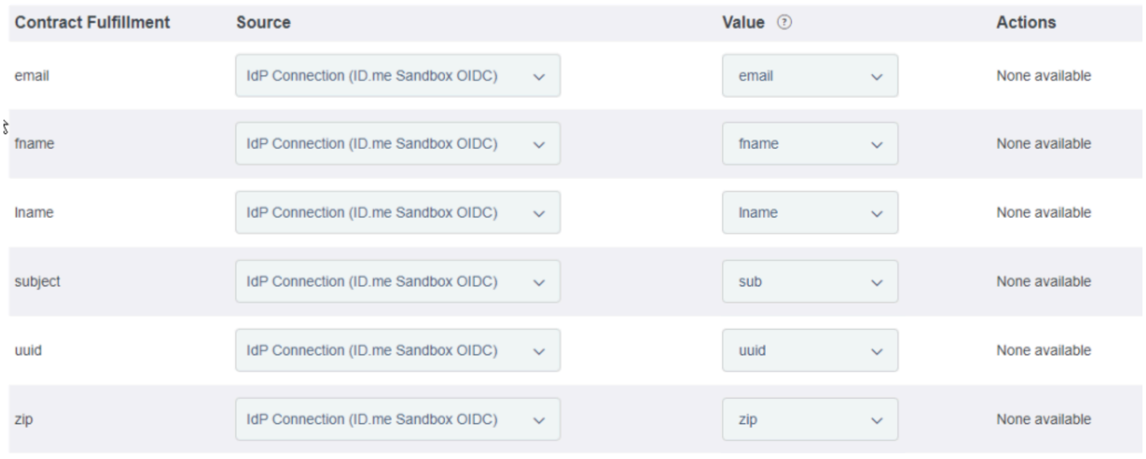
ID	NAME
idp.id.me	ID.me Sandbox
https://api.idmelabs.com/...	ID.me Sandbox OIDC

- ☐ Under **Fail**, select Done
- ☐ Under **Success**, select the dropdown for Policy Contracts and Select your Contract



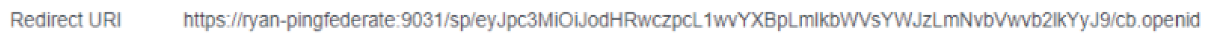
ID	NAME
7IfNIpf5eDpqli1Y	IDme Sandbox

- ☐ Select **Contract Mapping**
- ☐ Select **Next**
- ☐ Under **Source**, Select **IdP Connection** and map the appropriate values



Contract Fulfillment	Source	Value	Actions
email	IdP Connection (ID.me Sandbox OIDC)	email	None available
fname	IdP Connection (ID.me Sandbox OIDC)	fname	None available
lname	IdP Connection (ID.me Sandbox OIDC)	lname	None available
subject	IdP Connection (ID.me Sandbox OIDC)	sub	None available
uuid	IdP Connection (ID.me Sandbox OIDC)	uuid	None available
zip	IdP Connection (ID.me Sandbox OIDC)	zip	None available

- ☐ Select **Next**
- ☐ Select **Next**
- ☐ Select **Done**
- ☐ Select **Done**
- ☐ Select **Save**
- ☐ Navigate to **Authentication -> IdP Connections** and select your **OIDC Connection**
- ☐ Enter your **Redirect URI** shown here on your **ID.me Developer Account** or send the Redirect URI to your **ID.me Solutions Consultant**

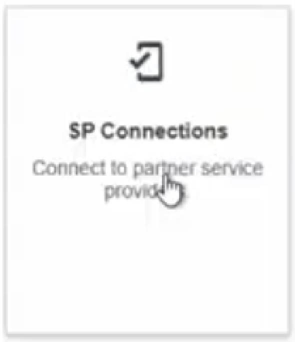


Redirect URI https://ryan-pingfederate:9031/sp/eyJpc3MiOiJodHRwczpcL1wvYXBpLmlkbWVsYWJzLmNvbVwvb2lkYyJ9/cb.openid

Step 4: Create a PingFederate Test SP Application *[optional]*

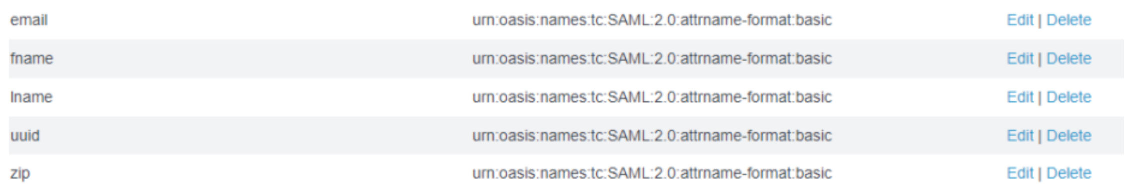
This section will walk you through creating a sample SP application to test the ID.me IdP Connection.

- ☐ Navigate to your **PingFederate Admin console**
- ☐ Navigate to **Applications -> SP Connections**



- ☐ Select **Create Connection**
- ☐ Select “Do Not Use a Template for the Connection” and click **Next**

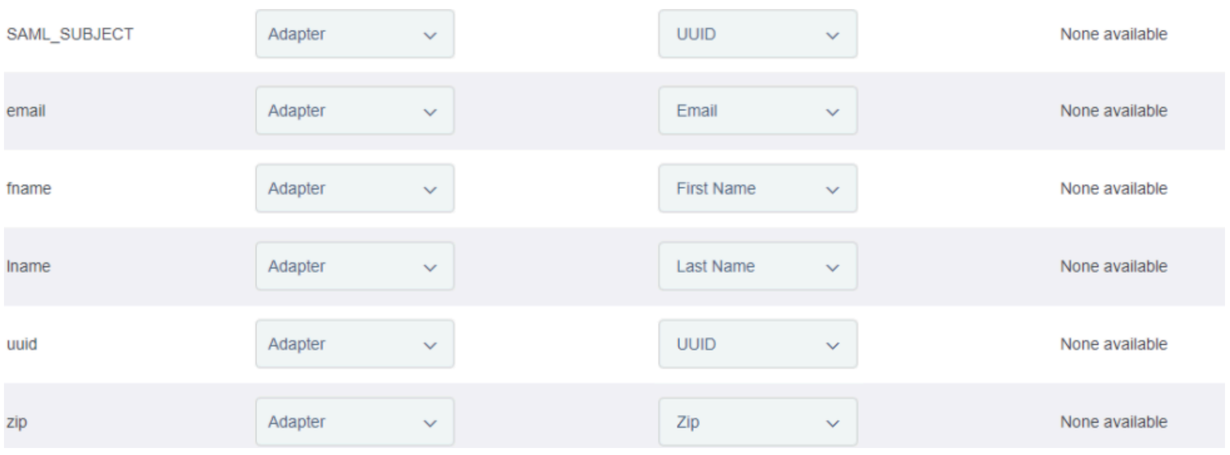
☐ Select **Browser SSO Profiles -> SAML 2.0** and click Next
☐ For Metadata, select **URL**
☐ Select **Manage Partner Metadata URLs**
☐ Select **Add New URL** and enter the following details:
    ☐ **Name**: IAMShowcase
    ☐ **URL**: https://sptest.iamshowcase.com/testsp_metadata.xml
☐ Select **Load Metadata**
☐ Click **Next**
☐ Click **Next**
☐ Click **Save**
☐ Under **Metadata URL**, select **IAMShowcase** and select **Load Metadata**
☐ Click **Next**
☐ Click **Next**
☐ Select **Configure Browser SSO**
☐ Check **IdP-Initiated SSO** and **SP-Initiated SSO**
☐ Click **Next**
☐ Click **Next**
☐ Select **Configure Assertion Creation**
☐ Select **Next**
☐ Enter the following values for **extending the contract** :



email	urn:oasis:names:tc:SAML:2.0:attrname-format:basic	Edit | Delete
fname	urn:oasis:names:tc:SAML:2.0:attrname-format:basic	Edit | Delete
lname	urn:oasis:names:tc:SAML:2.0:attrname-format:basic	Edit | Delete
uuid	urn:oasis:names:tc:SAML:2.0:attrname-format:basic	Edit | Delete
zip	urn:oasis:names:tc:SAML:2.0:attrname-format:basic	Edit | Delete

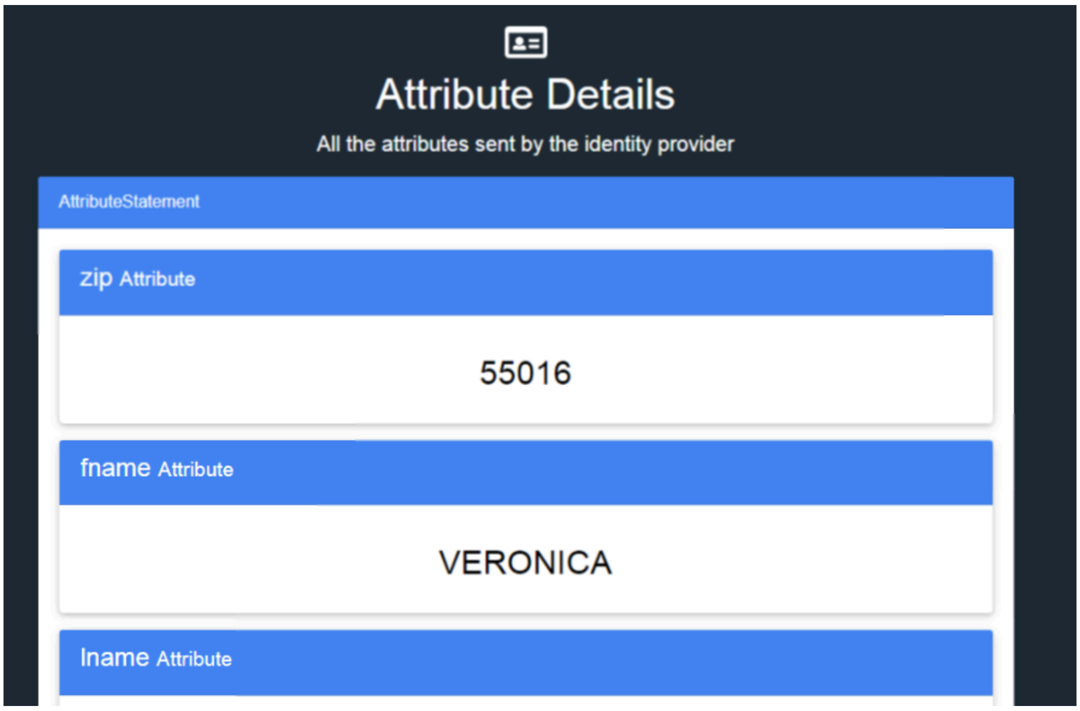
Note: you may want to add additional attributes based on your testing

☐ Select **Next**
☐ Select **Map New Adapter Instance**
☐ Select **SimpleForm**
☐ Click **Next**
☐ Click **Next**
☐ Under **Source**, select **Adapter** for each attribute contract
☐ Select the corresponding **attribute** to each contract, as shown below:



SAML_SUBJECT	Adapter	UUID	None available
email	Adapter	Email	None available
fname	Adapter	First Name	None available
lname	Adapter	Last Name	None available
uuid	Adapter	UUID	None available
zip	Adapter	Zip	None available

- ☐ Click **Next**
- ☐ Click **Next**
- ☐ Click **Done**
- ☐ Click **Next**
- ☐ Click **Done**
- ☐ Click **Next**
- ☐ Click **Configure Protocol Settings**
- ☐ Click **Next**
- ☐ Deselect **Artifact** and **SOAP** (POST and REDIRECT should be checked)
- ☐ Click **Next**
- ☐ Click **Next**
- ☐ Click **Done**
- ☐ Click **Next**
- ☐ Click **Done**
- ☐ Click **Next**
- ☐ Click **Next**
- ☐ Click **Save**
- ☐ Select **IAMShowcase** under **Connection Name**
- ☐ Click the SSO Application Endpoint URL
- ☐ Testing:
    - ☐ Log into ID.me successfully
    - ☐ On the success page, you should see the attributes mapped under Subject Information and Attribute Details



Attribute Details

All the attributes sent by the identity provider

AttributeStatement

zip Attribute

55016

fname Attribute

VERONICA

lname Attribute

Step 5: Configure JIT Provisioning *[optional]*

PingFederate's just-in-time (JIT) provisioning allows service providers (SPs) to create user accounts on the fly during single sign-on (SSO) events, based on attributes received in SSO tokens from identity providers (IdPs).

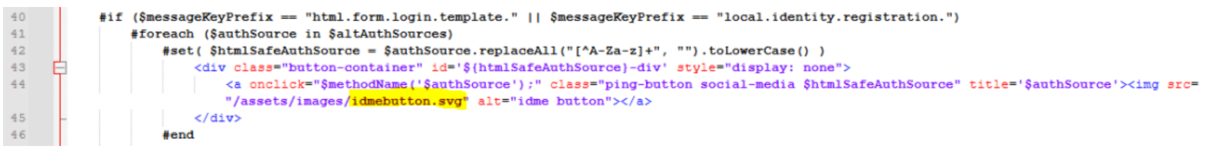
Just-in-time provisioning is highly dependent based on the user store you configured within your PingFederate environment. As there are many iterations of JIT provisioning, please folle the guide on Ping Identity's website: https://docs.pingidentity.com/r/en-us/pingfederate-103/help_idpconnectionconfigtasklet_userprovisioningstate

Step 6: Customize the Login Page

How to add the green ID.me Button :

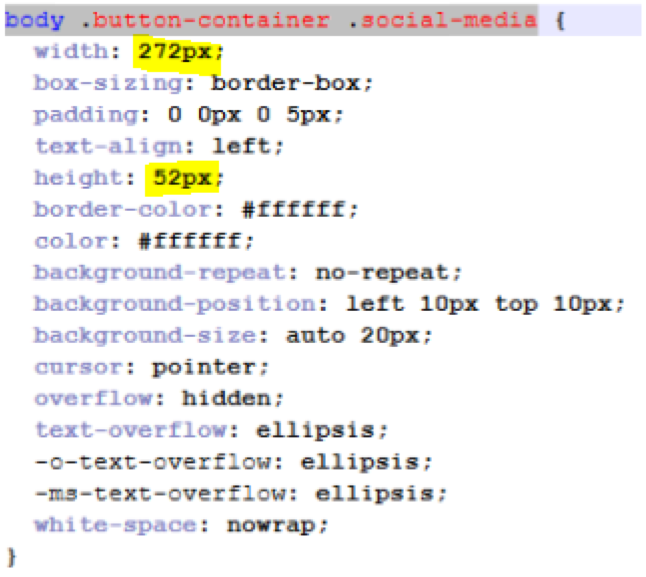
- ☐ Open **File Explorer**
- ☐ Navigate to **C:\Program Files\Ping Identity\pingfederate-11.2.2\pingfederate\server\default\conf\template\assets\images**
- ☐ Save this button as "**idmebutton.svg**" and upload it to this repository

- ☐ Navigate to **C:\Program Files\Ping Identity\pingfederate-11.2.2\pingfederate\server\default\conf\template**
- ☐ Open **alt-authn-source.template** in a text editor
- ☐ Update **line 44** with the following code snippet:
    - ☐ *<a onclick="$methodName('$authSource');" class="ping-button social-media $htmlSafeAuthSource" title='$authSource'><img src="/assets/images/idmebutton.svg" alt="idme button"></a>*



```
40  #if ($messageKeyPrefix == "html.form.login.template." || $messageKeyPrefix == "local.identity.registration.")
41      #foreach ($authSource in $altAuthSources)
42          #set( $htmlSafeAuthSource = $authSource.replaceAll("[^A-Za-z]+", "").toLowerCase() )
43              <div class="button-container" id='${htmlSafeAuthSource}-div' style="display: none">
44                  <a onclick="$methodName('$authSource');" class="ping-button social-media $htmlSafeAuthSource" title='$authSource'><img src=
                    "/assets/images/idmebutton.svg" alt="idme button"></a>
45              </div>
46          #end
```

- ☐ Save the **file**
- ☐ Restart your **PingFederate Service**
- ☐ Refresh your **Login Page**, you should see the ID.me button
- ☐ Next, you'll want to update the **dimensions** for the button
- ☐ Navigate to **C:\Program Files\Ping Identity\pingfederate-11.2.2\pingfederate\server\default\conf\template\assets\css**
- ☐ Open **main.css**
- ☐ Navigate to **line 4188: body .button-container .social-media**
- ☐ Change the width to 272px and the height to 52px



```
body .button-container .social-media {
  width: 272px;
  box-sizing: border-box;
  padding: 0 0px 0 5px;
  text-align: left;
  height: 52px;
  border-color: #ffffff;
  color: #ffffff;
  background-repeat: no-repeat;
  background-position: left 10px top 10px;
  background-size: auto 20px;
  cursor: pointer;
  overflow: hidden;
  text-overflow: ellipsis;
  -o-text-overflow: ellipsis;
  -ms-text-overflow: ellipsis;
  white-space: nowrap;
}
```

- ☐ Save the **file**
- ☐ Restart your **PingFederate Service**
- ☐ Refresh your **Login Page**, you should see the ID.me button resized

<u>How to update the Titles:</u>

- ☐ Open **File Explorer**

- ☐ Navigate to **C:\Program Files\Ping Identity\pingfederate-11.2.2\pingfederate\server\default\conf\language-packs**
- ☐ Open **pingfederate-messages.properties** in a text editor
- ☐ Navigate to the comment: **# html.form.login.template.html**
- ☐ This section is where you can update the **Login Page titles**
    - ☐ The **Social Login title** is under html.form.login.template.loginWithButtonTitle={title}
- ☐ To test your changes, **restart the PingFederate Service** and refresh the login page

Payload Specification

Handle	Name	Max Length	Null Possible?	Comment
email	Email	255	No	
uuid	Unique Identifier	32	No	
fname	First Name	255	**Yes**	Individuals with only 1 legal name would have that name placed in the Last Name attribute
mname	Middle Name	255	**Yes**	
lname	Last Name	255	No	
birth_date	Birth Date	10	No	Date format: YYYY-MM-DD
social	Full SSN	9	**Yes**	
itin	Full ITIN	9	**Yes**	
ssn_itin	Full SSN or ITIN	9	No	
street	Street	255	No	123 Main St Apt 1a
street1	Street1	255	**Yes**	123 Main St
street2	Street2	255	**Yes**	Apt 1a

city	City	255	No	
state	State	255	No	Abbreviation or full state
zip	Zip Code	10	No	US Zip format: 00000, 00000-0000, or 000000000
phone	Phone	11	**Yes**	Phone format: 10000000000
identity_sub groups	Identity Subgroup	255	No	Returns ID.me identity policy

Note
Additional attributes may be available. Discuss your use case with your ID.me team.

Best Practices

To efficiently and effectively integrate with ID.me it is recommended to use one of the Web Access Management Software Configurations listed above in the Resources section of this document. Those configurations will allow you to generate SP metadata and be able to ingest IdP metadata with ease vs when attempting a custom SAML implementation the above details must be used when defining and designing the integration.

Matching

When matching attributes in the ID.me payload to user data on your side, do not rely on a single attribute. Best practice would be to match multiple attributes such as SSN/ITIN, DOB and Last Name. Matching on SSN/ITIN as the first attribute to establish uniqueness, followed by Date of Birth, Last Name, and so on will increase assurance of uniqueness. With Last Name, issues can arise for hyphenated names or legal name changes. If First Name is leveraged, consideration should be taken to allow variations such as Thomas/Tom through the use of fuzzy logic.
Upon subsequent logins from a user, the UUID from ID.me should be incorporated into the logic. Additionally, additional attributes may have been updated. For example, last names may change due to marriage status or legal name changes.

Storing User Attributes

Storing key attributes about the user is vital to a seamless digital identity verification experience.
It is recommended to store the returned attributes in a separate table within your database with some relation to the user record. Remember, the ID.me provided UUID will remain constant for the life of that user’s ID.me account.
When extracting User attributes from JSON it is recommended that the handle_name be used to read the attributes. Relying on attribute order is susceptible to errors and issues as the platform and policies evolve.

Using User Attributes

It is recommended that the user attributes be used to pre-fill information into any additional forms requested in the workflow in the same session to improve UX. Consider locking fields that are pre-filled with attribute data, where appropriate. For example, First Name, Last Name and Social Security Number be locked. A "Prefered Name" field can be leveraged to allow for nicknames.
Care should be taken to handle unexpected payload changes such as the addition or removal of an attribute, change in attribute ordering, or an attribute containing an unexpected special character.

Exception Handling

If the user denies the access request, or if the request is invalid, the client will be informed using the parameters in the following table, appended to the AssertionConsumerService:

https://developers.id.me/documentation/identity/saml/overview#:~:text=Mode%22%20to%20users.-,ERRORS,-If%20the%20user

Code	Description
urn:oasis:names:tc:SAML:2.0:status: AuthnFailed	The responding provider was unable to successfully authenticate the principal.
urn:oasis:names:tc:SAML:2.0:status: Requester	The request could not be performed due to an error on the part of the requester.
urn:oasis:names:tc:SAML:2.0:status: Responder	The request could not be performed due to an error on the part of the SAML responder or SAML authority.
urn:oasis:names:tc:SAML:2.0:status: VersionMismatch	The SAML responder could not process the request because the version of the request message was incorrect.
urn:oasis:names:tc:SAML:2.0:status: InvalidAttrNameOrValue	Unexpected or invalid content was encountered within an element.
urn:oasis:names:tc:SAML:2.0:status: InvalidNameIDPolicy	The responding provider cannot or will not support the requested name identifier policy.

urn:oasis:names:tc:SAML:2.0:status: NoAuthnContext	The specified authentication context requirements cannot be met by the responder.
urn:oasis:names:tc:SAML:2.0:status: NoAvailableIDP	Used by an intermediary to indicate that none of the supported identity provider elements in an can be resolved or that none of the supported identity providers are available.
urn:oasis:names:tc:SAML:2.0:status: NoPassive	Indicates the responding provider cannot authenticate the principal passively, as has been requested.
urn:oasis:names:tc:SAML:2.0:status: NoSupportedID	Used by an intermediary to indicate that none of the identity providers in an are supported by the intermediary.
urn:oasis:names:tc:SAML:2.0:status: PartialLogout	Used by a session authority to indicate to a session participant that it was not able to propagate logout to all other session participants.
urn:oasis:names:tc:SAML:2.0:status: ProxyCountExceeded	Indicates that a responding provider cannot authenticate the principal directly and is not permitted to proxy the request further.
urn:oasis:names:tc:SAML:2.0:status: RequestDenied	The SAML responder or SAML authority is able to process the request but has chosen not to respond. This status code MAY be used when there is concern about the security context of the request message or the sequence of request messages received from a particular requester.
urn:oasis:names:tc:SAML:2.0:status: RequestUnsupported	The SAML responder or SAML authority does not support the request.
urn:oasis:names:tc:SAML:2.0:status: RequestVersionDeprecated	The SAML responder cannot process any requests with the protocol version specified in the request.
urn:oasis:names:tc:SAML:2.0:status: RequestVersionTooHigh	The SAML responder cannot process the request because the protocol version specified in the request message is a major upgrade from the highest protocol version supported by the responder.
urn:oasis:names:tc:SAML:2.0:status: RequestVersionTooLow	The SAML responder cannot process the request because the protocol version specified in the request message is too low.

urn:oasis:names:tc:SAML:2.0:status: ResourceNotRecognized	The resource value provided in the request message is invalid or unrecognized.
urn:oasis:names:tc:SAML:2.0:status: TooManyResponses	The response message would contain more elements than the SAML responder is able to return.
urn:oasis:names:tc:SAML:2.0:status: UnknownAttrProfile	An entity that has no knowledge of a particular attribute profile has been presented with an attribute drawn from that profile.
urn:oasis:names:tc:SAML:2.0:status: UnknownPrincipal	The responding provider does not recognize the principal specified or implied by the request.
urn:oasis:names:tc:SAML:2.0:status: UnsupportedBinding	The SAML responder cannot properly fulfill the request using the protocol binding specified in the request.