Introduction

Single sign-on (SSO) allows a third party to authenticate a user for the Qualtrics System. The Qualtrics System supports four basic types of SSO authentication. The first method is through the CAS central authentication service and requires that the third-party has a CAS server. The second is LDAP and requires that the third-party has an LDAP server. The third is SAML/Shibboleth and requires that the third-party has a working SAML/Shibboleth Identity Provider implemented. The fourth is a token based system where the third-party generates a secure token that allows the user (if validated) to automatically login.

Implementation Considerations

Qualtrics offers several options for providing users of an organization a Qualtrics account based on their needs and usage. When considering implementation of a SSO solution, the following information is important in determining how SSO is configured for the organization:

- **Qualtrics has never been used by the organization**

  Implementing an SSO solution for a new organization is considerably simpler than for an established organization. Once implemented, users are created automatically upon initial login. If a small number of user accounts existed (for demo or testing purposes), these accounts can be migrated manually by Qualtrics.

- **Qualtrics has been used by the organization prior to SSO integration**

  If user accounts have already been established in Qualtrics for an organization, those accounts should be migrated to use SSO authentication. Existing users have login credentials for Qualtrics that may not match the SSO provider’s credentials. For organizations in which Qualtrics user accounts already exist, Qualtrics provides the option to migrate users through the following process:

  1. The user arrives at the Qualtrics login page (ie. mybrand.qualtrics.com).
  2. The user authenticates using the SSO authentication method configured for the organization. For certain SSO configurations such as CAS, Shibboleth, and SAML, this involves the user being transparently redirected to the organization’s login page.
  3. Once the user has authenticated using SSO, they are redirected back to Qualtrics.
  4. Qualtrics determines based on the user ID passed back through SSO whether the user exists in Qualtrics.
  5. If the user exists, they are logged directly into their account.

     If the user does not exist in Qualtrics they are presented the following options:

     1. Create a new account – this is used by users that have never logged in to Qualtrics
     2. Migrate an existing account – for users that already have a Qualtrics account, their account can be updated to use the SSO configuration. If this option is selected, the user is presented with a form to authenticate using their existing Qualtrics user name and password. Upon successful authentication, the user’s account is migrated and they are logged in to Qualtrics.

     In both cases, subsequent SSO login attempts will not present the user with the migration options. It should be noted that migrated users will not be able to use their old username and password, and can only login through their organization’s SSO.
Handling multiple Qualtrics licenses for groups inside the organization:

If the organization is moving from multiple licenses to a site-wide license, users should be migrated to a single ‘brand’ or account within the Qualtrics system. Either a new brand can be created, or an existing brand can be used. If a new brand will be created, the organization should decide on a new brand ID to identify their organization in the login URL (ie. mybrand.qualtrics.com). If an existing brand will be used, the organization should decide on which brand to use (consider number of users and brand ID). An option is available to allow existing users to migrate directly from one brand to another. This is done on a brand by brand basis, so the organization should determine which brand(s) users can be migrated from. When completing the integration process, Qualtrics can redirect users from one brand’s login page to the new SSO login page to ensure that all users arrive in the correct brand.

SSO & User Types

Most SSO solutions allow the third-party to pass forward multiple attributes for a user to Qualtrics. Among these attributes an organization can specify the user’s user type. Qualtrics determines permissioning within an account based on a user’s type. By mapping a user’s type from an SSO service to Qualtrics, new users can be created with access to a different set of features based on the user type. Additionally, user account creation can be restricted to user types passed from the SSO service. Prior to SSO setup, the third-party should consider the implications of user types within Qualtrics as well as restrictions on users within their organization. If user type mappings will be used, the third-party should configure user types within the Qualtrics Brand Administration panel, as well as determine how the user types should map to the user attributes received from the SSO service. Currently, Qualtrics only supports the specification of user type at the time of account creation (ie. first login).

CAS Introduction

CAS provides enterprise single sign-on service and is supported by the JA-SIG Central Authentication Service. More information about CAS can be found at their website, http://www.ja-sig.org/products/cas/. The Qualtrics System can act as a CAS client allowing the user to authenticate via CAS and login to the Qualtrics system.

CAS Setup Process for a Third-Party

To setup CAS SSO it is assumed that the third-party has a working CAS server using the CAS 2.0 protocol. The Qualtrics System needs to know the following about the CAS server.

1. CAS server hostname
2. CAS server port
3. URI to the CAS system on the host

Once that information is provided to Qualtrics, CAS SSO will be turned on for the organization allowing user to authenticate via CAS and log into the Qualtrics System transparently.

LDAP Introduction

LDAP (Lightweight Directory Access Protocol) is a directory service against which a third party can authenticate. The Qualtrics System can be set up to automatically authenticate against an LDAP server when a user logs in to the Qualtrics System.
LDAP Setup Process for a Third-Party

To setup LDAP integration it is assumed that the third-party has a working LDAPv3 server. The following configuration must also be provided:

1. LDAP Server hostname
2. LDAP Server port
3. Base Distinguished Name (DN) (ex. o=organization)
4. Authentication filter parameters (ex. uid=%username%)

The following configuration parameters are optional:

1. Bind DN (Must use if your LDAP server does not allow unauthenticated search)
2. Bind Password (Must use if your LDAP server does not allow unauthenticated search)
3. Attribute containing the user’s first name (ex. givenname)
4. Attribute containing the user’s last name (ex. sn)
5. Attribute containing the user’s email address (ex. mail)

Once that information is provided to Qualtrics, LDAP integration will be turned on for the organization allowing users to authenticate via LDAP.

SAML/Shibboleth Introduction

Security Assertion Markup Language (SAML) is an XML-based standard for exchanging authentication and authorization data between security domains, that is, between an identity provider (a producer of assertions) and a service provider (a consumer of assertions). The Qualtrics System can be set up to automatically authenticate through a third party identity provider using SAML when a user logs in to the Qualtrics System.

SAML/Shibboleth Setup Process for a Third-Party

To setup SAML integration it is assumed that the third-party has a working SAML 1/2.0 or Shibboleth 1.3/2.0 implementation. The following configuration must also be provided:

1. Identity Provider Metadata in XML
2. Attribute containing the user’s unique username or ID.

The following configuration parameters are optional:

1. Attribute containing the user’s first name (ex. firstname)
2. Attribute containing the user’s last name (ex. sn)
3. Attribute containing the user’s full common name (ex. cn)
4. Attribute containing the user’s email address (ex. mail)

Once the above information is provided, Qualtrics can arrange to set up Shibboleth/SAML integration with the third-party.
**SSO Token Introduction**

Single sign-on using a secure token allows a third party to authenticate a user for the Qualtrics System. From this point on the documentation will refer to the token based SSO implementation as simply SSO. The following diagram shows the overall process to login using SSO.

![Diagram of SSO Process]

Steps for SSO:

1. User authenticates on third-party website using the third-party’s username and password
2. Third-party website generates a secure token only if login is successful with a unique id for each user
3. The third-party website presents a link to log into Qualtrics System with the secure token or can redirect the user to the Qualtrics System. The secure token must be in the URL or posted to the Qualtrics System.
4. Qualtrics System verifies that the secure token is correct and automatically logs in the user.
   a. If the first name, last name, or email is present in the token it will automatically update the user’s information in the Qualtrics System with the new values.
   b. SSO can be setup to create a new user if that user doesn’t already exist in the Qualtrics System.

**SSO Token Setup Process for a Third-Party**

To setup SSO for a third-party there are two phases of deployment. The first phase is the setup phase. The second is the deployment phase.

The setup phase consists of the third-party generating an SSO token and testing it on Qualtrics’ test site. The steps for this phase are as follows:

1. Third-party implements a site that generates an SSO token
   a. See “Example SSO Token Generation” for a site that will generate a token for you, allowing you to compare your implementation.
   b. Example SSO token generation, in several different programming languages, is available at the end of this document.
2. Third-party tests their SSO token against the Qualtrics Servers
   a. See “Testing the SSO Token” on how to test your SSO token and the URL for the test site.
   b. Send the successful SSO Test Results in an email to Qualtrics support to verify that you have successfully generated a token.
The deployment phase consists of a secret key exchange, and Qualtrics setting up and enabling SSO for the third-party organization. The steps for this phase are as follows:

1. Secret key, Encryption and MAC method exchange between Qualtrics and the third-party.
   a. Third-party sends Qualtrics the following:
      i. Encryption Method
      ii. Mac Method
      iii. The Success Code from ssoTest.php
   b. Qualtrics sends the third-party the following:
      i. Secure secret key
   c. Note:
      A secret key has to be established between Qualtrics and the third-party for the encryption of the token. This key is very important and is how Qualtrics identifies the creator of the secure token. It is the equivalent of a password for your entire organization. Please take care to keep this key secret.

2. Qualtrics enables SSO for the third-party organization
   a. Before SSO will work for your organization it must be enabled in the Qualtrics system.

3. Third-party tests the secure token on the Qualtrics production system.
   a. See “Qualtrics System URL” for the url and what data needs to be sent on the url.
   b. Please note: You will no longer pass the encryption method, MAC method, and key on the URL. Those values are only passed to the test site.

4. Qualtrics sets up SSO to meet the specific needs of the third-party
   a. There are multiple configuration options that may be setup by Qualtrics according to the third-party’s need. They include (but are not limited to):
      i. Automatic user creation
         1. Will only create a user automatically if the first name, last name, and email address is present
      ii. Logout redirection
      iii. Login page customization

5. Third-party deploys the SSO authentication on their network for the Qualtrics system.

Once the above steps are completed the third-party will be able to use SSO to login to the Qualtrics system.

---

_qualtrics system url_

The token will be sent to the Qualtrics System URL and will only contain the Brand field and the ssotoken field. No other fields should be present in the URL. Example:

http://new.qualtrics.com/ControlPanel/?Brand=somebrand&ssotoken=2Fnt1G6T1...

Alternatively the Brand field may be omitted if the SSO token is sent to the third-party’s domain. Example:

http://somebrand.qualtrics.com/ControlPanel/?ssotoken=2Fnt1G6T1...
Secure Token

A secure token will be sent to Qualtrics via the URL parameters or post data. The secure token name is ssotoken and the token’s data will be represented as encrypted name/value pairs.

Name/value pairs will be represented as follows (similar to a query string):

name1=value1&name2=value2&name3=value3...

## Secure Token Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Yes</td>
<td>Unique identifier for each and every user</td>
</tr>
<tr>
<td>timestamp</td>
<td>Yes</td>
<td>When the token was created in UTC time. Format: yyyy-mm-ddThh:mm:ss</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: 2008-07-16T15:42:51</td>
</tr>
<tr>
<td>expiration</td>
<td>Yes</td>
<td>When the token will expire in UTC time (same format as the timestamp). Due to security concerns, this should be no later than 1 hour after the timestamp.</td>
</tr>
<tr>
<td>mac</td>
<td>Yes</td>
<td>Message authentication code that accepts the secret key and the token fields (similar to a checksum over the secret key and token fields). Provides message integrity and authenticity. Specifically we are using an HMAC. Should be base64 encoded. Additional information on MAC is available here: <a href="http://en.wikipedia.org/wiki/Message_authentication_code">http://en.wikipedia.org/wiki/Message_authentication_code</a></td>
</tr>
<tr>
<td>firstname</td>
<td>No</td>
<td>The user’s first name. Used to auto-update the user’s data in the Qualtrics System.</td>
</tr>
<tr>
<td>lastname</td>
<td>No</td>
<td>The user’s last name. Used to auto-update the user’s data in the Qualtrics System.</td>
</tr>
<tr>
<td>email</td>
<td>No</td>
<td>The user’s email. Used to auto-update the user’s data in the Qualtrics System.</td>
</tr>
</tbody>
</table>

## Token Encryption Methods

The token’s data should be encrypted with one of the following algorithms:

- AES 128 / Rijndael 128 (ECB)
- BLOWFISH (ECB)
- TRIPLE DES (ECB)

## MAC Methods

The MAC will be an HMAC (keyed-hash message authentication code) using one of the following hash algorithms:

- sha512
- sha256
- sha1
- md5

HMAC functions are available in most languages. Additional information on HMAC and how they are computed is available at: http://en.wikipedia.org/wiki/HMAC
**Token Algorithm and Encoding**

The general algorithm is as follows:

```javascript
// The following functions are used in the example,
// but not part of the algorithm itself.
function base64Encode(string) {
    return encodedString;
}

function encrypt(token, key) {
    return encryptedMessage;
}
```

```javascript
// The general algorithm is as follows:

token = keyValuePairs + "&mac=" + base64Encode(hmac(keyValuePairs, key));
encToken = base64Encode(encrypt(token, key));
```

Once the encrypted token has been created, it should be **base64 encoded** and sent as the value for the ssotoken name.

---

**Example SSO Token Generation**

To help develop code to generate the SSO token, a token generation website is available. This website creates an SSO token from scratch and shows the token at each step of the algorithm allowing you to compare your token to a valid token.


Please note there are several different padding techniques that can be used when encrypting the token. Qualtrics will successfully decode the token regardless of the technique used. The example SSO Token page uses a null padding which may differ from other encryption libraries.

---

**Testing the SSO Token**

To ensure that the encrypted token is being generated correctly and that it can be decrypted by Qualtrics an SSO test site is available. The test site will attempt to decrypt the token and will show you the decrypted values or report any errors. The test site URL is:


At a minimum the query parameter ssotoken will need to be sent on the URL or posted. In addition to the token you can tell the test page what key, encryption method, and MAC method to use. The following table shows what query string parameters you can pass to the test page. The default values for each parameter is in bold.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssotoken</td>
<td>Yes</td>
<td>The encrypted token</td>
</tr>
<tr>
<td>key</td>
<td>No</td>
<td>The key to use in the decryption, the default key is <strong>generickey</strong></td>
</tr>
<tr>
<td>enc</td>
<td>No</td>
<td>The encryption/decryption method. The following are available: <strong>rijndael-128</strong>, <strong>blowfish</strong>, <strong>tripledes</strong></td>
</tr>
<tr>
<td>mac</td>
<td>No</td>
<td>The MAC method. The following are available: <strong>md5</strong>, <strong>sha1</strong>, <strong>sha256</strong>, <strong>sha512</strong></td>
</tr>
</tbody>
</table>
The following example would test the token with the key "ssoexamplekey123", the triple des encryption algorithm and a md5 MAC.

http://new.qualtrics.com/ControlPanel/ssoTest.php?key=ssoexample123&enc=tripledes&mac=md5&ssotoken=...

Parameter ssotoken would have the appropriately encrypted token as its value.
The following PHP code will produce a secure token that could be sent on the URL, posted to the Qualtrics System, or used in a redirect.

```php
//An example of how to generate a secure token for Qualtrics SSO

$brand = 'somebrand';

//the data to be put in the token
$tokenData = array(
    'id' => 'jeffw',
    'timestamp' => '2008-07-16T15:42:51',
    'expiration' => '2008-07-16T15:47:51',
    'firstname' => 'Jeff',
    'lastname' => 'Whiting',
    'email' => 'jeffw@qualtrics.com',
);

//generate the query parameters
$token = ''; $i = 0; foreach($tokenData as $key => $value)
{
    if ($value != '')
    {
        if ($i != 0)
            $token .= '&';
        $token .= "$key=$value";
        $i++; 
    }
}

//generate the secure token
$secret = 'SOMESECRETSHAREDKEY';
$encMethod = 'rijndael-128';
$macMethod = 'md5';

//generate the hmac
$hash = base64_encode( hash_hmac($macMethod, $token, $secret, true) );
$token .= '&mac='.$hash;

//encrypt the token
$ivSize = mcrypt_get_iv_size($encMethod, MCRYPT_MODE_ECB);
$iv = mcrypt_create_iv($ivSize, MCRYPT_RAND);
$encData = base64 Encode( mcrypt_encrypt( $encMethod, $secret, $token, MCRYPT_MODE_ECB, $iv));

//generate the link
$Brand = 'somebrand';
$ssotoken = $encData;

echo "SSO link and url<br><br>";
echo "link: <a href='http://new.qualtrics.com/CP/$query'>Secure Login Link</a><br><br>";
echo "url: http://new.qualtrics.com/CP/$query<br>";
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace SSO_example
{
    ///
    /// SSO Token class
    /// This class encapsulates the query parameters used to create the SSO Token
    ///
    /// class SSO_Token
    ///
    /// public Dictionary<string, string> parameters;
    /// public SSO_Token(Dictionary<string, string> parameters)
    /// {
    ///     this.parameters = parameters;
    /// }
    /// /**
    ///  * Overrides the ToString method to generate a query from the key value pairs
    ///  * @returns the token data as a string
    ///  */
    /// public override string ToString()
    /// {
    ///     string token = "";
    ///     int i = 0;
    ///     foreach(KeyValuePair<string, string> kvp in this.parameters)
    ///     {
    ///         if (i != 0)
    ///             token += ";";
    ///         token += kvp.Key+"=\"\"+kvp.Value;
    ///         i++;
    ///     }
    ///     return token;
    /// }
    /// /**
    ///  * @return byte[] The token data as a byte array
    ///  */
    /// public byte[] toByteArray()
    /// {
    ///     System.Text.ASCIIEncoding encoding = new System.Text.ASCIIEncoding();
    ///     return encoding.GetBytes(this.ToString());
    /// }
    /// }
}

class SSO
{
    string brand;
    string encData;
    SSO_Token ssotoken;
    ///
    /// SSO Constructor
    /// The constructor generates the secure token using the input parameters
    /// @param string brand The brand id
    /// @param string id The user id
    /// @param string timestamp The token timestamp
    /// @param string expiration The token expiration
    /// @param string firstname The user's first name
    /// @param string lastname The user's last name
    /// @param string email The user's email
    ///
}
* @param string key The brand’s secret key
*/
public SSO(string brand, string id, string timestamp, string expiration,
string firstname, string lastname, string email, string key)
{
    //This data structure contains all of the necessary parameters for the
token to be generated
Dictionary<string, string> sso_parameters = new Dictionary<string,
string>();
sso_parameters.Add("id", id);
sso_parameters.Add("timestamp", timestamp);
sso_parameters.Add("expiration", expiration);
if (firstname != null && firstname !="")
    sso_parameters.Add("firstname", firstname);
if (lastname != null && lastname !="")
    sso_parameters.Add("lastname", lastname);
if (email != null && email !="")
    sso_parameters.Add("email", email);
this.ssotoken = new SSO_Token(sso_parameters);
System.Text.ASCIIEncoding keyEncoding = new System.Text.ASCIIEncoding();
this.encData = generateSecureToken(this.ssotoken.toByteArray(),
keyEncoding.GetBytes(key));
this.brand = brand;
}
/**
 * This function generates the secure token string
 * @param byte[] token The unencrypted token converted to a byte array
 * @param byte[] key The brand’s secret key converted to a byte array
 * @return string The base64 encoded secure token
 */
private string generateSecureToken(byte[] token, byte[] key)
{
    //generate the HMAC (this implementation uses SHA1)
    HMACSHA1 hmac = new HMACSHA1(key);
    string hash = Convert.ToBase64String(hmac.ComputeHash(token));
    //the HMAC is converted to a byte array and appended to the existing
    //token
    System.Text.ASCIIEncoding tokenEncoding = new System.Text.ASCIIEncoding();
    byte[] tokenSecondHalf = tokenEncoding.GetBytes(newTokenString);
    byte[] newTokenByteArray = new byte[token.Length +
tokenSecondHalf.Length];
token.CopyTo(newTokenByteArray, 0);
tokenSecondHalf.CopyTo(newTokenByteArray, token.Length);
    //Generate the secure token (this implementation uses 3DES)
    TripleDESCryptoServiceProvider tdes = new
    TripleDESCryptoServiceProvider();
    //the key must be a valid length so we pad it until it is (it checks
    //number of bits)
    while (key.Length * 8 < tdes.KeySize)
    {
        byte[] tmp = new byte[key.Length + 1];
        key.CopyTo(tmp, 0);
        tmp[tmp.Length - 1] = (byte) '\0';
        key = tmp;
    }
tdes.Key = key;
tdes.Mode = CipherMode.ECB;
tdes.Padding = PaddingMode.Zeros;
ICryptoTransform ict = tdes.CreateEncryptor();
byte[] result = ict.TransformFinalBlock(newTokenByteArray, 0,
newTokenByteArray.Length); //convert the secure token to a Base64 string to return return Convert.ToBase64String(result);
}/**
 * Generate the query string using the brand and secure token
 * @return string The query string that includes the sso token
 */
private string generateQuery()
{
   return "?Brand=" + this.brand + "&ssotoken=" + this.encData;
}
/**
 * Generates the full URL for the SSO
 * @return string Full SSO URL
 */
public string generateURL()
{
   return "http://new.qualtrics.com/CP/" + this.generateQuery();
}
}

An Example of the SSO Class Usage

private string generateURL()
{
   string brand = "somebrand";
   string id = "userID";
   string timestamp = "2008-07-16T15:42:51";
   string expiration = "2008-07-16T15:47:51";
   string firstname = "John";
   string lastname = "Doe";
   string email = "johndoe@example.com";
   string key = "SOMESECRETSHAREDKEY";
   SSO sso = new SSO(brand, id, timestamp, expiration, firstname, lastname, email, key);
   string url = sso.generateURL();
   return url;
}
Perl Example

#!/usr/bin/perl -w
use strict;

use Digest::HMAC;
use Digest::SHA1;
use Crypt::ECB;
use MIME::Base64;

my $key = 'SOMESECRETKEY111'; # Must be 16 bytes for Rijndael ECB
my $brand = 'examplebrand';

my %tokendata = (
  'id' => 'jeffw',
  'timestamp' => '2008-07-16T15:42:51',
  'expiration' => '2008-07-16T15:47:51',
  'firstname' => 'Jeff',
  'lastname' => 'Whiting',
  'email' => 'jeffw@qualtrics.com'
);

my $hashidx;
my $token = '';
my $i = 0;
foreach $hashidx (keys %tokendata) {
  if ($i > 0) {
    $token .= '&' . $hashidx . '=' . $tokendata{$hashidx};
  } else {
    $token = $hashidx . '=' . $tokendata{$hashidx};
  }
  $i++;
}

my $hmac = Digest::HMAC->new($key, "Digest::SHA1");
$hmac->add($token);
$token .= '&mac=' . $hmac->b64digest;

my $cipher = Crypt::ECB->new;
$cipher->padding(PADDING_AUTO);
$cipher->cipher('Rijndael');
$cipher->key($key);

my $enctoken = $cipher->encrypt($token);

my $query = '?Brand=' . $brand . '&ssotoken=' . encode_base64($enctoken, '');

print "SSO URL: \n";
print "http://new.qualtrics.com/CP/\${query}\n";