

BLUEVOLT USER MANAGEMENT AND SINGLE SIGN-ON API OPTIONS

V5.4 11/04/2019




Table of Contents

| | |
|--|----|
| Introduction to the User Management and SSO API Options | 2 |
| How to get started: | 2 |
| User Management | 2 |
| Synchronous User Management..... | 2 |
| Asynchronous User Management..... | 5 |
| Single Sign-On - SAML | 7 |
| Process Overview | 7 |
| IdP-Initiated SSO: Post Binding..... | 7 |
| SAML URI's | 8 |
| Fields | 8 |
| Conditions | 8 |
| Certificate..... | 9 |
| Identity Provider Implementation..... | 9 |
| Saml References..... | 10 |
| SAML Single Sign-On Deep Linking..... | 10 |
| Setting Up SAML SSO..... | 10 |
| OPTION: SAML Single Sign-On with Provisioning | 11 |
| SAML Examples | 12 |
| Single Sign-On – OAUTH 2..... | 15 |
| OAUTH Roles | 15 |
| Tokens..... | 15 |
| Redirect URIs..... | 15 |
| Client ID and Secret..... | 15 |
| Things to provide to BlueVolt:..... | 16 |
| Overview of BlueVolt’s APIs and Integration Applications | 20 |
| User Management | 20 |
| Report Management:..... | 20 |
| Developer’s FAQ..... | 21 |

Introduction to the User Management and SSO API Options

The BlueVolt User Management and SSO API options makes functionality of the BlueVolt LMS available to your organization by exposing features of the BlueVolt platform via webservice.

The **user management** portion of the API allows you to maintain consistent user data that syncs with an existing system, such as an HRIS or CRM, which contains information about user *authorization*. The user management piece includes 3 end points for updating user data (Add/Sync/Delete). The user management piece allows you to control an individual's group roles and sends welcome emails. Data updates occur when the user is update in the originating system.

The **SSO component** allows for user *authentication* (login) from an existing system. Users are automatically logged in to the LMS, and do not have to login or create an account in the LMS to gain access to training.

The following chart illustrates the above options:

| Feature | SAML SSO (Authentication only) | SAML SSO with Provisioning | OAUTH SSO (Authentication only) | OAUTH SSO with Provisioning | User Management API |
|--|--------------------------------|----------------------------|---------------------------------|-----------------------------|---------------------|
| User Authentication | x | X | X | X | |
| Create / Update User Profile Information | | X | | X | X |
| Update User Group(s) Membership and Role | | X | | X | X |
| Create New Groups | | X | | X | X |
| Inactivate User | | | | | X |

How to get started:

Contact your account representative and a technical meeting will be scheduled with technical resources from both parties. After the technical discussion, API keys will be sent to you and a test environment in BlueVolt will be set up for you to test the integration(s).

Synchronous User Management

Users are managed synchronously through the API by calling one of three endpoints: AddUsers, SyncUsers, or Remove Users.

Adding/Updating Users

URL: [https://go.bluevolt.com/api/v1/UserSyncApi.svc/\[xml/\]AddUsers](https://go.bluevolt.com/api/v1/UserSyncApi.svc/[xml/]AddUsers)

Using the AddUsers endpoint allows you to add or update one or more users at a time by passing in data for just those users, with no need to know about the university's other users. See Payload for more information.

Adding new users or updating existing ones is fairly straightforward. Let's assume that a university exists that is already populated with 100 users. To add/update a user, we simply POST a JSON object of the following format to the AddUsers endpoint:

Example JSON:

```
{
  "ApiKey" : " Your API Key ",
  "GroupRolesType" : 2,
  "Profiles" : [{
    "Address1" : "123 nw burnside",
    "Address2" : "",
    "City" : "portland",
    "CompanyName" : null,
    "Country" : "US",
    "CourseRoleList" : [],
    "CustomFieldList" : [{
      "Value" : "",
      "name" : "Branch Number"
    }, {
      "Value" : "ABC Corporation",
      "name" : "Company Name"
    }, {
      "Value" : "",
      "name" : "Custom Display Information"
    }, {
      "Value" : "Supervisor",
      "name" : "Job Title"
    }, {
      "Value" : "North",
      "name" : "Region"
    }, {
      "Value" : "10\11\2012",
      "name" : "Date Hired"
    }, {
      "Value" : "",
      "name" : "Region 1"
    }, {
      "Value" : "",
      "name" : "Branch City"
    }
  ],
  "Email" : "johndoe@gmail.com",
  "EmployeeCode" : null,
  "FirstName" : "John",
  "GroupRoleList": [
    { "Path": "Branches|Oregon",
      "Role": [1]
    }, {
      "Path": "By Job Title|IT",
      "Role": [0,1],
      "Primary": true,
      "PrimarySpecified": true,
      "Hidden": true,
      "HiddenSpecified": true
    }
  ],
  "JobTitle" : null,
  "LastName" : "Smith",
  "Password" : "jWJA&H54#%GA",
  "State" : "OR",
  "UserName" : "johndoe",
  "WorkPhone" : null,
  "Zip" : "97229"
},
"SendCollisionEmails" : false,
"SendWelcomeEmails" : false,
```

Example XML:

```
<ApiUserSync_Payload xmlns="http://schemas.datacontract.org/2004/07/www.Api.v1"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
  <ApiKey>Your API Key</ApiKey>
  <GroupRolesType>GroupRoleList</GroupRolesType>
  <Profiles xmlns:a="http://schemas.datacontract.org/2004/07/Business.SingleSignOn">
    <a:BlueVoltSingleSignOnResponseProfile>
      <a:Address1 i:nil="true"/>
      <a:Address2 i:nil="true"/>
      <a:City i:nil="true"/>
      <a:CompanyName i:nil="true"/>
      <a:Country i:nil="true"/>
      <a:CourseRoleList i:nil="true"/>
      <a:CustomFieldList>
        <a:BlueVoltSingleSignOnResponseProfileCustomField>
          <a:Value>ACME Supply Group</a:Value>
          <a:name>Company Name</a:name>
        </a:BlueVoltSingleSignOnResponseProfileCustomField>
        <a:BlueVoltSingleSignOnResponseProfileCustomField>
          <a:Value>Training Manager</a:Value>
          <a:name>Job Title</a:name>
        </a:BlueVoltSingleSignOnResponseProfileCustomField>
      </a:CustomFieldList>
      <a:Email>jdoe@example.com</a:Email>
      <a:EmployeeCode i:nil="true"/>
      <a:FirstName>Jon</a:FirstName>
      <a:GroupRoleList>
        <a:BlueVoltSingleSignOnResponseProfileGroup>
          <a:Hidden>>false</a:Hidden>
          <a:HiddenSpecified>>true</a:HiddenSpecified>
          <a:Path>Branches|Oregon</a:Path>
          <a:Primary>>true</a:Primary>
          <a:PrimarySpecified>>true</a:PrimarySpecified>
          <a:Role>
            <a:groupRole>Member</a:groupRole>
            <a:groupRole>ReportViewer</a:groupRole>
          </a:Role>
        </a:BlueVoltSingleSignOnResponseProfileGroup>
        <a:BlueVoltSingleSignOnResponseProfileGroup>
          <a:Hidden>>false</a:Hidden>
          <a:HiddenSpecified>>true</a:HiddenSpecified>
          <a:Path>By Job Title|IT</a:Path>
          <a:Primary>>false</a:Primary>
          <a:PrimarySpecified>>true</a:PrimarySpecified>
          <a:Role>
            <a:groupRole>Member</a:groupRole>
          </a:Role>
        </a:BlueVoltSingleSignOnResponseProfileGroup>
      </a:GroupRoleList>
      <a:JobTitle i:nil="true"/>
      <a:LastName>Doe</a:LastName>
      <a:State i:nil="true"/>
      <a:UserName>jDoe</a:UserName>
      <a>Password>jWJA&H54#%GA</a>Password>
      <a:WorkPhone i:nil="true"/>
      <a:Zip i:nil="true"/>
    </a:BlueVoltSingleSignOnResponseProfile>
  </Profiles>
  <SendCollisionEmails>>true</SendCollisionEmails>
  <SendWelcomeEmails>>true</SendWelcomeEmails>
</ApiUserSync_Payload>
```

Syncing Users

URL: [https://go.bluevolt.com/api/v1/UserSyncApi.svc/\[xml/\]SyncUsers](https://go.bluevolt.com/api/v1/UserSyncApi.svc/[xml/]SyncUsers)

Using the SyncUsers endpoint is similar to using the AddUsers endpoint, except that it requires you to know about ALL of the existing users in the university; to use it, you must pass in data representing all of the university members that currently exist, **minus** any users to be removed (and **plus** users to be added, but this is the same as with AddUsers). The API will then sync the university's users with your data. To illustrate: let's assume a university is populated with 100 users. If you POST 99 of those users' data in the correct format to the SyncUsers endpoint, the 100th user will be removed from your university (the one who is **not** in the incoming payload).

Note: There is a **Sync Remove Lock** feature on the Edit User page in the admin area of your university that allows you to prevent a user from being removed from a university via the SyncUsers integration.

Removing Users

URL: [https://go.bluevolt.com/api/v1/UserSyncApi.svc/\[xml/\]RemoveUser](https://go.bluevolt.com/api/v1/UserSyncApi.svc/[xml/]RemoveUser)

To remove a user, POST a JSON object to the RemoveUser endpoint that contains that user's information including their email address. If successful, the API will return the user's information both in the usersRemoved and also userStatusRows portions of the response (see Response below).

```
{"ApiKey":"5b1d975d-effa-4df0-ba44-0678b21ba4d0","Profiles":[{"Email":"tony.asche@bluevolt.com"}]}
```

Response

The API will return a JSON or XML response to let you know the results of its operations. This will essentially consist of a status code indicating success or failure for the entire batch, a list of all user rows provided with a result string describing the outcome for each user, and also a list of users who were removed.

Example response (JSON):

Below is an example of a return payload after a successful addition of one user.

```
{
  "statusCode" : "Success"
  "userStatusRows" : [
    {
      "UserRow" : "0"
      "UserResult" : "successfully added"
    }
  ]
  "usersRemoved" : [{}]
```

Asynchronous User Management

FTP or SFTP the Payload to our server. The location will be <https://api.go.bluevolt.com> or <sftp://api.go.bluevolt.com> & we will supply username & password for customers requesting this feature.

The asynchronous user management API accepts CSV or XML files. Refer to the Synchronous User Management above for an example XML file. CSV files require the following information:

CSV Required Columns

- FirstName
- LastName
- EmailAddress

CSV Optional Columns

- EmployeeCode
- UserName
- Address1
- Address2
- City
- StateProvince (For countries United States and Canada, you must enter the 2-character state/province/territory code or name matching the United Nations/ISO 3166 standard)
- Country (Leave field blank if "United States". For other countries, you must enter the 2-character country code or name matching the United Nations/ISO 3166 standard)
- Zip
- Custom Fields (The column name for each custom field should match the name of your fields)
- GroupMemberList (Use the tilde character "~" to separate individual groups for each group path. Use the pipe character "|" to separate individual group paths.)
- GroupAdminList (Use the tilde character "~" to separate individual groups for each group path. Use the pipe character "|" to separate individual group paths.)
- ReportViewerList (Use the tilde character "~" to separate individual groups for each group path. Use the pipe character "|" to separate individual group paths.)

Example CSV (spreadsheet view):

| | A | B | C | D | E | F | G | H | I |
|---|-----------|-------------|-----------------------------|-----------------------------|----------------------|-----------------------|----------------------|----------------|----------------|
| 1 | FirstName | LastName | EmailAddress | UserName | GroupMemberList | GroupAdminList | ReportViewerList | Custom Field A | Custom Field B |
| 2 | Mercie | Signore | msignore0@zimbio.com | msignore0@zimbio.com | Root Group~Warehouse | Root Group~Management | Root Group~Reporting | Female | 195.109.49.208 |
| 3 | Karlene | Leaming | kleaming1@fda.gov | kleaming1@fda.gov | Root Group~Warehouse | Root Group~Management | | Female | 52.229.208.234 |
| 4 | Calida | Frowen | cfrowen2@qq.com | cfrowen2@qq.com | Root Group~Warehouse | | | Female | 104.184.1.97 |
| 5 | Florrie | Kinman | fkinman3@4shared.com | fkinman3@4shared.com | Root Group~Warehouse | Root Group~Management | | Female | 38.251.249.209 |
| 6 | Horace | Veracrussse | hveracrussse4@123-reg.co.uk | hveracrussse4@123-reg.co.uk | Root Group~Warehouse | Root Group~Management | Root Group~Reporting | Male | 52.98.184.73 |

Single Sign-On - SAML

The Security Assertion Markup Language (SAML) standard defines a framework for exchanging security information between online business partners. The LMS implements version 2.0 of SAML

Process Overview

SAML defines a number of profiles to describe and constrain the use of SAML protocol messages and assertions. This describes the expected implementation for BlueVolt LMS partners.

IdP-Initiated SSO: Post Binding

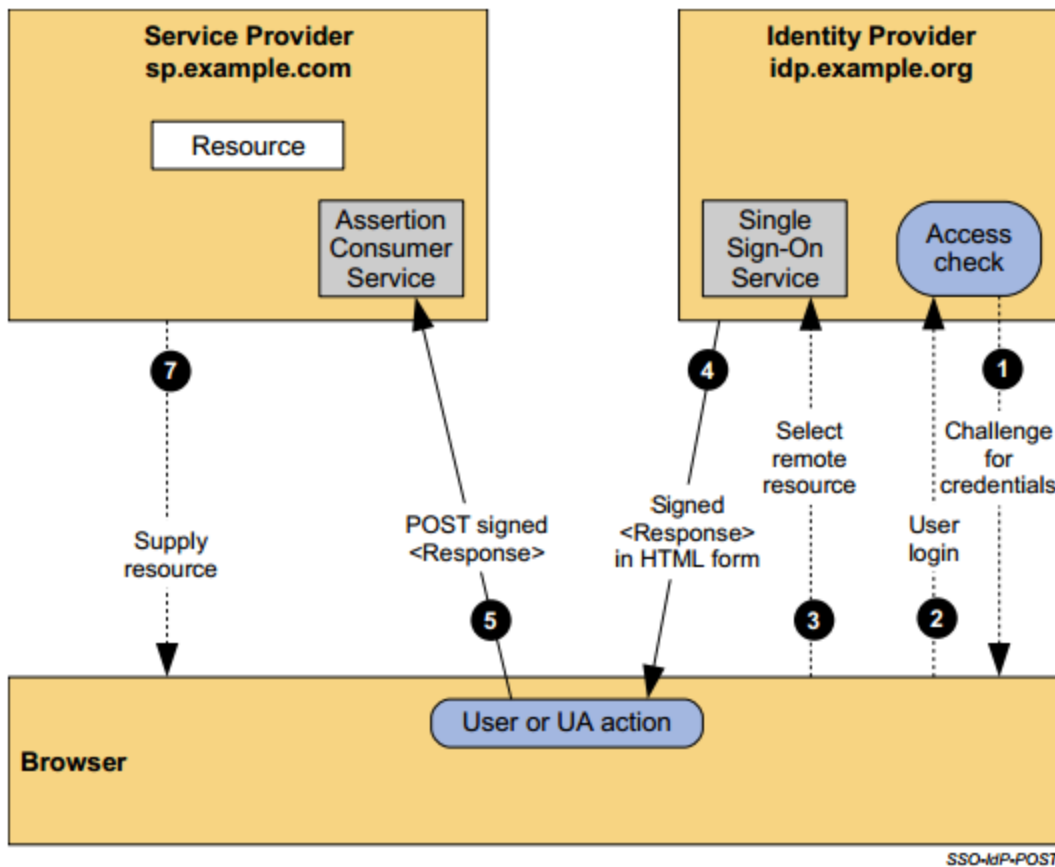


Figure 14: IdP-Initiated SSO with POST Binding

The processing is as follows:

1. If the user does not have a valid local security context at the IdP, at some point the user will be challenged to supply their credentials to the IdP site, idp.example.org.
2. The user provides valid credentials and a local logon security context is created for the user at the IdP.
3. The user selects a menu option or link on the IdP to request access to an SP web site, sp.example.com. This causes the IdP's Single Sign-On Service to be called.
4. The Single Sign-On Service builds a SAML assertion representing the user's logon security context. Since a POST binding is going to be used, the assertion is digitally signed before it is placed within a SAML <Response> message. The <Response> message is then placed within an HTML FORM as a hidden form control named SAMLResponse. (If the convention for identifying a specific application resource at the SP is supported at the IdP and SP, the resource URL at the

SP is also encoded into the form using a hidden form control named RelayState.) The Single Sign-On Service sends the HTML form back to the browser in the HTTP response. For ease-of-use purposes, the HTML FORM typically will contain script code that will automatically post the form to the destination site.

5. The browser, due either to a user action or execution of an “auto-submit” script, issues an HTTP POST request to send the form to the SP's Assertion Consumer Service. The service provider's Assertion Consumer Service obtains the <Response> message from the HTML FORM for processing. The digital signature on the SAML assertion must first be validated and then the assertion contents are processed in order to create a local logon security context for the user at the SP. Once this completes, the SP retrieves the RelayState data (if any) to determine the desired application resource URL and sends an HTTP redirect response to the browser directing it to access the requested resource (not shown).

6. An access check is made to establish whether the user has the correct authorization to access the resource. If the access check passes, the resource is then returned to the browser.

SAML URI's

The Assertion Consumer Service endpoint URI <https://go.bluevolt.com/SamlSSO/Login>

This endpoint expects an SAML Response to be Base64 Encoded in a HTML Form Field called samlResponse.

Fields

The Unique Identifier that identifies the user in the IdP should be included in the Assertions Subjects NameID element. This identifier **should be unique and should not ever change** at the IdP, and is stored in the LMS as the “Single Sign On Username”.

NOTE: It is strongly recommended that you do not use email addresses or combination of firstname, lastname as the unique identifier. Also, recycling of a unique identifier from a previously deleted account is not recommended.

```
<saml:Subject>
  <saml:NameID NameQualifier="http://go.bluevolt.com">f1079b5a-4ba0-4085-b996-
1ece20c2b323</saml:NameID>
  <saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
    <saml:SubjectConfirmationData NotOnOrAfter="2013-09-10T17:27:41.9926060Z"
Recipient="go.bluevolt.com/SamlSSO/Login" />
  </saml:SubjectConfirmation>
</saml:Subject>
```

The Assertions <Issuer> Element will be used to identify which LMS university the user is signing into. The Issuer should be a URI that Identifies the IdP. This URI should be provided to BlueVolt during the setup of the LMS University when implementing SAML SSO.

Conditions

The conditions are checked as a security measure. Values for NotBefore and NotOnOrAfter attributes of SSO assertions SHOULD have the shortest possible validity period consistent with successful communication of the assertion from Identity Provider to Service Provider site. This is typically on the order of a few minutes. This ensures that a stolen assertion can only be used successfully within a small time window.

NotBefore: The time that the assertion can start being used.

NotOnOrAfter: The time that the assertion can no longer be used.

Audience: Should always be <https://go.bluevolt.com/SamlSSO/Login> in production or <https://beta.bluevolt.com/SamlSSO/Login> during testing.

Certificate

The SAML Response should be digitally signed with X.509 certificate and include the XML Signature. The signature should include the certificate so that signature can be verified by the LMS. The Thumbprint of the certificate should be provided to BlueVolt during the setup of the LMS University when implementing SAML SSO.

Identity Provider Implementation

If Identity provider portal is implemented in C#, ASP.net then this code sample would be helpful.

The following is a sample call from Idp portal to Service Provider. Please see the parameter comments for reference.

```
samlResponse.Value =  
SamlHelper.GetPostSamlResponse(  
    "https://go.bluevolt.com/SamlSSO/Login",  
    "myportal.com", //this should be Identity Provider portal URL  
    "go.bluevolt.com", // SAML SSO URL for Bluevolt.  
    "localuserid", //you can use some userid for your portal though not necessary  
    StoreLocation.LocalMachine, StoreName.Root, X509FindType.FindByThumbprint, @"<Path of .pfx  
file for your certificate>", "<Certificate password>",  
    "<Thumbprint>", attrs);  
//Set Form Action  
  
this.frmSSO.Action = "https://go.bluevolt.com/SamlSSO/Login";
```

The following is a sample web form to post the data to Service provider:

```
<form id="frmSSO" runat="server" enableviewstate="False">  
    <center></center>  
    <center><asp:Label ID="lblMessage" runat="server"  
        Text="Redirecting to external site.." EnableViewState="False"></asp:Label>  
        <br />  
    </center>  
    <div style="display:none" >  
        <input id="samlResponse" type="text" runat="server" enableviewstate="False"/>  
        <input id="RelayState" type="text" runat="server" enableviewstate="False"/>  
    </div>  
</form>
```

More ASP.Net and C# sample code is available. Please ask account representative if required.

Saml References

Saml Tech overview: <https://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-overview-2.0-cd-02.pdf>

SAML Single Sign-On Deep Linking

When the LMS system sends out email the links in the emails will be to the SingleSignOn Sign in URI with a Query Parameter of “RelayState” with value of an encoded URL where the browser should be redirected back to after the user has signed in. So, if you give BlueVolt a Sign In URI of *http://www.example.com/SSOSignIn* and you create an Enrollment Email in the LMS that uses the [[CourseLink]] token, then the email that gets sent to the user will have a link that looks like:

http://www.example.com/SSOSignIn?RelayState=https%3a%2f%2fgo.bluevolt.com%2ftestSSO%2fCourseDetail%2fCourseName%2f20

The user clicks the link and should open the URL, after the user signs into the IdP system they should be redirected to:

<https://go.bluevolt.com/testSSO/CourseDetail/CourseName/20>

Setting Up SAML SSO

The attached code sample is written in [asp.net](#) and C#. For Identity provider, we use [asp.net](#) out of the box and then add the Identity provider specific controller and its methods. There are two projects in this solution:

1. Saml2IdPSample – Identity provider
2. Saml2 – Code Library for generating the Assertion

To use this sample, the changes should be made to Controllers/SamlIdPServiceController.cs:

- Change SiteHostName to whatever your domain is.
- Create your own certificate and use it rather than the included sample Certificate.

You can store it on the file system, or put it in the computer certificate store.

You also need to send to BlueVolt the information for:

- Assertion Issuer: `_issuer` in attached sample
- Certificate Thumbprint
- The identity provider URL (the SiteHostName in attached sample)

This information is used for validating the Assertions that are sent to us.

OPTION: SAML Single Sign-On with Provisioning

Provisioning can be accomplished using the User Management API (recommended option) or through the SAML Single Sign on with Provisioning option (this option requires the learner to login via SSO to create or update their account whereas the User Management API keeps accounts in sync without requiring the learner to login).

During the SSO process provisioned data will need to be sent in the SAML AttributeStatement. If the user account is not provisioned before a Single Sign-On response by the API, then an account can be created in the LMS University, based on the information provided within the SAML AttributeStatement.

Information provided in the SAML AttributeStatement should contain university required fields and optional fields.

- Conditionally Required Fields (provider may opt to allow these fields to be provided by the user if provider does not maintain full user profiles)
 - FirstName
 - LastName
 - Email
- Optional Fields
 - Address1
 - Address2
 - City
 - State
 - Zip
 - WorkPhone
 - EmployeeCode (An identifier that you supply that is unique to each employee or student)
 - Country (A 2 letter code designating the employee country, as defined in ISO 3166-1 alpha-2, our system is synchronized with [<http://www.iso.org/iso/list-en1-semic-3.txt>])
 - GroupRoleList (see below)
 - CustomFieldList (see below)
 - EnrollmentList (see below)

Group Role List Section

The group role list section is optional, and allows you to automatically control a user's roles in single-sign-on-controlled groups. A group is single-sign-on-controlled if it was created via the group role list section. Single-sign-on-controlled groups, and their roles, are read-only in the BlueVolt admin. You administer them exclusively through single-sign-on.

If provided, the group role list specifies the set of group roles the user will have after they log in. During login, we synchronize the user's roles to match the provided list. We will both add and remove roles as necessary to bring them up to date with the list. Therefore, you must always send the full list of all roles in computer-controlled groups for this user.

The login process creates and deletes single-sign-on-controlled groups as needed. When the first user logs in with a role in a new group, we will create that as a single-sign-on-controlled group. When the last person is removed from a group by logging in with group roles excluding that group, we will delete the single-sign-on-controlled group. When we delete a single-sign-on-controlled group, we will also delete any single-sign-on-controlled parent groups which are now empty. We will never delete a group that is not single-sign-on-controlled, even if the group becomes empty.

GroupRoleList (*optional*): This is a list of group roles within the university that the user is to be put into. A hierarchical path must be specified for each group.

Each element of the list can have the following properties:

- Path: the hierarchical path must be specified for each group.
- Role: the user's role within the group
 - GroupAdmin – Admin
 - GroupMember – Member
 - ReportViewer – ReportViewer

You have 3 options regarding the group role list section:

- Section not provided.
 - Do not change the user's roles in any way. This is the default.
- Section provided, but no groups provided.
 - If the EnrollmentList tag is present & empty, then we will remove all of the user's roles in single-sign-on-controlled groups.
- Section provided, with data.
 - Each Group tag represents one single-sign-on-controlled group in which the user has at least one role. Each tag contains a Role tag for each of the user's roles in that group. After login, the user will have exactly those roles in those groups. He will have no roles in any other single-sign-on-controlled groups in your university.

Custom Field List Section

Matching custom fields must be configured on the university first. If a field is not provided or provided as nil, any existing value will not be modified. If a blank value is provided, any existing value will be cleared.

EnrollmentList Section

You can add list of objects with a "CourseCode" field. This code refers to the course's ID which can be obtained via the reporting API.

If you add the EnrollmentList field you must also add the "EnrollmentSyncType" field with it. The value should be 1 if you want to sync the user's enrollments or 2 if you want to only add the included enrollments.

SAML Examples

```
<samlp:Response xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchemainstance"
ID="_b38aedd1-413e-45d5-ad86-55ad96685f14" Version="2.0" IssueInstant="2013-09-
10T22:23:39.3806629Z"
Destination="https://beta.bluevolt.com/SamlSSO/Login"
xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol">
  <saml:Issuer
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">http://my.domain.com/IDPService</saml:
Issuer>
  <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
    <SignedInfo>
      <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-
c14n-20010315" />
      <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"
/>
      <Reference URI="#_b38aedd1-413e-45d5-ad86-55ad96685f14">
        <Transforms>
          <Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
```

```
<Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-
c14n-20010315" />
</Transforms>
<DigestMethod Algorithm="http://www.w3.org/2000/09/xmlsig#sha1"
/>
<DigestValue>cyuGVIBnT42pyrKuGwL0Drext6U=</DigestValue>
</Reference>
</SignedInfo>
<SignatureValue>H0rhBJhydxgSX+7ZGKapZgzv2wcVqAX7oLvPK0VQomEBYmJUZgsgZet+LB+3AUWFY1
r00miAGGReaAhr7
gcNsxd60C6pt72/ZvoIdFkZcTfakwxr+4qYgLMbjpDc87mZf2Yo8LWbVy0IvtliSbi+HmDVR5c0zdTbMS5A/ToYic
K982ycqF8huT
BT4RX6C3uIbFh6mdj2kqy1ThUUGGAqanoJTH0cppoTk4wi8isg0i5VyEbIsbVTT0ZiAZkzHgYVeuqE+r7voH3bGb
b70B2B+qPL
1TJBqIYm3eTpDwSkwW0H6ANRBdW/DNS4ETz5V5L4dVq5V/Qx10KMw6isMUOg==</SignatureValue>
yki1buoqi13VgA8laX3faxqEZf03SUKRkxkUSI2artNgA8zdWHkWFgAdZv/ARGLve9pFQ0w6hSV1ppKxW23sXYiM
S2E6ET9vs0
olqarDwO71TbasWmpU/wnitZFNkESTe8oYcu3T+pkLFGj2oVf9ezUq9yLwPPZpxOVp1fpw6ybaxBceK8fKJr0joYK
Ste3mJyXaEV
zSckT/JcmsdCu0CQIDAQABoyQwIjALBgNVHQ8EBAMCBDAwEwYDVR01BAAwCgYIKwYBBQUHAWEdQYJKoZIhvcNAQE
FBQ
ADggEBADE6r48EfYp87wbTmsQwy0HB55tDmKlR8hmB0LIK+KUZt36NQOEHBMD6rY8VkhTsGHD/DUIgt9WpI5d6SdU
t1Klc3z
R5tHDn10VIkCoYFRC/0054hu+nkX0+z3BP7vJRO12AZDEuOyBYXnUxf0bkDee92TrEUDqqK1nLm2NyN8sUD3AZo1v
AHLu24j
4DSmcwG1Mt+lS/G74dUPf3xry2ZUGz0u+8ZTuue0OhxOt5vOunvOvKi39ikNIEkLKcbz/Yj3y1mcgXkxucuE71qKk
AeJMkk+O9tv
HtfVbTDjvqk/Z0gWt1nd6KiHoN2yvf1DAYHM4BA/400AlNSZGitQ1lb5s=</X509Certificate>
</X509Data>
</KeyInfo>
</Signature>
<samlp:Status>
<samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success" />
</samlp:Status>
<saml:Assertion Version="2.0" ID="_6fc83f56-90f4-4a3c-af7e-6339757f2ca2"
IssueInstant="2013-09-10T22:23:39.3756596Z"
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
<saml:Issuer>http://my.domain.com/IDPService</saml:Issuer>
<saml:Subject>
<saml:NameID
NameQualifier="http://my.domain.com">Immutable Unique Username 12345</saml:NameID>
<saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
<saml:SubjectConfirmationData NotOnOrAfter="2013-09-
10T22:28:41.0491472Z" Recipient="https://beta.bluevolt.com/SamlSSO/Login" />
</saml:SubjectConfirmation>
</saml:Subject>
<saml:Conditions NotBefore="2013-09-10T22:23:39.3766609Z" NotOnOrAfter="2013-09-
10T22:28:39.3766609Z">
<saml:AudienceRestriction>
<saml:Audience>https://beta.bluevolt.com/SamlSSO/Login</saml:Audience>
</saml:AudienceRestriction>
<saml:OneTimeUse />
</saml:Conditions>
<saml:AuthnStatement AuthnInstant="2013-09-10T22:23:39.3736586Z">
<saml:AuthnContext>
<saml:AuthnContextClassRef>AuthnContextClassRef</saml:AuthnContextClassRef>
</saml:AuthnContext>
</saml:AuthnStatement>
<saml:AttributeStatement>
```

```

        <saml:Attribute Name="FirstName"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
            <saml:AttributeValue xsi:type="xsd:string">Jimmy</saml:AttributeValue>
        </saml:Attribute>
        <saml:Attribute Name="LastName"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
            <saml:AttributeValue
xsi:type="xsd:string">Roberts</saml:AttributeValue>
        </saml:Attribute>
        <saml:Attribute Name="Email"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
            <saml:AttributeValue
xsi:type="xsd:string">jimroberts@mymail.com</saml:AttributeValue>
        </saml:Attribute>
        <saml:Attribute Name="Address1"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
            <saml:AttributeValue xsi:type="xsd:string">123 main
street</saml:AttributeValue>
        </saml:Attribute>
        <saml:Attribute Name="Address2"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
            <saml:AttributeValue xsi:type="xsd:string">#311</saml:AttributeValue>
        </saml:Attribute>
        <saml:Attribute Name="City" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:basic">
            <saml:AttributeValue
xsi:type="xsd:string">Portland</saml:AttributeValue>
        </saml:Attribute>
        <saml:Attribute Name="State"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
            <saml:AttributeValue xsi:type="xsd:string">OR</saml:AttributeValue>
        </saml:Attribute>
        <saml:Attribute Name="Zip" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:basic">
            <saml:AttributeValue xsi:type="xsd:string">97201</saml:AttributeValue>
        </saml:Attribute>
        <saml:Attribute Name="HomePhone"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
            <saml:AttributeValue
xsi:type="xsd:string">9715552234</saml:AttributeValue>
        </saml:Attribute>
        <saml:Attribute Name="WorkPhone"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
            <saml:AttributeValue
xsi:type="xsd:string">5035551234</saml:AttributeValue>
        </saml:Attribute>
        <saml:Attribute Name="EmployeeCode"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
            <saml:AttributeValue xsi:type="xsd:string">fc06c78a-c6ab-48ae-ac08-
edc1c6794b9a</saml:AttributeValue>
        </saml:Attribute>
        <saml:Attribute Name="GroupMember"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
            <saml:AttributeValue
xsi:type="xsd:string">group4|Customer</saml:AttributeValue>
            <saml:AttributeValue
xsi:type="xsd:string">group1|group2|group3</saml:AttributeValue>
        </saml:Attribute>
        <saml:Attribute Name="GroupAdmin"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
            <saml:AttributeValue
xsi:type="xsd:string">group1|group2</saml:AttributeValue>

```

```

        </saml:Attribute>
        <saml:Attribute Name="ReportViewer"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
            <saml:AttributeValue xsi:type="xsd:string">group1|group2
|group3</saml:AttributeValue>
        </saml:Attribute>
        <saml:Attribute Name="Job Title"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
            <saml:AttributeValue xsi:type="xsd:string">Sales</saml:AttributeValue>
        </saml:Attribute>
        <saml:Attribute Name="Date Added"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
            <saml:AttributeValue
xsi:type="xsd:string">1/30/2010</saml:AttributeValue>
        </saml:Attribute>
    </saml:AttributeStatement>
</saml:Assertion>
</samlp:Response>

```

Single Sign-On – OAUTH 2

Oauth is a standard way for accessing protected resources.

OAUTH Roles

Oauth 2 defines 4 roles:

Resource Owner: generally yourself (the User)

Resource Server: server hosting protected data (hosting the user's information)

Client: application requesting access to a resource server. The BlueVolt LMS.

Authorization Server: server issuing access token to the client.

Tokens

Tokens are random strings generated by the authorization server and issued when the client requests them.

Redirect URIs

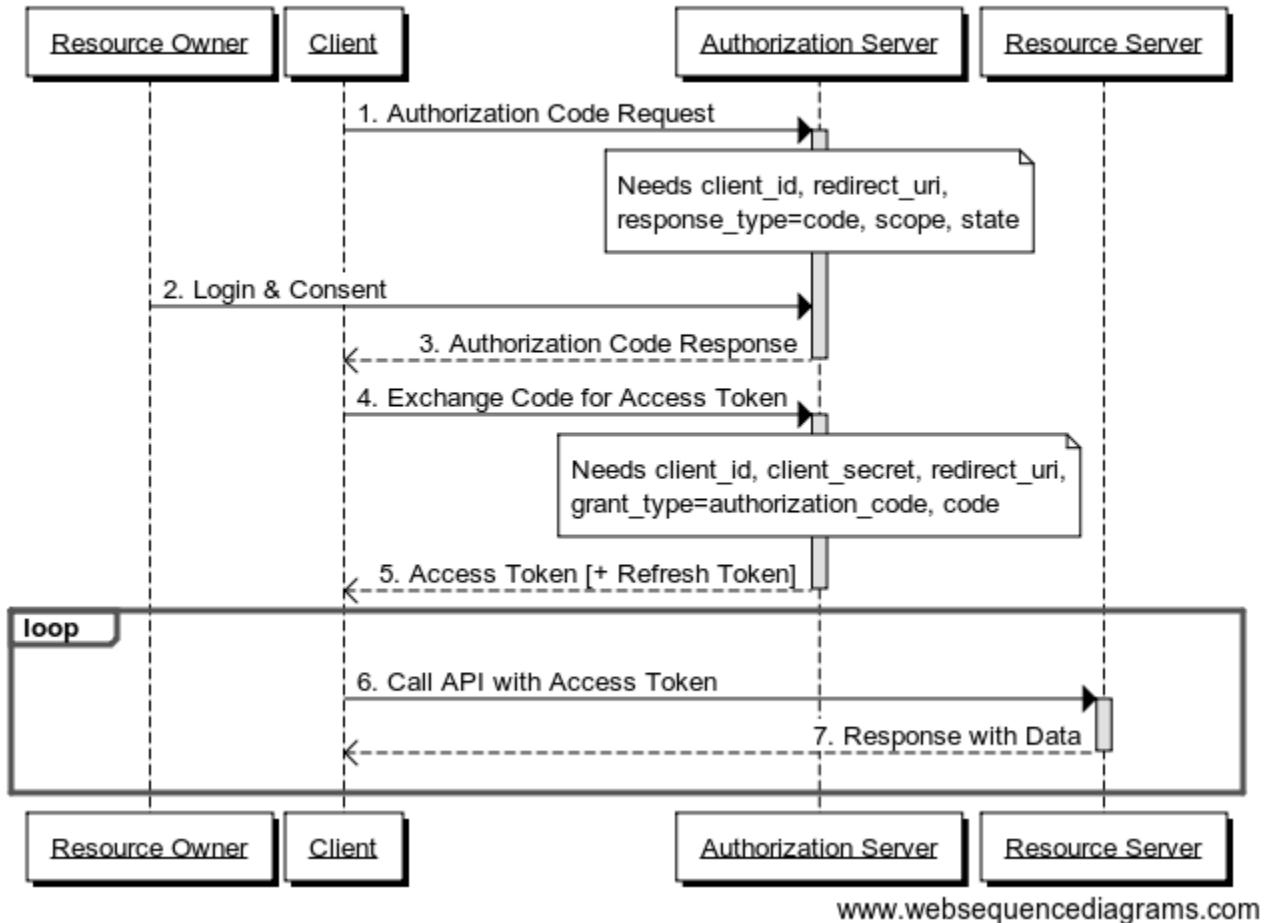
The authorization server should only redirect users to a registered URI.

Any HTTP redirects URIs must be protected with TLS security. This prevents tokens from being intercepted during the authorization process.

Client ID and Secret

The Auth server should generate a client ID and a secret for the client application.

Authorization Code Grant Flow



Things to provide to BlueVolt:

1. AuthorizationEndpoint: The Authorization Code request URL
2. Client_ID:
3. Client Secret:
4. Token Endpoint: The URL where the code is exchanged for the Access token.
5. Profile Endpoint: The URL of the where we get the users unique identifier.
6. The unique identifier field name: "userID".

When initiating the Single Sign-on, the users browser will be directed to [https://go.bluevolt.com/oauth2/?brand=\[UniversityName\]&ClientID=\[Client_ID\]](https://go.bluevolt.com/oauth2/?brand=[UniversityName]&ClientID=[Client_ID]) in production or [https://beta.bluevolt.com/oauth2/?brand=\[UniversityName\]&ClientID=\[Client_ID\]](https://beta.bluevolt.com/oauth2/?brand=[UniversityName]&ClientID=[Client_ID]) during testing, and the redirect Url will be the same. And the user should be redirected back to there after login & connect.

OPTION: OAuth Single Sign-On with Provisioning

Provisioning can be accomplished using the User Management API (recommended option) or through the OAuth Single Sign on with Provisioning option (this option requires the learner to login via SSO to create or update their account whereas the User Management API keeps accounts in sync without requiring the learner to login).

During the SSO process provisioned data will need to be sent in the OAuth AttributeStatement. If the user account is not provisioned before a Single Sign-On response by the API, then an account can be created in the LMS University, based on the information provided within the OAuth AttributeStatement.

Information provided in the OAuth AttributeStatement should contain university required fields and optional fields.

- Conditionally Required Fields (provider may opt to allow these fields to be provided by the user if provider does not maintain full user profiles)
 - FirstName
 - LastName
 - Email
- Optional Fields
 - Address1
 - Address2
 - City
 - State
 - Zip
 - WorkPhone
 - EmployeeCode (An identifier that you supply that is unique to each employee or student)
 - Country (A 2 letter code designating the employee country, as defined in ISO 3166-1 alpha-2, our system is synchronized with [<http://www.iso.org/iso/list-en1-semic-3.txt>])
 - GroupRoleList (see below)
 - CustomFieldList (see below)
 - EnrollmentList (see below)

Group Role List Section

The group role list section is optional, and allows you to automatically control a user's roles in single-sign-on-controlled groups. A group is single-sign-on-controlled if it was created via the group role list section. Single-sign-on-controlled groups, and their roles, are read-only in the BlueVolt admin. You administer them exclusively through single-sign-on.

If provided, the group role list specifies the set of group roles the user will have after they log in. During login, we synchronize the user's roles to match the provided list. We will both add and remove roles as necessary to bring them up to date with the list. Therefore, you must always send the full list of all roles in computer-controlled groups for this user.

The login process creates and deletes single-sign-on-controlled groups as needed. When the first user logs in with a role in a new group, we will create that as a single-sign-on-controlled group. When the last person is removed from a group by logging in with group roles excluding that group, we will delete the single-sign-on-controlled group. When we delete a single-sign-on-controlled group, we will also delete any single-sign-on-controlled parent groups which are now empty. We will never delete a group that is not single-sign-on-controlled, even if the group becomes empty.

GroupRoleList (*optional*): This is a list of group roles within the university that the user is to be put into. A hierarchical path must be specified for each group.

Each element of the list can have the following properties:

- Path: the hierarchical path must be specified for each group.
- Role: the user's role within the group
 - GroupAdmin – Admin

- GroupMember – Member
- ReportViewer – ReportViewer

You have 3 options regarding the group role list section:

- Section not provided.
 - Do not change the user's roles in any way. This is the default.
- Section provided, but no groups provided.
 - If the EnrollmentList tag is present & empty, then we will remove all of the user's roles in single-sign-on-controlled groups.
- Section provided, with data.
 - Each Group tag represents one single-sign-on-controlled group in which the user has at least one role. Each tag contains a Role tag for each of the user's roles in that group. After login, the user will have exactly those roles in those groups. He will have no roles in any other single-sign-on-controlled groups in your university.

Custom Field List Section

Matching custom fields must be configured on the university first. If a field is not provided or provided as nil, any existing value will not be modified. If a blank value is provided, any existing value will be cleared.

EnrollmentList Section

You can add list of objects with a "CourseCode" field. This code refers to the course's ID which can be obtained via the reporting API.

If you add the EnrollmentList field you must also add the "EnrollmentSyncType" field with it. The value should be 1 if you want to sync the user's enrollments or 2 if you want to only add the included enrollments.

Sample Profile from Profile Endpoint

```
{
  "ssusername": "987-ABC-123",
  "email": "tony.asche@bluevolt.com",
  "firstname": "Tony",
  "lastname": "Asche",
  "address1": "123 SW Main st.",
  "address2": "Suite 208",
  "city": "Portland",
  "state": "OR",
  "zip": "97201",
  "country": "US",
  "workphone": "(555) 123-1234",
  "employeeid": "987-ABC-123",
  "Favorite Color": "Black",
  "Birth Date": "3/1/1990",
  "GroupRolesType": 2,
  "GroupRoleList": [
    {"Path": "Job Type|Sales Desk",
      "Role": "GroupMember"},
    {"Path": "Branches|NW|Oregon",
      "Role": "Member"},
    {"Path": "Branches|NW|Washington",
      "Role": "Member"},
    {"Path": "Managers",
      "Role": "Admin"},
    {"Path": "Branches|NW|Oregon",
      "Role": "ReportViewer"}
  ],
  "EnrollmentSyncType": 2,
  "EnrollmentList": [
    {
      "CourseCode": "8675309"
    },
    {
      "CourseCode": "90210"
    },
    {
      "CourseCode": "3333333"
    }
  ]
}
```

Overview of BlueVolt's APIs and Integration Applications

BlueVolt provides several APIs that make functionality of the BlueVolt LMS directly available to your organization by exposing features of the platform. This allows organizations using the BlueVolt LMS to have direct access to their data. The current APIs and integrated applications provided by BlueVolt include the following:

User Management

- **User Management:** The user management portion of the API allows you to maintain consistent user data that syncs with an existing system, such as an HRIS or CRM, which contains information about user *authorization*. The user management piece includes 3 end points for updating user data (Add/Sync/Delete). It also allows you to control an individual's group roles, and sending of welcome emails. (via JSON or XML)
- **Single Sign-On (SAML SSO):** The SSO component allows for user *authentication* (login) from an existing system. Users are automatically logged in to the LMS, and do not have to login or create an account in the LMS to gain access to training. Data updates occur when the user comes through the SSO. (via XML)

Report Management:

- **Report Service:** The report service component allows you to automatically pull reporting data out of the BlueVolt LMS. It includes reporting data on course enrollments and completions, including dates and times. (via XML)
- **Salesforce Application:** The BlueVolt integration into Salesforce allows you to seamlessly import your LMS data into Salesforce. It allows you to create Contacts from LMS Learner accounts (optional), import Course metadata, and the Enrollment/Completion data for Contacts enrolled in those courses.
- **REST API:** The REST API enables you to pull your university data into your own Business Intelligence middleware in an automated fashion without having to manually run a course enrollment totals report and then have to import the data.

Developer's FAQ

1. How are the web services implemented?

The web services support many SOAP methods; these are described in an accompanying WSDL file that can be imported into your favorite web services programming environment.

2. What is a web service?

A web service is something you can call over the web from a program. For more background on web services, see http://en.wikipedia.org/wiki/Web_services

<http://www.w3schools.com/webservices/default.asp>

3. What is SOAP?

SOAP is the Simple Object Access Protocol. It is used for information exchange and RPC, usually (but not necessarily) over HTTP. More information can be found at:

<http://www.soaprpc.com/faqs/SoapFAQ.html>

<http://www.w3.org/TR/SOAP/>

4. Does this service work through HTTP proxies?

Sometimes. Many HTTP proxies do not correctly forward SOAP. If possible your server should bypass the proxy and communicate directly with BlueVolt.

5. What programming languages are supported?

The web services have been tested using C# and Visual Basic (Microsoft Visual Studio 2005) and the force.com APEX language. The service has not been tested with clients in other languages, but it should work with any language with web services support such as Java (Apache SOAP and Apache Axis) or PHP (NuSoap).

6. Can I see some example SOAP messages?

Our services are published using Microsoft .NET technologies and therefore come complete with accompanying .asmx description pages showing examples of the SOAP inputs and outputs used in each call. We also have a sample C# implementation available. Please contact us to get a copy.