IBM® Security Verify Access 10.0.0.0

IBM® Security Access Manager 9.0.7.0 and above

IBM Security Verify Access Extension for IBM Security Trusteer Pinpoint Detect

Integration Guide

October 2020

Contents

Preface

IBM Knowledge Center

The information in this guide is complemented by the documentation available for IBM Security Verify Access on the IBM Knowledge Center at https://www.ibm.com/support/knowledgecenter/SSPREK_10.0.0/

IBM Terminology website

The IBM Terminology website consolidates terminology for product libraries in one location. You can access the Terminology website at http://www.ibm.com/software/globalization/terminology.

Accessibility

Accessibility features help users with a physical disability, such as restricted mobility or limited vision, to use software products successfully. With this product, you can use assistive technologies to hear and navigate the interface. You can also use the keyboard instead of the mouse to operate all features of the graphical user interface.

Technical Training

For technical training information, see the following IBM Education website at

http://www.securitylearningacademy.com/

Support information

IBM Support provides assistance with code-related problems and routine, short duration installation or usage questions. You can directly access the IBM Software Support site at http://www.ibm.com/support/home

Statement of Good Security Practices

IT system security involves protecting systems and information through prevention, detection and response to improper access from within and outside your enterprise. Improper access can result in information being altered, destroyed, misappropriated or misused or can result in damage to or misuse of your systems, including for use in attacks on others. No IT system or product should be considered completely secure and no single product, service or security measure can be completely effective in preventing improper use or access. IBM systems, products and services are designed to be part of a comprehensive security approach, which will necessarily involve additional operational procedures, and may require other systems, products or services to be most effective. IBM DOES NOT WARRANT THAT ANY SYSTEMS, PRODUCTS OR SERVICES ARE IMMUNE FROM, OR WILL MAKE YOUR ENTERPRISE IMMUNE FROM, THE MALICIOUS OR ILLEGAL CONDUCT OF ANY PARTY.

Product name updates

This publication was first established for IBM Security Access Manager. IBM Security Access Manager has since been superseded by IBM Security Verify Access. Similarly, this integration first worked with Trusteer Pinpoint Criminal Detect (“CD”) capabilities which are now part of IBM Security Trusteer Pinpoint Detect.

Wherever in this guide, any figures and graphics that contain or refer to IBM Security Access Manager, the use of IBM Security Verify Access is implied. For any reference to Trusteer Criminal Detect or CD, the use of IBM Security Trusteer Pinpoint Detect is implied.

Introducing the Integration

Introduction

This guide describes the steps that are required to integrate IBM Security Verify Access with IBM Security Trusteer Pinpoint Detect for protecting web applications. It makes use of an IBM-provided extension package published on the IBM Security App Exchange.

Overview

The main purpose of this integration is to combine the risk determinations of IBM Security Trusteer Pinpoint Detect (Pinpoint Detect) with the access enforcement, multi-factor authentication, and session management capabilities of IBM Security Verify Access (Verify Access).

Verify Access has its own basic risk detection capabilities but applications that are sensitive, or subject to increased risk, require the sophisticated capabilities of Pinpoint Detect.

Verify Access can be used to allow a web application to enjoy the accurate risk determinations and security protection of Pinpoint Detect, while reducing (or even removing) the need to modify the web application directly.

Those who implement integration between Verify Access and Pinpoint Detect should consider these points:

- There are many ways in which applications, Verify Access, and Pinpoint Detect can be integrated. This guide describes a recommended and tested initial integration pattern which can form the foundation of a fully customized integration. Your deployment may require variation based on how sessions, user identifiers, and transaction data are handled by your application.
- Pinpoint Detect deployment choices should follow the recommendations of the Trusteer Deployment team in the Trusteer lab. The use of integration between Verify Access and Pinpoint Detect, to allow Verify Access to leverage Pinpoint Detect risk determinations, should not change this.
- Customers who have integrated Verify Access and Pinpoint Detect using other integration patterns (the use of a sub-domain or proxy mode for snippet traffic for example) do not have to migrate. Other integration patterns, based on standard product capabilities, remain supported.
- It is important that both Trusteer and Verify Access SMEs are part of the project, at least initially, to understand the required capabilities and discuss implementation options.

It is assumed that the reader has a reasonable understanding of both Verify Access and Pinpoint Detect.

IBM provides extensive documentation to customers for both products (although Pinpoint Detect documentation is available only to Pinpoint Detect customers). There are many capabilities in Verify Access and Pinpoint Detect which are beyond the scope of this document.

Integration Product Version Information

Functionality documented in this guide is for the following product versions:

- IBM Security Verify Access 10.0.0.0 or above
- IBM Security Access Manager 9.0.7.0 or above
- IBM Security Trusteer Pinpoint Detect versions supporting the Pinpoint V5 API.

Note: Previous versions of this integration supported the Pinpoint Detect V2, V3, and V4 API, but the V5 API response format is different enough such that that a new PIP version was built to support the V5 API. Any new deployment of Verify Access integration with Pinpoint Detect should use a version of Pinpoint running the Pinpoint V5 API.

Integration Package Download

You should always obtain the latest version of the integration package from IBM Security App Exchange. Here is a short link: https://ibm.biz/isvatrusteer.

Integration Components

Verify Access Reverse Proxy

The Verify Access Reverse Proxy (a web reverse proxy) protects web applications and can be configured to perform the following tasks:

- Retrieve and enforce access decisions - which can be based on risk intelligence returned by the Pinpoint API.
- Optionally, inject client-side JavaScript snippets for Pinpoint Detect into application pages without the need to modify the web application.
- Provide a Permanent User ID (PUID) to Pinpoint for consistent session and user identification.
- Generate a unique Customer Session ID (CSID) for each user session (if necessary)
- Cache access decisions for a limited time to manage periodic re-evaluation
- Perform internal request redirection to display native web application error or status pages to provide a consistent user experience throughout the application access.

Verify Access Advanced Access Control (Authentication)

The Authentication Service in Verify Access Advanced Access Control (AAC) provides customizable authentication policies to be created. These are used to:

- Create an initial login policy which allows a Customer Session ID (CSID) from Pinpoint to be read via a Callback Handler on the login page.
- Create additional multi-factor authentication policies which are invoked when risk is detected by Trusteer

Verify Access Advanced Access Control (Authorization)

The context-based access functionality in Verify Access Advanced Access Control (AAC) adds advanced authorization capabilities.

The Verify Access AAC Trusteer Policy Information Point (Trusteer PIP) allows AAC Policies to initiate calls to the Pinpoint API, send session information, and request risk determinations. Once this integration component is configured, Verify Access AAC policies can be written which reference the risk determinations returned by Pinpoint Detect.

The Trusteer PIP supports retrieval of a risk determination at login time or at any point during a session when a sensitive action is being performed (updating contact information for example). The Trusteer PIP also supports requesting a risk determination for application-specific actions such as creating a new account, adding a payee, or triggering a financial transaction. For those Trusteer deployments which use Trusteer Pinpoint login confirmation, the Trusteer PIP supports an option where it will call the Pinpoint login confirmation API after a user completes a 2FA mandated because Trusteer returned a recommendation to authenticate.

In addition to basic function, the PIP and Verify Access Policy can work together to provide:

- Retrieval of values from external sources via HTTP/REST, Database, fixed mapping, etc. for Pinpoint API inputs such as action-specific data when these are not available in the client request.
- Runtime selection of the Pinpoint API Snippet ID for cases where multiple applications are deployed.
- Monitor-only mode where no 2FA access requirement is implemented, however the Pinpoint API is still invoked, creating Pinpoint alerts when Pinpoint API calls detect risks.
- Generation of a Verify Access audit record for each Pinpoint API call which can be forwarded to a remote syslog server, such as IBM QRadar, for further analysis and auditing.

Pinpoint Detect Snippet Service

The Pinpoint Detect Snippet Service hosts the JavaScript snippets that are loaded to the end user's browser and return data to Pinpoint for use in its risk analysis. The Snippet Service is also the destination for the data gathered by the snippets.

In the recommended integration pattern described in this document, access to the Snippet Service is NOT routed via the Verify Access proxy. An alternative deployment (where snippet traffic is proxied via the Verify Access Reverse Proxy) is described under Advanced Configuration.

Pinpoint API

The Pinpoint API is a RESTful API endpoint which clients can call in order to provide information about an active session and/or request a risk determination for a specific activity in the active session.

When Verify Access is integrated with Pinpoint Detect, the Trusteer PIP running in the Verify Access Advanced Access Control component manages all communication with the Pinpoint API.

Component Interaction

The following diagram shows the integration between the user's client device, Verify Access, Pinpoint Detect and the web application protected by Verify Access:

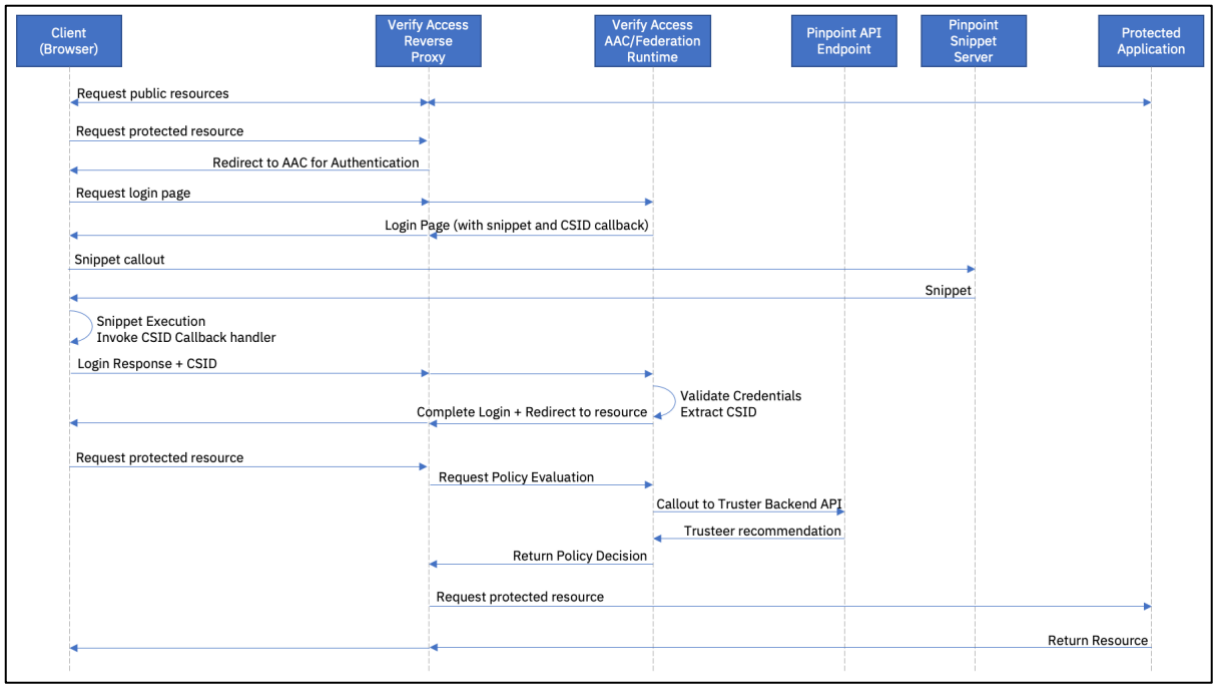

Figure 1 Trusteer Pinpoint Detect – Verify Access communication flow

1. The user accesses public resources within a protected application. The traffic is proxied via the Verify Access Reverse Proxy.
2. The user attempts to access a protected resource within the protected application. Verify Access returns its login page. This has the login page snippet imbedded and also a callback handler to receive a session ID (CSID) from Pinpoint.
3. The snippet code is executed in the browser. The snippet loads detection and collection JavaScript. These requests are made directly to the Pinpoint snippet servers.
4. The Pinpoint JavaScript code collects data and sends it to the Pinpoint snippet servers.
5. When the callback handler receives the CSID from Pinpoint, it adds the CSID to a hidden input in the login page.
6. The user submits authentication data to Verify Access. Authentication validation is completed in Verify Access. The CSID is populated to the user's Verify Access credential along with a Permanent User ID (PUID) associated with the user's account.
7. A Verify Access AAC policy is invoked which refers to the Pinpoint login recommendation. The Trusteer PIP is called to provide this recommendation and it calls out to the Pinpoint API to obtain a risk determination. The call includes the CSID and PUID.
8. The Pinpoint API looks up the session using the CSID and associates the provided PUID with the session. It returns a risk determination which includes a recommendation. This recommendation is returned to the AAC Policy Engine.

9. Verify Access AAC Policy is evaluated. The Policy may permit access, deny access, or trigger additional (multi-factor) authentication. If additional authentication is required, this must be performed before the policy will grant access.
10. Once the policy grants access, Verify Access requests the user-requested protected page from the protected web application.
11. The web application returns the requested page response.
12. Verify Access forwards the response to the client.

Pinpoint API Recommendation Mapping

Whilst the Pinpoint API response provides rich context, many of the attributes including the risk score are mapped to the single recommendation response item. The Verify Access JavaScript PIP for Pinpoint maps the recommendation received in the Pinpoint risk determination to a recommendation which can be used in the Verify Access AAC Policy.

When authoring Advanced Access Control (AAC) policy, use the literal string from the Verify Access AAC PIP Result from the table below. These default mappings produce PIP results in Verify Access AAC vocabulary, as shown in *Table 1 Pinpoint Recommendation Mappings*.

Pinpoint API Recommendation	Verify Access AAC PIP Result
allow_login	permit
allow_login_restrict	allow_login_restrict
authenticate	authenticate
authorize	authenticate
allow_transaction	permit
deny_transaction	error
alert	permit
allow	permit
deny_login	deny
deny	error
N/A	unavailable

Table 1 Pinpoint Recommendation Mappings

You can update the mappings as required by modifying the JavaScript Policy Information Point rule file and importing it again, replacing the contents of the previous version.

The PIP detects errors reported by the Pinpoint API and returns a mapped result to enable the use of Obligation Handlers allowing a path for error recovery, such as rerunning the detection snippets.

Pinpoint API Error	Verify Access AAC PIP Result
customer_session_id not found	unavailable
permanent_user_id not found	error

Table 2 Pinpoint Error Mappings

Planning the integration

This section describes the steps required when planning for integrating a protected application with IBM Security Verify Access and IBM Security Trusteer Pinpoint Detect.

DNS and Traffic Routing

In order for Verify Access to provide in-line protection of an application, all application traffic must be routed via the Verify Access Reverse Proxy.

The integration described in this document can work with both Standard and Virtual Host Junctions. However, where possible, it is recommended to use a Virtual Host Junction for the protected application. This approach will minimize integration complexity.

The following diagram shows how junctions are configured to route application traffic via the Verify Access Reverse Proxy:

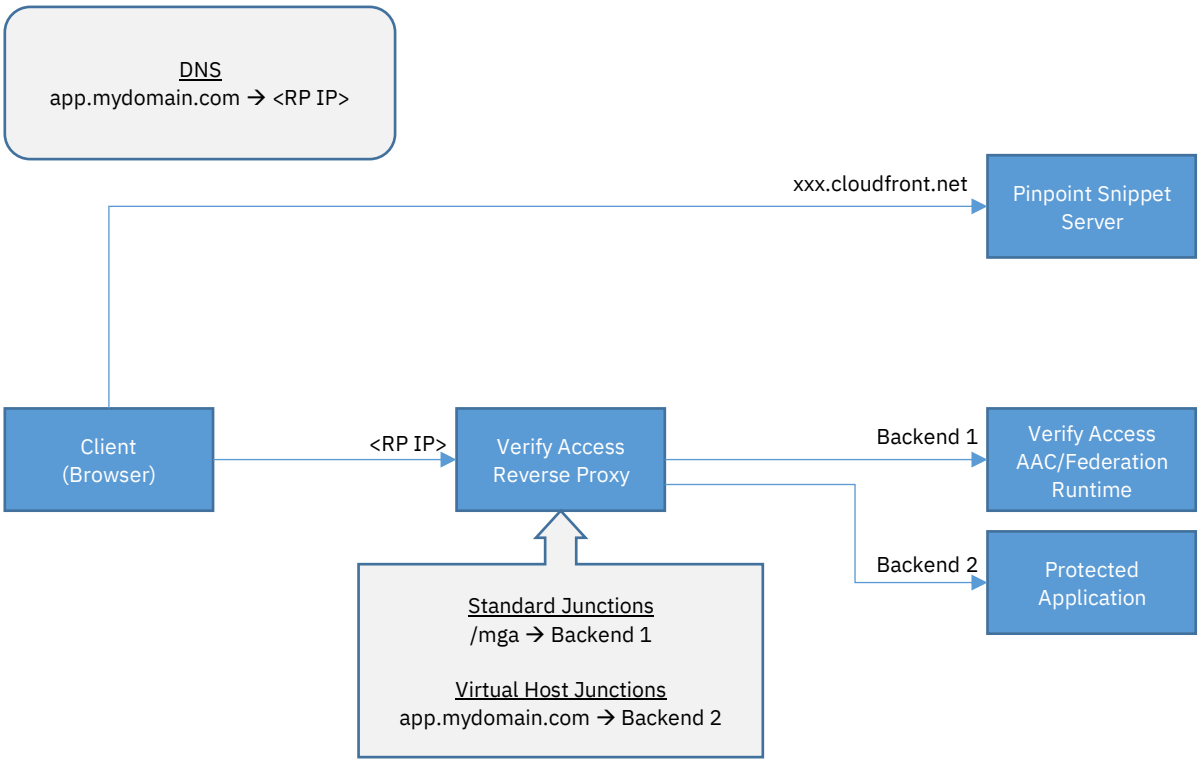

Figure 2 Request routing when NOT proxying snippet data

In the example above, the following URLs would be used to access resources:

- Protected Application: https://app.example.com/...
- Advanced Access Control Runtime: https://app.example.com/mga/...
- Pinpoint Snippet Server: https://xxx.cloudfront.net/...

To prevent browser warnings, you will need to obtain a CA issued certificate for the public fully qualified host name used to connect to the Reverse Proxy.

Required information and where to get it

The following information will be needed to configure the integration:

Item	Where to get it
Pinpoint snippet server DNS hostname.	Trusteer to supply
Pinpoint API server DNS hostname	Trusteer to supply
Pinpoint snippet ID	Trusteer to supply
PKCS#12 with Pinpoint API Client Certificate	Trusteer to supply (or sign certificate request)
Signer certificate for Pinpoint API Client Certificate	Trusteer to supply
Root CA certificate to validate Pinpoint API server	Trusteer to supply (or export from browser)
JavaScript snippets to be inserted into login page and protected application pages	Trusteer to supply
Locations (page and place in page) where snippets should be inserted.	Trusteer to supply

Table 3 Information required for integration

Verify Access Pre-requisite Configuration

This guide does not cover Verify Access configuration outside the scope of Verify Access and Pinpoint Detect integration. It is assumed that you have a deployed Verify Access environment and that the applications you want to protect are already integrated with the Verify Access Reverse Proxy. Help with setting up a Verify Access system can be found in the Verify Access Knowledge Center and on the IBM Security Learning Academy site.

Verify Access Knowledge Center:

- https://www.ibm.com/support/knowledgecenter/SSPREK/welcome.html

IBM Security Learning Academy courses:

- https://www.securitylearningacademy.com/course/view.php?id=4464
- https://www.securitylearningacademy.com/course/view.php?id=4412

The Verify Access system you integrate with Pinpoint Detect will need outbound Internet connectivity to the Pinpoint API server from the AAC Runtime component.

Recommended initial setup for an integration test

Anyone preparing to test this integration for the first time is advised to first install Verify Access and configure the Advanced Access Control (AAC) component. Next integrate a simple test application so that Verify Access performs its initial authentication and manages authenticated user sessions for the application.

Define at least two users in Verify Access who can log into the test application. Then configure Verify Access AAC so that it can force the users to do some sort of additional authentication in order to access at least one of the test application's pages. Perform all these steps before attempting to add integration with Pinpoint Detect.

Integration Package Installation

This section will describe the steps for installation of the IBM Security Verify Access Extension for IBM Security Trusteer Pinpoint Detect ("the extension").

The extension package (downloaded from the IBM Security App Exchange) is a ZIP file which contains the following files:

File Name	Description
`Trusteer Verify Access PIP Configuration Guide.pdf`	This Pinpoint Detect Integration Guide.
`trusteerPIPv3.0.1.ext`	The extension file (to be installed into an IBM Security Verify Access or IBM Security Access Manager system).

When the extension file is installed into a Verify Access (or Access Manager) system, it will perform the following actions:

- Create a new JavaScript Policy Information Point
- Create a set of Attributes for use in context-based access policies
- Add the PIP JavaScript and a set of sample files to the "File Downloads" directory

Installation

To install the extension, complete the following steps:

1. Extract the extension package ZIP file so that you have access the extension file.
2. In the Verify Access LMI, select **System→Updates and Licensing: Extensions** from the mega-menu.
3. Click **New**
    a. Click **Browse package**
    b. Select the **trusteerPIPv3.0.1.ext** file
    c. Click **Next**
    d. Click **Install**
4. **Review** and **deploy** pending changes

The extension installation writes to a system log file. On Verify Access, this file can be viewed by navigating to **Monitor→Application Log Files** and looking for **system→extension_install.log**.

On Access Manager system logs cannot be viewed via the LMI. However, they are included in a generated support file if it is configured to include system logs.

Validation

Policy Information Point

1. Navigate to **AAC→Policy: Information Points** in the LMI mega-menu. You should see the Trusteer PIP v3.0.1 listed. The description shows the build version of the extension.
2. Select the PIP and click the **modify** icon.
    a. Select the **Properties** tab. You should see a set of properties have been defined. Some are completed with default values and some are completed with NOT_SET.
    b. Click **Cancel** to close the modify dialog.
3. Click the **JavaScript** icon and select **View** from the drop-down menu.
    a. You should see the JavaScript of the PIP populated here.
    b. Click **Close** to close the view script dialog.

Attributes

1. Navigate to **AAC→Policy: Attributes** in the LMI mega-menu. You should see the following attributes defined:
    a. Pinpoint Detect Add Payee Recommendation
    b. Pinpoint Detect In-Session Recommendation
    c. Pinpoint Detect Login Recommendation
    d. Pinpoint Detect New Account Recommendation
    e. Pinpoint Detect Transaction Recommendation
2. Each of these attributes should be in the **Environment** category and have a data-type of **String**.

File Downloads

1. Navigate to **System→Secure Settings: File Downloads** in the LMI mega-menu.
2. Expand **access_control→examples→pip→trusteerPIPv3.0.1**. This folder was created by the extension installation.
    a. The **TrusteerPinpointPIP.js** file is a copy of the JavaScript used in the PIP. You can use this if you make changes to the PIP and want to restore a clean version.
    b. The **TrusteerPinpointPIP-Samples.zip** file contains samples referenced in this guide. To access, export the file and then expand it on your local machine. It contains the following files:

File Name	Description
`sample-aac-login-page.html`	Sample replacement login page for AAC password authentication mechanism. Includes logic for acquiring CSID via callback and placeholder for login snippets.
`sample-getCSID-auth-infomap.js`	Sample JavaScript for a custom authentication mechanism which captures CSID sent with login form. To be used with sample login page above.
`sample-transaction-pip.js`	Sample JavaScript PIP to collect transaction attributes for Transaction Recommendation assessment.
`sample-add-payee-pip.js`	Sample JavaScript PIP to collect payee attributes for Add Payee Recommendation assessment.
`sample-new-account-pip.js`	Sample JavaScript PIP to collect new account attributes for New Account Recommendation assessment.
`sample-dynamic-snippetid-pip.js`	Sample JavaScript PIP to enable dynamic resolution of the Snippet ID for multiple application support rather than a single configured value.
`sample-innocent-path-transform.xsl`	Sample Transformation rule which inserts the snippet ID into requests arriving at the snippet junction.

Table 4 Files in TrusteerPinpointPIP-Samples.ZIP

Uninstallation

If you want to uninstall the extension, complete these steps:

1. In the Verify Access LMI, select **System→Updates and Licensing: Extensions** from the mega-menu.
2. Select the **trusteerPIPv3.0.1** entry and click **Delete**.
    a. Click **Yes** to confirm the deletion.

The uninstallation performs the following actions:

- Remove the trusteerPinpointPIP directory from File Downloads
- Attempt to remove attributes added during installation
    - Any attribute still referenced in a policy will NOT be removed

Note: The installation does NOT remove the Policy Information Point (in order to retain configuration and customization). It can easily be manually removed from the Policy Information Point screen.

Basic Integration Configuration

Obtain and store required certificates

Your Verify Access system will make TLS connections to the Pinpoint API server.

To allow Verify Access to validate this Pinpoint server, the root CA Certificate used to sign the Pinpoint server certificate must be obtained. You should be able to download this CA certificate from the Pinpoint server using a browser (or other tools).

The Pinpoint API server uses client certificate authentication to authenticate connections. Before you can configure Pinpoint Detect integration, you must have a Pinpoint API client certificate and the associated Trusteer CA certificate. These will be provided by Trusteer.

To complete this section, you will need the following:

- The root CA that validates the Pinpoint API server certificate
- The Trusteer CA Certificate that signed the Pinpoint API client certificate
- A PKCS#12 file containing a signed Pinpoint API client certificate and private key
- The password protecting the PKCS#12 file

Create and open a new certificate database

You should create a new certificate database in Verify Access for the Trusteer PIP. This will contain the Pinpoint API client certificate, the CA root certificate that validates it, and the root CA certificate needed for Pinpoint API server validation.

Complete the following steps:

3. In the Verify Access LMI, select **System→Secure Settings: SSL Certificates** from the mega-menu.
4. Click **New**
    e. Enter a name for the keystore (e.g. *pinpoint*)
    f. Click **Save**
5. Select the certificate database you just created
6. Click **Manage→Edit SSL Certificate Database**

Import certificates for the Trusteer PIP

1. On the *Signer Certificates* tab, select **Manage→Import**
    a. Select the Root CA certificate for the Pinpoint API server certificate
    b. Enter a label (of your choice)
    c. Click **Import**
2. On the *Signer Certificates* tab, select **Manage→Import** again
    a. Select the Trusteer CA Certificate
    b. Enter a label (of your choice)
    c. Click **Import**

3. Select the *Personal Certificates tab.*
4. Select **Manage→Import**
    a. Select the PKCS#12 file containing your Pinpoint client certificate
    b. Enter the password for the file (if applicable)
    c. Click **Import**

Note: When importing a PKCS#12 file the label for the imported client certificate is taken from the *friendly name* in the PKCS#12 file. You can see the label in the UI after import, but it cannot be changed within the Verify Access LMI.

Configure AAC Authentication for CSID capture

In a deployment where Pinpoint Detect will generate the CSID, Verify Access must be able to capture this using a callback in the login page.

For username/password login, it is recommended to use AAC authentication, rather than the built-in Reverse Proxy login, so that the Customer Session ID (CSID) can be captured and added to user credentials.

Follow these steps to configure the username password authentication mechanism in AAC:

1. In the Verify Access LMI, select **AAC→Policy: Authentication** from the mega-menu.
2. Select the **Mechanisms** tab
3. Select the **Username Password** mechanism and click the **modify** icon.
    a. Select **Properties** tab
    b. Set the properties for you Verify Access directory
    c. Click **Save**

Follow these steps to create a new authentication mechanism which will collect the CSID (from a POST attribute named *qid* and add to token attributes:

1. In the Verify Access LMI, select **AAC→Global Settings: Mapping Rules** from the mega-menu.
2. Click **Add**
    a. Enter the following into the text box:

```
var csid =
context.get(Scope.REQUEST,"urn:ibm:security:asf:request:parameter","qid");
context.set(Scope.SESSION,

"urn:ibm:security:asf:response:token:attributes","csid",csid);
success.setValue(true);
```

    b. Enter **getCSID** as the Name
    c. Select **InfoMap** as the Category
    d. Click **Save**

3. **Deploy** pending changes
4. In the Verify Access LMI, select **AAC→Policy: Authentication** from the mega-menu.
5. Select **Mechanisms** tab
6. Click **New** icon and select **Info Map Authentication** from the drop-down menu.
    a. Enter **GetCSID** as the Name
    b. Enter **getcsid** as the Identifier
    c. Select the **Properties** tab
    d. Set the Mapping Rule to **getCSID**
    e. Click **Save**

Follow these steps to add a new authentication policy which performs username password authentication and captures the CSID:

1. In the Verify Access LMI, select **AAC→Policy: Authentication** from the mega-menu.
2. Click **New**
    a. Enter **Login_With_CSID** as the Name
    b. Enter **gologin** as the Identifier
    c. Click **Add Step**
        i. Select **Username Password**
        ii. Click **OK**
    d. Click **Add Step**
        i. Select **GetCSID**
        ii. Click **OK**
    e. Click **Save** (at the top of the screen)

Follow these steps to modify the login page HTML:

1. In the Verify Access LMI, select **AAC→Global Settings: Template Files** from the mega-menu.
2. Expand **C→authsvc→authenticator→password** folders
3. Select **login.html** from the password folder and click **Edit**
    a. Add the following callback function inside the first *<script>* section. Note that the name of this callback function (e.g. *thegenr*) will be provided by Trusteer:

```
function thegenr(qid) {
 document.getElementById("qid").value = qid;
}
```

    b. Add the following to the login form HTML:

```
<input type="hidden" id="qid" name="qid" value="">
```

   c. Click **Save**

4. **Deploy** pending changes

Reverse Proxy: Configure for AAC

Follow these steps to set up the Reverse Proxy to use AAC for Authentication and Context-based Access:

1. In the Verify Access LMI, select **Web→Manage: Reverse Proxy** from the mega-menu.
2. Select the Reverse Proxy and click **Manage→AAC and Federation Configuration→Authentication and Context-Based Access Configuration**.
   a. Select the **AAC Runtime** tab and enter details for your AAC runtime
   b. Click **Finish**
3. **Deploy** pending changes

AAC: Configure path kickoff for authentication

By default, authentication policies are triggered using a policyId query string. For new deployments it is recommended to use the path option instead. This is configured in AAC Advanced Configuration. Complete the following steps:

1. In the Verify Access LMI, select **AAC→Global Settings: Advanced Configuration** from the mega-menu.
2. Locate the **sps.authService.policyKickoffMethod** and set the value to **path**.
3. **Deploy** pending changes.

Reverse Proxy: General Configuration

Configure local response redirect for login

By default, the Verify Access Reverse Proxy will show its own internal login page when authentication is required. You will now configure it to use the AAC policy you created above instead. Make the following changes to the Reverse Proxy configuration file:

```
enable-local-response-redirect = yes

[local-response-redirect]
local-response-redirect-uri = [login] /mga/sps/authsvc/policy/gologin
```

Modify junction matching configuration

By default, the Verify Access Reverse Proxy will not match standard junctions if the requested virtual host matches a configured Virtual Host Junction. To allow requests to the Pinpoint snippet server (and the Verify Access AAC runtime) to work when connecting to an application using a Virtual Host Junction, make the following change in the Reverse Proxy configuration file:

```
match-vhj-first = no
```

With this configuration, requests where the URL path matches a configured standard junction will be routed to that junction instead of to the Virtual Host Junction.

Allow Session Sharing across Virtual Host Junctions

By default, the Reverse Proxy maintains independent sessions for each Virtual Host Junction and for the standard junctions. In order to allow a single session to be used across these different junctions, set the following in the Reverse Proxy configuration file:

```
shared-domain-cookie = yes
```

Note: This parameter should NOT be used if you enabled the Distributed Session Cache. In this case the Distributed Session Cache will handle session sharing as long as your Virtual Host Junctions are in the *default* DSC realm.

Reverse Proxy: Add snippets to login page

It is likely that you will want to include Trusteer login and carbon copy snippets on your login page. If Verify Access AAC is performing authentication, these snippets will need to be added to the Verify Access AAC login page.

Follow these steps to add the login snippet to the Verify Access AAC username password login page:

1. In the Verify Access LMI, select **AAC→Global Settings: Template Files** from the mega-menu.
2. Expand **C→authsvc→authenticator→password** folders
3. Select **login.html** from the password folder and click **Edit**
    a. Paste the snippets into the file at the appropriate locations
    b. Click **Save.**

Note: While updating the login page, Trusteer deployment technical staff may also direct you to make other changes necessary to optimize integration. For example, the addition of a **name** attribute to the login **<form**> element of the login page so that the form can be identified for keystroke analysis.

Reverse Proxy: Inject snippets into application pages

Load Snippets

To complete this configuration, you will need:

- The JavaScript snippets to be inserted into your protected application

Trusteer deployment technical staff will provide this information after analyzing the application to which Pinpoint Detect protection will be added.

Follow these steps to load the snippet files:

1. In the Verify Access LMI, select **Web→Manage: Reverse Proxy** from the mega-menu
2. Select the Reverse Proxy and select **Manage→Management Root**

3. Select the **snippets** folder and select **Manage→Import**
4. Select the snippet file to load and click **Import**
5. Repeat steps 3 and 4 for any additional snippet files

Figure 3 below shows the management root with some snippet files loaded:

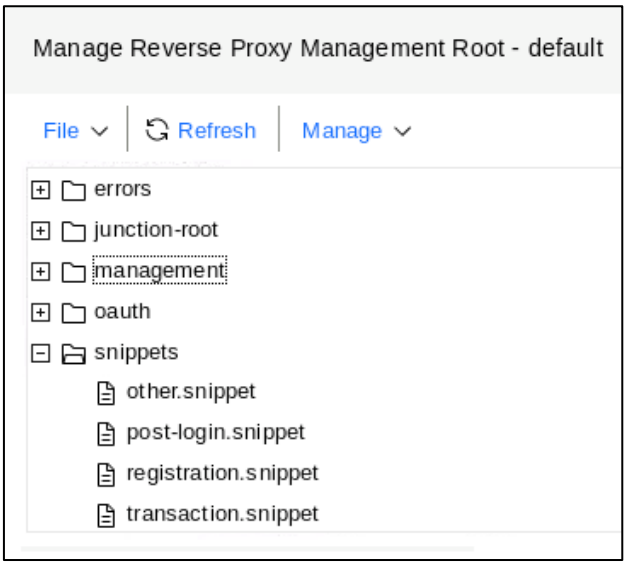

Figure 3 Snippet files available for insertion, under a Verify Access Reverse Proxy instance

Note: Snippets that will be added to the login page do not need to be loaded here as these will be directly added to the HTML for the Verify Access login page (see below).

Inject snippets into application pages

To add a Trusteer Pinpoint snippet to an application page, configure the Reverse Proxy to match the URL and then configure where in the page text the snippet should be inserted.

To complete this configuration, you will need:

- Locations (page and place in page) where snippets should be injected

In Verify Access (and Access Manager 9.0.7.0 and above), snippets can be injected within a line and can even replace existing content. These additional capabilities are often necessary if injecting into minimized or optimized pages.

Snippet injection is configured in the Verify Access Reverse Proxy configuration file. For each resource where snippets are to be injected, a stanza must be created. The name of the stanza includes the server-relative URI of the resource.

There is no special format for Virtual Host Junctions. The server-relative URI will match regardless of the hostname requested. Wildcards are not supported for the server-relative URI.

Whole-line match injection

When using whole-line matching, the snippet is injected into the returned content immediately before the matched line. Create a stanza with the following name in the Reverse Proxy configuration file:

[snippet-filter:<*server-relative URI*>**]**

For example:

[snipppet-filter:/myapp/homepage.html**]**

Within the snippet stanza there must be one or more entries of the form:

<pattern> **=** <*snippet filename*>

For example:

</head> = postlogin.snippet

The pattern can (and likely will) include one or more * wildcard characters. If the pattern includes an = it must be escaped to \=.

When the snippet filter matches the server-relative URI in a snippet-filter stanza, the Reverse Proxy looks for the first line in the returned content that matches the pattern of the first entry in the stanza. If found, the specified snippet is injected immediately before the matching line. The remaining lines of the returned content are then searched for the next entry in the stanza (if there is one).

Partial-line match injection

When using partial-line matching, the snippet is injected into the returned content immediately before (or replacing) the matching text. Create a stanza with the following name in the Reverse Proxy configuration file:

[snippet-filter:<*server-relative URI*>**:partial-line-match]**

For example:

[snipppet-filter:/myapp/homepage.html**:partial-line-match]**

Within the snippet stanza there must be one or more entries of the form:

<match-text> **=** <snippet filename> or
<match-text> **=** <snippet filename>**::replace**

Wildcard characters are not supported in the match text. If the pattern includes an = it must be escaped to \=.

When the snippet filter matches the server-relative URI in a snippet-filter stanza, the Reverse Proxy looks for the match-text of the first entry in the stanza. If found, the specified snippet is either injected immediately before, or replaces, the first instance of the match-text. The rest of the returned content is then searched for the next entry in the stanza (if there is one).

Trusteer PIP: Basic Configuration

The Trusteer PIP gets its configuration from properties configured in the Policy Information Point definition. Among other things, this configuration controls connection to the Pinpoint API server and information on how required attributes (such as CSID and PUID) are obtained.

The installation of the extension has created these properties, but values need to be provided.

To complete this section, you will need the following:

- The Hostname (or IP Address) of the Pinpoint API server (e.g. *1234.a.trusteer.com*)
- The keystore containing CA certificate(s) for server connection validation (e.g. *pinpoint*)
- The name of the key store containing the client certificate (e.g. *pinpoint*)
- The label of the client certificate to be used for authentication to the Pinpoint API server
- The snippet ID for your integrated application (e.g. *123456*)

Complete the following steps:

1. In the Verify Access LMI, select **AAC→Policy: Information Points** from the mega-menu.
2. Select the **Trusteer PIP** and click the **Modify** icon.
3. Select the **Properties** tab.
4. For each property (from table 3 below):
    a. Select the entry for the property
    b. Click the **modify** icon
    c. Enter the Value
    d. Click **OK**
5. Click **Save**.

Property Name	Value	Notes
backend_api_host	*e.g. 1234.a.trusteer.com*	
backend_api_request_timeout	***3***	
log_trusteer_response	**true**	When true, write each API response body to an audit record. **IBM recommends "true" both in lower environments and in production**.
backend_api_request_audit	**true**	When true, write each API request body to an audit record. **IBM recommends "true" both in lower environments and in production**.
cert_auth_client_key	*e.g. my_cert_label*	This is the label. If PKCS#12 file entry has no *friendly name* the Subject DN is set as the label on import. Format may be

Property Name	Value	Notes
		CN=acme,OU=Security,O=Trusteer Inc,ST=MA,C=US. This label must exist within key store identified by *cert_auth_keystore* below.
cert_auth_keystore	*e.g. pinpoint*	User-defined value set when creating key store.
customer_session_id_attribute	**csid**	
permanent_user_id_attribute	**AZN_CRED_PRINCIPAL_UUID**	This works when using standard users in Verify Access. For basic or external users, an alternative approach is needed.
snippet_id	*e.g. 123456*	
tls_channel_keystore	*e.g. pinpoint*	User-defined value set when creating key store.

Table 5 Trusteer Basic JavaScript PIP Properties

Perform Basic Integration Test

Pinpoint Detect integration with Verify Access is driven by Access Control policies. These are configured within the Verify Access Advanced Access Control (AAC) component.

In this section you will create a simple AAC Policy which will drive the Pinpoint APIs so that basic integration functionality can be tested and verified.

A policy will be created for which the access decision is dependent on the recommended action from Pinpoint Detect. This policy will be attached to a URL which is accessed after authentication.

When the recommendation from Pinpoint Detect is for additional authentication, Verify Access will enforce this requirement. This document shows use of Time-based One-Time Password as a 2nd Factor because it requires no configuration. You can use a different 2nd Factor if you prefer.

It is assumed that you have one or more test users defined in Verify Access that are registered for the 2nd Factor authentication you will use for this test.

Create Test Policy

You can now create the test Access Control Policy. This policy will trigger the Trusteer PIP which will make a REST call to the Pinpoint API to request a recommendation for the current session. Based on the recommendation, the policy will either allow access, trigger authentication, or deny access.

Note that the PolicyId and Authentication Policy used here must match what was specified for *required_auth_policy* when configuring the PIP.

Complete the following steps:

1. In the Verify Access LMI, select **AAC→Policy: Access Control** from the mega-menu.
2. Click the **Create** icon.
3. Give your policy a name (e.g. *Test-Login-Policy)*
4. Change the **Precedence** to **First**.
5. Click **Add Rule**.
    a. Add this rule:
    If **All** are true (
    Pinpoint Detect Login Recommendation = **permit**
) Then **Permit**
    b. Click **OK**.
6. Click **Add Rule**.
    a. Add this rule:
    If **All** are true (
    Pinpoint Detect Login Recommendation = authenticate
) Then **Permit with *Authentication TOTP One-time Password***
    b. Click **OK**.
7. Click dropdown on **Add Rule** button and select **Unconditional rule**.
    a. Add this rule:
    Deny

b. Click **OK**.

8. Scroll to the top of the page and click **Save**.

If you view the policy, it should look like this:

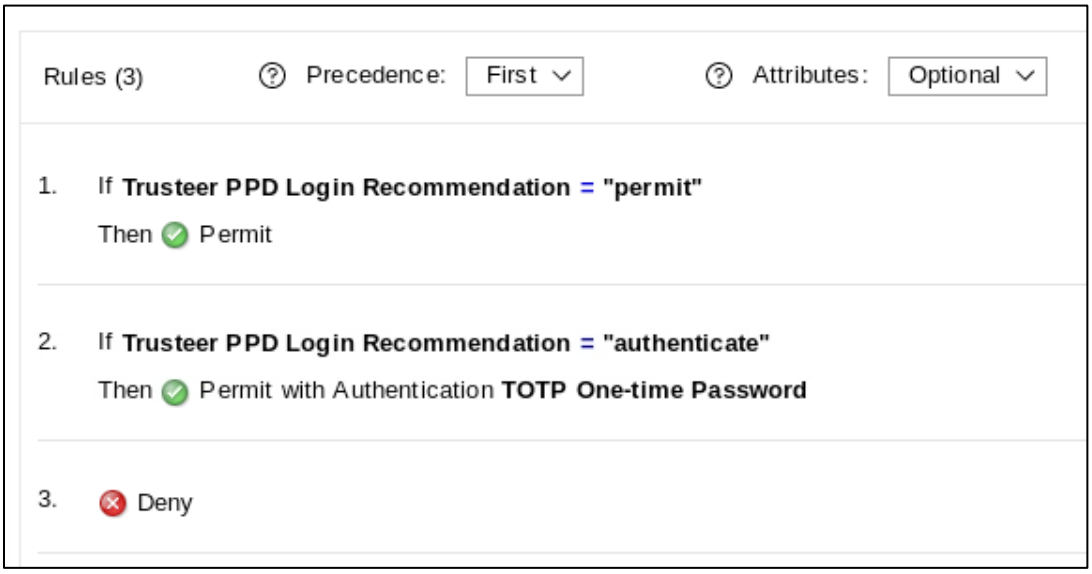

Figure 4 Sample AAC Policy using Pinpoint Detect recommendation

Attach Test Policy

An AAC Access Control Policy is invoked when the resource to which it is attached, or a child resource with no other policy attached, is requested. You will need to identify a resource (either a page or folder) where you want the Trusteer policy to be invoked.

Add Resource

Complete the following steps:

1. In the Verify Access LMI, select **AAC→Policy: Access Control** from the mega-menu.
2. Select the **Resources** tab.
3. Click the **Add** icon. If this is the first time you've added a resource, you will be required to provide the credentials for a Verify Access administrator (e.g. *sec_master*)
    a. Check Type is **Reverse Proxy**
    b. Select the Reverse Proxy Instance that protects your application
    c. Enter the path for the resource where the policy should be attached. For an application connected with a Virtual Host Junction, this will have the form: **/@***<junction name>*/<resource path>
    d. Click **Save**

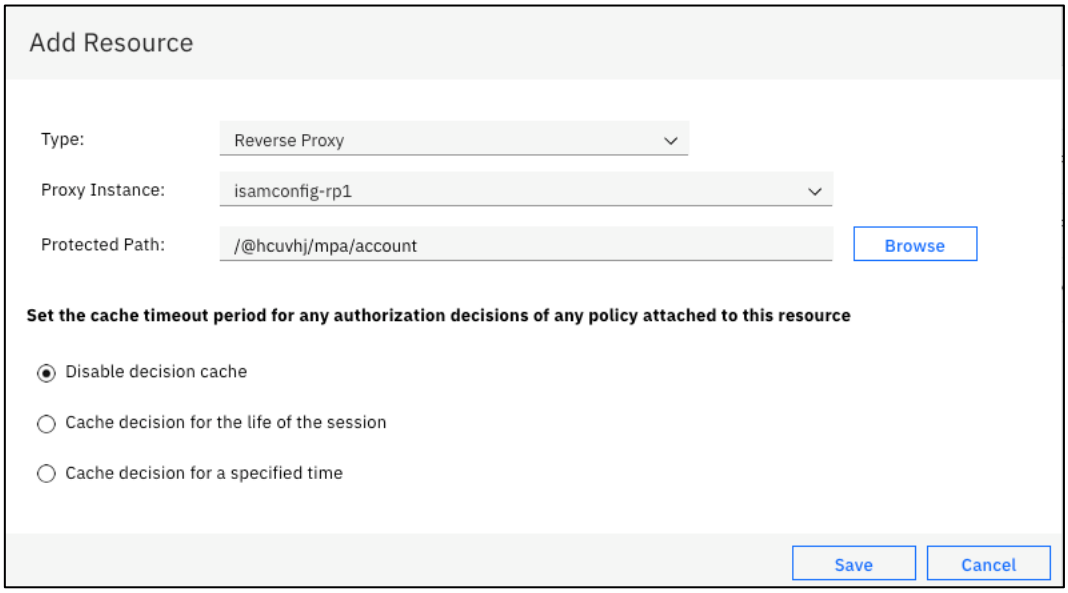

Figure 5 Create Policy Attachment Resource

Attach Test Policy and Publish

Complete the following steps:

1. Select the newly added resource
2. Click the **Attach** button
    a. Select you test policy from the list
    b. Click **OK**
3. Click **Publish** (and confirm).

Figure 6 Pinpoint Detect Login Recommendation Policy Attachment

Run Test

You are now ready to test Verify Access integration by accessing your test application, authenticating on the Verify Access login page, and then accessing the page where the test policy is

attached. It is assumed that you have already tested your environment to ensure that basic AAC functionality is working.

During the evaluation of the Test Policy, the Trusteer PIP will be invoked to request a recommendation from Pinpoint Detect. Depending on this recommendation you may be permitted access, denied access, or you may receive a challenge for 2nd Factor Authentication.

Trusteer deployment staff can advise on ways to trigger different recommendations for testing.

Complete the following steps:

1. Open a new browser window and enable browser web developer tools so you can see the network requests being made by the browser.
2. Navigate to a protected page of your test application via Verify Access. (e.g. *https://app.domain.com/protected/page.html*).

   You should receive the Verify Access login page. If you don't see the Verify Access login page, make sure your DNS is configured so that your traffic is routing to the Verify Access Reverse Proxy.
3. Review the traffic shown in the browser web developer tools.

   You should see Pinpoint Detect collection snippets being loaded and collected data being sent. This traffic can be identified by its use of the innocent path in the URL.
4. Login to Verify Access and request the resource that triggers the Test Policy.

   You should be permitted access, denied access, or challenged for 2nd Factor Authentication based on the recommendation received from Pinpoint Detect.
5. Logout of Verify Access by navigating to *https://app.domain.com/pkmslogout*.

Test In-Session Recommendation

You can test the In-Session Recommendation by creating a similar policy as the one described above except that it should reference the in-session recommendation attribute instead of the login recommendation attribute.

Create a resource to represent the URL where you want to enforce the recommendation, attach the policy to it, and publish the policy.

You can then test the policy by navigating to the URL.

Additional Configuration

This section contains additional configuration that you might want to configure once you have the basic integration tested and working.

Specifying Specific Pinpoint API version level

By default, and under typical usage, one calls the Trusteer Pinpoint API (the V5 API) without specifying any particular level of the Pinpoint V5 API, and the most current level will be used. In certain cases, Trusteer may instruct you to specify a certain release level of the V5 API. To do this you can use the optional PIP property **backend_api_version**. As an example, setting this property to "v5/1" will cause the older release 1 level of the Trusteer V5 API to be used. This property should only be used (and is only needed) when given direct guidance from Trusteer to do so.

Enable Login Confirmation

When to use Login Confirmation

Pinpoint Detect can support receipt of an explicit login confirmation notification when an end user successfully completes a required 2nd Factor Authentication challenge. This allows Pinpoint Detect to know when a user has completed 2FA, or to imply when a user has not completed a required 2FA, and to adjust its risk assessment accordingly.

Your Trusteer deployment team will inform you if login confirmation is enabled for your Trusteer-protected application.

Enabling Login Confirmation

If login confirmation is enabled in Pinpoint Detect, you must also enable it for the Trusteer PIP. In the Trusteer PIP configuration properties, set the **login_confirmation_enabled** property to **true**. In most cases, this is all you need to do.

When this PIP option is active, if your Verify Access AAC policy requires the user to complete a 2FA, and the user does so successfully, and Trusteer Pinpoint returned a recommendation of "authenticate", the Trusteer PIP will call the Trusteer Pinpoint login confirmation API.

The Trusteer PIP will not call the login confirmation API when a recommendation other than "authenticate" is returned by Trusteer Pinpoint, even if the user completes a 2FA for other reasons. (This would not be considered a 2FA that was driven by a Trusteer recommendation).

Note that Trusteer Pinpoint can return a recommendation of "allow_login_restrict" to indicate a risk level higher than "authenticate". This provides guidance to not allow access even if a user completes a 2FA. The Trusteer PIP does not call the Trusteer login confirmation API if your AAC policy permits access after requiring a user to complete a 2FA when the recommendation is "allow_login_restrict".

Login Confirmation Advanced topics

If your AAC policy uses Trusteer recommendations in a conventional fashion, and requires 2FA only based on whether Trusteer Pinpoint returns a recommendation of "authenticate", then you do not need to read the rest of this paragraph. But if your AAC policy requires the user to complete a 2FA for some other reason, and the Trusteer recommendation was not "authenticate" (or mapped to

"authenticate" via a customized PIP recommendation mapping table), then the Trusteer PIP will not call the Trusteer login confirmation API.

The Login Confirmation function of the Trusteer PIP stores DMAP entries in the Access Verify runtime database, also known as the HVDB. The Trusteer PIP also uses these DMAP entries to prevent unnecessary extra calls to the Trusteer API after a user completes a 2FA, even if the Trusteer PIP's Login Confirmation capability is not enabled. It is particularly important to run performance tests of the solution before going live in production if the Verify Access runtime database (HVDB) has not been externalized to a database such as DB2.

Trusteer Recommendation Types and Reuse (Advanced)

If your AAC policies do not permit access without requiring 2FA, when Trusteer returns a recommendation of authenticate, then you do not need to read or understand this section. Most decision caching scenarios should use the method described in "AAC Policy Decision Cache".

Because of how the Trusteer PIP uses DMAP entries (see "Login Confirmation Advanced Topics"), if you use an AAC policy which does **not** require 2FA for a Trusteer recommendation of authenticate, and you use the same type of recommendation (login, in-session, or action-specific) in two different AAC policies with two AAC resources, this will enable a narrow type of caching: The Trusteer recommendation of authenticate can be reused when the user visits the second AAC resource.

This function exists to support a specific use case, when it is desired to not immediately enforce 2FA at the time of login, even though moderate risk was detected. Careful planning should be used when choosing to use an AAC policy which permits access without requiring 2FA when Trusteer returns a recommendation of authenticate. But you can make things simple, even for this case, if you do not use a the same recommendation type (i.e. "login") in any of your other AAC policies under the same Trusteer snippet ID.

AAC Policy Decision Cache

For performance, and to eliminate the invocation of the Pinpoint APIs on each page load, the AAC Policy Decision Cache in Verify Access should be configured for the login and in-session policies.

The policy decision cache is enabled at the AAC resource level. When the policy decision cache is enabled for an AAC resource, permit and deny decisions associated with that resource will be cached so that the attached policies do not need to be re-evaluated until the cache expires.

Note: Only simple permit and deny results are cached. Decisions that include an authentication requirement or an obligation are not cached.

The policy decision cache can be tuned for each resource. A decision can be cached for the lifetime of the session or for a fixed time period.

- Decisions based on the login recommendation should be cached for the session lifetime.
- Decisions based on in-session recommendations may be cached for a short time so that they are re-evaluated regularly (but not on every single resource load). Whether to use the AAC policy decision cache for in-session recommendations may depend on the specifics of the application, and what the in-session recommendation from Trusteer is protecting.
- Decisions based on action-specific recommendations should not be cached. This will ensure that the Pinpoint Detect API is called again, so that risk is re-evaluated for every action.

To enable the policy decision cache, follow these steps:

1. In the Verify Access LMI, select **AAC→Policy: Access Control** from the mega-menu.
2. Select the **Resources** tab.
3. Select the resource for which you want to change the decision cache settings
4. Click the **Modify Resource** button
    a. Select the cache option you want to use
    b. Click **Save**.
5. Click **Publish** to publish the updated policy.

Returning Custom Content

In order to provide a better user experience, or better integration with an application, you may want to return custom content to the client rather than a standard HTML page when access is not being permitted. Verify Access supports this through the use of obligations.

Returning an obligation to the Reverse Proxy tells it to redirect to a configured URL. This redirect can be performed via a 302 to the client or it can be followed internally so the client is not aware of it.

To use an obligation, the following steps must be completed:

- (optionally) upload content to return as a static template page.
- Register an obligation (name and identifier) in Advanced Access Control
- Create a mapping from obligation identifier to URL in Reverse Proxy configuration
- (optionally) configure internal redirection

Optional: Upload static content to be returned

If you want to return a static page to the client in the event of access being denied, you can load this page into the Advanced Access Control template pages.

Complete the following steps to import a static resource:

1. In the Verify Access LMI, select **AAC→Global Settings: Template Files** from the mega-menu.
2. Expand **C** folder
3. Select the **static** folder
4. Click **Manage** and select **Import** from the drop-down menu
    a. Select the file to upload
    b. Click **Import**

Note: the file suffix is used to set the content-type when sending this file to a client. Ensure that the file has an appropriate file suffix (e.g. .html for an HTML file or .json for a JSON file). Suffix to content-type mapping can be customized in the **C/metadata.xml** file.

Create an Obligation Definition

Each unique location you want to redirect to from an Advanced Access Control policy must have a unique obligation definition.

Complete the following steps to create an Obligation definition:

1. In the Verify Access LMI, select **AAC→Policy: Obligations** from the mega-menu.
2. Click **New Obligation** icon and select **Enforcement Point** in the drop-down menu.
    a. Enter a human-readable name. This is shown in the Policy Editor (e.g. *Show Error Page*)
    b. Enter an identifier. It is recommended to use a URN format to avoid name collisions (e.g. *urn:example:trusteer:error*)
    c. Click **Save**.

Map Obligation Identifier to Redirect URL

The mapping from obligation identifier to redirect URL is held in the [obligations-urls-mapping] stanza of the Reverse Proxy configuration file. Entries have the form:

```
<identifier pattern> = <redirect URL>
```

where identifier pattern can include ***** as a wildcard and redirect URL can be a server relative or absolute URL. To refer to resources loaded as static Template pages, use **/mga/sps/static/<*file*>**.

For example:

```
[obligations-urls-mapping]
urn:example:trusteer:error = /mga/sps/static/error.html
```

Optional: Enable internal redirects

Note: this configuration requires a fix available from IBM Support if the error page is hosted as a static file on the AAC.

By default, an obligation will result in the Reverse Proxy generating a 302 Redirect to the specified URL. However, there may be cases where you don't want a redirect to occur (either because you want to hide it or the client cannot handle it).

It is possible to configure the Reverse Proxy to follow the redirect internally and only return the resulting response to the client.

First, enable internal redirects by setting the number of redirects to follow to something greater than zero. Then add a *follow-redirects-for* entry to the [server] stanza.

Make the following change to the Reverse Proxy configuration file:

```
maximum-followed-redirects = 1

follow-redirects-for = !LOCATION! /mga/sps/static/error.html
```

Update and re-publish policy

To return the custom error page, modify the AAC Authorization Policy to return **Deny with Obligation** instead of just **Deny**. Select the obligation you created above.

After saving the updated policy, go to the resources tab and re-publish.

Configuring Action-specific Recommendations

In addition to requesting a basic "login" or "in-session" recommendation, Pinpoint Detect can also accept additional information regarding targeted application actions and provide action-specific recommendations based on this additional data. Typical actions related to banking use-cases are:

- Create a New Account
- Add a Payee
- Perform a Transaction

These action-specific recommendations are requested (via the Trusteer PIP) in the same way as a login or in-session recommendation but Verify Access must provide additional action-specific information in the call to the Pinpoint API.

The action-specific information is provided by the following process:

- The Reverse Proxy extracts attributes from the incoming HTTP request and sends to the AAC Authorization engine.
    - The Reverse Proxy must be configured to identify the required attributes
- The AAC Authorization engine receives the attributes as part of the decision request context
    - The Attributes must be defined in the AAC Authorization Engine
- The attributes must be transformed to a Trusteer-defined format
    - A Custom JavaScript PIP must be defined which can perform the required formatting
- The Trusteer PIP will request the standard format attribute during it's processing
    - The standard format attribute must be defined

Every application is different. This section can only provide examples; you will need to modify them to work with your particular application.

Reverse Proxy: Extract attributes from requests

Identify Attributes

Before attributes can be extracted from the application flow, you will need to understand how they are presented. Access Manger can extract from the following sources:

- HTTP Headers
- Cookies
- URI query parameters
- Post Data (url-form-encoded or JSON)

A good way to investigate the data available in HTTP requests is to use the built-in web developer tools provided with your browser. These will usually have a "Network" view which allow you to inspect the headers, cookies, and body of all requests made by the browser.

Here is an example of a POST request with a JSON body that might be used when adding a payee:

```
POST /bankapi/addpayee

{"payee":{"country":"GB","name":"John Smith","sortcode":"12-34-
56","acc":"01234567"}}
```

Configure Attribute Locations

In the Reverse Proxy configuration file, add entries for each attribute you need to extract. When working with JSON data, you can choose to pull in each element individually or just get a parent element (and parse it later).

It is recommended to use a URN format for attribute names to avoid name collisions.

To extract data from the example JSON body above, you would add the following to the Reverse Proxy configuration file:

```
[azn-decision-info]
urn:example:payee:country = post-data:/"payee"/"country"
urn:example:payee:bankid = post-data:/"payee"/"sortcode"
urn:example:payee:accountid = post-data:/"payee"/"acc"
```

Set Attribute Data Type and Category

Each attribute that the Reverse Proxy sends in the decision request context must have a data type and category. In most cases the type used will be *string* and the category will be *Resource*.

For the example data above, you would add the following to the Reverse Proxy configuration file:

```
[user-attribute-definitions]
urn:example:payee:country.type = string
urn:example:payee:country.category = Resource
urn:example:payee:bankid.type = string
urn:example:payee:bankid.category = Resource
urn:example:payee:accountid.type = string
urn:example:payee:accountid.category = Resource
```

Note: The name, type, and category of each attribute will be needed when defining the attributes for use in Advanced Access Control.

AAC: Create custom PIP to format data

When the Trusteer PIP receives a request for a recommendation which requires context data to be sent in the request to the Pinpoint API, it will make a request for the attribute configured for the requested action:

Action	Default Attribute
Add a payee	urn:ibm:security:trusteer:pinpoint:cd:payee:data
Perform a transaction	urn:ibm:security:trusteer:pinpoint:cd:transaction:data

Action	Default Attribute
Create a new account	urn:ibm:security:trusteer:pinpoint:cd:newaccount:data

Table 6 Default attribute names for context-based recommendation data

You will need to create a custom JavaScript PIP which will return the required attribute when it is called. A sample JavaScript is provided for each of the above attributes but it will need modification to work with your application.

Write JavaScript code

The JavaScript for the PIP must expose two methods:

- [Boolean] function hasAttribute (requestedAttribute, category)
- function getAttributes (context, requestedAttribute, category)

The *hasAttribute* function can be very simple. It just needs to check whether the requested attribute matches the attribute it can supply. For example:

```
function hasAttribute (requestedAttribute, category) {
    if ("urn:ibm:security:trusteer:pinpoint:cd:payee:data"
         .equals(requestedAttribute.getURI()))
    {
         return true;
    }
    return false;
}
```

The *getAttribute* function will be application specific. It must request the decision context attributes that hold the input data (the ones that are extracted and sent by the Reverse Proxy) and then add an attribute to the supplied context with a value correctly formatted for the Pinpoint API.

This is an example of the format required by Pinpoint Detect for the *Add Payee* data:

```
[{
  "target_country_code": "…",
  "target_bank_identifier": "…",
  "target_account_identifier": "…"
}]
```

A helper class is available to retrieve attributes from the decision context. The following call would obtain the value of the *urn:example:payee:bankid* attribute used in the examples above.

```
var bankId = getContextAttributeValue(
                context,
                "urn:example:payee:bankid",
                Attribute.DataType.STRING,
                null,
```

```
                    Attribute.Category.RESOURCE);
```

Here is some example code which would create a string of required data format:

```
var lDataBody = [];
var lData = {};
lData["target_country_code"] = countryCode;
lData["target_bank_identifier"] = bankId;
lData["target_account_identifier"] = accountId;
lDataBody.push(lData);
var dataString = JSON.stringify(lDataBody);
```

Here is example code to create an attribute containing this string and add it to the input context:

```
  var addPayeeDataAttributeIdentifier = new AttributeIdentifier(
            "urn:ibm:security:trusteer:pinpoint:cd:payee:data",
            Attribute.DataType.STRING,
            null);
  context.addAttribute(addPayeeDataAttributeIdentifier, [dataString]);
```

Create JavaScript PIP

Once you have created the JavaScript which implements the PIP interface, you can use it to create a JavaScript PIP.

Complete the following steps:

1. In the Verify Access LMI, select **AAC→Policy: Information Points** from the mega-menu.
2. Click the **New** icon and select **JavaScript** from the drop-down menu.
    a. Enter a name (of your choice)
    b. Browse for, and select, the JavaScript file
    c. Click **Save**

Testing

Testing this policy is similar to testing the login policy.

You must create a Resource in Advanced Access Control for the URL which triggers the action you are protecting. This is also the URL to which the parameters you are extracting in the Reverse Proxy are being sent.

You must create an Advanced Access Control Policy which queries the value of the recommendation for the application action. This is one of:

- Pinpoint Detect New Account Recommendation
- Pinpoint Detect Add Payee Recommendation
- Pinpoint Detect Transaction Recommendation

Apart from the use of this action-specific recommendation, the policy can be the same as the one used at login time.

Advanced Functionality

This section describes additional functionality that you may need to implement in order to integrate with applications that do not work with the basic integration techniques described above. This section assumes a more advanced knowledge of IBM Security Verify Access and IBM Security Trusteer Pinpoint Detect.

Verify Access Proxies Snippet Traffic

In some cases, it may be desirable to route snippet traffic via the Verify Access Reverse Proxy. The following diagram shows how DNS is configured to route all traffic via the Verify Access Reverse Proxy:

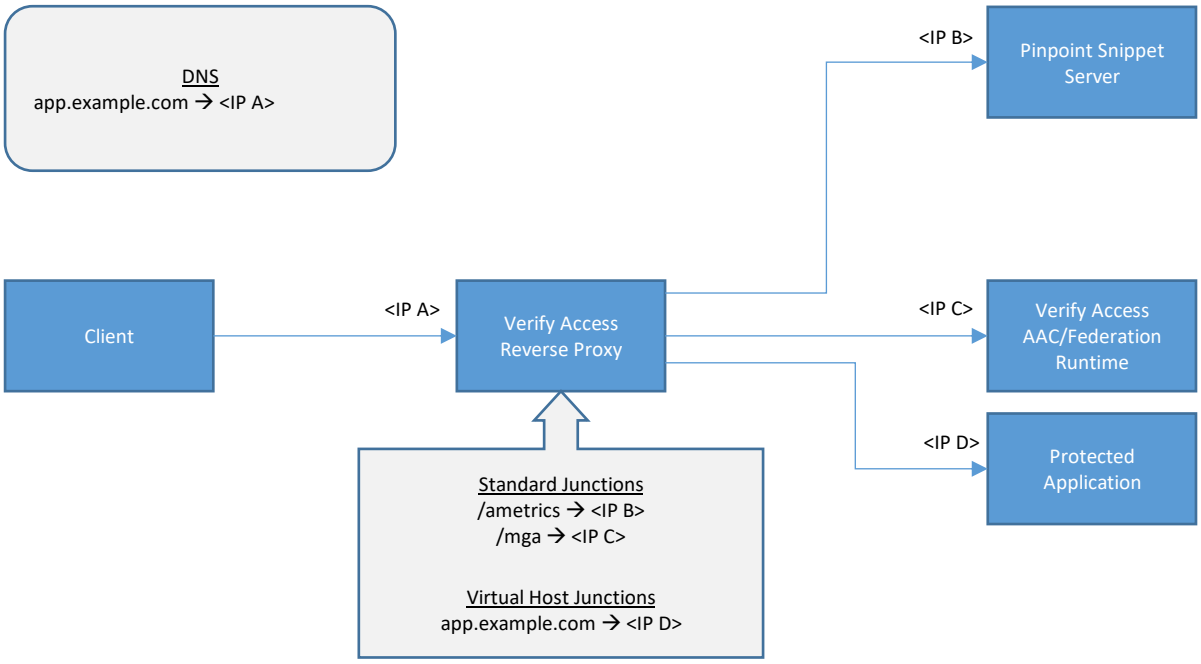

Figure 7 Request routing when proxying snippet data

In the example above, the following URLs would be used to access resources:

- Protected Application: https://app.example.com/...
- Pinpoint Snippet Server: https://app.example.com/ametrics/...
- Advanced Access Control Runtime: https://app.example.com/mga/...

The path prefix used in the Pinpoint Snippet Server URL (*ametrics)* is just an example. Usually a unique string will be chosen. This string must not clash with URLs used by the protected application and should not make it obvious that Pinpoint Detect is in use. It is known as the "innocent path".

Import certificates for the Reverse Proxy

The Reverse Proxy will be pre-configured with a certificate database (default is *pdsrv).*

Complete the following steps to load the required Trusteer certificates to this file:

1. In the Verify Access LMI, select **Manage System Settings→Secure Settings: SSL Certificates** from the mega-menu.
2. Select the Reverse Proxy certificate database (e.g. *pdsrv*)
3. Click **Manage→Edit SSL Certificate Database**
4. On the *Signer Certificates* tab, select **Manage→Import**
    a. Select the Root CA certificate for the Pinpoint snippet server certificate
    b. Enter a label (of your choice)
    c. Click **Import**
5. On the *Signer Certificates* tab, select **Manage→Import** again
    a. Select the Trusteer CA Certificate
    b. Enter a label (of your choice)
    c. Click **Import**
6. Select the *Personal Certificates tab.*
7. Select **Manage→Import**
    d. Select the PKCS#12 file containing your Pinpoint client certificate.
    e. Enter the password for the file (if applicable)
    f. Click **Import**

Reverse Proxy: Connect Pinpoint snippet server

To allow the Verify Access Reverse Proxy to route snippet traffic (for script download and data upload) you must create a junction to the snippet server. This should be a standard (non-transparent) junction. The junction name is the "innocent path" agreed with Trusteer.

To complete this configuration, you will need:

- The DNS address (or IP address) of the Pinpoint snippet server (e.g. *1234.a.trusteer.com*)
- The "innocent path" to be used with the Pinpoint snippet server

Follow these steps to create the junction:

1. In the Verify Access LMI, select **Secure Web Settings→Manage: Reverse Proxy** from the mega-menu
2. Select the Reverse Proxy and select **Manage→Junction Management**
3. Select **New→Standard Junction**
    a. Select **SSL** as the Junction Type
    b. Enter ***/<innocent-path>*** as the Junction Name (e.g. */ametrics*)
4. Select the **Servers** tab and click **New**.
    a. Enter the DNS address of the Pinpoint snippet server
    b. Click **Save**.
5. Select the **Basic Authentication** tab

a. Check the **Enable mutual authentication to junctioned WebSEAL servers** checkbox.

b. Select your Pinpoint client certificate from the Key Label drop-down list (the list may take a few seconds to populate).

6. Select the **Identity** tab

   a. Set HTTP Basic Authentication Header to **Ignore**.

7. Click **Save**.

Support IP address change for snippet service

When connecting to internet-facing services, it is wise to support the case where the DNS address of the service may be dynamically directed to a new IP address. To support this for the snippet service, add the following stanza to the Reverse Proxy configuration file (replacing */ametrics* with your own "innocent path"):

```
[junction:/ametrics]
dynamic-addresses = yes
dynamic-addresses-ttl = 30
```

The time-to-live (ttl) is set to 30 seconds in the example above. You may wish to tune this for your environment balancing the number of DNS calls with the time taken to identify a change.

Allow unauthenticated access

Pinpoint Detect snippet download and data upload is performed before users are authenticated to Verify Access. This means that Verify Access security policy must allow unauthenticated access to the Trusteer snippet server.

Perform the following actions in Verify Access:

1. Create a new Access Control List which grants the following permissions:
   - sec_master TcmdbsvaBRrxl
   - iv-admin TcmdbsvaBRrxl
   - webseal-servers Tgmdbxrxl
   - Any-other Trx
   - Unauthenticated Trx
2. Attach this ACL to /WebSEAL/<*webseal-server*>/<*innocent-path*>

Validation

When using a standard (non-transparent) junction, Verify Access removes the junction name from the requested URL path when forwarding to the backend server. For example, a request for */ametrics/123456/somescript.js* becomes */123456/somescript.js* when forwarded to the Pinpoint snippet server.

Using a browser, attempt to retrieve a resource via the new Junction (e.g. *https://app.example.com/ametrics/123456/somescript.js*). You should, at least, get back an empty response. If you specify the URL of a valid snippet resource, you should receive that file.

If you receive a 400 "SSL Connection Error" error, check that you have imported your Pinpoint client certificate into the Reverse Proxy key store and set it for use with mutual authentication in the junction configuration.

Forward Client IP address

Pinpoint Detect uses the source IP address received by the snippet server within its risk calculations. The Verify Access Reverse Proxy must be configured to forward the source IP address it receives at the network level in the *X-Forwarded-For* HTTP header.

Make the following change in the Reverse Proxy configuration file:

```
[header-names]
client-ip-v4 = +X-Forwarded-For
```

Alternative CSID handling

Pinpoint Detect links requests made to the Pinpoint API with a particular browser session using a "Customer Session Identifier" (CSID). This must be transferred over the browser session (so it is linked with the data gathered by the snippets) and must also be available to whatever is calling the Pinpoint API (i.e. the Trusteer PIP).

In the recommended integration, Pinpoint Detect creates the CSID and executes a callback on the login page so that it can be captured during the login process. This method works for most use cases where an interactive initial login is performed.

The sections below detail alternative approaches to CSID handling which might be more suited to some environments (as per the recommendation of the Trusteer deployment team).

Verify Access is using built-in Reverse Proxy login form

If Verify Access is using the built-in Reverse Proxy login form, it is not possible to add the CSID to the user credential during the login process. In this case, the callback on the login form must set a cookie containing the CSID which can then be extracted by the Reverse Proxy and sent to the AAC authorization engine in subsequent calls that require risk determination.

This JavaScript must be added to the header of the login.html page that is presented by the Reverse Proxy before the other login page snippets. The name of this callback must be agreed with Trusteer.

```
<script>
  function mycallback(qid) {
    document.cookie = "qid=" + qid;
  }
</script>
```

This script sets a cookie (named *qid*) with the value of the CSID. Extracting and passing the value of this cookie to the Trusteer PIP is achieved in the same way as described for passing request data for application-specific recommendations. A default attribute, *urn:ibm:security:trusteer:pinpoint:csid* is already defined for this purpose.

In the Reverse Proxy configuration file, add an entry for the cookie you need to extract.

```
[azn-decision-info]
urn:ibm:security:trusteer:pinpoint:csid = cookie:qid
```

In the Trusteer PIP configuration properties, set the **customer_session_id_attribute** property to **urn:ibm:security:trusteer:pinpoint:csid**.

Verify Access is using a custom EAI authentication method

If you are using a custom authentication application integrated using the External Authentication Interface (EAI), then the preferred method of integration would be to modify this application so that it collects the CSID in the same way as described in the recommended integration approach. That means adding a hidden input to the login page which is populated with the CSID from Trusteer using a callback handler. The custom authentication application would then extract the CSID from the login POST and add it as an extended attribute when returning the EAI response.

If the custom authentication application cannot be modified, you should use the Reverse Proxy snippet functionality to inject a callback handler into the login page to set a cookie holding the CSID. This can then be extracted by the Reverse Proxy and sent to the AAC authorization engine in subsequent calls that require risk determination. This is the same approach as described for the Reverse Proxy built-in login form.

Verify Access owns the CSID

In some cases, Verify Access may own the CSID. In this case, the CSID must be generated and made available to the Pinpoint Detect snippets.

The CSID will usually be the **tagvalue_session_index** attribute maintained by Verify Access.

Enable unauthenticated sessions

To enable the Reverse Proxy to maintain a session index for all sessions, which is established on first connection and is consistent until the session expires or the user logs out, make the following change in the Reverse Proxy configuration file:

```
create-unauth-sessions = yes
```

Set the CSID attribute in Trusteer PIP configuration

In the Trusteer PIP configuration properties, set the **customer_session_id_attribute** property to **tagvalue_session_index**.

Option: Pass session index to snippet server in HTTP Header

If proxying all snippet traffic via Verify Access, the Pinpoint snippet server can consume a CSID which is received in the **x-tr-pp-csid** HTTP Header. This removes the need to use callbacks for this purpose. You will need to configure the Reverse Proxy to pass this header.

Complete the following steps:

1. Navigate to **Secure Web Settings→Manage: Policy Administration** in the LMI.
2. Login as a management user (e.g. *sec_master*).
3. Expand **Object Space** and select **Browse Object Space** in the Task List.

4. Expand **WebSEAL** and your Reverse Proxy server in the object space browser.
5. Click the link for your snippet server junction (e.g. *ametrics*).
6. Select the **Extended Attributes** tab.
7. Click **Create...**
    a. Enter **HTTP-Tag-Value** as the Attribute Name
    b. Enter **tagvalue_session_index=x-tr-pp-csid** as the Attribute Value.
    c. Click **Apply**.

Option: Pass CSID to Pinpoint Detect using a callback function

A call-back on the login page can be used to pass the CSID to Pinpoint Detect snippets running in the browser. The Trusteer JavaScript will call an agreed JavaScript function which will be implemented on the Verify Access login page. This function will return the CSID. Usually, Verify Access will set a cookie containing the CSID and the call-back must read and return the value of this cookie.

Use HTTP Transformation to set cookie

Refer to the included sample *trusteer_ppcd_response* HTTP Transformation Rule which sets a cookie named *user_csid* with the value of *tagvalue_session_index* credential attribute.

Modify the Verify Access Reverse Proxy configuration file to invoke the HTTP Transformation Rule on GET requests which will return pages where the login snippets will execute.

```
[http-transformations]
trusteer_ppcd_response = trusteer_ppcd_response

[http-transformations:trusteer_ppcd_response]
cred-attr-name = tagvalue_session_index
request-match = response:GET /myapp*
```

Add call-back script to login page

This script could be manually added to the login page or injected into the login page using a snippet filter. The *getCookie* function is a generic function to read the value of a cookie. The *mycallback* function is the one that will be called by the Pinpoint Detect snippets. The name of this function must be agreed with Trusteer.

```
<script type = "text/javascript">
  function getCookie(cname) {
    var name = cname + "=";
    var ca = document.cookie.split(';');
    for (var i = 0; i < ca.length; i++) {
      var c = ca[i];
      while (c.charAt(0) == ' ') c = c.substring(1);
      if (c.indexOf(name) == 0) return c.substring(name.length, c.length);
    }
    return "";
  }

function mycallback() {
```

```
  var result = {
    "c": getCookie("user_csid")
  }
  //alert('mycallback called - user_csid: ' + result);
  return result;
}
</script>
```

Application owns the CSID

In some cases, the protected application may want to use its session identifier as the CSID. In this case, the application must provide the session identifier to Pinpoint Detect and make it available to the Security Verify Trusteer PIP so that it can send it in requests to the Pinpoint API.

Usually the easiest way to achieve this is to have the application set the session identifier in a cookie. This cookie can then be read via a call-back (as described above) and can also be read by the Reverse Proxy and provided to the Trusteer PIP via the decision context.

Extracting and passing the value of a cookie to the Trusteer PIP is achieved in the same way as described for passing request data for application-specific recommendations. A default attribute, *urn:ibm:security:trusteer:pinpoint:csid* is already defined for this purpose.

In the Reverse Proxy configuration file, add an entry for the cookie you need to extract.

```
[azn-decision-info]
urn:ibm:security:trusteer:pinpoint:csid = cookie:mysessioncookie
```

In the Trusteer PIP configuration properties, set the **customer_session_id_attribute** property to **urn:ibm:security:trusteer:pinpoint:csid**.

Alternative PUID handling

Pinpoint Detect indexes all sessions and actions for an individual user using a Permanent User Identifier (PUID). Usually the PUID becomes associated with a session as a result of calls made to the Pinpoint API. When a call is made to the Pinpoint API which includes both a CSID and PUID, the PUID is associated with the session.

In the recommended integration pattern, this identifier is managed by Verify Access and is populated in the user's AZN_CRED_PRINICIAL_UUID credential attribute. In some cases, this attribute may not be available for all users. It may also be desirable for the application to own this identifier.

Verify Access is using Basic Users

The recommended integration pattern assumes that all users are standard Verify Access Users. This means that they have a Unique User Identifier (UUID) which is managed by Verify Access. If you are using Basic Users, this UUID will not exist and an alternative is required.

Many LDAP vendors provide a built-in unique entry identifier which can be used as the Pinpoint Detect PUID. For example, OpenLDAP allocates an *entryUUID* attribute for every object in the directory.

Make the following change to the Reverse Proxy configuration file:

```
[TAM_CRED_ATTRS_SVC:eperson]
emailAddress = mail
mobileNumber = mobile
puid = entryUUID
```

Then, set the **permanent_user_id_attribute** configuration property of the Trusteer PIP to **puid**.

Verify Access is using External Users

The recommended integration pattern assumes that all users are standard Verify Access Users. This means that they have a Unique User Identifier (UUID) which is managed by Verify Access. If you are using External Users, this UUID will not exist and an alternative is required.

External Users are used when an identity is asserted to Verify Access by a federation partner or a custom authentication mechanism. You will need to identify an asserted attribute which can be used as a PUID.

Once you have identified the attribute, identify the attribute name that it is mapped to in the local Verify Access credential and set the **permanent_user_id_attribute** configuration property of the Trusteer PIP to this name.

If no suitable attribute is available, you may need the application to own the PUID (see below).

Application owns PUID

If the application wants to own the PUID, it must make this available to Security Verify so that it can be sent to Pinpoint Detect in Pinpoint API requests. Usually the easiest way to achieve this is to have the application set the PUID in a cookie. This cookie can then be read by the Reverse Proxy and provided to the Trusteer PIP via the decision context.

Extracting and passing the value of a cookie to the Trusteer PIP is achieved in the same way as described for passing request data for application-specific recommendations. A default attribute, *urn:ibm:security:trusteer:pinpoint:puid* is already defined for this purpose.

In the Reverse Proxy configuration file, add an entry for the cookie you need to extract.

```
[azn-decision-info]
urn:ibm:security:trusteer:pinpoint:puid = cookie:myuuidcookie
```

In the Trusteer PIP configuration properties, set the **permanent_user_id_attribute** property to **urn:ibm:security:trusteer:pinpoint:puid**.

Providing a User ID

In normal operation, all Pinpoint Detect operations use only the PUID. This prevents leaking of PII from the Pinpoint system. If a real user identity is required, this could be looked up, out of band, in the PUID source (likely the application or Verify Access user registry).

In the case where analysts or helpdesk staff need to be able to get the real user identity directly from the Trusteer Management Application (TMA) it is possible to add a real user identity to session information.

Usually the real User ID is populated using a call-back which executes, along with Pinpoint Detect snippets, on a page accessed after authentication. This call-back could read a cookie containing the

user identity (set up in a similar way to the CSID or PUID call-back) or it could make a callout to an API endpoint which will return the user identity.

Verify Access HTTP Transformation rules can be used to create a "Mock API" endpoint which returns the authenticated user in response to a request to a particular URL.

A Sample HTTP Transformation, *UserIDResponse*, is provided which returns the username (from *AZN_CRED_PRINCIPAL_NAME* attribute) as a fixed response to any matching request.

Load this transformation rule and then use the following configuration to trigger it:

```
[http-transformations]
UserIDResponse = UserIDResponse

[http-transformations:UserIDResponse]
request-match = response:GET /api/some/unique/url*
cred-attr-name = AZN_CRED_PRINCIPAL_NAME
```

The following call-back shows how to call this endpoint from client-side JavaScript to retrieve the username and return it to the calling function:

```
function mycallback(callback) {
   var xhr = new XMLHttpRequest();
   xhr.open('GET', "/api/some/unique/url", true);
   xhr.send();
   xhr.onreadystatechange = function () {
        if (xhr.readyState == 4 && xhr.status == 200) {
            returned_data = xhr.responseText;
            callback.apply(this,[returned_data]);
   };
}
```

Dynamic Snippet ID

Pinpoint Detect identifies applications by their snippet ID. This snippet ID is used when communicating with the Pinpoint snippet server (directly or via the innocent path junction) and is also used when making calls to the Pinpoint API.

When a Security Verify system is protecting a single application (or a set of applications that share an authenticated session), a static snippet ID is configured in the PIP configuration (using the *snippet_id* parameter).

If multiple applications (with independent sessions) are being protected by a single Verify Access system, each should be protected by a different Reverse Proxy instance so that authentication is independent. Different snippets (with a different snippet ID) would be used with each application.

In this case, the backend AAC and Trusteer PIP service is shared. In order to make sure that requests to the Trusteer PIP use the appropriate snippet ID for the application being accessed, the snippet ID must set dynamically at runtime.

To set the Snippet ID dynamically, create a custom JavaScipt PIP for the **urn:ibm:security:trusteer:pinpoint:snippet:id** attribute. You can base this PIP on the provided sample which maps from the Host of the request (available in the decision context) to a snippet ID.

Remove the **snippet_id** parameter from the Trusteer PIP configuration. This will cause the Trusteer PIP to request the **urn:ibm:security:trusteer:pinpoint:snippet:id** to obtain the snippet ID.

Override default dynamic data attribute names

Usually, to keep things simple, you should use the default attribute names when working with the Trusteer PIP. However, if you need to override these attribute names, you can do so by adding the following attributes to the Trusteer PIP configuration:

Property	Default
snippet_id_attribute	urn:ibm:security:trusteer:pinpoint:snippet:id
add_payee_data_attribute	urn:ibm:security:trusteer:pinpoint:cd:payee:data
transaction_data_attribute	urn:ibm:security:trusteer:pinpoint:cd:transaction:data
new_account_data_attribute	urn:ibm:security:trusteer:pinpoint:cd:newaccount:data

Table 7 Trusteer JavaScript PIP dynamic data attribute name override properties

Note: To use the **snippet_id_attribute** you must also delete the **snippet_id** configuration property.

Troubleshooting

Advanced Access Control Logging

Use the audit options provided by the Trusteer PIP

IBM recommends setting the PIP options “backend_api_request_audit” and “log_trusteer_response” to true, starting in your lower environments, and always keeping these options turned on in production.

Trace is a more resource intensive method as compared to auditing. Trace can only be activated in most customer production environments with oversight and planning, and only for limited periods of time. If you are preparing to activate a trace, it is probably to troubleshoot a problem. If you always have the PIP write the audit messages containing the full request body for each Trusteer API call, and the full response body for the response from each Trusteer API call, this may lessen the need to activate a trace, or provide more information so that the trace can be better targeted at the issue. The audit messages written by the PIP include the Trusteer PUID, CSID, the Trusteer snippet ID and the URL to which the Verify Access AAC policy was attached. (These last two items help identify the application, if using more than one Trusteer-protected application). These audit records can help identify or narrow down the cause of many issues, including issues that might not be an error in either Trusteer or in Verify Access.

These audit messages can be forwarded to an SIEM tool such as IBM QRadar to enable analysis and reporting.

Enabling Trace

1. Click the Monitor Analysis and Diagnostics menu icon
2. Click the **Runtime Tracing** from the Logs menu list
3. Select or manually enter com.tivoli.am.rba.extensions.PluginUtils=ALL into the **Tracing Specification** field for the PIP trace
4. When debugging the connection errors with the Pinpoint API callout, append com.ibm.security.access.httpclient.*=ALL

   For example:
   com.tivoli.am.rba.extensions.PluginUtils=ALL:com.ibm.security.access.httpclient.*=ALL

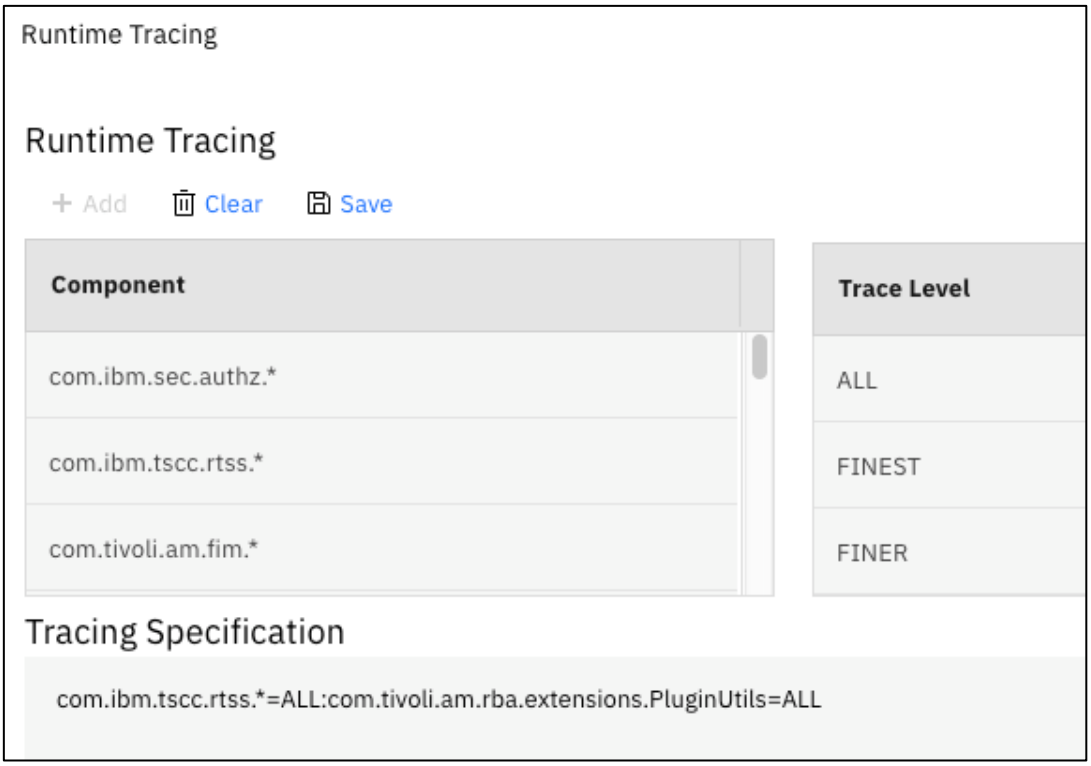

Figure 8 JavaScript PIP Tracing

5. Deploy the pending changes

Viewing Trace

1. Click the Monitor Analysis and Diagnostics menu icon
2. Click the **Application Log File** from the Logs menu list
3. Expand the **access_control** folder
4. Expand the **runtime** subfolder
5. Select the **trace.log** file
6. Click the **View** button

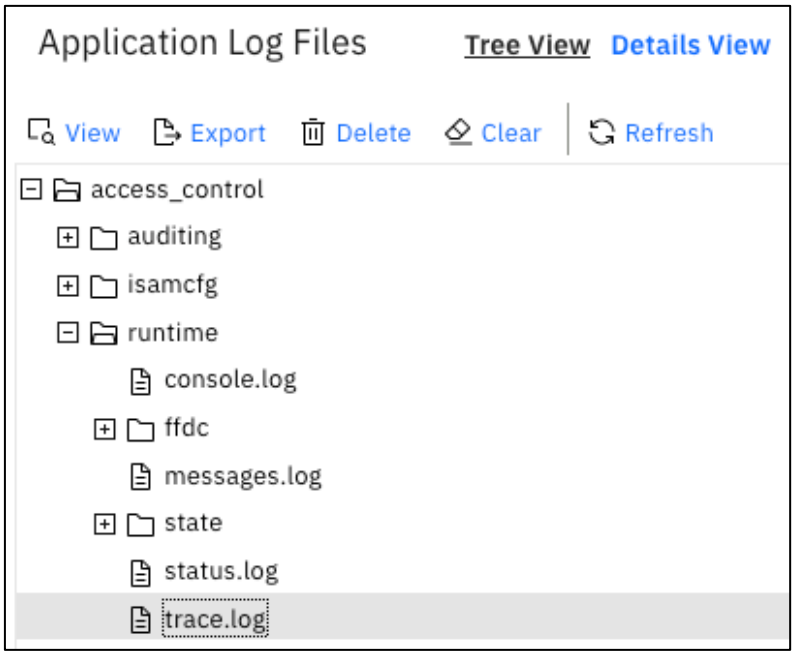

Figure 9 AAC Trace file location

7. Review any errors in the log.
8. Clear the log (do not Delete) each time you re-execute a test.
9. Disable the trace once you have resolved the issue to reduce performance impact of verbose logging in the trace.log.

Common Issues and Solutions

Certificate Signer Errors

If the HTTP client trace returns errors from an unknown or untrusted signer, despite having loaded these in to the keystore, it is likely the Certificate Revocation List (CRL) check is failing.

To manually disable:

1. Click the AAC menu icon
2. Click the **Advanced Configuration** from the Global Settings menu list
3. Manually locate or filter for **kess.crlEnabled**
4. Set the Value to **false**
5. **Deploy** the pending changes

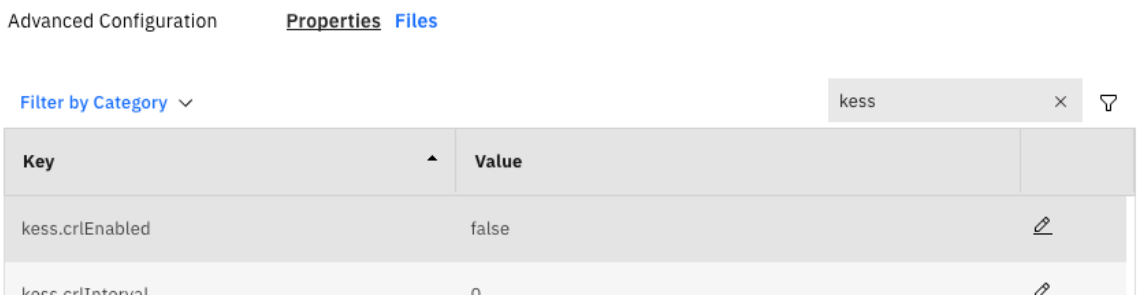

Figure 10 Disabling CRL checking

Verify Access Reverse Proxy Errors

The difference in behaviour between large and small POSTs is that WebSEAL is able to cache the smaller POST in order to send it to a second server. This caching limit is set by the following parameters in the WebSEAL configuration file:

- max-client-read
- request-body-max-read
- request-max-cache

When snippet traffic is proxied, these configuration items may need increased maximum limits to accommodate the transactions between the client and the Pinpoint Detect server.

The size of these settings will depend on the payloads generated by your particular Pinpoint Detect deployment.

To determine appropriate values, inspecting the content-length when accessing the protected web app behind WebSEAL with the snippet enabled.

DPWWA1100W POST request larger than request-body-max-read

When using the Verify Access Reverse Proxy to proxy the snippet traffic you may see –

```
0x38cf044c Error Text: DPWWA1100W POST request larger than request-body-
max-read, cannot apply dynurl matching.
```

If you see this error, update the Verify Access Reverse Proxy configuration file -

```
dynurl-allow-large-posts = yes
```

Configuration Sample 1 DynURL Large Post support

TLS Version Issues

In older versions of Verify Access, the TLS version used by its HTTP Client code (which is used by the Trusteer PIP) defaults to TLS v1.0. This setting may remain unchanged if you upgrade from an older version of Verify Access. Depending on the Pinpoint API server configuration, this may not be acceptable.

Complete the following steps to change the default TLS version:

1. Navigate to **AAC→Advanced Configuration** in the LMI

2. Locate the *util.httpClient.defaultSSLProtocol* key and click the edit icon.
    a. Select TLSv1.2 from the drop-down menu
    b. Click **Save**.

Appendix

In this appendix are configuration steps which are performed by the installation of the IBM Security Verify Access Extension for IBM Security Trusteer Pinpoint Detect. They are included here in case you need to perform them manually (having acquired a copy of the PIP JavaScript from an installed system).

AAC: Create Trusteer Attributes

The recommendation attributes used by the updated Trusteer PIP are not pre-defined in Advanced Access Control. They must be manually defined so that they are available for use in policies.

Create Attribute for Login Recommendation

Complete the following steps:

1. In the Verify Access LMI, select **AAC→Policy: Attributes** from the mega-menu.
2. Click the **New Attribute** icon.
    a. Enter **Pinpoint Detect Login Recommendation** as the Name
    b. Enter **urn:ibm:security:trusteer:pinpoint:cd:login:recommendation:recommendation** as the Identifier.
    c. Enter **urn:ibm:security:trusteer:pinpoint:cd** as the Issuer
    d. Change the Category to **Environment**
    e. Select **Policy** checkbox under Type.
    f. Click **Save**.

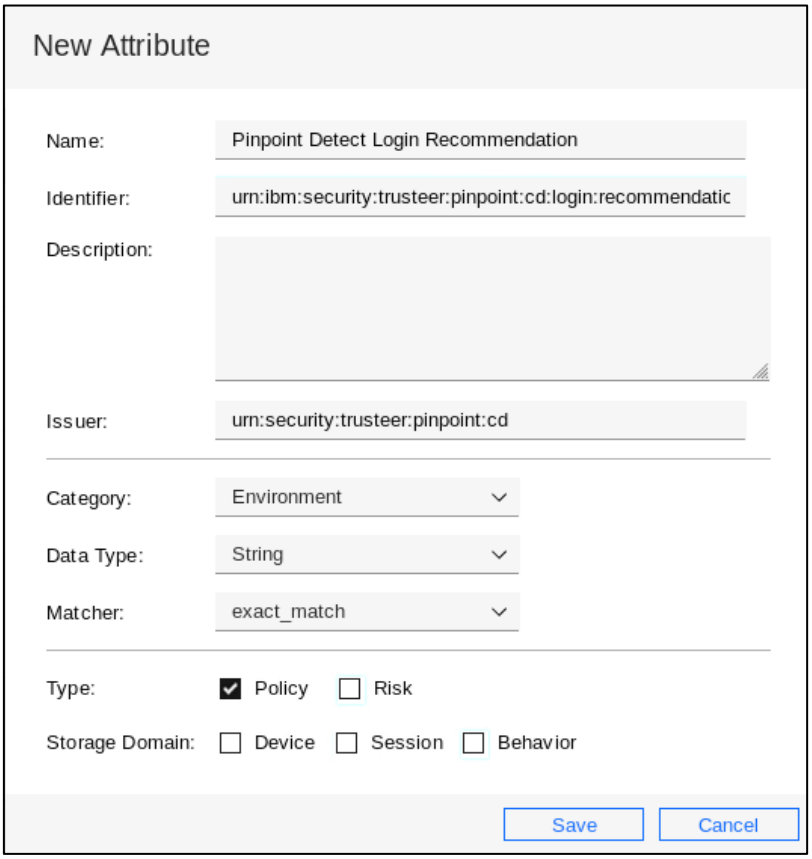

Figure 11 Trusteer JavaScript PIP Login Recommendation Attribute

Create Attribute for In-Session Recommendation

Complete the following steps:

1. In the Verify Access LMI, select **AAC→Policy: Attributes** from the mega-menu.
2. Click the **New Attribute** icon.
    a. Enter **Pinpoint Detect In-Session Recommendation** as the Name
    b. Enter **urn:ibm:security:trusteer:pinpoint:cd:session:recommendation:recommendation** as the Identifier.
    c. Enter **urn:ibm:security:trusteer:pinpoint:cd** as the Issuer
    d. Change the Category to **Environment**
    e. Select **Policy** checkbox under Type.
    f. Click **Save**.

Create Action-Specific Recommendation Attributes

For each action-specific recommendation you will use, you will need to create the associated recommendation attribute. These attributes have the same properties as the login and in-session recommendation attributes. Only the name and identifier will be different.

Name	Identifier
Pinpoint Detect New Account Recommendation	urn:ibm:security:trusteer:pinpoint:cd:newaccount:recommendation:recommendation
Pinpoint Detect Add Payee Recommendation	urn:ibm:security:trusteer:pinpoint:cd:payee:recommendation:recommendation
Pinpoint Detect Transaction Recommendation	urn:ibm:security:trusteer:pinpoint:cd:transaction:recommendation:recommendation

Table 8 Recommendation attribute identifiers

Complete the following steps for each action-specific recommendation you will use:

1. In the Verify Access LMI, select **AAC→Policy: Attributes** from the mega-menu.
2. Click the **New Attribute** icon.
    a. Enter the Name as per the table above.
    b. Enter the Identifier as per the table above.
    c. Enter **urn:ibm:security:trusteer:pinpoint:cd** as the Issuer
    d. Change the Category to **Environment**
    e. Select **Policy** checkbox under Type.
    f. Click **Save**.

Install Trusteer PIP

In this section you will install the Trusteer PIP. There are different steps for Verify Access v10.0.0.0 and Access Manager v9.0.7.x.

To complete this section, you will need the following:

- The latest Trusteer PIP JavaScript

Option 1: Verify Access v10.0.0.0

When using a new Verify Access v10.0.0.0 system, you will need to create a new JavaScript PIP using the JavaScript provided in this integration package.

Complete the following steps:

1. In the Verify Access LMI, select **AAC→Policy: Information Points** from the mega-menu.
2. Click the **New** icon and select **JavaScript** from the drop-down menu.
    a. Enter **Trusteer PIP** as the Name
    b. Enter a Description (give a name that indicates the integration package version)
    c. Browse and select the updated Trusteer PIP JavaScript file.

d. Click **Save**

Option 2: Access Manager v9.0.7.x (or upgraded system)

When using Access Manager v9.0.7.x, or a system upgraded from v9.0.7.x, you will replace the JavaScript of the built-in Trusteer PIP with the updated JavaScript provided in this integration package. Re-using the built-in PIP definition is necessary because it is associated with a set of built-in policy attributes which would otherwise have to be re-created.

Complete the following steps:

1. In the Verify Access LMI, select **AAC→Policy: Information Points** from the mega-menu.
2. Select the Trusteer PIP
3. Click the **JavaScipt** icon and select **Import** from the drop-down menu.
    a. Browse and select the updated Trusteer PIP JavaScript file.
    b. Click **OK.**
4. Click the **Modify** icon.
    a. Update the Description of the PIP to indicate it has been changed.

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing

IBM Corporation

North Castle Drive

Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing

Legal and Intellectual Property Law

IBM Japan, Ltd.

19-21, Nihonbashi-Hakozakicho, Chuo-ku

Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement might not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
2Z4A/101
11400 Burnet Road
Austin, TX 78758 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© IBM 2015, 2019. Portions of this code are derived from IBM Corp. Sample Programs. ©Copyright IBM Corp 2010, 2015, 2019. All rights reserved.

If you are viewing this information in softcopy form, the photographs and color illustrations might not be displayed.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and trademark information; at www.ibm.com/legal/copytrade.shtml.

Adobe, Acrobat, PostScript and all Adobe-based trademarks are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Other company, product, and service names may be trademarks or service marks of others.