Delegated Authentication Implementation Guide

Delegated authentication allows a user to login to Projector using your company's own authentication servers. Many companies using LDAP leverage this feature so that users only need to remember a single set of credentials and so that you can enforce your own security policies like password complexity and expiration dates. This document explains how DA is implemented and how you can start using it in your installation.

We XML encode the password when passing it to your web service. That means special characters like <, >, &, ', and " need to be properly decoded by your XML parser in order for your end users to have those characters in their passwords.

Projector also supports a technology called Single Sign On (SSO) which is different from Delegated Authentication. SSO allows a user to sign into many different web applications using a single set of credentials. While Delegated Authentication works with Management Portal, Projector Web, and Projector Mobile; SSO only works with Projector Web. See Single Sign On (SSO) Implementation Guide.

How it Works

Delegated Authentication uses web services to pass the login credentials around. You will need to implement your own publicly accessible, SSL encrypted web service that Projector can query. The information flow works like so:

1. A user enters their username/pass into Projector's login screen and attempts to connect
2. Projector checks the user's profile to see if they use DA
3. If DA is enabled then we pass the username/password combo to your web service
4. Your web service returns true/false to Projector
5. Projector allows/disallows access to our systems

Enable DA

There are four steps to start using Delegated Authentication

Implement Web Service

Implement a web service with a method named `OpsAuthenticate` that is accessible via https. The method should accept two parameters, `userName` and `password`, and return a boolean.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>userName</td>
</tr>
<tr>
<td>input</td>
<td>password</td>
</tr>
<tr>
<td>return</td>
<td>boolean</td>
</tr>
</tbody>
</table>

Provide Authentication URL

Send your authentication URL to Projector. This URL will be saved in our database. If your authentication server URL changes, notify Projector support to have it updated. An SSL url (https) is a requirement.

Enable DA On Users

DA is controlled on a per user basis. It is most likely that you want to enable it for everyone in your installation. In this case it is fastest to edit a common user type and enable it.
You can also turn DA on/off for individual users from the user editor.

Set DA Name

This last step is optional. Projector needs to be able to tie an individual Projector account with your authentication server account. This is done through the username. By default, Projector assumes your username is your email address. If your internal system uses something else, then you will need to go in and explicitly set the login name for each person. In the example below I have changed the login name from tom@projectors.com to just Tom.

Implementation

The following are code examples you can use as a reference for implementing your own DA system.

C# Example Code

A code example of implementing the OpsAuthenticate service.
using System.Web;
using System.Web.Services;

namespace Ops.Projector.WebSvc
{
    /// <summary>
    /// Summary description for OpsAuthenticationSvc.
    /// </summary>
        Name="OpsAuthenticationSvcSoap", Namespace="http://www.opsplanning.com/webservices/private/auth/OpsAuthenticate")]
    {
        /// <summary>
        /// Constructor
        /// </summary>
        public OpsAuthenticationSvc ()
        {
            //CODEGEN: This call is required by the ASP.NET Web Services Designer
            InitializeComponent();
            return;
        }

        #region Component Designer generated code

        //Required by the Web Services Designer
        private IContainer components = null;

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
        }

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        protected override void Dispose( bool disposing )
        {
            if(disposing && components != null)
            {
                components.Dispose();
            }

            base.Dispose(disposing);
        }

        #endregion

        #region OpsAuthenticate

        /// <summary>
        /// Authenticate a user
        /// </summary>
        /// <param name="userName">User name</param>
        /// <param name="password">Password</param>
        /// <returns>True if authenticated</returns>
        [WebMethod]
        public bool OpsAuthenticate (string userName, string password)
        {
            // Do your authentication here
            return true_or_false;
        }

        #endregion
    }
}
Sample XML Request

The following is an example request that we would send your web service.

**XML Request**

```xml
POST /OpsProjectorWebSvc/OpsAuthenticationSvc.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://www.opsplanning.com/webservices/private/auth/OpsAuthenticate"

<?xml version="1.0" encoding="utf-8"?>
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:OpsAuthenticate="http://www.opsplanning.com/webservices/private/auth">
    <soap:Body>
        <OpsAuthenticate xmlns="http://www.opsplanning.com/webservices/private/auth">
            <userName>string</userName>
            <password>string</password>
        </OpsAuthenticate>
    </soap:Body>
</soap:Envelope>
```

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

```xml
<?xml version="1.0" encoding="utf-8"?>
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:OpsAuthenticate="http://www.opsplanning.com/webservices/private/auth">
    <soap:Body>
        <OpsAuthenticateResponse xmlns="http://www.opsplanning.com/webservices/private/auth">
            <OpsAuthenticateResult>boolean</OpsAuthenticateResult>
        </OpsAuthenticateResponse>
    </soap:Body>
</soap:Envelope>
```

Sample WSDL

The WSDL service description for the sample web service implemented in C# above.
WSDL

<?xml version="1.0" encoding="utf-8" ?>
  <wsdl:types>
    <s:schema elementFormDefault="qualified" targetNamespace="http://www.opsplanning.com/webservices/private/auth">
      <s:element name="OpsAuthenticate">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="userName" type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="password" type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="OpsAuthenticateResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="OpsAuthenticateResult" type="s:boolean" />
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:schema>
  </wsdl:types>
  <wsdl:message name="OpsAuthenticateSoapIn">
    <wsdl:part name="parameters" element="s0:OpsAuthenticate" />
  </wsdl:message>
  <wsdl:message name="OpsAuthenticateSoapOut">
    <wsdl:part name="parameters" element="s0:OpsAuthenticateResponse" />
  </wsdl:message>
  <wsdl:portType name="OpsAuthenticationSvcSoap">
    <wsdl:operation name="OpsAuthenticate">
      <wsdl:input message="tns:OpsAuthenticateSoapIn" />
      <wsdl:output message="tns:OpsAuthenticateSoapOut" />
    </wsdl:operation>
  </wsdl:portType>
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
  <wsdl:operation name="OpsAuthenticate">
    <wsdl:input>
      <soap:body use="literal" />  
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />  
    </wsdl:output>
  </wsdl:operation>
  <wsdl:service name="OpsAuthenticationSvc">
    <documentation xmlns="http://schemas.xmlsoap.org/wsdl/" />
    <wsdl:port name="OpsAuthenticationSvcSoap" binding="tns:OpsAuthenticationSvcSoap">
      <soap:address location="http://localhost/OpsProjectorWebSvc/OpsAuthenticationSvc.asmx" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>

Saved Passwords

Projector allows you to save your login credentials for the Management Portal. A saved login stores an authentication ticket (random string) at Projector. In this case your servers will not be re-queried until the ticket expires or it is cleared from memory.
Web Service Down

If your authentication URL becomes inaccessible or is otherwise "broken," then Projector reports to the user that their username/password is invalid. In the case that your authentication service is down and you really need to get to your data, Projector allows one special user to fall back to our internal authentication service. This is the Projector Administrator. You can identify the admin by going to your users list and finding the bolded name. As you won't be able to access your installation if the authentication server is down, you should note your admin and make sure they know their fallback password before a problem arises.

If your service is down and nobody can get into the system please contact Projector Support for assistance.

Unknown macro: 'sidebartoc'