Trusted Source SSO

Document version 2.3
Last updated: 30/10/2017

www.iamcloud.com

TABLE OF CONTENTS

1 Introduction

Trusted Source SSO is a feature of the IAM Cloud platform, available on the Advanced and Expert plans. It provides a mechanism to bridge the gap between a non-federated application and any federated application supported by IAM Cloud.

A typical use-case is where a customer has an internal staff portal that is not a federated application but authenticates directly against the on-premises Active Directory or an on-premises database. The user would log into the portal, and then the user is presented with a link to a federated application like Office 365, they would now be greeted with a second login page and must perform a second login because the federation service used for the federated application would be disconnected from the non-federated application (the internal staff portal).

If the non-federated application is using Trusted Source SSO, the user would still authenticate with the staff portal as they do now, but clicking the link to their federated application would now give them a seamless SSO experience straight into that application.

In other words, Trusted Source can turn your own staff portal or Intranet, for example, into a full SSO portal. This means you can get the significant benefits of federated access and single sign-on, without making changes to your end user experience or existing infrastructure.

Trusted Source SSO communicates over https on standard port 443 and leverages Short Lived Tokens (SLT) to perform authentications with IAM Cloud. It also uses normal ReST web services for generating the token so supports many languages however we have examples for Java, PHP, C# and ASP.NET

2 Prerequisites

2.1 Agent

If the customers is running Active Directory and they want the experience seemelss when a user changes their password on-premises they should be running our latest Agent, which supports hashing technology. Agent has extensibility to support other source of passwords such as SQL and other third parties. This means your users will not see any login for the first time and will use the password from the source system.

** Please note**, without this** if the user resets their password on Active Directory and we have password sync enabled. The next time they try and log into the federated application it will fail require the user to log in that once. This will rebuild the profile on IAM Cloud and then will be seamless SSO until the next password reset.

2.2 Smart Link

Once the application is configured within the IAM Cloud portal and the SAML authentication is working you must then configure a smart link – this smart link is then used as the redirection point after trusted source authentication has occurred.

A smart link gives a direct way of accessing an application or a specific place within an application. For example, if you want to have your users go to a specific SharePoint site but do not want to provide complicated crafted links you can provide a short name on your own domain. For example xxx.mydomain.com. This is then configured as a CNAME within your own DNS pointing to our redirection service. We will interpret this link and once authentication has occurred (via trusted source or normally) it will redirect directly to your specific location. We leverage this technology to given flexibility when redirecting after authentication has occurred.

2.3 Required Information to setup Trusted Source SSO

2.3.1 Supplied Key

The Supplied Key is a private key that is entered by the customer during the installation. This Supplied Key should be treated like an administrator password or PIN, and should not be disclosed to anyone outside of the organisation, including IAM Cloud. This key is used to encrypt data with, meaning that IAM Cloud or anyone else cannot tamper with it or decrypt it.

This key is essential to the Trusted Source SSO mechanisms.

2.3.2 DNS Records

To use Trusted Source SSO, customer's must have a smart link which requires a single DNS record of your choice. For example if you are wanting the user to log into application A after the authentication, then you would create a smart link of

xxx.mydomain.com which would be a CNAME within your external DNS system pointing to our redirection service – remember, you can also get us to redirect to specific locations within the application.

2.3.3 API Service Account Username and Password

IAM Cloud will supply customers with an API Service Account. This account is used to authenticate with the API when generating the SLT token.

2.3.4 Tenancy Identifier

Every customer has a tenancy identifier when running on our G3 platform. This is used as the authentication endpoints for all your SAML and Smart Link logins. For customers that have our Fully Branded package, this can be a full vanity URL for example "idp.mydomain.com" however for most customers this would be XXX.iamcloud.net. The tenancy Identifier is referenced in the code.

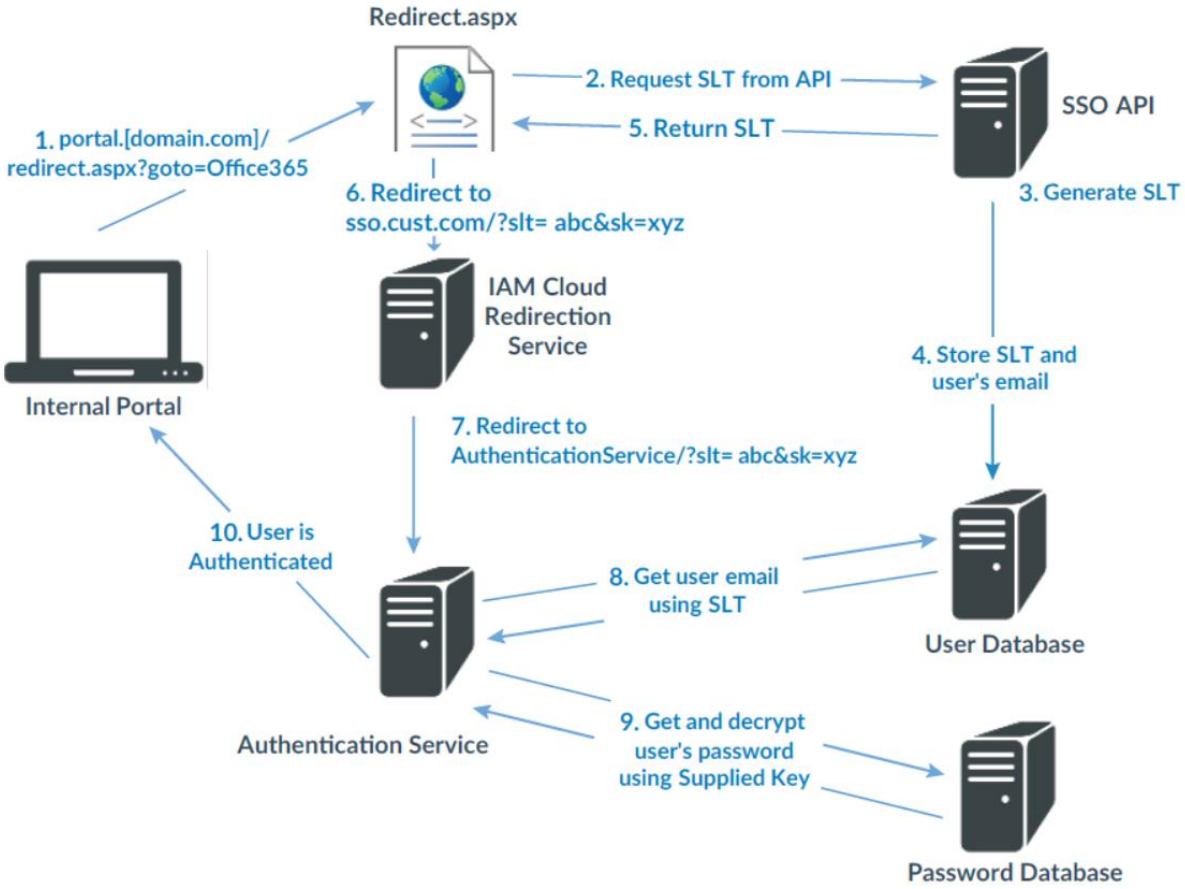
2.3.5 Application ID

In addition to the ID of the smart link, you will also require the ID of the application. These can both found in our portal at portal.iamcloud.net and are referenced in the code. This dictates the application you with to log into after authentication and the ID of the Smart Link to redirect to after a successful authentication.

3 System Overview

3.1 Process flows

3.1.1 Visio Representation



3.1.2 Visio Representation Explained

1. Because the SLT has a finite life span, it is recommended to change the federated application link to 'point' to a new page on your non-federated application. E.g. Redirect.aspx?goto=Email. This makes it easier to manage but isn't necessary.
2. A small section of code is required on the non-federated application to talk with the IAM Cloud API to use Trusted Source SSO. Currently ASP.NET, PHP MVC and Java clients are supported. The code samples can be found in section 5. 2 calls are made to make to the API. The first is to authenticate with it and the second to request the SLT.
3. The API is requested to generate a unique SLT for a user which is stored and available for 1 time use within 10 minutes of creation.

4. The SLT is then returned back to the calling client (which is the server that initiated the request not the browser client)
5. Now the first browser redirection occurs - the request is passed on to the Authentication Service.
6. The SLT is now used to retrieve the user who we are attempting to authenticate.
7. The username is used to retrieve the authentication details. These details were securely sent by Agent/SPS using the supplied key or will be built on first login - and encrypted with IAM Cloud's secure encryption algorithms. We use the Supplied Key to perform a check against the hashed results we have. This means that the password is never in a clear text state and is salted with a key we only know during the transaction and that the customer has ownership of. If successful then it will continue. If however there is of the following conditions met it will fail:
    a. Failed Authentication
    b. TouchPoint requiring user interactions
    c. We detect it as a hacking attempt
    d. Login Control policy has been applied
8. Once authentication is completed and we have produced the federated token for the end application we use the smart link to redirect to the application which will then show the application to the user. Please note, applications that support IDP logins will have one less redirection for the user at this point.

4 Integration

4.1.1 PHP

```
<?php

//username
$Username = 'test_user@yourdomain.com';

$ApiBaseAddress = "https://api.iamcloud.net";
$ServiceAccount = 'xxx';
$ServicePassword = 'xxx';

$TenancyIdentifier = 'xxx.iamcloud.net';
$AppID = '12345';
$SmartLinkID = '678';

$curl = curl_init($ApiBaseAddress);

curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);

//authenticate with api
$RequestUrl = '/api/authenticate/';
curl_setopt($curl, CURLOPT_URL, $ApiBaseAddress . $RequestUrl);

$postfields = array("Username" => $ServiceAccount, "Password" => $ServicePassword);
curl_setopt($curl, CURLOPT_POST, true);
curl_setopt($curl, CURLOPT_POSTFIELDS, http_build_query($postfields));

$AuthToken = curl_exec($curl);

$j = json_decode($AuthToken);

//get atsso slt
$RequestUrl = '/api/Atsso/';
curl_setopt($curl, CURLOPT_URL, $ApiBaseAddress . $RequestUrl);

$atssopostfields = array("Username" => $Username);
curl_setopt($curl, CURLOPT_POSTFIELDS, http_build_query($atssopostfields));

curl_setopt($curl, CURLOPT_HTTPHEADER, array(
   'Authorization: Bearer '. $j->access_token
   ));

$curl_response = curl_exec($curl);

curl_close($curl);

$JsonResponse = json_decode($curl_response);

$Slt = trim($JsonResponse->Token, '"'); //short lived token to use for authenticating the user

$SuppliedKey = 'xxx'; //this is a private key used by the SPS software

//pass the Supplied Key and Slt to IAM Cloud's federation servers ...
header ('Location: https://'.$TenancyIdentifier'/atsso/p-
auth/?slt='.$Slt.'&sk='.$SuppliedKey'&appid='.$AppID'&smartlinkid='.$SmartLinkID');

?>
```

4.1.2 Java

```
package com.trustedsource.servlets;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.net.URI;
import java.net.URLEncoder;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.http.HttpResponse;
import org.apache.http.HttpStatus;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClientBuilder;

import com.google.gson.Gson;

@WebServlet(description = "Get Trusted Source short lived token and redirect to IAM Cloud Authentication Service",
urlPatterns = { "/" })
public class Redirector extends HttpServlet {

        private static final long serialVersionUID = 1L;

        protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
                Redirect(response);
        }

        /**
         * Method for getting Trusted Source short lived token and redirecting to IAM Cloud Authentication Service
         * @param redirectResponse {@link HttpServletResponse} used by servlet
         * @throws IOException if an input or output error is detected
```

```
 */
protected void Redirect(HttpServletResponse redirectResponse) throws IOException {

        String username = "test_user@yourdomain.com";
        String apiBaseAddress = "https://api.iamcloud.net/api";
        String serviceUsername = "xxx";
        String servicePassword = "xxx";

        String TenancyIdentifier = "xxx.iamcloud.net";
        String AppID = "12345";
        String SmartLinkID = "678";

        String suppliedKey = "xxx"; // this is a private key used by the SPS software

        BearerToken apiBearerToken = new BearerToken();
        AtssoToken atssoToken = new AtssoToken();

        // for parsing objects to/from JSON
        Gson gson = new Gson();

        BufferedReader reader = null;
        try (CloseableHttpClient cleint = HttpClientBuilder.create().build()) {

                HttpPost request = new HttpPost(apiBaseAddress + "/authenticate");
                request.addHeader("content-type", "application/json");

                // API AUTH
                AuthenticationPost authPost = new AuthenticationPost();
                authPost.setUsername(serviceUsername);
                authPost.setPassword(servicePassword);

                // request bearer token
                StringEntity params = new StringEntity(gson.toJson(authPost));
                request.setEntity(params);
                HttpResponse response = cleint.execute(request);
                //

                if (response.getStatusLine().getStatusCode() == HttpStatus.SC_OK) {
                        reader = new BufferedReader(new
InputStreamReader(response.getEntity().getContent()));
                        apiBearerToken = gson.fromJson(reader, BearerToken.class);
```

```
                                        // ATSSO TOKEN
                                        AtssoPost atssoPost = new AtssoPost();
                                        atssoPost.setUsername(username);

                                        // add auth header
                                        request.addHeader("Authorization", "Bearer " +
apiBearerToken.getAccess_token());

                                        // request atsso token
                                        params = new StringEntity(gson.toJson(atssoPost));
                                        request.setEntity(params);
                                        request.setURI(URI.create(apiBaseAddress + "/Atsso"));
                                        response = cleint.execute(request);
                                        //

                                        if (response.getStatusLine().getStatusCode() == HttpStatus.SC_OK) {
                                                reader = new BufferedReader(new
InputStreamReader(response.getEntity().getContent()));
                                                atssoToken = gson.fromJson(reader, AtssoToken.class);
                                        }
                                }
                        } finally {
                                if (reader != null) {
                                        reader.close();
                                }
                        }

                        String url = null;
                        try {
                                url = https:// + TenancyIdentifier + "/atsso/p-auth/?slt="
                                                + URLEncoder.encode(atssoToken.getToken(), "UTF-8") + "&sk="
+ URLEncoder.encode(suppliedKey, "UTF-8") + "&appid=" + AppID + "&smartlinkid=" + SmartLinkID;
                        } catch (UnsupportedEncodingException e) {
                                e.printStackTrace();
                        }

                        // redirect using servlet's response
                        redirectResponse.sendRedirect(url);

                        // redirect using apache httpClient
```

```
                        /*HttpClient instance = HttpClientBuilder.create().setRedirectStrategy(new
LaxRedirectStrategy()).build();
                        instance.execute(new HttpGet(url));*/
                }
}

/**
 * Class for holding service username and password
 *
 */
class AuthenticationPost {

        private String username;

        private String password;

        public String getUsername() {
                return username;
        }

        public void setUsername(String username) {
                this.username = username;
        }

        public String getPassword() {
                return password;
        }

        public void setPassword(String password) {
                this.password = password;
        }
}

/**
 * Class for holding data after requesting Bearer token.
 */
class BearerToken {

   public String access_token;
```

```
    public String token_type ;

    public int expires_in;

    public String refresh_token;

        public String getAccess_token() {
            return access_token;
        }

        public void setAccess_token(String access_token) {
            this.access_token = access_token;
        }

        public String getToken_type() {
            return token_type;
        }

        public void setToken_type(String token_type) {
            this.token_type = token_type;
        }

        public int getExpires_in() {
            return expires_in;
        }

        public void setExpires_in(int expires_in) {
            this.expires_in = expires_in;
        }

        public String getRefresh_token() {
            return refresh_token;
        }

        public void setRefresh_token(String refresh_token) {
            this.refresh_token = refresh_token;
        }
}

/**
```

```
 * Class for holding username
 */
class AtssoPost {
        private String username;

        public String getUsername() {
                return username;
        }

        public void setUsername(String username) {
                this.username = username;
        }

}

/**
 * Class for holding ATSSO TOKEN
 */
class AtssoToken
{
   public String Token;

        public String getToken() {
                return Token;
        }

        public void setToken(String token) {
                Token = token;
        }
}
```

4.1.3 ASP.NET 4

```
public void Redirect()
{
        string Username = "test_user@yourdomain.com";

        string ApiBaseAddress = "https://api.iamcloud.net/api";
        string ServiceUsername = "xxx";
        string ServicePassword = "xxx";

        String TenancyIdentifier = “xxx.iamcloud.net”;
        String AppID = “12345”;
        String SmartLinkID = “678”;

        string SuppliedKey = "xxx";  //this is a private key used by the SPS software

        BearerToken ApiBearerToken = new BearerToken();
        AtssoToken AtssoToken = new AtssoToken();

        using (var client = new WebClient())
        {
                client.BaseAddress = ApiBaseAddress;

                //API AUTH
                NameValueCollection AuthPost = new NameValueCollection();
                AuthPost.Add("Username", ServiceUsername);
                AuthPost.Add("Password", ServicePassword);

                //request bearer token
                byte[] result = client.UploadValues(ApiBaseAddress + "/authenticate", "POST", AuthPost);
                string response = client.Encoding.GetString(result);

                if (response != null && response.Any())
                {
                        JavaScriptSerializer parser = new JavaScriptSerializer();

                        var Result = client.Encoding.GetString(result);
                        var Token = parser.Deserialize<BearerToken>(Result);

                        //add bearer to this request
                        client.Headers.Add(HttpRequestHeader.Authorization, "Bearer " + Token.access_token);
```

```
                        //ATSSO TOKEN
                        var AtssoPost = new NameValueCollection();
                        AtssoPost.Add("Username", Username);

                        //request atsso token
                        var AtssoTokenResult = client.UploadValues(ApiBaseAddress + "/Atsso", "POST",
AtssoPost);

                        if (AtssoTokenResult.Any())
                        {
                                var AtssoTokenResponse = client.Encoding.GetString(AtssoTokenResult);
                                AtssoToken = parser.Deserialize<AtssoToken>(AtssoTokenResponse);
                        }
                }
        }

        string Url = "https://" + TenancyIdentifier + "/atsso/p-auth/?slt=" + Server.UrlEncode(AtssoToken.Token) +
"&sk=" + Server.UrlEncode(SuppliedKey) + "&appid=" + AppID + "&smartlinkid" + SmartLinkID;

        Response.Redirect(Url);
}
public class BearerToken
{
        public string access_token { get; set; }
}

public class AtssoToken
{
        public string Token { get; set; }
}
```

4.1.4 .NET 4.5

```
public void Redirect()
{
	string Username = "test_user@yourdomain.com";
	string ApiBaseAddress = "https://staging-api.iamcloud.net/api";
	string ServiceUsername = "xxx";
	string ServicePassword = "xxx";

	String TenancyIdentifier = “xxx.iamcloud.net”;
	String AppID = “12345”;
	String SmartLinkID = “678”;

	string SuppliedKey = "xxx"; //this is a private key used by the SPS software

	BearerToken ApiBearerToken = new BearerToken();
	AtssoPostOutput AtssoToken = new AtssoPostOutput();

	using (var client = new HttpClient())
	{
		client.BaseAddress = new Uri(ApiBaseAddress);
		client.DefaultRequestHeaders.Accept.Clear();
		client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("application/json"));

		//API AUTH
		AuthenticationPost AuthPost = new AuthenticationPost();
		AuthPost.Username = ServiceUsername;
		AuthPost.Password = ServicePassword;

		//request bearer token
		var result = await client.PostAsJsonAsync(ApiBaseAddress + "/Contacts", AuthPost);

		if (result.IsSuccessStatusCode)
		{
			ApiBearerToken = await result.Content.ReadAsAsync<BearerToken>();

			//ATSSO TOKEN
			var AtssoPost = new AtssoPost();
			AtssoPost.Username = Username;

			//add auth header
			client.DefaultRequestHeaders.Authorization =
new AuthenticationHeaderValue("Bearer", ApiBearerToken.AccessToken);

			//request atsso token
			var AtssoTokenResult = await client.PostAsJsonAsync<AtssoPost>(ApiBaseAddress +
"/Atsso", AtssoPost);

			if (AtssoTokenResult.IsSuccessStatusCode)
			{
				AtssoToken = await
AtssoTokenResult.Content.ReadAsAsync<AtssoPostOutput>();
			}
		}
	}

	string Url = “https://” + TenancyIdentifier + “/atsso/p-auth/?slt=" + Server.UrlEncode(AtssoToken.Token) +
"&sk=" + Server.UrlEncode(SuppliedKey) + “&appid=” + AppID + “&smartlinkid” + SmartLinkID;

	Response.Redirect(Url);
```

```
}

public class AuthenticationPost
{
        public string Username { get; set; }

        public string Password { get; set; }
}

public class AtssoPostOutput
{
        public string Token { get; set; }
}

public class BearerToken
{
        [JsonProperty(PropertyName = "access_token")]
        public string AccessToken { get; set; }


        [JsonProperty(PropertyName = "token_type")]
        public string TokenType { get; set; }


        [JsonProperty(PropertyName = "expires_in")]
        public int ExpiresIn { get; set; }


        [JsonProperty(PropertyName = "refresh_token")]
        public string RefreshToken { get; set; }
}

public class AtssoPost
{
        public string Username { get; set; }
}

public class AtssoToken
{
        public string ShortLivedToken { get; set; }
}
```

COPYRIGHT NOTICE