



SINGLE SIGN ON API USING CONVIO AS MASTER AUTHENTICATOR

Overview

This document describes a Single Sign-On (SSO) protocol with signed URL redirects that uses Convio as the master authenticator. The criteria for using Convio as master are:

- integrating with a third-party CMS that does not have a lot of personalization capabilities
- integrating with a Community or other web application that will cater to a subset of your online users (in which case, the Convio database will be a superset of your online users)

In general, when a new user comes to any of your web applications, you will want to get them registered with Convio for email marketing and will manage the account there. This scenario is appreciably more common than the Convio-as-slave configuration, which is documented in the Single Sign On API material on Convio Open.

This Convio-as-master version of SSO uses the same basic signed redirect design, as was previously documented in [Single Sign-On with Signed URL Redirects](#), but all pieces are implemented in the Convio Open APIs. The salient changes include:

- new methods in the constituent API
- new features for using all client APIs
- coverage of a full set of scenarios around login processing

In the original process, you signed redirect URLs created using the NEXTURL feature of the UserLogin process. The construction of the signature was handled using some esoteric template tags within a PageBuilder page. In the new process.

The API method `CRConsAPI?method=login` is used instead of the UserLogin servlet. Because the APIs now contain native support for generating signatures on redirect URLs, no PageBuilder page is needed for the redirects, which are passed on the API call.

The advantages to this model are:

- completely productized and supported
- partner site controls error processing
- no need for PageBuilder hack
- no visible client-side redirection during login process

- support for logout and forgotten password
- support for testing to see if user is already logged in or has “remember me” cookie
- leverages same basic flow, making it easy to convert existing SSO using signed redirects
- login API is secure and POST-only, keeping passwords confidential
- proper use automatically establishes session cookies on both secure and insecure channels

New Constituent API Methods

The Constituent API methods supported for SSO are:

- login
- logout
- loginTest
- authenticateUser

Login

The login method takes a user_name and password argument and authenticates them against the Convio database. The client version of the API will actually create a logged-in session and return the cons_id to the caller (via the redirect URL).

Login (send_user_name=true)

This variation on the login method takes an email address argument and sends to that address an email containing the user_name and password (if the user has previously registered). If the email address is not registered, the system still sends the email indicating that fact. In this way, it does not allow an unscrupulous user to test to see if email addresses are registered.

Logout

The logout method logs the user out of the Convio system.

LoginTest

The loginTest method is client-only. It returns a cons_id if the user is already logged in or could be logged in because of a “remember me” cookie. The method supports HTTP GET requests so that it can be invoked via a redirect. Note: loginTest should be invoked on an insecure channel since that is the most likely channel for a session already established.

AuthenticateUser

Similar to the login method, authenticateUser uses an organization-specific combination of attributes (such as membership number and zip code) to authenticate the user.

New Features in All Client APIs

The new features in client APIs are identifiable in four basic categories:

1. Signing redirect URLs
2. Automatic redirection via insecure channel
3. Substitution of request params in redirect URLs
4. Substitution of response elements in redirect URLs

These new features apply to all of the APIs and are useful in other contexts. They are, however, essential to the implementation of single signon that is described in this document.

Signing Redirect URLs

All client APIs provide for specifying redirect URLs. As a result, AJAX is not needed to invoke the APIs. Both `success_redirect` and an `error_redirect` options can be specified. Since single signon is inherently a cross-domain activity that requires establishing a connection between the user's browser and the two sites involved in the single signon, it is almost always done by processing redirects between the two sites.

With the `sign_redirects` option, the target can verify that the request originated with the Convio-powered system.

Previously, it was impossible to trust that the page being served in response after an action was completed (e.g. a successful login or a completed donation) was being requested as a byproduct of completing that action or if it was just coming from any other source (e.g. a link or direct entry into the browser address bar).

The URL is signed by adding a timestamp to the query string. Then an MD5 hash of the URL query string plus the `CONVIO_API_SECRET_KEY` are calculated and the result is appended to the query string. The `CONVIO_API_SECRET_KEY` should be a value that is not easily guessed and should never be stored or communicated in clear text. This is in direct contrast to the `CONVIO_API_KEY`, which can be communicated in clear text and is frequently visible in URLs or in the source code view of a web page.

In the next release, a site will be configurable to use SHA-1 encryption for a more secure encryption if desired. This must be handled by Convio Support and is done by setting the Site Data Parameter named `CONVIO_API_ENCRYPTION_TYPE` to the value `SHA_v1`.

The process for verifying the signature involves these steps.

1. Pull that portion of the URL query string after the "?" and up to the argument "&signature=".
2. Append the secret key to that portion of the query string.
3. Generate the appropriate hash (MD5 or SHA-1) for that string.
4. Compare the hash with the value of the "signature" argument.
5. Assuming that the values match, you can be confident that the URL was generated by Convio.
6. Compare the value of the "ts" argument in the URL with the current time from your system in seconds since 1/1/1970. The values should be close enough (within a few seconds provided your system clock is accurate) to ensure that this is not a replay attack.

Automatic Redirection Via Insecure Channel

A common problem with calling the secure API has been that it does not establish a session on the insecure channel. This could result in a constituent being created or logged in by the API, then going to a page on the Convio site, connecting to a different session and not being logged in. That limitation has been overcome.

Now, when a redirect is specified to a non-Convio domain, the API will first check to see if it has already pushed a session cookie on the insecure channel. If not, it will actually generate 2 consecutive redirects to ensure that the session cookie is passed on the insecure channel. This is particularly important in single signon implementations to ensure that links from the partner system connect back to the same session that authenticated the user originally.

Substitution of Request Params in Redirect URLs

Request parameters can be substituted into the redirect URL by specifying `${paramname}` as the value of a query string parameter in the URL. This functionality is useful for redisplaying the data that was entered by the user. It does not allow password, credit_card, and other highly sensitive data to be passed in the URL.

Substitution of Response Elements in Redirect URLs

Any value that would normally be returned in the response XML can be substituted into the redirect URL. The response values use xpath syntax to identify the value from the response XML (for example, `error_code=${errorResponse/code}`). This feature is necessary for error handling, displaying confirmation codes, and retrieving `cons_id` from the return. Because the response is a structured XML document, Xpath syntax for accessing the variables from the response. Typically, the response documents are fairly simple and the path used to access a variable will just be a simple directory path as in the examples.

Usage Scenarios

Six common scenarios are described below. These are:

1. Establishing a Session
2. Accessing Restricted Content
3. Logging In
4. Logging Out
5. Retrieving Username and Password
6. Alternate Authentication

Establishing a Session

When a user comes to the partner system for the first time, the system establishes an initial session. The partner system then wants to check with Convio to determine if the user can be logged in with a "remember me" cookie, or whether the user is already logged in to the Convio system.

The solution in this scenario is to redirect the user to

```
http://[org]/site/CRConsAPI?method=loginTest&api_key=<key>&v=1.0
&sign_redirects=true&success_redirect=<page>
?cons_id=${loginResponse/cons_id}&error_redirect=<page>?cons_id=0
```

Upon the user's return, verify the signature and, if `cons_id != 0`, log in the user. If the `cons_id` on the return is zero, the partner system should treat that user as an anonymous user.

Accessing Restricted Content

When a user who is not logged in makes a request for content available only to registered users, the partner system will want to check with Convio to see either if the user can be logged in with a "remember me" cookie, or if the user has already logged in to the Convio system.

The solution in this scenario is to redirect the user to

```
http://[org]/site/CRConsAPI?method=loginTest&api_key=<key>&v=1.0
&sign_redirects=true&success_redirect=<page>
?cons_id=${loginResponse/cons_id}&error_redirect=login page>
```

Upon the user's return, verify the signature and, if `cons_id != 0`, log the user in. If the user is not already logged in, the redirect loads a login page on the partner system. That login page will then follow the steps for the next scenario.

Logging In

When a user enters a username and password on a partner site's login page, the partner wants to authenticate with Convio and redirect that user to a content page if successful, or back to the login page if the authentication was unsuccessful.

The solution in this scenario is to design a secure form that uses POST to the login method at this URL:

```
https://securex.convio.net/[org]/site/CRConsAPI
```

The form should look something like this:

```
<form method="post"
action="https://securex.convio.net/yoursite/site/CRConsAPI">
  <input type="hidden" name="api_key" id="api_key" value="open" />
  <input type="hidden" name="v" id="v" value="1.0" />
  <input type="hidden" name="method" id="method" value="login" />
  <input type="hidden" name="success_redirect"
value="http://yourothersite.org/login_page.html?cons_id=${loginResponse/co
ns_id}" >
  <input type="hidden" name="error_redirect"
value="http://yourothersite.org/login_page.html?code=${errorResponse/code}
&message=${errorResponse/message}" >
  <input type="hidden" name="sign_redirects" id="sign_redirects"
value="true" />
  <table>
  <tr>
    <td>User Name:</td>
    <td><input name=user_name type=text size=15 maxlength="100"></td>
```

```

    </tr>
    <tr>
        <td>Password:</td>
        <td><input name=password type=password size=15 maxlength="100"></td>
    </tr>
    <tr>
        <td>Remember Me:</td>
        <td><input name=remember_me type=checkbox></td>
    </tr>
</table>

    <input type="submit" value="Submit" name="Submit" />
</form>

```

When the form is submitted, the data will be posted to the API. In the event of a correct username/password combination, a session will be established on Convio and the user's browser will be redirected back to the partner system. The redirect will contain the cons_id of the user and the signature. The partner system should then log that user in and display the content provided that the user is authorized. In the event of a login failure, the user's browser will be redirected back to the login page with an error message in the URL. The login page should then take that error message from the URL and display it on the page to the user.

Logging Out

When a user wants to log out of the partner system, they expect to be logged out of the Convio system at the same time.

The solution in this scenario is to design a secure form that uses POST to this URL:

```
https://securex.convio.net/[org]/site/CRConsAPI
```

The form should look something like this:

```

<form method="post"
action="https://securex.convio.net/yoursite/site/CRConsAPI">
    <input type="hidden" name="api_key" id="api_key" value="open" />
    <input type="hidden" name="v" id="v" value="1.0" />
    <input type="hidden" name="method" id="method" value="logout" />
    <input type="hidden" name="success_redirect"
value="http://yourothersite.org/logout.html" >
    <input type="hidden" name="error_redirect"
value="http://yourothersite.org/logout.html?code=${errorResponse/code}&mes
sage=${errorResponse/message}" >
    <input type="hidden" name="sign_redirects" id="sign_redirects"
value="true" />

    <input type="submit" value="Logout" name="Submit" />
</form>

```

Upon the user's return, partner system should process its logout and then redirect the user to an appropriate page.

Retrieving Username and Password

When a user has lost or forgotten their username and password, they expect to be provided some means to retrieve that information from the system, typically through email.

The solution in this scenario is to design a secure form that uses POST to the login method with the option `send_user_name=true` at this URL:

```
https://securex.convio.net/[org]/site/CRConsAPI
```

The form should look something like this:

```
<form method="post"
action="https://securex.convio.net/yoursite/site/CRConsAPI">
  <input type="hidden" name="api_key" id="api_key" value="open" />
  <input type="hidden" name="v" id="v" value="1.0" />
  <input type="hidden" name="method" id="method" value="login" />
  <input type="hidden" name="send_user_name" id="send_user_name"
value="true" />
  <input type="hidden" name="success_redirect"
value="http://yourothersite.org/login_page.html?user_name_sent=true" >
  <input type="hidden" name="error_redirect"
value="http://yourothersite.org/login_page.html?code=${errorResponse/code}
&message=${errorResponse/message}" >
  <input type="hidden" name="sign_redirects" id="sign_redirects"
value="true" />
  <table>
    <tr>
      <td>Email you registered with:</td>
      <td><input name=email type=text size=30 maxlength="200"></td>
    </tr>
  </table>

  <input type="submit" value="Submit" name="Submit" />
</form>
```

If an unregistered email address is provided, the API does not respond with an error. Instead, it emails that address saying that the user is not registered. This response is intended to prevent attempts to mine constituents.

Alternate Authentication

A user may be recorded in the database without ever having logged in to the system. They have a membership card that uniquely identifies the account information and that is considered relatively secure. In actual usage, it is a best practice to require at least 2 pieces of information from the user to authenticate (e.g. last name and membership number).

The solution in this scenario is to design a secure form that uses POST to the `authenticateUser` method at this URL:

```
https://securex.convio.net/[org]/site/CRConsAPI
```

The form should look something like this:

```
<form method="post"
action="https://securex.convio.net/yoursite/site/CRConsAPI">
  <input type="hidden" name="api_key" id="api_key" value="open" />
  <input type="hidden" name="v" id="v" value="1.0" />
```

```

    <input type="hidden" name="method" id="method"
value="authenticateUser" />
    <input type="hidden" name="success_redirect"
value="http://yourothersite.org/login_page.html?cons_id=${loginResponse/co
ns_id}" >
    <input type="hidden" name="error_redirect"
value="http://yourothersite.org/login_page.html?code=${errorResponse/code}
&message=${errorResponse/message}" >
    <input type="hidden" name="sign_redirects" id="sign_redirects"
value="true" />
    <table>
    <tr>
    <td>Last Name:</td>
    <td><input name=last_name type=text size=15 maxlength="100"></td>
    <tr>
    <td>Membership Number:</td>
    <td><input name=membership_number type=text size=15
maxlength="100"></td>

    </tr>
    </table>

    <input type="submit" value="Submit" name="Submit" />
</form>

```

The Input fields displayed to the user will be based on site-specific configuration in the Constituent Database (used for Authentication flag). Typically, this configuration should be done by Convio Support.

Error Codes

New error codes are associated with the methods discussed in this document.

CODE	CODE NUMBER	DESCRIPTION
LOGIN_MISSING_USER_NAME	200	Missing user name on login request
LOGIN_INVALID_USER_NAME_OR_PASSWORD	201	Missing password on login request
LOGIN_INVALID_USER_NAME_OR_PASSWORD	202	invalid user name or password on login request
LOGIN_UNABLE_TO_EMAIL_USER_NAME	203	unable to email user his user_name
LOGIN_USER_NOT_LOGGED_IN	204	missing user name on login request

Maintaining Logged-in Status on the Convio-Powered System

One issue with any single sign-on system is maintaining the logged-in state on the two systems. Systems typically expire a session after some period of inactivity. On Convio-powered systems, this time limit is 15 minutes. To keep the Convio-powered session alive, the partner web site only needs to include an image tag that references a servlet on the Convio powered site:

```
<img src=http://www.foo.org/site/PixelServer />
```

This will render a 1x1 clear GIF image. It is desirable to only include this once the user has been logged in to the Convio powered system.

Smooth Sailing after Login

Because the session cookie has been pushed to the user's browser, any links to the insecure domain (for example, www.foo.org) will automatically connect to the correct session and, as long as the partner site maintains the logged in status, will operate seamlessly for the user. Links directly to the secure URL (for example, <https://secure2.convio.net/foo>) should not be used from the partner site as it is possible that a session cookie has not been pushed yet. If the partner site links to a page that should be secure, such as a donation form over the insecure domain, the Convio code will automatically redirect and push the secure session cookie at that time.

Access to the Constituent API will need to be done from the server code. The server code will have to use an API administrator username and password on each call and will need to be from an authorized IP address, but other than those standard requirements should be able to use the `cons_id` from the login redirect response to read and update the constituent's profile.