Employee Giving and Employee Volunteering

Single Sign-On (SSO)

Clients interested in streamlining their users experience can utilize single sign-on (SSO) capabilities to federate user identities.

Single sign-on allows Blackbaud to configure a client's application to redirect their users to a branded, secure client login page. Upon authenticating themselves via the login page by entering their company-issued credentials users are then seamlessly logged into the application.

Table of Contents

Support

If you have questions or need assistance in any way, please contact Support.

Single Sign On

The standard Single Sign On (SSO) solution method described in this document is secure and relatively simple to implement. This is appropriate for clients that do not have an existing corporate standard SSO implementation and do not have an identity provider that supports the more involved SAML protocol.

The solution uses a digital signature that is sent with the login request and verified using a pre-shared public key. This allows the solution to confirm that the data and signature received originated from the SSO portal.

The tables below describe the types of SSO implementations and experience available.

Types of SSO implementations we offer:

Standard SSO	**SHA1/RSA, SAML 1.1, SAML 2.0, Workday***
Custom SSO	**LDAP or other**

NOTE: For Workday SSO implementation, the client must contact their Workday administrator to initially set up SSO with Workday. Workday authenticated users can then be authenticated via SSO to the application . More details and requirements will be provided when implementation is purchased.

Types of SSO experience we offer:

Flavor	What happens when a user visits the site:
SSO Only	**Unauthenticated users are always forwarded to the client's SSO portal for authentication.**
Reverse-Hybrid SSO	**By default, unauthenticated users are forwarded to the client's SSO portal for authentication. Via a specially formed URL unauthenticated users can be driven to the login page.**

NOTE: This document describes the Standard SSO implementation. For Custom SSO implementations, please contact an Account Manager with your customization requirements.

Basic Steps for SSO Implementation

The basic steps for SSO implementation are described below:

1. Determine the type of SSO implementation you need (SHA1-RSA, SAML 1.1 or 2.0., custom, etc.).
2. **Provide us with an X-509 certificate**. You will be using this certificate to digitally sign the SSO authentication request and we will use the key to ensure that it came from you. It can be one certificate (shared on staging and production environments) or different ones for staging and production.
3. **Provide your SSO Portal URL**. This is required regardless of the type of SSO being used and must be the URL that will be used in production. Without this URL set, the SSO will not work. Testing the SSO without a Portal URL will not be possible.

   NOTE: Your Portal URL must be the same for staging and production.
4. **Provide your desired logout URL**. This is also required regardless of the type of SSO being used. However, as it does not disrupt functionality, it can be set to a "reasonable default" URL until the final one is available.
5. **SSO is set up on our staging environment**. We will create one user on our staging environment with an appropriate external ID and sysadmin access. This sysadmin login can then be used to create additional user accounts for testing. You can login either using your implemented SSO or, if not yet set up, through the login screen with SSO in hybrid mode.
6. **Set the staging Post URL**. The POST URL that will be used by your system to automatically "post" or send authentication requests during the testing stage will be: https://clientname.staging.angelpointsevs.com/login.sso.
7. **Test the SSO on the staging environment**. Provide us with feedback and approval to move to the production environment. At this point you may submit a new X-509 certificate for the production environment or decide to use the same one from staging.
8. **SSO functionality is copied to production**. SSO functionality is turned on in the production environment and any necessary modifications are made.
9. **Set the production Post URL**. Update the production POST URL to: https://clientname.angelpointsevs.com/login.sso.

SSO Implementation Notes:

- If your site is already live, the production site will initially be set to a hybrid mode for testing.
- If your site is not currently live and you are going to be SSO only, then it will be set as SSO only from the start.
- To minimize downtime, the final SSO Portal and Logout URLs must be set correctly and a time will need to be coordinated with us for the switch over.

1) SSO Authentication Process

The standard SSO authentication process has 3 required steps and 2 optional steps. They are, in the order in which they take place:

1. **Optional Step:** User visits the application site. This is an optional step since the user can be sent to the SSO portal without first visiting the site. However, this is required for navigating directly to a specific page within the application. If this step does not occur, the user will initially be shown the default page.
2. **Optional Step:** The application redirects the user to the client SSO portal URL. Again, this is optional and will only take place when the user's first request is to the application.
3. The SSO portal authenticates the user through whatever mechanism is appropriate. This could be an HTML form, one of the HTTP authentication mechanisms or even authentication against a Windows Active Directory or LDAP/Kerberos server.
4. Upon successful authentication of the user's credentials, the user is sent an HTML page with JavaScript that submits an HTML form to the application SSO URL. The elements of the HTML form are discussed below. One note: it should be possible to use an HTTP redirect in lieu of an automatically-submitted HTML form. However, this has yet to be tested. If using this method, be aware that each parameter passed must be URL encoded (the browser does this automatically when the form is submitted).
5. The SSO implementation verifies the digital signature contained in the request from the user to confirm that the data in the login request was signed by the SSO portal. Upon successful verification of the signature, the user will be shown the initial URL from step 1, if that step occurred or the default page. The case where verification fails is covered below in the "Error Conditions" section.

The SHA-1/RSA Login Request

The login request is usually done in step 4 of the SSO authentication process. It is specific to SHA-1/RSA and consists of the following parameters:

Userid:

This is the identifier assigned to the user that is being authenticated. Please refer to documentation on the data feed format for more information on the value of this parameter.

Timeout:

This is the expiration time for the login attempt. This prevents someone from reusing a login attempt beyond when it is initially intended.

The value must be sent in UTC format (e.g. 2008-01-01T18:30:00). Values should be adjusted to UTC time. UTC is roughly equivalent to GMT time, though there are slight differences.

NOTE: For SAML2.0, this would be the **NotOnOrAfter** condition.

See: http://en.wikipedia.org/wiki/UTC

Digsig:

This is the digital signature of the other two fields. The value of this is the digital signature of the data in the format userid|timeout (the user's id followed by the pipe ('|') character followed by the value of the timeout parameter. The signature is computed using the SHA-1 and RSA algorithms and then encoded using base-64.

Most platforms have implementations for all of these algorithms, so it is unlikely that a custom implementation would be needed. We can provide sample code in most languages that will compute the signature.

The above parameters will be posted to:

https://yoursite.angelpointsevs.com/login.sso

(for clients without a custom domain, replacing *yoursite* with the actual value for your site) OR

https://yourcustomdomain.com/login.sso

(for clients with custom domains, replacing *yourcustomdomain.com* with the actual value for your site)

SHA/RSA Example:

To authenticate the user jdoe123 at 10:22pm EST on January 1st, 2008, the HTML page would be:

```
<html>
    <body    onLoad="document.forms[0].submit();">
        <form method="POST"
        action="https://example.angelpointsevs.com/login.sso">
        <input type="hidden" name="userid" value="jdoe123"/>
        <input type="hidden" name="timeout" value="2008-01-01T15:22:00"/>
        <input type="hidden" name="digsig" value=" NcYINU…dHX+d3m9tQk=">
        </form>
    </body>
</html>
```

Things to note:

- The timestamp has been adjusted from EST to UTC time and has been incremented by 5 minutes. By setting the expiration time to 5 minutes in the future, it decreases the chance that a legitimate request will be deemed expired due to inconsistent system times on the two servers involved in the SSO process.
- The actual value for the digital signature will be significantly larger than the one above. However due to the nature of the algorithms used, the last character of the signature will always be the '=' character.

- The digital signature was performed on the data:
  jdoe123|2008-01-01T15:22:00
- A SAML 2.0 Assertion Response

Below is a SAML 2.0 XML Assertion example.

SAML 2.0 Response XML Example
(Signature Omitted)

```
 <saml2p:Response xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
Destination="http://angelpoints.com/login.sso" ID="responseID"
IssueInstant="2010-01-04T17:31:50.668Z"
Version="2.0"
>
<saml2:Issuer xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"> http://identity-provider.example.com
</saml2:Issuer>
<saml2:Assertion xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion" IssueInstant="2010-01-
04T17:31:50.668Z"
Version="2.0"
>
<saml2:Subject>
<saml2:NameID
Format="urn:oasis:names:tc:SAML:2.0:nameid-format:unspecified"

EXTERNAL_ID
</saml2:NameID>
</saml2:Subject>
<saml2:Conditions
NotBefore="2010-01-04T17:31:50.668Z" NotOnOrAfter="2010-01-04T18:01:50.840Z"
></saml2:Conditions>
<saml2:AuthnStatement AuthnInstant="2010-01-04T17:31:50.668Z">
<saml2:AuthnContext>
<saml2:AuthnContextClassRef> urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
</saml2:AuthnContextClassRef>
</saml2:AuthnContext>
</saml2:AuthnStatement>
</saml2:Assertion>
</saml2p:Response>
```

NOTE: The External ID is used and maps to the external ID in the user import.

2) Error Conditions

There are 6 error conditions that can occur during the process. By default, each error condition results in the user seeing a short error page that is part of the application. However, each of these error conditions can be customized to redirect the user to a custom page hosted by our clients. This allows clients to provide a much more detailed description of what happened and/or what is the next step for the user. It also gives clients the ability to track errors based on the number of times each error page is viewed.

Each error page is configured individually, so there is no need to configure custom pages for every error condition. This customization can be made at any time, so it need not happen during the initial SSO implementation time period.

The possible errors are:

1. *No Such User:*

This error happens when the SSO id does not correspond to an existing id in our database. Accounts must be created prior to the SSO request being made and can be created either a) by a manager using the site, b) through a data feed sent periodically to the application or c) through our Web Services API.

2. *Expired User:*

This error happens when the SSO id corresponds to a user that's marked as expired.

Each of the 3 methods indicated above for creating users can also be used to mark a user as being expired. This is typically done when an employee leaves the company and should no longer have access to the site.

3. *Expired Request:*

This error happens when the timeout timestamp (NotOnOrAfter) is before the current time on the server. This can happen if the server becomes out of synch with the client's SSO server. While the application provides for a small window beyond when the request is initially issued to prevent a request from expiring before it can be sent, some clients add an additional window to the timeout to reduce the chances of this error occurring.

4. *Invalid Request:*

This error happens when the data in the request is invalid. This is the expected error condition when the digital signature verification fails, which can result from someone trying to hack the request to either provide a different user id or timestamp.

5. *Invalid Request Format:*

This error happens when the format of the request is invalid. This can include an improperly-formatted timestamp, a signature that is not properly base-64 encoded or when any of the three parameters are missing. This is generally indicative of a problem in the client's implementation of the SSO process.

6. *Invalid Configuration*

This error happens when there is something that is incorrectly configured on the application side. We are committed to ensuring that this error never happens, but we still account for it and give clients the ability to provide a custom error page.

We can provide the text for the default error messages to help in making the decision on which conditions need custom error pages.

3) Security Concerns

While we consider this solution to be secure, there are some security considerations to be made.

The data (userid and timeout) are not encrypted directly as part of the SSO handshake. While the connection is made over SSL to prevent the data from being intercepted in transit, the entire request is visible to the user that is authenticating. For this reason, requests with timestamps far into the future should be avoided since they would allow a user to re-authenticate without using the SSO portal.

As with any public-key encryption, the private key must be protected from falling into the wrong hands. If the private key is compromised, we encourage you to immediately generate a new one and send us the corresponding X.509 certificate. We can work quickly to update our side of the SSO to use the new key.

For the initial key exchange, we do require that clients generate the key pair themselves and send us the resulting X.509 certificate. We can provide instructions on how to do this, but to ensure that none of our staff are able to craft unapproved login requests, our policy is to avoid ever being exposed to the private key. No employee should ever ask for access to your private key and, if asked, you should refuse.