IBM Security Verify Access
Version 10.0.7
December 2023

Advanced Access Control Administration

Contents

Figures

Tables

Chapter 1. Overview of context-based access

Context-based access provides access decision and enforcement that is based on a dynamic risk assessment or confidence level of a transaction. Context-based access uses behavioral and contextual data analytics to calculate risk.

Context-based access:

- Improves security during authentication and authorization of business transactions.
- Assesses risk based on static, contextual, and analytically calculated attributes.
- Calculates a risk score based on multiple weighted attributes.
- Provides policy rules that determine whether an access request must be permitted, denied, or challenged.

You can configure context-based access to:

- Silently register or require users to register devices that they commonly use.
- Associate the registered devices with user credentials.
- Present a challenge or request additional authentication, if the user attempts to authenticate with the same credentials from another unregistered device.
- Enforce specific authentication mechanisms to access a particular protected resource.
- Use the behavioral patterns of the user as a factor in risk score calculation. For example, a user might attempt to access a protected resource at a time outside of normal business hours. You can configure the context-based access policy to deny access or force the user access to authenticate with a secondary challenge.

Business scenarios

Business transactions that have an increased security risk factor can benefit by implementing context-based access.

The following examples are some scenarios where you can use context-based access to provide a higher level of confidence for the transaction:

- A user tries to access sensitive information where a simple user ID and password authentication is not sufficient. However, the data is not sensitive enough to use a more complex authentication mechanism, such as token IDs.
- Users require access from remote locations that are not trusted and they use devices such as mobile devices and notebooks. To ensure that mobile users are authenticated sufficiently, the business requires a second factor authentication.
- Users need to access an application that provides sensitive business information. They might access the information outside of their regular work patterns.
- A user accesses a resource from a device that the user previously used and maintains typical usage patterns. Context-based access improves the user experience by limiting secondary authentication mechanisms.

Features

Context-based access provides several capabilities to identify potential risk and limit the ability for an attacker to use stolen credentials.

- Silent device registration where the system does not require any user interaction.
- Ready-to-use, predefined policy attributes that are specific to context-based access.
- Scenario-based, predefined risk profiles.

- A risk-scoring engine that calculates a risk score for the current transaction based on the active risk profile. The risk score is based on configurable weights that are assigned to context attributes and behavior attributes. If the risk score is high, further challenges are presented to the user or access is denied. If the risk score is low, the user is permitted access.

Functional overview

Context-based access includes an external authorization service (EAS), runtime authorization service, and attribute collection service.

The following diagram illustrates the architecture of context-based access. The diagram also shows how the various components plug into WebSEAL. WebSEAL is a component available in the following IBM products:

- IBM Web Gateway Appliance
- IBM Security Verify Access
- Tivoli Verify Access for e-business

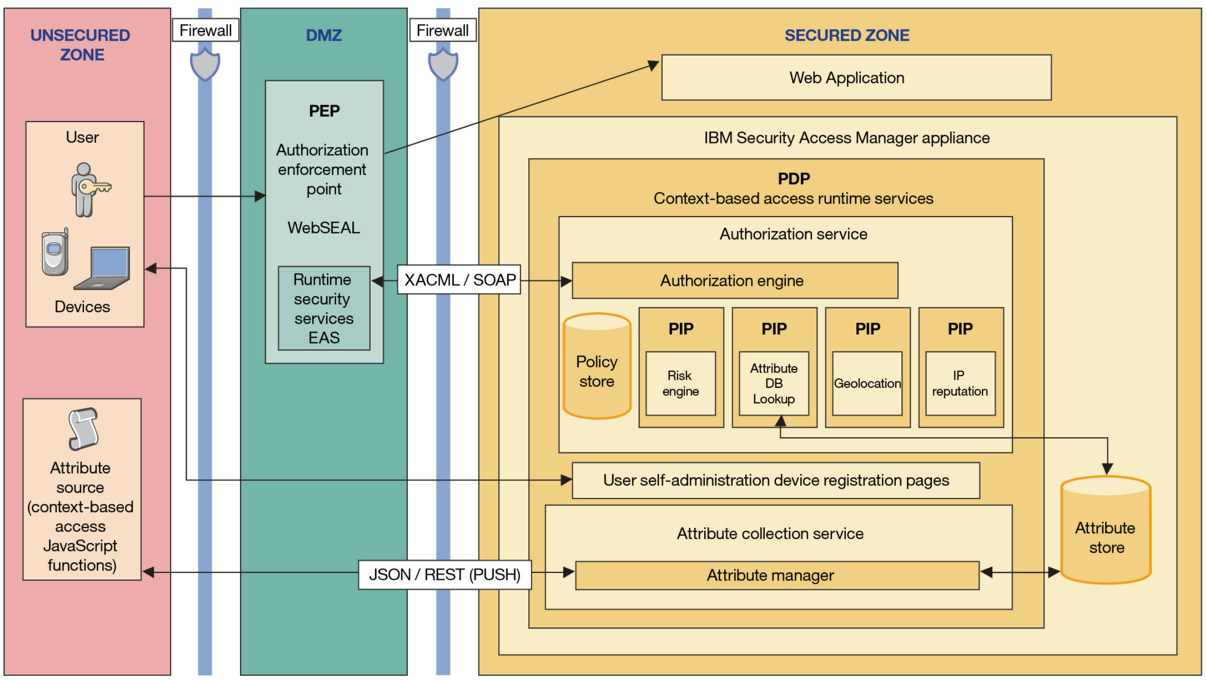

Figure 1. Context-based access architecture

Context-based access runtime services

Context-based access provides the following runtime services:

Authorization service

The context-based access authorization service is a component of the runtime environment. The runtime authorization service stores the policy, calculates the risk score, and makes the access decision. The authorization service exposes an XACML over SOAP web service that third-party enforcement points can call to get authorization decisions.

Attribute collection service

The attribute collection service is a Representational State Transfer (REST) service that collects web browser and location attributes from the user. The attribute collection service is a push service. You can configure the context-based access runtime service to use the collected attributes as the policy attributes for calculating risk. You can also use the Java™ ADK to plug in your custom implementation for a pull service that retrieves attributes from the user.

Risk-scoring engine

The risk-scoring engine calculates the risk or confidence level. It provides a single integer that represents the risk score for the current transaction in the form of a percentage. The risk score is calculated based on the weights that are assigned to one or more of the following policy attributes that are part of the active risk profile:

- Device identification or fingerprint, such as details of hardware, IP address, location information, IP address reputation, operating system, web browser type, web browser version, web browser plug-ins, and screen resolution.
- Behavioral patterns, such as frequency of login, time of access, frequency of access, and type of transactions.
- Custom attributes that you can configure and manage through a pluggable interface. The context-based access authorization service is extensible and can also include external sources for attributes.

The risk engine returns the final risk score as a policy attribute, which is the basis of the final authorization decision.

Policy enforcement point (PEP)
WebSEAL is the policy enforcement point for context-based access. Context-based access integrates with the existing WebSEAL authentication mechanisms, such as cross domain authentication service (CDAS) and external authentication interface (EAI).

External authorization service
The runtime security services EAS plug-in for WebSEAL enforces the policy decision. The EAS takes the request data and sends an authorization decision request to the context-based access authorization service. The authorization service maps the authorization decision response to the appropriate WebSEAL action, such as permit, deny, or step-up authentication. You can manage the EAS with entries in the `webseald.conf` file with the WebSEAL stanza syntax. The **`isamcfg`** tool automates the configuration of the EAS for the predefined scenarios provided with the product.

Policy information points (PIPs)
Policy information points are components of the context-based access authorization service. They provide all the policy attributes that are not provided in the initial access request. The risk score and attributes that are pushed to the attribute collection service are provided to the authorization service through PIPs.

Context-based access includes ready-to-use PIP implementations that provide the policy attributes that are required. You can also provide custom policy attributes to the authorization service through a custom PIP.

Policy decision point (PDP)
The runtime authorization service is the policy decision point for context-based access. This service is configured to use the PEP context-based access plug-in. The authorization decision is based on an authorization policy that uses policy attributes and PIPs that are specific to context-based access. The PIPs provide information, such as risk score, user location, and device type.

The policy administration point (PAP)
The IBM Security Verify Access appliance is the policy administration point for context-based access. Context-based access provides an administrative console for configuring and managing the policies, risk profiles, attributes, and weights, which are required for calculating risk.

Transaction flow

In a typical context-based transaction, the user requests access to a protected resource. Context-based access calculates the risk score and determines whether access is permitted, denied, or permitted with an obligation.

In the following example of a context-based access transaction, the risk score indicates that the user must be presented with a challenge. The user successfully completes the challenge and receives permission to access the protected resource.

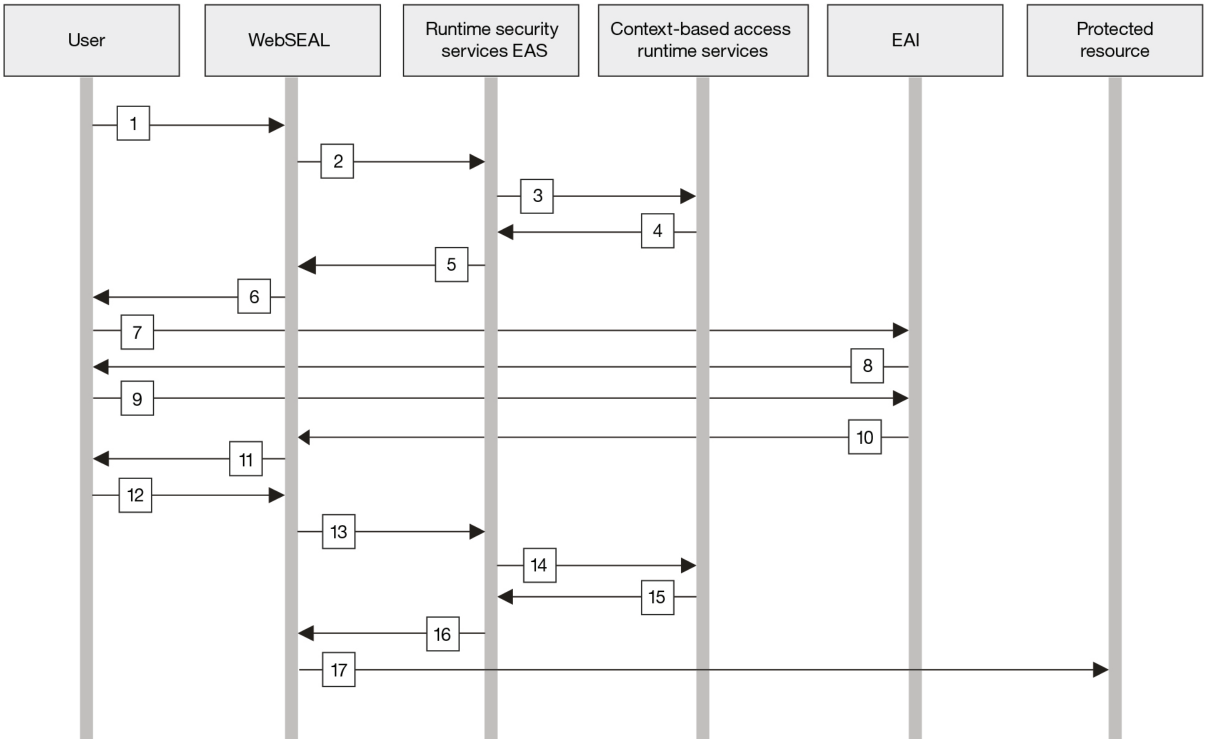

Figure 2. Context-based access transaction flow example

The following process explains the flow of the transaction in the example scenario:

1. The user interacts through a web browser to submit authentication information and requests access to a protected resource. The protected resource is a junctioned web application that WebSEAL protects.
2. WebSEAL inspects the request.

   WebSEAL is the reverse proxy server that interacts with all transactions. For the protected resource that the user requests, the WebSEAL policy (POP) is configured to call the runtime security services EAS plug-in to authorize the request. The EAS is a shared library plug-in, which is internal to the WebSEAL process, so there is no on-the-wire callout between WebSEAL and the EAS.
3. The EAS first checks the local Verify Access policy.
   a. If the Verify Access denies access, then the EAS returns the response and does not continue with a forbidden response.
   b. If the Verify Access permits access, the EAS collects the context information about the user and the request. The WebSEAL **azn-decision-info** stanza has the specifications to create an XACML over SOAP authorization decision request.
   c. The EAS sends the request to the context-based access runtime authorization service for the authorization decision.
4. The runtime authorization service (PDP) runs the configured policy and calls the appropriate PIPs based on the current policy.
   a. If a risk score policy attribute is requested, then the risk engine is called.
   b. The risk engine takes the context details, requests additional policy attributes, if required, and then calculates a risk score.
   c. The authorization service applies the risk score policy attribute against the authorization policy and returns an appropriate decision response to the runtime security services EAS.
   d. The EAS supports three decision types: permit, deny, and permit with obligation. In this example, a decision to permit with obligation is returned.

i) If the policy decision is a simple permit or deny, the decision is mapped to the WebSEAL `permit` and `not_permit` decisions. The WebSEAL decisions are returned from the **`azn_svc_decision_access_allowed_ext`** call.

ii) If the response is to permit access, WebSEAL permits the request to continue to the requested resource.

iii) If the response is to deny access, WebSEAL displays an error page, which indicates that access is forbidden.

5. The runtime security services EAS parses the response and returns one of the following EAS responses to WebSEAL:

   - Permit
   - Deny
   - Step-up authentication levels
   - Browser redirect to a location configured on the WebSEAL configuration file

   In this example, the response is a redirect to a location on the appliance runtime, so that WebSEAL can enforce the appropriate authentication challenge.

   The context-based access EAS also provides a configuration to map the returned obligation to a specific WebSEAL external authentication interface (EAI) mechanism. The EAI can be either a WebSEAL EAI or CDAS external authentication mechanism, which the customer or IBM® Business Partner can implement.
6. WebSEAL specifies the appropriate authentication mechanism to the user.
7. The user is redirected to the authentication mechanism, which builds a challenge response.
8. The challenge response is presented to the user.
9. The user responds to the challenge.
10. If the response of the user to the challenge request is successful, the challenge is processed. The EAI application returns the credentials of the user and a successful step-up level in the HTTP response to WebSEAL.
11. WebSEAL updates the session of the user with the new credential details that the EAI application provides and redirects the user to the protected resource.
12. The request from the user to access the protected resource is sent via 302 redirect. The request does not require any user interaction.
13. WebSEAL inspects the request and directs the request to the runtime security services EAS for authorization.
14. The EAS collects all context information about the user and the request and creates an XACML over SOAP decision request. The EAS sends the request to the runtime authorization service.
15. The context-based access runtime service takes the context and other policy attributes and calculates a risk score. The runtime service applies the risk score against the configured authorization policy and returns a policy decision response to permit access.
16. The runtime security services EAS interprets the response and returns the permit decision to WebSEAL.
17. WebSEAL permits the original request to continue without going back to the web browser of the user. The user is not aware of the transaction process.

Chapter 2. Risk management overview

Context-based access policy decisions can be based on the risk score. The risk score is calculated based on the active risk profile attributes that are retrieved from the user.

The system allows for multiple risk profiles to be defined, but only one is active at run time.

Each attribute included on a risk profile has an assigned weight to be used while calculating the risk score of a given request. The active risk profile attributes are evaluated to determine whether a user should be granted access to a protected resource. A policy author can rely on the risk score to enforce stronger authentication mechanisms or to perform device registration.

To get started setting up context-based access control for your installation, work with:

1. **Attributes**: The product provides a predefined set of attributes that are ready to use without any customization. Optionally, you can add attributes that can come from:
   - Standard HTTP headers
   - HTTP FORM parameters
   - Client-side JavaScript files that are collected into the attribute collection service
   - Custom attributes you define by writing custom JavaScript files
2. **Obligations**: The product provides a predefined set of obligations that are ready to use without any customization. Optionally, you can update or define your own obligations.
3. **Risk profiles**: The product provides a predefined set of risk profiles that are ready to use without any customization. Optionally, you can update or define your own risk profiles to calculate the risk score.

   Note: A default risk profile is set as active when you configure the appliance. This risk profile is not intended to be used in a production environment. Set a different risk profile before using risk profiles in your production environment.
4. **Policies**: Create policies that evaluate requests that are based on attributes and obligations that you defined and the risk decisions that you want to make.

Risk score calculation

Risk score calculation is the process by which the risk engine determines a risk score. The *risk score* demonstrates the level of risk that is associated with permitting a request to access the resource. This risk score is compared to a *threshold score* that is set in a policy. A decision is made based on the result of this comparison.

Overview

The risk engine determines a risk score by comparing sets of attributes that identify devices. These sets of attributes are called *device fingerprints*. Device fingerprint attributes include items such as IP address, location, and screen size. Each registered device has one device fingerprint. Because the user accesses the resource in different locations and on different devices, the user can have many registered devices.

The following process describes how risk assessment works:

1. The incoming device requests access to the resource.
2. The risk engine collects as many device fingerprint attributes as it can from the request device.
3. After the attributes are collected, the risk engine:
   - Determines the device fingerprint.
   - Calculates the risk score. The risk score
     - Is a number.
     - Represents the amount of risk that is associated with the incoming request.

- Indicates the likelihood that the incoming request represents the user.

4. The risk engine:
   - Compares the incoming fingerprint with each registered device fingerprint.
   - Uses the attributes that are contained in the larger fingerprint for each comparison.
   - Calculates a risk score for each comparison.
5. To determine the final risk score, the risk engine:
   - Chooses the lowest risk score of the comparisons between the incoming fingerprint and the registered fingerprint.
   - Measures the final risk score against a threshold score or range that the administrator sets in a policy.
6. Depending on the way the administrator writes the policy, one of the following outcomes occurs:

   Permit
   The risk score for the incoming request is well below the threshold score. The user is granted access to the resource. For example, the risk score is `30`, and the threshold score that is set by the administrator is `40`.

   Permit with obligation or authentication
   The user is asked to complete an extra security measure, such as step up authentication. For example, the risk score is `40`, and the policy that the administrator wrote requires users that operate devices with scores `30 - 90` to step up.

   Deny
   The risk score for the incoming request is above the threshold score or range. The user is denied access to the resource. For example, the risk score is `50`, and the threshold score that is set by the administrator is `40`.

The risk score is calculated through the following formula:

```
Risk Score = (total weight of mismatched attributes /
total weight of all attributes) × 100
```

When the values that belong to the incoming device fingerprint and the registered device fingerprint are the same, the values are matched. When the values that belong to the incoming device fingerprint and the registered device fingerprint are not the same, the values are mismatched.

Sometimes, the fingerprints contain attributes that are not matched or mismatched. These attributes are called indeterminate attributes. When there are indeterminate attributes present, the following formula is used to calculate the risk score:

```
Risk Score = (total weight of mismatched attributes /
(total weight of all attributes-total weight of indeterminate attributes)) × 100
```

Scenarios

The following example scenarios demonstrate risk score calculation.

All three of the scenarios assume that the administrator

- Wrote a policy that specifies that any risk score at or below `40` is permitted, and any risk score above `40` is denied.
- Gave equal weight values to all of the attributes in the tables.
  - The attributes in the tables have the same weight value of `10`.

Scenario 1: Authentication permitted

The total weight of the unequal device fingerprint values that belongs to one attribute is not significant enough to prohibit authentication.

The example information in the table is used to calculate the risk score.			
Attribute names	**Weight values**	**Incoming device fingerprint values**	**Registered device fingerprint values**
`colorDepth`	10	32	32
`deviceLanguage`	10	en-US	en-US
`devicePlatform`	10	Win32	Win32
`http:userAgent`	10	Mozilla/5.0 (Windows NT 6.1; WOW64; rv:15.0) Gecko/ 20120427 Firefox/ 15.0a1	Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/28.0.1468.0 Safari/537.36
`ipAddress`	10	42.29.144.5	42.29.144.5
`screenHeight`	10	1080	1080
`screenWidth`	10	1920	1920

- All of the device fingerprint values match except for the incoming device fingerprint value and existing device fingerprint value for `http:userAgent`.
- Because `http:userAgent` is the only attribute that has any mismatched values, the total weight of the mismatched attributes is `10`.
- The total weight of all of the attributes is `70` because each attribute has a weight value of `10`.
- According to the risk score calculation formula: `(10/70)×100=14`. Therefore, the risk score is `14`.
- Because the risk score is below `40`, authentication is permitted.

Scenario 2: Authentication denied with multiple significant attributes

The total weight of the unequal device fingerprint values that belongs to 6 out of 7 of the attributes is significant enough to prohibit authentication.

The example information in the table is used to calculate the risk score.			
Attribute names	**Weight values**	**Incoming device fingerprint values**	**Registered device fingerprint values**
`colorDepth`	10	24	32
`deviceLanguage`	10	en-US	en-US
`devicePlatform`	10	Linux	Win32
`http:userAgent`	10	Mozilla/5.0 (X11; Linux i686 (x86_64)) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/27.0.1453.93 Safari/537.36	Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/28.0.1468.0 Safari/537.36
`ipAddress`	10	9.53.18.164	42.29.144.5
`screenHeight`	10	1050	1080
`screenWidth`	10	1680	1920

- None of the device fingerprint values match except for the incoming device fingerprint value and existing device fingerprint value for `deviceLanguage`.
- Because all of the attributes except for `deviceLanguage` have mismatched values, the collective weight of the mismatched attributes is `60`.
- The total weight of all of the attributes is `70` because each attribute has a weight value of `10`.
- According to the risk score calculation formula: `(60/70)×100=86`. Therefore, the risk score is 86.
- Because the risk score is above `40`, authentication is denied.

Scenario 3: Authentication denied with one significant attribute

The total weight of the unequal device fingerprint values that belongs to one attribute is significant enough to prohibit authentication.

The example information in the table is used to calculate the risk score.

Attribute names	Weight values	Incoming device fingerprint values	Registered device fingerprint values
`devicePlatform`	5	`Android`	`Android`
`geoLocation`	85	`51.499444, -0.1275, 10`	`30.283611, -97.7325, 10`
`screenHeight`	5	`800`	`800`
`screenWidth`	5	`480`	`480`

- In addition to the previous assumptions, this scenario prohibits any distance greater than 40 kilometers.
- All of the device fingerprint values match except for the incoming device fingerprint value and the existing device fingerprint value for the `geoLocation` attribute. The `geoLocation` attribute contains the values that the risk engine uses to calculate the distance between the incoming device fingerprint and the registered device fingerprint. In this instance, the distance between the two device fingerprints is 7909 kilometers.
- Because the `geoLocation` attribute is the only attribute with mismatched values, the weight of the mismatched attributes is 85.
- The total weight of all of the attributes is `100` because the `geoLocation` attribute has a weight value of 85. `devicePlatform`, `screenHeight`, and `screenWidth` each have weight values of 5.
- According to the risk score calculation formula: `(85/100)×100=85`. Therefore, the risk score is 85.
- Because the risk score is above `40`, authentication is denied.

Note: Authentication can be denied if the incoming fingerprint value and registered device fingerprint value for just one attribute indicate a large enough discrepancy. In this scenario, the distance between the incoming device fingerprint value and the registered device fingerprint value is too large for authentication to be permitted.

Risk reports

Risk reports illustrate the comparison between the incoming device fingerprint and the registered device fingerprint for each risk score calculation.

Each risk report contains the incoming device fingerprint that attempts to access a protected resource as compared to the registered device fingerprint. Analysis of each risk report provides the following information about the corresponding risk score calculation:

- Outcome of the risk score calculation.
- Matcher that the risk engine used for the comparison between the device fingerprints.
- Attributes that were compared between the incoming fingerprint and the registered fingerprint.

- Other calculations that were made during the risk score determination.

The administrator can use risk reports to complete the following objectives:

- Write more effective risk profiles.
- Monitor system activity.

Related concepts

Risk score calculation

Risk score calculation is the process by which the risk engine determines a risk score. The *risk score* demonstrates the level of risk that is associated with permitting a request to access the resource. This risk score is compared to a *threshold score* that is set in a policy. A decision is made based on the result of this comparison.

Related tasks

Accessing risk reports

The administrator can access the risk reports that are generated during risk score calculation scenarios.

Configuring risk reports

The risk engine generates risk reports to analyze risk score calculation scenarios. The administrator can configure the `riskEngine.reportsEnabled` property and the `riskEngine.reportsMaxStored` property to enable the risk engine to store a specified number of these reports.

About this task

The following properties are necessary for the generation of risk reports:

`riskEngine.reportsEnabled`

Enables the risk engine to produce risk reports after the risk engine generates the risk score for each incoming request.

The `riskEngine.reportsEnabled` property defaults to disabled, so the administrator must configure `riskEngine.reportsEnabled` to enable risk reporting.

`riskEngine.reportsMaxStored`

Enables the risk engine to store a maximum number of risk reports as specified by the administrator.

The administrator can set the `riskEngine.reportsMaxStored` property. However, the property defaults to five reports. The risk engine deletes the oldest report when it produces a report that exceeds the maximum number of reports that it is specified to store.

To configure the `riskEngine.reportsEnabled` property, complete the following steps.

Procedure

1. Log in to the local management interface.
2. Click **AAC** > **Global Settings** > **Advanced Configuration**.
   - Configure the `riskEngine.reportsEnabled` property
     a. Under **Key**, find `riskEngine.reportsEnabled`.
     b. Click the edit button.
     c. Complete one of the following actions:

        Enable risk reporting

        i) Select the **Enabled** box.
        ii) Click **Save**.

        Note: Enabling risk reporting sets the `riskEngine.reportsEnabled` property to **true**.

        Disable risk reporting

        i) Clear the **Enabled** box.

ii) Click **Save**.

Note: Disabling risk reporting sets the `riskEngine.reportsEnabled` property to **false**.

- Configure the `riskEngine.reportsMaxStored` property
    a. Under **Key**, find `riskEngine.reportsMaxStored`.
    b. Click the edit button.
    c. Specify the number of reports for the risk engine to store.
    d. Click **Save**.

Related tasks

Accessing risk reports
The administrator can access the risk reports that are generated during risk score calculation scenarios.

Accessing risk reports

The administrator can access the risk reports that are generated during risk score calculation scenarios.

About this task

The administrator can access the risk reports specific to your business by completing the following steps:

Procedure

1. Log in to the local management interface.
2. Click **Monitor**.
3. Click **Application Log Files**.
4. Click the **+** next to the **access_control** folder.
5. Click the **+** next to the **risk_reports** folder. The administrator can access the risk reports after the risk engine generates the first report and the **risk_reports** folder is created.
6. Find the necessary report, and click **View**.

Related concepts

Risk reports
Risk reports illustrate the comparison between the incoming device fingerprint and the registered device fingerprint for each risk score calculation.

Related tasks

Configuring risk reports
The risk engine generates risk reports to analyze risk score calculation scenarios. The administrator can configure the `riskEngine.reportsEnabled` property and the `riskEngine.reportsMaxStored` property to enable the risk engine to store a specified number of these reports.

Chapter 3. Attributes

Attributes specify the context of a request that you want to be evaluated as part of an access decision. For example, an attribute might be information in a request, such as a user name, or information in an external source, such as a user's age in a user registry or an information type in a database.

Requests contain one or more of the following categories of attributes:

Action
Indicates the user action.

Environment
Indicates when and how the user is trying to access the resource.

Resource
Gives information about what the user is trying to access.

Subject
Indicates who is trying to access the resource.

When you author policies or risk profiles, you select attributes that you want to be evaluated. Commonly used attributes are predefined. You can also create your own to meet the needs of your environment.

Managing attributes

Attributes represent unique information about a request. You use attributes in policy rules and risk profiles to match the attributes in a request. You can view, add, modify, and delete attributes.

Procedure

1. Log in to the local management interface.
2. Click **AAC**.
3. Under **Policy**, click **Attributes**.
4. Perform one or more of the following actions:

   View and filter attributes

   Take any of the following actions to filter your view:

   - Select the **Details View** to view attribute name, category, and data type.
   - Select the **List View** to view only the name of the attribute.
   - Type a term, such as an attribute name, category, or data type in the **Filter** field to list attributes that use that term.

     Note: The filter searches all attribute properties fields, including descriptions, for the alphanumeric characters you type in the **Filter** field. For example, if you type `header` in the **Filter** field, all attributes that contain `header` in their properties are shown in the attributes list.

     Click **x** to clear the **Filter** field.
   - Sort the attribute list by column with the up or down arrow on each column. For example, you can view the list of attributes that are sorted by the **Data type** column in ascending order by clicking the up arrow.

   Add an attribute

   a. Click .
   b. Complete the properties for the attribute.

Note: The attribute name must begin with an alphabetic character. Do not use control characters, leading and trailing blanks, and the following special characters ~ ! @ # $ % ^ & * ( ) + | ` = \ ; " ' < > ? , [ ] { } / anywhere in the name.

c. Click **Save**.

Modify an attribute

Attention: Ensure that the modification does not affect a current policy or configuration. If you modify an attribute that is in-use, the policy or configuration that uses the attribute might stop working.

a. Click .

b. Complete the properties for the attribute.

Note: The attribute name must begin with an alphabetic character. Do not use control characters, leading and trailing blanks, and the following special characters ~ ! @ # $ % ^ & * ( ) + | ` = \ ; " ' < > ? , [ ] { } / anywhere in the name.

c. Click **Save**.

Delete an attribute

a. Select an attribute or press and hold the Ctrl key and select multiple attributes to remove.

b. Click . Confirm the deletion. Click **OK** to continue or click **Cancel**.

The attribute is removed.

5. When you add, modify or delete an attribute, a message indicates that there are changes to deploy. If you are finished with the changes, deploy them.

For more information, see Chapter 14, “Deploying pending changes,” on page 157.

Related reference

“Predefined attributes” on page 19

An appliance with Advanced Access Control uses attributes to provide information about users and devices that try to access a protected resource. The appliance also includes a set of commonly used attributes called *predefined attributes*.

Updating location attributes

To define policy that is based on geolocation, you must update the geolocation database with appropriate location and IP data.

About this task

When a request is received, a GeoLocator policy information point (PIP) determines the location of the device that made the request. The device IP address as determined by the point of contact server is the input to the PIP. The PIP reads the geolocation database to determine the device location.

All location attributes stored in the database are shown as environment attributes that you can use to author policies.

Location attributes include:

- Country
- State or region
- City

Attention: A sample geo-location database is available from the MaxMind site (https://www.maxmind.com). In order to be able to download the database you need to sign up for a free GeoLite2 account. After the account has been activated you will be granted access to the GeoLite2 database (CSV format).

Procedure

1. Obtain or create an appropriate geolocation data file in ZIP format.

   The file or files you must use depend on whether you want support for IPv4 addresses, IPv6 addresses, or both.

   Note: Both IPv4 and IPv6 files must be in the import ZIP, even if only one of the two data sets are going to be used.

   - **For IPv4:** The file must contain two CSV files. One file contains all of the possible locations and the other contains the IP blocks and their corresponding locations.

     Locations file

     GeoIP (version 1 database)

     Each line in the locations file corresponds to one location and is in the following format:

     ```
     location id,country,region,city,,,,,
     ```

     Attention: You must include the 5 commas after `city`commas in your locations file for version 1 data.

     GeoIP (version 2 database)

     Each line in the locations file corresponds to one location and is in the following format:

     ```
     geoname id,,,,country iso code,,subdivision 1 iso code,,,,city name,,,
     ```

     Attention: You must include the four commas separating the `geoname id` and `country iso code`, the two commas separating the `country iso code` and `subdivision (region) iso code`, the four commas separating the `subdivision iso code` and `city name` and the three commas after the `city` name in your location file for version 2 data.

     `country`

     A two-letter country code. For assistance with locating a country code, see `geoCountryCode` in "Predefined attributes" on page 19.

     `region`

     A two-character region code. For assistance with locating a region code, see `geoRegionCode` in "Predefined attributes" on page 19.

     `city`

     The name of a city.

     The locations file must have `Location` in its file name. The sample provided is named: `GeoLiteCity-Location.csv`

     IP blocks file

     GeoIP (version 1 database)

     Each line in the IP blocks file corresponds to one IP block and is in the following format:

     ```
     startip,endip,location id
     ```

     GeoIP (version 2 database)

     Each line in the IP block file corresponds to one IP block and is in the following format:

     ```
     network (CIDR format),geoname_id,,,,,,latitude,longitude,
     ```

     Attention: You must include the 6 commas after `geoname_id` and the trailing commas in your IPv4 log block files for version 2 data.

     `startip`

     The first IP address in the block that is represented as an integer.

endip
: The last IP address in the block that is represented as an integer.

location id
: The integer that is defined in the locations file that corresponds with the IP block.

The IP blocks file must have `Blocks` in its name. The sample provided is named: `GeoLiteCity-Blocks.csv`

⚠ **Attention:** Ensure that the CSV files contain all of the data that you want to load in the database. When you import the file, the existing data is removed and replaced with the data in the file.

- **For IPv6:** The file must contain one CSV file that contains all of the location and IP block information.

  Each line in the file corresponds to one location and IP block combination:

  GeoIP (version 1 database):
  Each line in the file corresponds to one location and IP block combination in the format:

  ```
  startip string, endip string, startip int, endip int, country,
    region, city,,,,,
  ```

  ⚠ **Attention:** You must include the 5 commas after `city` in your IPv6 block file.

  GeoIP (version 2 database):
  Each line in the file corresponds to one location and IP block combination in the format:

  ```
  network (CIDR format),geoname_id,,,,,,,,
  ```

  ⚠ **Attention:** You must include the 8 trailing commas after the `geoname_id` in your ipv6 blocks file. For GeoIP2 data the *country*, *region* and *city* attributes are read from the corresponding `geoname_id` in the locations file

  startip *string*
  : The first IP address in the block that is represented as a hexadecimal string. For example, a IPv6 string might be `2001:200:ffff:ffff:ffff:ffff:ffff:ffff`

  endip *string*
  : The last IP address in the block that is represented as a hexadecimal string.

  startip *integer*
  : The first IP address in the block that is represented as an integer. For example, the IPv6 integer that corresponds to `startip 2001:200:ffff:ffff:ffff:ffff:ffff:ffff` might be 42540528806023212578155541913346768895.

  endip *integer*
  : The last IP address in the block that is represented as an integer.

  country
  : A two-letter country code. For assistance with locating a country code, see `geoCountryCode` in "Predefined attributes" on page 19.

  region
  : A two-character region code. For assistance with locating a region code, see `geoRegionCode` in "Predefined attributes" on page 19.

  city
  : The name of a city.

  The file must have v6 in its file name. The sample provided is named: `GeoLiteCityv6.csv`

2. Create a file in ZIP format that contains the files you want to upload.

   For example, if you want to use both IPv4 and IPv6, include all the CSV files that you created in one ZIP formatted file.

3. Log in to the local management interface.

4. Click **System**.
5. Under **Updates and Licensing**, click **Geolocation Database**.
6. Click **Import**.
7. Select the geolocation file in ZIP format.
8. Click **Import**.

   Note: Importing the data can take more than 20 minutes to complete.
9. Click **Refresh Status** to check the import process.

   When the process is complete, the status says **Loaded**.

Attribute properties

When you add or modify an attribute, you specify properties that make that attribute unique.

Adding attributes

Specify the following properties when you add an attribute:

Name
: A unique name for the attribute.

Description
: A description of the attribute.

Identifier
: The internal name of the attribute that is used in the generated XACML policy.

Issuer
: The identifier of the policy information point from which the value of the attribute is retrieved. If an attribute can be returned from multiple policy information points, the issuer property specifies which policy information point to use.

  Note: Use this field only if you are using a policy information point. Otherwise, leave this field blank.

Type
: Indicates whether the attribute is used for policies or risk profiles or both. If neither check box is selected, the attribute is not available for policies or risk profiles.

Category
: The part of the XACML request that the attribute value comes from.

Data type
: The type of values that the attribute can handle. In a policy rule with an attribute, the data type indicates how the attribute can be compared to a value. In a risk profile, the risk matchers compare attribute values that have the same data type.

Matcher
: An attribute matcher compares the values of a specified attribute in the incoming device fingerprint with the existing device fingerprint of the user.

Storage Domain
: The storage domain indicates whether the attribute is stored as a `device`, `session`, or `behavior` attribute.

  Device fingerprint data
  : Consists of attributes that are stored when a device is registered. The incoming device fingerprint is compared against this stored repository of trusted device fingerprints.

  Session data
  : Consists of the session attributes of the user that are stored temporarily until the session times out. However, if the device is registered, the session attributes are also stored as part of the device fingerprint. If `session` is selected, the attribute is collected in the user's session.

Behavior data

Is historic data that is stored in the database and used for behavior-based attribute matching. For example, the login timestamps of the user over the previous three months. If an attribute is included in a risk profile configuration and the storage domain is not specified, the default storage domain is `device`.

Modifying attributes

All the properties for an attribute are displayed. However, you can modify only some of attribute properties. Also, if an attribute is included in a policy, you cannot make further updates to the attribute.

You can modify the following properties:

Editable properties of predefined attributes

Storage Domain

The storage domain indicates whether the attribute is stored as a `device`, `session`, or `behavior` attribute. If `session` is selected, the attribute is collected in the user's session. If an attribute is included in a risk profile configuration and the storage domain is not specified, the default storage domain is `device`.

Editable properties of custom attributes

Name

A unique name for the attribute.

Description

A description of the attribute.

Identifier

The internal name of the attribute that is used in the generated XACML policy.

Issuer

The identifier of the policy information point from which the value of the attribute is retrieved. If an attribute can be returned from multiple policy information points, the issuer property specifies which policy information point to use.

Note: Use this field only if you are using a policy information point. Otherwise, leave this field blank.

Type

Indicates whether the attribute is used for policies or risk profiles or both. If neither check box is selected, the attribute is not available for policies or risk profiles.

Category

The part of the XACML request that the attribute value comes from.

Data type

The type of values that the attribute can handle. In a policy rule with an attribute, the data type indicates how the attribute can be compared to a value. In a risk profile, the risk matchers compare attribute values that have the same data type.

Matcher

An attribute matcher compares the values of a specified attribute in the incoming device fingerprint with the existing device fingerprint of the user.

Storage Domain

The storage domain indicates whether the attribute is stored as a `device`, `session`, or `behavior` attribute. If `session` is selected, the attribute is collected in the user's session. If an attribute is included in a risk profile configuration and the storage domain is not specified, the default storage domain is `device`.

Related tasks

"Managing attributes" on page 13

Attributes represent unique information about a request. You use attributes in policy rules and risk profiles to match the attributes in a request. You can view, add, modify, and delete attributes.

Predefined attributes

An appliance with Advanced Access Control uses attributes to provide information about users and devices that try to access a protected resource. The appliance also includes a set of commonly used attributes called *predefined attributes*.

Five values describe each predefined attribute:

- Category
- Type
- Data type
- Source type
- Source

Table 1. Predefined attribute categories. Categories indicate the type of information that each attribute conveys.

Category	**Category description**
Action	Indicates the user action.
Environment	Indicates when and how the user is trying to access the resource.
Resource	Gives information about what the user is trying to access.
Subject	Indicates who is trying to access the resource.

Table 2. Predefined attribute types

Type	**Type description**
Access policy	The administrator uses policy attributes to create policies.
Risk profile	The administrator uses risk attributes to create risk profiles.

Table 3. Predefined attribute data types. Each predefined attribute has a data type. Data types are classifications that identify the possible values for each type of attribute.

Data type	**Data type description**
Boolean	Condition that refers to two possible values: • True • False
Date	Date of the request.
Integer	Number that can be written without a fractional or decimal component.
String	Sequence of characters.
Time	Time of the request.
X500Name	Values with distinguished names.

Table 4. Predefined attribute source types. Source types indicate the source of each attribute.	
Source type	**Source type description**
Active	Collected by the attribute collection service. The administrator must add JavaScript to the application so that active attributes can be collected. For example: system fonts.
Derived	Generated by a policy information point (PIP). For example: risk score.
Passive	Collected from the browser by the external authorization service (EAS) and placed into an XACML request. Attributes with this source type are collected by the policy enforcement point (PEP) without installing more software or challenging the client to provide more details. For example: user-agent HTTP header and client IP address.

Table 5. Predefined attribute sources. Sources indicate where the attributes originate from.	
Source	**Source description**
Attribute collection service	Collects information about the user device such as browser information, the operating system of the device, and the language of the device.
Consent external authentication interface	Asks the user for a device registration decision.
Device fingerprint count PIP	Counts the number of devices that are associated with the user.
Fiberlink MaaS360 PIP	Retrieves device attributes from the registered MaaS360 device inventory.
Geolocation PIP	Looks up the location of the user that is based on the IP address.
HTTP headers	Provides information about the request.
IP reputation PIP	Generates the IP reputation. See IP reputation for more information about IP reputation.
POST data	Collects information about the user and sends it to the external authorization service (EAS) as POST data. The EAS inserts this POST data into the decision request.
Risk engine	Generates the risk score. See Risk score calculation for more information about risk score calculation.
System time	Keeps the time of the system.
Security Verify Access credential	Collects information about the user from Security Verify Access.
Worklight JavaScript PIP	Parses the POST data from a Worklight adapter invocation and returns custom attributes that are created from the data that is contained within the POST from the `parameters` element.

accessTime

The **accessTime** attribute contains the record of the past login times that belongs to the user.

The **accessTime** attribute uses the ISO 8601 standard: *YYYY-MM-DD'T'hh:mm:ss'Z'*. For example: `2013-06-07T04:02:33Z`

Category	Type	Data type	Source type	Source
Environment	Risk profile	Time	Passive	System time

action

The **action** attribute indicates the action that is performed on the resource

The **action** attribute includes one of the following values:

- `CONNECT`
- `DELETE`
- `GET`
- `HEAD`
- `OPTIONS`
- `PATCH`
- `POST`
- `PUT`
- `TRACE`

Category	Type	Data type	Source type	Source
Action	Access policy, Risk profile	String	Passive	HTTP headers

authenticationLevel

The **authenticationLevel** attribute is a numeric value that specifies the authentication level of a user.

It increases as the levels of authentication that belong to the user increase. For example: A possible authentication level is 2.

Category	Type	Data type	Source type	Source
Subject	Access policy	Integer	Passive	Security Verify Access credential

authenticationMechanism

The **authenticationMechanism** attribute indicates the location of the user registry that belongs to the requesting user.

For example: `LDAP Registry`

Category	Type	Data type	Source type	Source
Subject	Access policy, Risk profile	String	Passive	Security Verify Access credential

authenticationMechanismTypes

The **authenticationMechanismTypes** attribute indicates the types of authentication mechanisms that the user completes during the current authenticated session.

The **authenticationMechanismTypes** attribute values consist of the unique identifiers for the completed authentication mechanisms. The following values are predefined:

- `urn:ibm:security:authentication:asf:mechanism:hotp`
- `urn:ibm:security:authentication:asf:mechanism:macotp`
- `urn:ibm:security:authentication:asf:mechanism:rsa`
- `urn:ibm:security:authentication:asf:mechanism:totp`
- `urn:ibm:security:authentication:asf:mechanism:consent_register_device`
- `urn:ibm:security:authentication:asf:mechanism:otp`
- `urn:ibm:security:authentication:asf:mechanism:http_redirect`
- `urn:ibm:security:authentication:asf:mechanism:password`

Category	Type	Data type	Source type	Source
Subject	Access policy	String	Passive	Security Verify Access credential

authenticationMethod

The **authenticationMethod** attribute indicates the method of authentication that is used, such as `password`.

The **authenticationMethod** attribute includes the following values:

- `ext-auth-interface`
- `password`
- `ssl`
- `token-card`
- `unauthenticated`

Category	Type	Data type	Source type	Source
Subject	Access policy, Risk profile	String	Passive	Security Verify Access credential

authenticationTypes

The **authenticationTypes** attribute indicates the types of authentication policies that the user completes during the current authenticated session.

The **authenticationTypes** attribute values consist of the unique identifiers for the completed authentication policies. The values can include the identifiers for predefined authentication policies or custom values that are created by the administrator. The following values are predefined:

- `urn:ibm:security:obligation:register_device`
- `urn:ibm:security:authentication:asf:otp`
- `urn:ibm:security:authentication:asf:hotp`
- `urn:ibm:security:authentication:asf:totp`
- `urn:ibm:security:authentication:asf:rsa`
- `urn:ibm:security:authentication:asf:email`
- `urn:ibm:security:authentication:asf:sms`
- `urn:ibm:security:authentication:asf:consent_register_device`

- `urn:ibm:security:authentication:asf:password`
- `urn:ibm:security:authentication:asf:macotp`
- `urn:ibm:security:authentication:asf:hotp`
- `urn:ibm:security:authentication:asf:otp`
- `urn:ibm:security:authentication:asf:totp`
- `urn:ibm:security:authentication:asf:sms`
- `urn:ibm:security:authentication:asf:email`
- `urn:ibm:security:authentication:asf:rsa`
- `urn:ibm:security:authentication:asf:consent_register_device`
- `urn:ibm:security:authentication:asf:http_redirect`
- `urn:ibm:security:authentication:asf:password_smsotp`
- `urn:ibm:security:authentication:asf:password_hotp`
- `urn:ibm:security:authentication:asf:password_macotp`
- `urn:ibm:security:authentication:asf:password_rsa`
- `urn:ibm:security:authentication:asf:password_emailotp`
- `urn:ibm:security:authentication:asf:password_totp`
- `urn:ibm:security:authentication:asf:password_otp`

Category	Type	Data type	Source type	Source
Subject	Access policy	String	Passive	Security Verify Access credential

browserPlugins

The **browserPlugins** attribute indicates all of the installed plug-ins in the browser in a comma-separated list.

For example: `Shockwave Flash,Chrome Remote Desktop Viewer,Widevine Content Decryption Module,Native Client,Chrome PDF Viewer,Java(TM) Plug-in 1.7.0,Citrix Receiver for Linux`

Category	Type	Data type	Source type	Source
Environment	Risk profile	String	Active	Attribute collection service

colorDepth

The **colorDepth** attribute indicates the bit depth of the color palette of the device.

For example: 32

Category	Type	Data type	Source type	Source
Environment	Risk profile	Integer	Active	Attribute collection service

currentDate

The **currentDate** attribute indicates the date of the request.

The **currentDate** attribute uses the format: *YYYY-MM-DDzzzzzz*. For example: `2013-05-20–06:00`

Category	Type	Data type	Source type	Source
Environment	Access policy	Date	Passive	System time

currentTime

The **currentTime** attribute indicates the time of the request.

It uses the format: *hh:mm:sszzzzzz*. For example: `12:15:26+01:00`

Category	Type	Data type	Source type	Source
Environment	Access policy	Time	Passive	System time

deviceFonts

The **deviceFonts** attribute indicates all of the installed fonts on the device in a comma-separated list.

For example: `Arial, Dingbats, Georgia, Tahoma, Times New Roman, Verdana`

Category	Type	Data type	Source type	Source
Environment	Risk profile	String	Active	Attribute collection service

deviceLanguage

The **deviceLanguage** attribute indicates the language of the device.

It uses the Internet Engineering Task Force (IETF) language tag standard. For example: `en-US`

Category	Type	Data type	Source type	Source
Environment	Access policy, Risk profile	String	Active	Attribute collection service

deviceName

The **deviceName** attribute indicates the name of the device.

For example: `My laptop`

Category	Type	Data type	Source type	Source
Environment	Access policy, Risk profile	String	Active	Consent external authentication interface

devicePlatform

The **devicePlatform** attribute indicates the operating system of the device.

For example: `Linux x86_64`

Category	Type	Data type	Source type	Source
Environment	Access policy, Risk profile	String	Active	Attribute collection service

fiberlink.maas360.device.compliance.state

The **fiberlink.maas360.device.compliance.state** attribute indicates the Fiberlink MaaS360 ownership of the registered device.

Valid values are:

- `In Ccompliance`
- `Out of Compliance`

Category	Type	Data type	Source type	Source
Subject	Policy, Risk	String	Derived	Fiberlink MaaS360 PIP

fiberlink.maas360.device.ids

The **fiberlink.maas360.device.ids** attribute indicates the Fiberlink MaaS360 registered device IDs.

Category	Type	Data type	Source type	Source
Environment	None	String	Derived	Fiberlink MaaS360 PIP

fiberlink.maas360.device.ownership

The **fiberlink.maas360.device.ownership** attribute indicates a Fiberlink MaaS360 ownership of the registered device.

Valid values are:

- `Corporate Owned`
- `Corporate Shared`
- `Corporate Third Party`
- `Employee Owned`
- `Not Defined`

Category	Type	Data type	Source type	Source
Subject	Policy, Risk	String	Derived	Fiberlink MaaS360 PIP

fiberlink.maas360.device.jailbroken

The **fiberlink.maas360.device.jailbroken** attribute indicates the Fiberlink MaaS360 jailbroken state of the registered device.

Valid values are:

- `Yes`
- `No`

Category	Type	Data type	Source type	Source
Subject	Policy, Risk	String	Derived	Fiberlink MaaS360 PIP

fiberlink.maas360.device.last.reported

The **fiberlink.maas360.device.last.reported** attribute indicates the Fiberlink MaaS360 last reported date and time of the device to the server.

The date and time uses the ISO 8601 standard: *YYYY-MM-DD*'T'*hh*:*mm*:*ss*. For example: `2014-06-07T04:02:33`

Category	Type	Data type	Source type	Source
Environment	Policy, Risk	String	Derived	Fiberlink MaaS360 PIP

fiberlink.maas360.device.managed.status

The **fiberlink.maas360.device.managed.status** attribute indicates the Fiberlink MaaS360 managed status of the registered device.

Valid values are:

- `Enrolled`
- `Not Enrolled`
- `User Removed Control`
- `Control Removed`
- `Pending Control Removal`

Category	Type	Data type	Source type	Source
Subject	Policy, Risk	String	Derived	Fiberlink MaaS360 PIP

fiberlink.maas360.device.match.found

The **fiberlink.maas360.device.match.found** attribute indicates a Fiberlink MaaS360 match against the attribute collection service to a registered device.

Valid values are:

- `True`
- `False`

Category	Type	Data type	Source type	Source
Environment	Policy	Boolean	Derived	Fiberlink MaaS360 PIP

geoCity

The **geoCity** attribute indicates the city in which the device is located.

For example: `Austin`

Category	Type	Data type	Source type	Source
Environment	Access policy, Risk profile	String	Derived	Geolocation PIP

geoCountryCode

The **geoCountryCode** attribute indicates the country in which the device is located.

It uses the two letter country code according to the ISO-3166-1 standard. For example: The country code for the United States is US.For a list of country codes, see the International Organization for Standardization homepage. To find your two letter country code:

1. On the right side of the page, click **ISO 3166**.
2. In the **HTML, Text and XML versions** section, choose a format for the list of country codes.

Category	Type	Data type	Source type	Source
Environment	Access policy, Risk profile	String	Derived	Geolocation PIP

geoLocation

The **geoLocation** attribute indicates the latitude, longitude, and accuracy of the requesting device that is obtained with the W3C geolocation standard.

It uses the format: *latitude, longitude, accuracy* For example: `60.170833, 24.9375, 98`

Category	Type	Data type	Source type	Source
Environment	Risk profile	String	Active	Attribute collection service

geoRegionCode

The **geoRegionCode** attribute indicates the state or province in which the device is located.

It uses the two letter code for the region. For the United States and Canada, the region codes are according to the ISO-3166-2 standard. For example: The region code for the province of Nova Scotia in Canada is NS.

For all other countries, the region codes are according to the FIPS 10-4 standard. Your region code is the last pair of characters in the four-character code that is assigned to your region. For a list of region codes, go to the National Geospatial Intelligence Agency homepage. To find your two-letter region code:

1. On the left side of the page, click **Country codes**.
2. In the **Geopolitical Entities and Codes (Formerly FIPS PUB 10-4)** section, find the most recent update and choose a format for a list of countries and region codes.

For example: The region code for Queensland in Australia is `04`.

Category	Type	Data type	Source type	Source
Environment	Access policy, Risk profile	String	Derived	Geolocation PIP

groups

The **groups** attribute lists the groups of which the user is a member.

The **`groups`** attribute is a multivalue attribute. See Attribute properties. For example: `engineering`

Category	Type	Data type	Source type	Source
Subject	Access policy	String	Passive	Security Verify Access credential

groupsDN

The **groupsDN** attribute lists the groups of which the user is a member.

The **`groupsDN`** attribute has a distinguished name format and is a multivalue attribute. For example: `CN=engineering, O=ibm, C=us`

Category	Type	Data type	Source type	Source
Subject	Access policy	X500Name	Passive	Security Verify Access credential

http:accept

The **http:accept** attribute indicates the acceptable content types.

For example: `text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8`

Category	Type	Data type	Source type	Source
Environment	Risk profile	String	Passive	HTTP headers

http:acceptEncoding

The **http:acceptEncoding** attribute indicates the acceptable encoding types.

For example: `gzip, deflate`

Category	Type	Data type	Source type	Source
Environment	Risk profile	String	Passive	HTTP headers

http:acceptLanguage

The **http:acceptLanguage** attribute indicates the acceptable human languages for response.

For example: `en-US,en;q=0.8`

Category	Type	Data type	Source type	Source
Environment	Access policy, Risk profile	String	Passive	HTTP headers

http:host

The **http:host** attribute indicates the host that is requested.

Category	Type	Data type	Source type	Source
Environment	Access policy	String	Passive	HTTP headers

http:uri

The **http:uri** attribute indicates the requested resource and the query string.

For example: `/mga/sps/ac/rest`

Category	Type	Data type	Source type	Source
Environment	Access policy	String	Passive	HTTP headers

http:userAgent

The **http:userAgent** attribute indicates the user agent.

For example: `Mozilla/5.0 (X11; Linux i686 (x86_64)) AppleWebKit/537.17 (KHTML, like Gecko) Chrome/24.0.1312.57 Safari/537.17`

Category	Type	Data type	Source type	Source
Environment	Access policy, Risk profile	String	Passive	HTTP headers

ipAddress

The **ipAddress** attribute indicates the string representation of the IP address.

Specify one of the following IP address formats depending on the network type:

IPv4 format
`192.9.200.1`

IPv6 format
`FEDC:BA98:7654:3210:FEDC:BA98:7654:3210`

Category	Type	Data type	Source type	Source
Subject	Access policy, Risk profile	String	Passive	Security Verify Access credential

ipReputation

The **ipReputation** attribute lists any combination of certain classifications whose IP reputation score is at or above the threshold.

The **ipReputation** attribute lists any combination of the following classifications:

- `Anonymous Proxies`
- `Botnet Command and Control Server`
- `Dynamic IPs`
- `Malware`
- `Scanning IPs`
- `Spam`

The user can either configure the threshold or use the default threshold of 50.

Category	Type	Data type	Source type	Source
Subject	Access policy	String	Derived	IP reputation PIP

oauthScopeResource

The **oauthScopeResource** attribute indicates authorization of an API call for OAuth. This value originates when the OAuth grant is created.

Category	Type	Data type	Source type	Source
Resource	Access policy	String	Passive	OAuth External Authorization Service (EAS)

oauthScopeSubject

The **oauthScopeSubject** attribute indicates the identity for the session for OAuth. This value originates when the OAuth grant is created.

This value is only present for access control policies when a user authenticates by using the OAuth mechanism. If a user authenticates another way, this value is not present for an access control decision.

Category	Type	Data type	Source type	Source
Subject	Access policy, Risk profile	String	Passive	OAuth authentication mechanism of the reverse proxy server

qop

The **qop** attribute indicates the quality of protection information.

The **qop** attribute includes the following valid values:

- `None`
- `Integrity`
- `Privacy`

Category	Type	Data type	Source type	Source
Subject	Access policy, Risk profile	String	Passive	Security Verify Access credential

qradar.uba.risk.score

The **`qradar.uba.risk.score`** attribute indicates the risk score of a user, which is returned by the QRadar User Behavior Analytics (UBA) app.

A value of 0 indicates that the user is not indulging in risky/malicious activities according to the QRadar UBA app. A non-zero value indicates that the user has been indulging in risky/malicious activities.

The risky behavior threshold is set in the QRadar UBA app. If the value of the **`qradar.uba.risk.score`** attribute is consistently being set to '0', this threshold value in the QRadar UBA app may require an adjustment.

Category	Type	Data type	Source type	Source
Source	Policy, Risk	Integer	Derived	QRadar UBA PIP

registeredDeviceCount

The **registeredDeviceCount** attribute indicates the number of devices that the user registers.

Category	Type	Data type	Source type	Source
Subject	Access policy	Integer	Derived	Device fingerprint count PIP

resource

The **resource** attribute indicates the Worklight adapter parameters URL encoded JSON array POST data.

It does not include the query string and is usually the same as the **`http:uri`** attribute.

For example: `/mga/sps/ac/rest`

Category	Type	Data type	Source type	Source
Resource	Access policy	String	Passive	HTTP headers

riskScore

The **riskScore** attribute indicates the calculated risk score, which is a number 0 - 100.

The risk score indicates the level of risk that is associated with authenticating a request device. See Risk score calculation.

Category	Type	Data type	Source type	Source
Subject	Access policy	Integer	Derived	Risk engine

scheme

The **scheme** attribute indicates the protocol that is used to access the resource.

For example: `http`, `https`, or `ftp`.

Category	Type	Data type	Source type	Source
Environment	Access policy	String	Passive	HTTP headers

screenAvailableHeight

The **screenAvailableHeight** attribute indicates the height of the screen of the requesting device without the toolbar.

For example: `1025`

Category	Type	Data type	Source type	Source
Environment	Risk profile	Integer	Active	Attribute collection service

screenAvailableWidth

The **screenAvailableWidth** attribute indicates the width of the screen of the requesting device without the toolbar.

For example: `1920`

Category	Type	Data type	Source type	Source
Environment	Risk profile	Integer	Active	Attribute collection service

screenHeight

The **screenHeight** attribute indicates the height of the screen of the requesting device.

For example: `1080`

Category	Type	Data type	Source type	Source
Environment	Risk profile	Integer	Active	Attribute collection service

screenWidth

The **screenWidth** attribute indicates the width of the screen of the requesting device.

For example: `1920`

Category	Type	Data type	Source type	Source
Environment	Risk profile	Integer	Active	Attribute collection service

userConsent

The **userConsent** attribute indicates the registration decision of the user.

A registered device is `true`. An unregistered device is `false`.

Category	Type	Data type	Source type	Source
Environment	Access policy	Boolean	Active	Consent external authentication interface

userDN

The **userDN** attribute indicates the distinguished name for the user in a registry.

For example: `CN=Mark Smith,o=IBM, L=Austin, S=Texas, C=US`

Category	Type	Data type	Source type	Source
Subject	Access policy, Risk profile	X500Name	Passive	Security Verify Access credential

username

The **username** attribute identifies the user.

For example: `sec_master`

Category	Type	Data type	Source type	Source
Subject	Access policy, Risk profile	String	Passive	Security Verify Access credential

worklight.adapter.adapter

The **worklight.adapter.adapter** attribute indicates the Worklight adapter adapter POST data.

Category	Type	Data type	Source type	Source
Environment	Access policy, Risk profile	String	Passive	POST data

worklight.adapter.balance.account

The **worklight.adapter.balance.account** attribute indicates the Worklight adapter BankingApprovals getBalance account.

Category	Type	Data type	Source type	Source
Environment	Access policy, Risk profile	String	Derived	Worklight JavaScript PIP

worklight.adapter.parameters

The **worklight.adapter.parameters** attribute indicates the Worklight adapter parameters URL encoded JSON array POST data.

Category	Type	Data type	Source type	Source
Environment	Access policy, Risk profile	String	Passive	POST data

worklight.adapter.procedure

The **worklight.adapter.procedure** attribute indicates the Worklight adapter procedure POST data.

Category	Type	Data type	Source type	Source
Environment	Access policy, Risk profile	String	Passive	POST data

worklight.adapter.transfer.account.from

The **worklight.adapter.transfer.account.from** attribute indicates the Worklight adapter BankingApprovals doTransfer from account.

Category	Type	Data type	Source type	Source
Environment	Access policy, Risk profile	String	Derived	Worklight JavaScript PIP

worklight.adapter.transfer.account.to

The **worklight.adapter.transfer.account.to** attribute indicates the Worklight adapter BankingApprovals doTransfer to account.

Category	Type	Data type	Source type	Source
Environment	Access policy, Risk profile	String	Derived	Worklight JavaScript PIP

worklight.adapter.transfer.amount

The **worklight.adapter.transfer.amount** attribute indicates the Worklight adapter BankingApprovals doTransfer amount.

Category	Type	Data type	Source type	Source
Environment	Access policy, Risk profile	Integer	Derived	Worklight JavaScript PIP

worklight.device.id

The **worklight.device.id** attribute indicates the unique ID of the device that is assigned by Worklight.

Category	Type	Data type	Source type	Source
Environment	Access policy, Risk profile	String	Passive	HTTP headers

worklight.version.app

The **worklight.version.app** attribute indicates the version of the Worklight application.

Category	Type	Data type	Source type	Source
Environment	Access policy, Risk profile	String	Passive	HTTP headers

worklight.version.native

The **worklight.version.native** attribute indicates the version of the Worklight application.

Category	Type	Data type	Source type	Source
Environment	Access policy, Risk profile	String	Passive	HTTP headers

worklight.version.platform

The **worklight.version.platform** attribute indicates the version of the Worklight Software Development Kit for the Advanced Access Control.

Category	Type	Data type	Source type	Source
Environment	Access policy, Risk profile	String	Passive	HTTP headers

Configuring the hash algorithm for attribute storage

Hashing encodes a character string as a fixed-length bit string for comparison. Context-based access hashes certain attributes by default. You can change the hash algorithm and specify additional attributes that you want to hash.

About this task

By default, when attributes are stored in the context-based access database, the attributes that exceed the maximum length according to the schema are hashed. You can also specify any other attribute that you require to be hashed. For example, you might want to hash values that are considered confidential or private.

The default hash algorithm that context-based access uses for storing these attributes is SHA256. Context-based access also uses the default when the hash algorithm is not configured properly. You can specify any other hash algorithm that Java Security supports.

Procedure

1. Log in to the local management interface.
2. Click **AAC**.
3. Under **Global Settings**, click **Advanced Configuration**.
4. Under **Key**, find the name of the property that you must work with.
5. Take one of the following actions:
   - Configure the `attributeCollection.attributesHashEnabled` property.
     a. Click the edit icon .
     b. Enter the Identifier names of the attributes that you want the `attributeCollection.attributesHashEnabled` property to hash. For example: `urn:ibm:security:environment:http:userAgent, urn:ibm:security:environment:deviceFonts, urn:ibm:security:environment:browserPlugins`

        To find the list of attributes that context-based access can hash, complete the following steps:

        i) Log in to your local management interface
        ii) Click **AAC**
        iii) Under **Policy**, click **Attributes**.
        iv) Select the name of an attribute, and click Modify attributes .
        v) Under **Modify Attribute**, find the Identifier of the attribute.
        vi) Use the Identifier of the attribute in your list of attributes that you want context-based access to hash.
        vii) Click **Cancel** to exit.
   - Configure the `attributeCollection.hashAlgorithm` property.
     a. Click the edit icon .
     b. Set the value for the `attributeCollection.hashAlgorithm` property to one of the following values:
        – SHA1
        – SHA512
        – SHA256
     c. Click **Save**.
6. When you make changes to the properties, the appliance displays a message that there are undeployed changes. If you are finished making changes, deploy them.

Dynamic attributes

You can capture additional context-based access information in dynamic attributes that you create or customize.

Attributes used to calculate risk score come from various sources. Context-based access can capture and use many default attributes. However, you might require more information to be captured. This data can come from:

- Data that is gathered when a session is established that is sent to the Attribute Collector Service. This data persists during the session and is stored in the session table in context-based access. This type of information is gathered by the `info.js` file.
- Data that is derived from a particular request for which authorization is being performed. This data can be stored in either the session or behavior tables.

- Data that is provided by a PIP during the authorization decision. This data can be stored in either the session or behavior tables.

This type of data can be captured in *dynamic attributes*.

Related concepts

"Scenarios for adding and manipulating attributes" on page 35
You can dynamically add attributes or manipulate the values of attributes for various purposes.

Scenarios for adding and manipulating attributes

You can dynamically add attributes or manipulate the values of attributes for various purposes.

The following topics provide specific instructions to create or update the JavaScript file for dynamic attributes:

1. "Coding the dynamic.attributes.js file" on page 35
2. "Updating and deploying the dynamic.attributes.js file" on page 37

The following scenarios show examples of the data you can dynamically capture:

Creating a device.name attribute

In this scenario, you can create a device name attribute that a user can easily identify. While configuring a system that only has hybrid applications, you can gather a few more attributes while using the `info.js` active collection library. For example, if you combine three separate device attributes that the client sends, you can uniquely identify the device. The attributes are:

- Device Model
- Device Model Version
- Device UUID (unique ID)

Combining these attributes into a single attribute with a zero device weight results in a meaningful attribute name that you can use to display the device. You can accomplish this by providing a server-side JavaScript rule that runs when an attribute collection service request arrives.

Adding the last risk score as a session attribute to be used by a third party

The risk score is a derived attribute using a combination of other attributes. It can be useful for third party authentication services.

In this scenario, you can implement a JavaScript rule that captures the current risk score and stores it as a session attribute. Later, when a third-party application is determining possible further action based on the current context, it can make a remote call to the attribute collection service using the REST API call to retrieve the sessions attributes, such as risk score.

Coding the dynamic.attributes.js file

Add JavaScript code to the `dynamic.attributes.js` file to dynamically add attributes or modify the values of attributes.

Before you begin

- Review the "Scenarios for adding and manipulating attributes" on page 35.
- Determine which attributes you want to modify and how to store them. Create a function for each one, which you will add to the `dynamic.attributes.js` file in the steps below.
- Deploy context-based access.

About this task

The `modifySessionAttributes` and `modifyBehaviorAttributes` functions are the only functions called by the context-based access processing. Therefore, any function you add must be called within one of these functions in order to be executed.

Procedure

1. Create a JavaScript file or edit the sample `dynamic.attributes.js` file.
2. Define the necessary import packages for the APIs you use. For example:

   ```
   importPackage(com.tivoli.am.rba.extensions);
   importClass(Packages.com.tivoli.am.rba.attributes.AttributeIdentifier);
   ```

3. Use the `modifySessionAttributes` function to add or manipulate the session attributes.

   ```
   function modifySessionAttributes(attributes, username, session)
   ```

4. Add code to work with session attributes.
   a) Create an attribute identifier. For example, to use the risk score value, the following line is required:

      ```
      var riskScoreIdentifier = new AttributeIdentifier("riskScore",
        "urn:ibm:security:subject:riskScore","Integer",
        "urn:ibm:security:issuer:RiskCalculator");
      ```

   b) Use the attribute identifier to get the value of this attribute. For example, to use the risk score, the following line is required:

      ```
      var riskScoreValue = session.getValue(riskScoreIdentifier);
      ```

   c) Call the function you created to process the attributes. Define the function either at the beginning or at the end of the `dynamic.attributes.js` file.
   d) Store your attribute to session storage. For example, to store the risk score, the following line is required:

      ```
      attributes.put(riskScoreIdentifier, riskScoreValue);
      ```

5. Use the `modifyBehaviorAttributes` function to add or manipulate behavior attributes.

   ```
   function modifyBehaviorAttributes(attributes, username, session)
   ```

6. Add code to work with behavior attributes.
   a) Create an attribute identifier.
   b) Use the attribute identifier to get the value of this attribute.
   c) Call the function you created to process the attributes. Define the function either at the beginning or at the end of the `dynamic.attributes.js` file.
   d) Store your attribute to behavioral storage.
7. Save the file.

Example

The following `dynamic.attributes.js` file is a sample that is available with Advanced Access Control. The code in this example JavaScript file captures the risk score. This is the only way you can save the risk score value.

```
/**
 * This script is executed after each request is processed by risk engine.
 * The intent is to allow users to capture attributes that don't get captured
 * automatically by the system. The attributes captured here will be stored
 * in either the session storage or the behavior storage (i.e., usage data, historical)
 * area of RBA, or both. The risk profile configuration dictates where the
 * attributes will be stored by the system.
 *
 * For any RBA specific API, necessary packages need to be imported as shown in this example.
```

```
 */


/**
 * Import RBA packages necessary for the script to execute.
 */
importPackage(com.tivoli.am.rba.extensions);
importClass(Packages.com.tivoli.am.rba.attributes.AttributeIdentifier);

/**
 * @param username - current user's name
 * @param attributes - java.util.Map where the 'dynamic' values need to be captured by
 *                     this javascript file.
 * @param session - object containing current values visible to incoming request context
 */
function modifySessionAttributes(attributes, username, session) {

    // creates an identifier with the attribute's name, URI, datatype, and the issuer
    var riskScoreIdentifier = new AttributeIdentifier("riskScore", "urn:ibm:security:subject:riskScore",
    "Integer", "urn:ibm:security:issuer:RiskCalculator");

    // retrieve the risk score
    var riskScoreValue = session.getValue(riskScoreIdentifier);

    // set the risk score to be stored as a session attribute
    attributes.put(riskScoreIdentifier, riskScoreValue);
}

/**
 * @param username - current user's name
 * @param attributes - java.util.Map where the 'dynamic' values need to be captured
 *                     by this javascript file.
 * @param session - RBA's com.tivoli.am.rba.fingerprinting.IValueContainer object
 *                  containing current values visible to incoming request context
 */
function modifyBehaviorAttributes(attributes, username, session) {

    // store any behavior attributes here
}
```

What to do next

Update and deploy the dynamic.attributes.js file.

Updating and deploying the dynamic.attributes.js file

To implement the `dynamic.attributes.js` file on your appliance, update the file and then deploy the change.

Before you begin

- Code the `dynamic.attributes.js` file to create the customizations you require.
- To use a custom attribute in a policy, create the attribute. See "Managing attributes" on page 13.

  You must set the following properties in the local management interface for the custom attribute:

  - Specify a name for the attribute.
  - Issuer: `urn:ibm:security:issuer:AttributeSession`
  - Category: `Environment`
  - Type: `Policy`
  - Storage Domain: `Session`

  Then, you must use the attribute you created in the rule section of a policy. See "Creating an access control policy" on page 100.

Procedure

1. Log in to the local management interface.
2. Click **AAC**.
3. Under **Global Settings**, click **Template Files**.
4. Locate the existing `dynamic.attributes.js` file in the `C/ac/javascript_rules` directory.

5. Select the `dynamic.attributes.js` file to edit or import:
    - Click **Manage** > **Edit** to update the existing file in an editing window by using copy and paste from the file you created. Click **Save**.
    - Click **Manage** > **Import** to import the script file you created and replace the existing file. Click **Browse** to select the location of the file, and click **Import** to replace the existing `dynamic.attributes.js` file.
6. Deploy the changes.

Custom attributes for the authorization service

You can modify the `[azn-decision-info]` stanza of the WebSEAL configuration file to make other data available to the external authorization service (EAS).

You can define your own attributes, called *custom attributes*, to specify in policies and for decision making. Update the WebSEAL configuration file with the attribute information so that the EAS can use these attributes. The WebSEAL EAS uses this attribute information to create the runtime security services request.

To update the WebSEAL configuration file for custom attributes:

1. Update the `[azn-decision-info]` stanza with the custom attribute information.
2. Define the type of data and category that the attribute represents using the `[user-attribute-definitions]` stanza.

[azn-decision-info] stanza

Add extra information from the HTTP request to the authorization decision information.

Syntax

```
<attr_ID> = <http_info>
```

Description

This stanza defines any extra information that is available to the authorization framework when it makes authorization decisions. This extra information can be obtained from various elements of the HTTP request, namely:

- HTTP method
- HTTP scheme
- HTTP cookies
- Request URI
- HTTP headers
- POST data
- Client IP address

If the requested element is not in the HTTP request, no corresponding attribute is added to the authorization decision information.

Options

attr_ID
The identifier of the attribute that contains the HTTP information.

http_info
The source of the information. It can be one of the following values:

- `method`
- `scheme`
- `uri`
- `client_ip`
- `header:`*<header-name>*

  where:

  <header-name>
  : The name of the header that contains information for WebSEAL to add to the authorization decision information. For example, `Host`.
- `cookie:`*<cookie-name>*

  where:

  <cookie-name>
  : The name of the cookie that contains information for WebSEAL to add to the authorization decision information.
- `post-data:`*<values>*,

  where the content and format of the *<values>* depends on the type of POST data.

  See "WebSEAL support for POST data" on page 39.

WebSEAL support for POST data

WebSEAL supports two types of POST data:

- Normal FORM data, which is the `application/x-www-form-urlencoded` content-type.

  To add normal FORM data to the HTTP request, use the following format for this entry:

  ```
  post-data:<post-data-name>
  ```

  where the *<post-data-name>* is the name of the selected form data field in the request. WebSEAL adds the corresponding value for this field to the authorization decision information.
- JavaScript Object Notation (JSON) data, which is the `application/json` content-type. For more information about the JSON syntax, see http://www.json.org.

  To search for a key in the JSON data and add its value to the HTTP request, use the following format:

  ```
  post-data:/"<JSON-node-id>"[[/"<JSON-node-id>"][<JSON-array-indx>]]...
  ```

  where:

  "<JSON-node-id>"
  : The name of a node in the JSON data.

    JSON data is essentially a hierarchy of name-value pairs. The forward slash character (/) that precedes each *"<JSON-node-id>"* identifies a level of the JSON hierarchy. You can repeatedly add *[/<JSON-node-id>]* elements to move through the JSON data hierarchy and identify the node that contains the value that you want WebSEAL to add to the authorization decision information.

    Each *<JSON-node-id>* must be:

    – Enclosed in double quotation marks.

    – Preceded by a forward slash character (/).

    – A case-sensitive match with a node in the JSON data hierarchy.

    If WebSEAL does not find a matching node name in the POST data, no corresponding attribute is added to the authorization decision information.

<JSON-array-indx>

The contents of a node in the JSON data might be a JSON array. If you configure WebSEAL to search for a JSON node that contains an array, specify the array index of the value that you want WebSEAL to use. Use a base of 0. In other words, the first entry in the array has an index of `0`.

Note: The *<JSON-array-indx>* is not enclosed in double quotation marks.

Usage notes:

- The square brackets ([]) in this syntax indicate an optional element. Do not include square brackets in your configuration entry. Similarly, the ellipsis (...) indicates that you can repeat the optional elements that precede it. Do not include the ellipsis in your configuration entry.
- WebSEAL returns only node values of the following JSON types:
  - `String`
  - `Number`
  - `true` or `false`
  - `null`

  If the value of the selected node is not one of the types in this list, WebSEAL does not return it as authorization decision information.

  Object and Array types cannot be added to the authorization decision information.

Usage

This stanza entry is optional.

Default value

None.

Example 1: Standard HTTP elements

```
HTTP_REQUEST_METHOD = method
HTTP_HOST_HEADER= header:Host
```

If these example configuration entries are set in the **`[azn-decision-info]`** stanza, WebSEAL adds the following attributes to the authorization decision information:

HTTP_REQUEST_METHOD
: Contains the HTTP method.

HTTP_HOST_HEADER
: Contains the data from the Host header.

Example 2: JSON POST data

For this example, consider the following JSON form data:

```
{ "userid": "jdoe",
  "transactionValue": "146.67",
  "accountBalances": {
        "chequing": "4345.45",
        "savings": "12432.23",
        "creditLine": "19999.12"
    }
}
```

The following configuration entries in the **[azn-decision-info]** stanza extract information from this JSON form data.

```
USERID = post-data:/"userid"
SAVINGS = post-data:/"accountBalances"/"savings"
```

The first entry prompts WebSEAL to search for the JSON node called **"userid"**. In this example, the value that is associated with the **"userid"** node is `jdoe`. WebSEAL adds this value to the HTTP request in an attribute called **USERID**.

When WebSEAL processes the second entry, it searches for a top-level JSON node called **"accountBalances"**. Under the **"accountBalances"** hierarchy, WebSEAL locates the **"savings"** JSON node. In the example data, the value that is associated with this node is `12432.23`. WebSEAL adds this value to the HTTP request in an attribute called **SAVINGS**.

WebSEAL adds the following attributes to the authorization decision information:

USERID
Contains the value `jdoe`.

SAVINGS
Contains the value `12432.23`.

Example 3: JSON POST data with a JSON array value

For this example, consider the following JSON form data:

```
{
    "userid": "jdoe",
    "transactionValue": "146.67",
    "accounts": [
        {"name": "chequing":, "balance": "4350.45"},
        {"name": "savings":, "balance": "4350.46"}
    ]
}
```

The following configuration entry is included in the **[azn-decision-info]** stanza:

```
SAVINGSBAL = post-data:/"accounts"/1/"balance"
```

WebSEAL processes this entry as follows:

1. Searches for a top-level node in the JSON data called **"accounts"**.
2. Locates the element in position 1 of the JSON array (base 0).
3. Searches for the **"balance"** name-value pair in this array element.
4. Adds the associated value to the authorization decision information.

In this example, WebSEAL adds the following attribute to the authorization decision information:

SAVINGSBAL
Contains the value `4350.46`.

[user-attribute-definitions] stanza

Use the `[user-attribute-definitions]` stanza to modify the data type, the category, or both of a custom attribute.

Syntax

```
attr_ID.datatype = data_type
attr_ID.category = category_name
```

Description

Use the appropriate stanza entry syntax depending on if you want to set the data type or category of a custom attribute from the default values.

Options

attr_ID
: Specify the attribute identifier for which you want to set the data type or category. The attribute ID must match the name that exists in the `[azn-decision-info]` stanza entry.

datatype *data_type*
: Set the data type of an attribute from the default of `string` to a specified *data_type*. Your choices are:

  - `string`
  - `boolean`
  - `integer`
  - `double`
  - `time`
  - `date`
  - `x500name`

category *category_name*
: Set the category of an attribute from the default of `Environment` to a specified *category_name*. Your choices are:

  - `Environment`
  - `Subject`
  - `Action`
  - `Resource`

Usage

This stanza entry is not required.

Default value

The default value for data type is `string`.

The default value for category is `Environment`.

Example: Updating the data type for JSON data

If you defined a custom attribute in the `[azn-decision-info]` stanza as:

```
urn:example:company:txn:value = post-data:/"accountBalances"/"savings"
```

Then, you can set the data type of `urn:example:company:txn:value` to `double` by using the following stanza and entry:

```
[user-attribute-definitions]
urn:example:company:txn:value.datatype = double
```

Example: Updating the category for form data

If you defined a custom attribute in the `[azn-decision-info]` stanza as:

```
urn:example:company:txn:userid = post-data:userid
```

Then, you can set the category of `urn:example:company:txn:userid` to `Subject` by using the following stanza and entry:

```
[user-attribute-definitions]
urn:example:company:txn:userid.category = Subject
```

Setting the data type or category of a custom attribute

Set the data type or category of a custom attribute that is being passed to the runtime security services. Setting the data type or category ensures that the runtime security services use the data accurately when evaluating the policy.

About this task

The default data type of an attribute is string. The default category of an attribute is Environment. Use the following procedure to change the type and the category.

Procedure

1. Open the WebSEAL configuration file.
2. Create one or more entries in the `[azn-decision-info]` stanza for custom attributes, if they do not already exist.
3. Set the data type or category of a custom attribute by using the `[user-attribute-definitions]` stanza and entry. Add the stanza if it does not already exist.

   data type

   Define the data type by using the following syntax:

   ```
   [user-attribute-definitions]
   attr_ID.datatype = data_type
   ```

   category

   Define the category by using the following syntax:

   ```
   [user-attribute-definitions]
   attr_ID.category = category_name
   ```

   The value you use for *`attr_ID`* must match the attribute identifier you defined in the `[azn-decision-info]` stanza.

   See "[user-attribute-definitions] stanza" on page 41 for the list of values you can use for data type and category.
4. Save the file.
5. Restart the WebSEAL server for the changes to take effect.

Related reference

"[user-attribute-definitions] stanza" on page 41
Use the `[user-attribute-definitions]` stanza to modify the data type, the category, or both of a custom attribute.

"[azn-decision-info] stanza" on page 38
Add extra information from the HTTP request to the authorization decision information.

Chapter 4. Attribute collection service

The attribute collection service is a Representational State Transfer (REST) service. It can collect web browser and location information from the user for calculating the risk score.

Process overview

The following process describes the attribute collection service and how to use it:

1. Make REST calls to store and delete attributes in the database. The initial request to the service receives a correlation ID. The correlation ID is used to make further REST calls.
2. Use JavaScript to collect the web browser attributes. You can place the **HTML** page that calls the JavaScript functions on any server.
   - Ajax collects information in the background. It does not slow down page loading.
   - You can make standard Ajax requests only to the same domain. With Cross Origin Resource Sharing (CORS), you can make Ajax requests across domains.
   - The CORS response header contains the settings for the following specifications:
     - The server from which requests are accepted.
     - The types of requests that are accepted.

Attributes that are configured as session attributes are collected automatically by the `info.js` file for risk score calculation.

Request types

GET and POST requests create a correlation ID to identify the session in the database. A correlation ID is a UUID that is stored in a cookie. The attribute collection service process uses the following request types:

GET

Retrieves information about an attribute session from the database. GET requests are disabled by default. Requests use a URL with a REST path, such as: `https://`*`webseal`*`/mga/sps/ac/rest/`*`UUID`*.

POST

Creates an attribute session in the database. POST requests use a URL such as `https://`*`webseal`*`/mga/sps/ac/`*`UUID`*.

The session attributes are sent as a JSON string with the request. In a response, the server sets a cookie that contains the correlation ID. For example, the POST `/sps/ac/9d37e806-24cf-4398-a3b9-d7f13fb2231f` request creates a session in the database with a UUID of `9d37e806-24cf-4398-a3b9-d7f13fb2231f`.

You can also configure the risk-based access properties to use an existing cookie

DELETE

Deletes an attribute session from the database.

Risk-based access runtime properties

Use the local management interface to configure the risk-based access properties that are required for attribute collection service.

The following properties specify information about the attribute collection service:

`attributeCollection.cookieName`

Correlation ID used by the attribute collector.

Data type: String

Example:

```
ac.uuid
```

attributeCollection.requestServer

Request server for attribute collector. A list of the allowable hosts where the ajaxRequest can be sent from.

Data type: String List

Example:

```
https://rbademo.example.com,https://rbaemo2.example.com
```

attributeCollection.serviceLocation

Location of the attribute collector.

Data type: String List

Example:

```
http://rbademo.example.com/mga
```

attributeCollection.sessionTimeout

Number of seconds in which sessions stored in context-based access will automatically expire, unless updated. If any attribute in the session is updated, the session expiry is extended by the specified number of seconds configured in this property. The default is `3600` seconds.

Data type: Integer

Example:

```
3600 seconds
```

attributeCollection.enableGetAttributes

Enables the REST GET method to return attributes.

Data type: Boolean

Example:

```
False
```

attributeCollection.getAttributesAllowedClients

A comma-separated list of clients that are allowed to access the ACS REST GET method.

If this property is not set and `attributeCollection.enableGetAttributes` is set to `true`, anyone can access the GET method. If this property is set but `attributeCollection.enableGetAttributes` is set to `false`, this property is ignored.

Data type: String List

Example:

```
hostname1, hostname2
```

attributeCollection.hashAlgorithm

The algorithm that is used to create the hash.

Data type: String

Example:

```
SHA256
```

attributeCollection.attributesHashEnabled

A comma-separated list of attribute URI values that have been configured for hashing.

Data type: String List

Example:

```
urn:ibm:security:environment:http:userAgent,
urn:ibm:security:environment:deviceFonts,
urn:ibm:security:environment:browserPlugins
```

attributeCollection.authenticationContextAttributes

Comma-separated lists of attribute names to be collected when performing an authentication service obligation.

Data type: String List

Example:

```
authenticationLevel, http:host
```

JavaScript functions

Use the JavaScript functions in the `C/ac/info.js` file to make requests to the server. Include the `info.js` JavaScript file in the HTML landing page of your application. When `info.js` is loaded, it calls the following functions:

sendSession()

Makes a POST request to the delegate service.

The `sendSession()` function collects the web browser attributes and sends them to the server. They are stored in the database. Call this function when a user logs in.

deleteSession()

Makes a DELETE request for a specified correlation ID.

The POST request from the `sendSession()` returns a correlation ID. Based on the correlation ID, the `deleteSession()` function deletes the attributes from the database. Call this function when the user logs out or when the current session times out.

getLocation()

Detects the location of the device from which the requests are made. If the location information is sent to the server, call the `getLocation()` function before the `sendSession()` function. The following web browsers support the detection of location: Mozilla Firefox, Google Chrome, Opera, Apple Safari, and Microsoft Internet Explorer 9 and 10.

Note: For the JavaScript functions to work in Microsoft Internet Explorer, include the following statement in the **HTML** page from which you call the function. The following statement forces Microsoft Internet Explorer to use the standards mode:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
     "http://www.w3.org/TR/html4/loose.dtd">
```

For configuration steps and examples, see "Configuring the attribute collection service" on page 47.

Configuring the attribute collection service

Before you can collect risk calculation information, you must specify the server and location of the collection service. You also must specify a JavaScript file to collect the session attributes.

Before you begin

Run the **isamcfg** tool to configure the runtime security services EAS. See Using the isamcfg tool.

Review Chapter 4, "Attribute collection service," on page 45.

Procedure

1. Optional: Configure the context-based access properties that you require for the attribute collection service. These properties are set by the **isamcfg** tool, but if you need to change them, use the following instructions:
   a) Configure `attributeCollection.requestServer` to specify the server from which requests are received using the local management interface:
      i) Select **AAC** > **Global Settings** > **Advanced Configuration**
      ii) Find the `attributeCollection.requestServer` key in the list and click the edit icon. A new window displays the name and the current value.
      iii) Edit the value of the request servers. The value is a space-separated list of WebSEAL host names from which requests are permitted. Host names musts begin with `http://` or `https://`. For example, type `http://mywebsealhost.company.com`.
   b) Configure `attributeCollection.serviceLocation` to specify the location using the local management interface:
      i) Select **AAC** > **Global Settings** > **Advanced Configuration**
      ii) Find the `attributeCollection.serviceLocation` key in the list and click the edit icon. A new window displays the name and the current value.
      iii) Edit the value of the location. Specify the location as:

      ```
      https://host_name/webseal-junction-name
      ```

      For example, type `https://mywebsealhost.company.com/mga`.

   You can configure any other attribute collection service attributes in a similar fashion, as necessary.
2. Add the URL of `info.js` to the <head> block in the HTML landing page of your application. The `info.js` file calls functions that are required to collect session attributes. Follow this format:

   ```
   <script src="https://host_name/webseal-junction-name/sps/ac/js/info.js"></script>
   ```

   Note: When the `info.js` file is included on an HTML page, attribute collection by Ajax calls can take time to complete. To avoid issues, attribute collection must end before moving away from the page. For example, if the attribute collection is still running, and a link is clicked, the policy fails to resolve session attributes. To prevent this issue, modify the JavaScript file to prevent the user from continuing until after the Ajax call completes.

Results

The basic configuration of the attribute collection service for context-based access is complete.

Configuring the REST service to GET session and behavior attributes

The client can use REST services in the attribute collection service to GET session and behavior attributes.

About this task

An administrator must enable the `attributeCollection.enableGetAttributes` property before using the REST services capability. If you do not enable this property, the following error message is returned when the attribute collection service or the client attempts to GET behavior or session attributes:

```
FBTRBA079E The attribute collection service GET method is not enabled.
```

The GET method is not enabled by default. To enable specific clients to GET attributes from the attribute collection service, you must add their corresponding host names to the `attributeCollection.getAttributesAllowedClients` property. If the administrator does not set this property, any client can GET attributes from the attribute collection service.

Procedure

1. Log in to the local management interface.
2. Click **AAC**.
3. Under **Global Settings**, click **Advanced Configuration**.
4. Enable the `attributeCollection.enableGetAttributes` property.

Viewing the JSON for behavior and session attributes

The administrator can view the JSON for behavior and session attributes for diagnostic purposes.

About this task

You can view the JSON for registered behavior or session attributes.

Procedure

1. Open a browser.
2. View the information for your registered attributes by entering one of the following URLs:

Attribute type	URL	Example information

Behavior	`https://`*hostname*`/mga/sps/ac/rest/behavior`	`[`
		`  {`
		`    "time":1399431163540,`
		`    "attributes":[`
		
		`{"urn:ibm:security:environment:geoLocation":"30.402806599999998,-97.714635,45"},`
		
		`{"urn:ibm:security:environment:accessTime":"2014-05-07T02:52:43Z"}`
		`     ]`
		`  },`
		`  {`
		`    "time":1399431202147,`
		`    "attributes":[`
		
		`{"urn:ibm:security:environment:geoLocation":"30.4028181,-97.7146326,44"},`
		
		`{"urn:ibm:security:environment:accessTime":"2014-05-07T02:53:22Z"}`
		`    ]`
		`  }`
		`]`

Session	`https://`*hostname*`/mga/sps/ac/rest/session`	`{`
		
		`"urn:ibm:security:subject:riskScore":"0",`
		
		`"urn:ibm:security:environment:devicePlatform":"Linuxx86_64",`
		
		`"urn:ibm:security:environment:screenAvailableWidth":"1920",`
		
		`"urn:ibm:security:environment:browserPlugins":"Shockwave Flash",`
		
		`"urn:ibm:security:environme`

Attribute type	URL	Example information
		nt:geoRegionCode":"", "urn:ibm:security:environme nt:screenHeight":"1080", "urn:ibm:security:environme nt:deviceLanguage":"en-US", "urn:ibm:security:environme nt:colorDepth":"24", "urn:ibm:security:environme nt:screenWidth":"1920", "urn:ibm:security:environme nt:geoLocation":"30.4029283 ,-97.7147197,49", "urn:ibm:security:environme nt:screenAvailableHeight":" 1025" }

Chapter 5. Attribute matchers

An attribute matcher compares the values of a specified attribute in the incoming device fingerprint with the existing device fingerprint of the user. Context-based access uses the information that is returned by the attribute matchers to calculate the risk score.

In some scenarios, multiple attributes or a set of composite attributes must be matched. For example, longitude, latitude, and accuracy are three attributes that are related to location. In a given scenario, two device fingerprints are considered a match if the distance between two location points is not greater than a specified threshold value. In this scenario, the comparison of only the longitude attribute does not provide accurate results. The matcher must do a more complex comparison or composite matching, where it matches multiple attributes from both fingerprints.

The matcher returns one of the following results after it compares the attributes values in the registered device fingerprint and the incoming device fingerprint:

`Matched`
: The decision that the matcher returns if the attribute value in the registered device fingerprint and the incoming device fingerprint value are the same or considered equivalent.

`Mismatched`
: The decision that the matcher returns if the attribute value in the registered device fingerprint and the incoming device fingerprint value are not the same or considered equivalent.

`Indeterminate`
: The decision that the matcher returns if it cannot gather enough attribute information to determine a result.

  Note: When the matcher returns `Indeterminate` as the result, the risk engine does not use the attribute in risk score calculations.

A mismatch increases the risk score that is based on the assigned weight of the attributes.

The matcher might not be used in the risk calculation in the following situations:

- The incoming device fingerprint does not contain the required attributes.
- The historical data is not available for a matcher to make a match or mismatch decision.

Risk-based access provides ready-to-use attribute matchers that compare composite attributes or analyze a range of attribute values. You can configure one or more of the attribute matchers that are described in the following sections.

Exact match matcher

The **exact_match** matcher checks whether the values of an attribute in a registered device and an incoming request exactly equal each other. Use this matcher if the more specialized matchers are not appropriate for the attribute.

IP address matcher

The IP address matcher (**ipaddr_matcher**) compares the IP address of a request with:

- A trusted list (inclusion list) of IP addresses
- An untrusted list (exclusion list) of IP addresses
- The historical IP addresses of the device
- The IP reputation of the device

The IP address matcher has the following properties:

Trusted addresses

IPV4 addresses

IP and Netmask: Specifies the IP address and its netmask to include. Include `X.X.X.X` as a value to compare the incoming IP address with the IP address with which the device is registered.

IPV6 addresses

IP and Prefix: Specifies the IP address and its prefix to include. Include `X:X:X:X:X:X:X:X` as a value to compare the incoming IP address with the IP address with which the device is registered.

Untrusted addresses

IPV4 addresses

IP and Netmask: Specifies the IP address and its netmask to exclude. Include `X.X.X.X` as a value to compare the incoming IP address with the IP address with which the device is registered.

IPV6 addresses

IP and Prefix: Specifies the IP address and its prefix to exclude. Include `X:X:X:X:X:X:X:X` as a value to compare the incoming IP address with the IP address with which the device is registered.

The IP address matcher returns one of the following decisions after it compares the incoming IP address with the IP address that belongs to the registered device:

`MISMATCHED`

The decision that the matcher returns if either of the following conditions are true:

- The incoming IP address is in the list of untrusted IP addresses.
- The incoming IP address is not in the list of trusted IP addresses, and the IP address has a reputation other than `Dynamic IPs`.

`MATCHED`

The decision that the matcher returns if the matcher finds the incoming IP address in the list of trusted IP addresses.

`INDETERMINATE`

The decision that the matcher returns if the following conditions are true:

- The IP address is not in the list of untrusted IP addresses.
- The IP address is not in the list of trusted IP addresses.
- The IP address qualifies for one of the following conditions:
  - Does not have a reputation.
  - Has a `Dynamic IPs` reputation.

PIP matcher

The policy information point (PIP) matcher (**pip_matcher**) uses the value of a single-valued attribute to determine one of the following results:

`Matched`

The value of the attribute is `MATCHED`.

`Mismatched`

The value of the attribute is `MISMATCHED`.

`Indeterminate`

The value of the attribute is `INDETERMINATE`.

Note: The PIP matcher supports only single-valued attributes with String data types.

Write and configure a JavaScript PIP with the following capabilities if you prefer to use the PIP matcher:

- The PIP determines attribute values.
- The PIP compares attribute values.
- The PIP returns match decisions that are based on the values of attributes that it compares.

Location matcher

The location matcher (**location_matcher**) checks whether the location of a device is within a specific distance from the previous known locations of the device. Configure the location matcher properties to specify the accuracy range and how to compare the location information.

Limitation: The retrieval of location attributes depends on the web browser and the settings that the user specifies in the web browser. The web browser must support the Geolocation API. An error might occur in some web browsers if a user tries to access a protected resource from a device with a wired internet connection.

The location-based analysis processes all three location attributes (longitude, latitude, and accuracy) collectively when it determines the match for the location. Though weights are assigned to all three attributes, the weight for only the longitude attribute is considered. The weights that are assigned to the supporting latitude and accuracy attributes are ignored.

The location matcher has two properties:

Comparison

Indicates how you want the attribute matcher to calculate the accuracy range of the location coordinates.

The following figure illustrates the closest points, midpoints, and farthest points of the accuracy ranges of two locations. In this figure, the circle represents the accuracy range and the center of the circle represents the location.

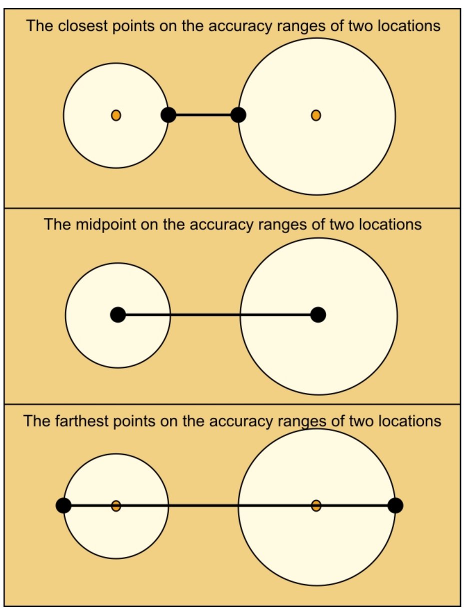

Figure 3. The closest points, midpoints, and farthest points on the accuracy ranges of two locations

Set the Comparison property to one of the following values:

- Specify the value as `closest` to calculate the distance between the closest points on the accuracy range of two locations. This calculation is the most restrictive calculation.
- Specify the value as `midpoint` to calculate the distance between the midpoints of the circles without considering accuracy.
- Specify the value as `farthest` to calculate the distance between the farthest points on the accuracy ranges of the two locations. This calculation is the least restrictive calculation.

Distance

The maximum distance between the new location and the historic locations. The unit of the numeric value is in kilometers. The default value is `40`.

Login time matcher

The login time matcher (**login_time_matcher**) compares and analyzes the historical login time data of the user with the current login time of the user. You must configure the attributes and properties that are required for login time analysis. The login time matcher primarily detects the logins per session. The first of the several access times that are captured within the session is considered the login time of the user. The result of the analysis determines the probability of a fraudulent user.

The login time matcher has one property:

Threshold
: Indicates the probability that a user might log in at a particular time. Valid values are `0` to `1`. The default value is `.3`. This default value indicates the probability that the user logs in approximately within an hour of the previous login times. If you set a lower value, the odds of the matcher returning true are higher and the risk score is lower. If you set a higher value, the odds of the matcher returning true are lower and risk score is higher. For example, if you set a value of 0.5, the matcher almost always returns false. The login time analysis collects data for eight login times before it provides input for risk score calculation.

Related concepts

IP reputation
The IP reputation policy information point (PIP) uses the IP reputation database to determine the reputation of the IP address of a request. Based on the IP address reputation, you can write a policy to grant or prohibit access to the requesting IP address. The IP reputation PIP is pre-configured with Advanced Access Control.

Related tasks

Modifying attribute matchers
Attribute matchers match incoming attributes to attributes in a device fingerprint. The predefined matchers are set to default values. You can modify those values to customize the risk calculations for your policies.

IP reputation

The IP reputation policy information point (PIP) uses the IP reputation database to determine the reputation of the IP address of a request. Based on the IP address reputation, you can write a policy to grant or prohibit access to the requesting IP address. The IP reputation PIP is pre-configured with Advanced Access Control.

Note: The IP reputation policy information point (PIP) capability is not available when the appliance is running in a Docker environment.

Possible IP reputations include the following classifications:

- Anonymous proxies
- Botnet Command and Control Server
- Dynamic IPs
- Malware
- Scanning IPs
- Spam

IP addresses that have reputations engage in certain activities that qualify them for reputations. The IP reputation database contains classification information about IP addresses. When the IP reputation PIP requests information, the database returns a score for each classification that the IP address might have. The score ranges from 0 - 100. As the value increases, the likelihood that the IP address has a reputation corresponding with that score increases.

The score for each classification is compared to a threshold score that you can configure. You can also use the default score of 50. The IP address has a reputation if the score that belongs to any number of classifications is greater than or equal to the threshold score. For example, if an IP address is suspected to be a spammer, the database may return a value of 95 for the spam classification. If this value is greater than or equal to the threshold score you choose to use, the IP reputation PIP returns a Spam classification for the requesting IP address.

You can write a policy to either grant access to or prohibit access from IP addresses with specified classifications. To use the IP reputation PIP, use the `ipReputation` attribute in a policy.

IP reputation can also be used to influence an access decision when the IP matcher is used.

Related tasks

Modifying attribute matchers
Attribute matchers match incoming attributes to attributes in a device fingerprint. The predefined matchers are set to default values. You can modify those values to customize the risk calculations for your policies.

Attribute matchers
An attribute matcher compares the values of a specified attribute in the incoming device fingerprint with the existing device fingerprint of the user. Context-based access uses the information that is returned by the attribute matchers to calculate the risk score.

Modifying attribute matchers

Attribute matchers match incoming attributes to attributes in a device fingerprint. The predefined matchers are set to default values. You can modify those values to customize the risk calculations for your policies.

About this task

Each predefined matcher uses specific properties.

exact_match
The **exact_match** matcher checks whether the values of an attribute in a registered device and an incoming request exactly equal each other. Use this matcher if the more specialized matchers are not appropriate for the attribute. This matcher cannot be modified.

location_matcher
The location matcher checks whether the location of a device is within a specific distance from the previous known locations of a device.

Comparison
Indicates how you want the attribute matcher to calculate the accuracy range of the location coordinates.

Distance
Specifies the maximum distance between the new location and the historic locations. The value is in kilometers. The default value is `40`.

login_time_matcher
The login matcher compares and analyzes the historical login time data for the user with the current login time of the user.

Threshold
Indicates the probability that a user might log in at a particular time. Valid values are `0` to `1`. The default value is `0.3`. This default value indicates the probability that the user logs in approximately within an hour of the previous login times. If you set a lower value, the odds of a return value of true are higher and the risk score is lower. If you set a higher value, the odds of a return value of true are lower and risk score is higher. For example, if you set a value of `0.5`, the matcher almost always returns false. The login time analysis collects data for eight login times before it provides input for risk score calculation.

ipaddr_matcher
The IP address matcher compares an inclusion list (trusted) or exclusion list (not trusted) of IP addresses with the historical IP addresses of the device.

Trusted addresses

IPV4 addresses

IP and Netmask: Specifies the IP address and its netmask to include. Include `X.X.X.X` as a value to compare the incoming IP address with the IP address with which the device is registered.

IPV6 addresses

IP and Prefix: Specifies the IP address and its prefix to include. Include `X:X:X:X:X:X:X:X` as a value to compare the incoming IP address with the IP address with which the device is registered.

Untrusted addresses

IPV4 addresses

IP and Netmask: Specifies the IP address and its netmask to exclude. Include `X.X.X.X` as a value to compare the incoming IP address with the IP address with which the device is registered.

IPV6 addresses

IP and Prefix: Specifies the IP address and its prefix to exclude. Include `X:X:X:X:X:X:X:X` as a value to compare the incoming IP address with the IP address with which the device is registered.

Use the IP reputation database for classification of IP addresses

Select this box to check the requesting IP address against the addresses in the IP Reputation database. Addresses in the database are associated with one or more classifications. If the requesting address matches an address in the database, the database returns a score for each classification that is associated with the address.

The IP reputation threshold for classifications

The score that is compared to the classification score of an IP address. Select a score between `0` and `100` below the **Untrusted** tab in **IP Address Matcher Properties**. The default value is `50`.

Procedure

1. Log in to the local management interface.
2. Click **AAC**.
3. Under **Policy**, click **Attributes**.
4. Click **Matchers**.
5. Click the  icon for the matcher.
6. Change the properties.
7. Click **Save**.
8. When you modify an attribute matcher, a message indicates that there are changes to deploy. If you are finished with the changes, deploy them.

   For more information, see Chapter 14, “Deploying pending changes,” on page 157.

Results

The modified attribute matcher is saved.

Related concepts

Attribute matchers

An attribute matcher compares the values of a specified attribute in the incoming device fingerprint with the existing device fingerprint of the user. Context-based access uses the information that is returned by the attribute matchers to calculate the risk score.

IP reputation

The IP reputation policy information point (PIP) uses the IP reputation database to determine the reputation of the IP address of a request. Based on the IP address reputation, you can write a policy to grant or prohibit access to the requesting IP address. The IP reputation PIP is pre-configured with Advanced Access Control.

Chapter 6. Obligations

Obligations are used in policies to inform the enforcement point that more actions are required before access is granted or denied to a protected resource.

Obligations are either of the following types:

- Actions that require the user to perform an operation.
- Actions that occur on the server without user involvement.

Predefined obligations are available by default. See "Predefined obligations" on page 61.

You can create, modify, or delete obligations.

When you create a policy, you can add an obligation. If the obligation has parameters, define the parameter details when you create the policy.

You can use the same obligation for different policies. Parameters customize the obligation for each policy. Parameter details are defined in the policy, not the obligation.

Obligations are saved in the local configuration database.

You can also map an obligation to a URL by defining it in the WebSEAL configuration file.

Managing obligations

Obligations are used for authoring policies. You can view, add, modify, and delete obligations.

About this task

The obligation name must begin with an alphabetic character. Do not use control characters, leading and trailing blanks, and the following special characters ~ ! @ # $ % ^ & * ( ) + | ` = \ ; : " ' < > ? , [ ] { } / anywhere in the name.

Procedure

1. Log in to the local management interface.
2. Click **AAC**.
3. Under **Policy**, click **Obligations**.
4. Click **Obligations**.
5. Perform any of the following actions:

   Add an obligation

   a. Determine whether you want to use an imported obligation extension or not.

      - If you want to use an imported obligation extension, click and select its associated obligation type.
      - If you do not want to use an imported obligation extension, click and select **Enforcement Point**.

   b. Complete the fields in the **General** tab.

   c. Click the **Properties** tab.

      i) Select a property that you want to configure.

      ii) Click .

      iii) Enter the value for that property.

      iv) Click **OK**.

d. Click **Save**.

Modify a custom obligation

Note:

- You cannot modify the identifier for obligations that are in use by a policy.
- You cannot change the extension that is used in the custom obligation. Only extension properties can be changed.

a. Select the obligation that you want to modify.

b. Click .

c. To add a parameter, click , specify the settings. Click **OK**.

d. To delete a parameter, highlight it, and click .

e. Modify the obligation properties.

i) Select a property that you want to configure.

ii) Click .

iii) Enter the value for that property.

iv) Click **OK**.

Note: You must enter the value for all required properties.

f. Click **Save**.

Delete a custom obligation

Note: You cannot delete predefined obligations or obligations that are in use by a policy.

a. Select an obligation from the list. To select multiple obligations, press and hold the Ctrl key and select several obligations

b. Click . A message prompts you to confirm the deletion.

c. Click **Delete**.

6. When you add, modify or delete an obligation, a message indicates that there are changes to deploy. If you are finished with the changes, deploy them.

   For more information, see Chapter 14, “Deploying pending changes,” on page 157.

What to do next

You can use any obligation in your policy. See Chapter 9, “Access control policies,” on page 99 for instructions on how to specify an obligation for use in your policy.

Related reference

“Predefined obligations” on page 61
Predefined obligations are provided for your use in policy authoring.

“Obligation properties” on page 60
When you add or modify an obligation, you specify properties that make that attribute unique.

Obligation properties

When you add or modify an obligation, you specify properties that make that attribute unique.

Specify the following properties when you add or modify an obligation:

Name
Specify a unique name for the obligation.

Description
Enter a description of the obligation. (Optional)

Identifier
Specify an obligation ID that is used within the XACML policy file to identify the obligation at run time. The identifier must be a known identifier that can be handled by the target enforcement point. This value is required. For example, `urn:ibm:security:obligation:myCustomObligation`.

Type
This field is read only. The type is **Obligation**.

Parameters

DataType
Select a data type for the parameter. Valid options include the following options:

Boolean
Represents the XML Boolean data type, `http://www.w3.org/2001/XMLSchema#boolean`.

Date
Represents the XML date data type, `http://www.w3.org/2001/XMLSchema#date`.

Double
Represents the XML double data type, `http://www.w3.org/2001/XMLSchema#double`.

Integer
Represents the XML integer data type, `http://www.w3.org/2001/XMLSchema#integer`.

String
Specifies the XML string data type, `http://www.w3.org/2001/XMLSchema#string`.

Time
Represents the XML time data type, `http://www.w3.org/2001/XMLSchema#time`.

X500Name
Represents the XACML X500 name data type, `urn:oasis:names:tc:xacml:1.0:data-type:x500Name`.

Related tasks
"Managing obligations" on page 59
Obligations are used for authoring policies. You can view, add, modify, and delete obligations.

Predefined obligations

Predefined obligations are provided for your use in policy authoring.

An obligation consists of an action that must occur to allow or deny access to a resource.

Register Device
The system registers the device that is used by the user to access the resource. No user interaction is required for this obligation.

The administrator can configure the following properties to affect device fingerprint registration:

`deviceRegistration.allowIncompleteFingerprints`
Specifies whether the risk engine registers devices with incomplete device fingerprints.

An *incomplete device fingerprint* is a device fingerprint that does not include all of the attributes that the administrator specifies in the risk profile.

`deviceRegistration.permitOnIncompleteFingerprint`
Specifies whether the risk engine grants access to request devices with incomplete device fingerprints.

If the administrator does not set the configuration properties, then the properties default to **false**. When the properties are set to **false**:

- The risk engine cannot register devices that have incomplete device fingerprints.
- The risk engine denies request devices with incomplete device fingerprints.

Mapping obligations to a URL

You can define the mapping between the obligation that the policy decision point (PDP) returns and the URL that attempts to satisfy the obligation.

Procedure

1. Open the WebSEAL configuration file.
2. Add entries to the `[obligations-urls-mapping]` stanza. These entries define the mapping between an obligation and the URL that attempts to satisfy that obligation. The following example contains a complete stanza and entry:

   ```
   [obligations-urls-mapping]
   obligation = URL
   ```

   Where:

   obligation
   : Defines the obligation string that is returned by runtime security services. This string, or *key*, is case-sensitive.

     You can also use wildcard obligations in this entry. Add an asterisk at the end of an obligation to indicate that all obligations found that match this entry, up to but not including the asterisk, are redirected to the URL value. Exact matches are searched for first. If no match is found, wildcard matches are used.

   URL
   : Defines the URL to which the user is redirected for authentication. The URL must point to an external authentication interface (EAI) application. See the WebSEAL documentation for information about the requirements for the EAI application.
3. Save the file.
4. Restart the WebSEAL server for the changes to take effect.

Results

When the runtime security services returns an obligation, the key is searched for in the configuration file in the following order:

1. `[obligations-urls-mapping]` entries
2. `[obligations-levels-mappings]` entries

The entries in the `[obligations-urls-mapping]` stanza must have unique keys as compared to the keys in the `[obligations-levels-mappings]` entries.

Example

The following entry specifies that an obligation named `auth1`. The value of `auth1` is a URL that is used to satisfy the obligation.

```
[obligations-urls-mapping]
auth1 = https://example.com
```

To redirect all obligations that start with `urn:example` to `http://www.example.com`, add the following entry:

```
urn:example:* = http://example.com
```

Suppose that you have the following entries in the **`[obligations-urls-mapping]`** stanza:

```
urn:example:sports = http://example.sports
urn:example:* = http://example
```

If runtime security services returns an obligation of `urn:example:sports`, the first entry is used to redirect the user to `http://example.sports`. In this case, both stanza entries apply to the obligation returned, but because there is an exact match, that obligation is used.

Chapter 7. Authentication policies

Authentication policies are workflows that dictate the authentication mechanisms to execute.

The access control policy that is attached to the resource can be used to determine the authentication policy with which the user must comply to access the resource. For example, the authentication policy can require the user to provide a one-time password value or authenticate with a user name and password whether or not an authenticated session exists.

Predefined authentication policies are available by default. See Predefined authentication policies.

You can create, modify, or delete authentication policies.

Attention: You cannot modify or delete predefined authentication policies.

Use an authentication policy as a permit condition of an access control policy.

Managing authentication policies

Authentication policies determine the order and conditions in which various authentication mechanisms are used to successfully authenticate a user. You can view, add, modify, and delete authentication policies.

About this task

Keep the following considerations in mind when you work with authentication policies:

- The authentication policy name must begin with an alphabetic character. Do not use control characters, leading and trailing blanks, and the following special characters ~ ! @ # $ % ^ & * ( ) + | ` = \ ; : " ' < > ? , [ ] { } / anywhere in the name.
- You cannot modify or delete predefined policies.

Procedure

1. Log in to the local management interface.
2. Click **AAC**.
3. Under **Policy**, click **Authentication**.
4. Perform any of the following actions:

   Add an authentication policy:

   Click . The Authentication Policy Editor opens. See "Creating an authentication policy" on page 66.

   Modify a custom authentication policy:

   a. Select the authentication policy that you want to modify.
   b. Click . The Authentication Policy Editor opens.
   c. Modify any of the authentication policy properties.
   d. Click **Save**.

   Delete a custom authentication policy:

   a. Select an authentication policy from the list. To select multiple authentication policies, press and hold the Ctrl key and select several authentication policies
   b. Click . A message prompts you to confirm the deletion.
   c. Click **Delete**.

What to do next

You can use any authentication policy in your access control policy. See "Creating an access control policy" on page 100.

Creating an authentication policy

Use the Authentication Policy Editor on the appliance local management interface to create and configure an authentication policy.

About this task

Each policy consists of one or more decisions and/or authentication mechanisms. A decision is a grouping of different branch paths that contain one or more authentication mechanisms. The specific branch path that is followed is dependent on the configured decision mapping rule. The mechanisms are modules that authenticate the user with a specific challenge or authentication technology, such as user name and password and one-time password. In the policy, the decisions and/or authentication mechanisms are grouped into a workflow. The workflow specifies the mechanism to use and the order in which each mechanism runs. The policy identifier (PolicyID) supplied as a parameter is used to initiate the authentication policy and can be supplied either with or without the standard prefix.

Procedure

1. Click .

   The Authentication Policy Editor opens.
2. Complete the **Name** and **Identifier** fields.
3. Provide a description in the **Description field**.
4. Click **Add Step** to add an authentication mechanism as a step in the policy workflow.
   a) Select an authentication mechanism. See Authentication.
   b) Click to review and select parameters that are supported by the mechanism. Not all authentication mechanisms support parameters. However, some configuration settings for authentication mechanisms can be customized with parameters on a per policy basis. If an authentication mechanism supports parameters, use the parameters settings to assign values to the parameters. See "Authentication policy parameters and credentials" on page 68.
   c) Click **OK**.
5. Click **Add Decision** to add a decision point to the policy workflow.
   a) Specify a unique name for the new decision.
   b) Select a mapping rule to be used by the new decision. Only mapping rules with the category "Decision" are valid.
   c) Optional: Specify a template file to be used by the new decision. If the template file is not specified, the default template file is used.
   d) Select whether or not to allow the workflow to return back to the decision point.
   e) Click **Add Branch** to add a new branch to the decision.
      i) Specify a unique name for the new branch.
      ii) Click **Add Step** to add an authentication mechanism as a step in the branch.

      Note: Click to review and select parameters that are supported by the mechanism. Not all authentication mechanisms support parameters. However, some configuration settings for authentication mechanisms can be customized with parameters on a per policy basis. If an authentication mechanism supports parameters, use the parameters settings to assign values to the parameters. See "Authentication policy parameters and credentials" on page 68.
   f) Continue with one of the following steps:
      i) Click **Add Branch** to add another branch to the decision.

ii) Click **Add Step** in the relevant branch to add another authentication mechanism as the first step in the branch.

iii) Click **Add Step** in a specific authentication mechanism to add a new authentication mechanism after the existing one.

iv) Click **Cancel** on a specific authentication mechanism to delete that authentication mechanism.

v) Click **Cancel** on a specific branch to delete that branch and its associated authentication mechanism steps.

vi) Click **OK** to finish defining the decision.

vii) Click **Cancel** to discard the decision and its branches.

6. Continue with one of the following steps:
   - Add another authentication mechanism to the workflow. Repeat the preceding steps.
   - After you add all authentication mechanisms, click if you want to customize the information that is included in the user credential. See "Authentication policy parameters and credentials" on page 68.
   - If you want to modify a specific workflow element in the list, use the toolbar associated with that element to perform the following tasks:

     a. Edit the element properties.
        - If the element is a decision, clicking **Edit** allows all decision elements to be edited.

     b. Add a new element directly after this element.
        - This is only available if the element is a top level authentication mechanism and allows a new top level authentication mechanism to be added.

     c. Delete an element.
        - If the element is a decision it also deletes all branches and branch steps.
        - If the element is a top level authentication mechanism it deletes only that element.

     d. Move element up.

        Note: The move is only within the level of the selected element.
        - Decisions and top level authentication mechanism steps can be moved with each other.
        - Branch steps can be moved with each other inside the same branch.

7. Click **OK**.

   Note: When you are viewing or editing an authentication policy, there are two panes shown in the user interface:
   - The first pane is the tree view of the workflow. This is where the policy elements can be managed.
   - The second pane is a flowchart view of the workflow. This flowchart gives a visual representation of the steps in the workflow. This pane has a button that allows the flowchart pane to be changed. The following options are available:
     - **Size**- Use the slider to set the size of the flowchart elements ranging from Small to Large.
     - **Orientation**- Use the radio buttons to set the flowchart to render either vertically or horizontally.
     - **Layout**- Use the radio buttons to set the flowchart pane to render beside or beneath the workflow pane.
     - **Ratio**- Use the slider to change the size of the workflow and flowchart panes relative to each other.

   The flowchart settings are restored by using cookies so that any subsequent uses of this page retains the same settings.

What to do next

Use this authentication policy as the Permit with authentication action in an access control policy. See "Creating an access control policy" on page 100.

Authentication policy parameters and credentials

When you add or modify an authentication policy, you specify parameters for the authentication mechanism and the attributes that you want in the credential. The credentials are evaluated as part of the access control decision.

Note: You cannot modify predefined authentication policies.

Parameters

Parameters pass policy configuration to the authentication mechanism. Parameters can be set for each workflow step. Parameter values can be a literal string that you provide in the parameter settings or they can be a context attribute reference. A context attribute consists of an attribute source, attribute namespace, and attribute ID. See Table 7 on page 82 for a list of context attributes that you can use.

Table 6. Authentication mechanism runtime parameters

Authentication Mechanism	Parameter Name	Default Value	Description
Username Password	`reauthenticate`	`true`	An authentication value that indicates whether the user must authenticate even if the user previously authenticated.
One-Time Password	`reauthenticate`	`true`	An authentication value that indicates whether the user must authenticate even if the user previously authenticated.
One-Time Password	`username`	No default value	The user name for OTP authentication. If the **Pass** check box is not checked, the OTP authentication mechanism retrieves the user name from the current authentication service credential.
HOTP One-Time Password	`reauthenticate`	`true`	An authentication value that indicates whether the user must authenticate even if the user previously authenticated.
HOTP One-Time Password	`username`	No default value	The user name in HOTP authentication. If the **Pass** check box is not checked, the HOTP authentication mechanism retrieves the user name from the current authentication service credential.

Authentication Mechanism	Parameter Name	Default Value	Description
HOTP One-Time Password	`secretKey`	No default value	The secret key in HOTP authentication. If the **Pass** check box is not checked, the HOTP authentication mechanism retrieves the secret key of the user from its internal database. Users can configure their own secret key on the **OTP Secret Keys management** page. See "Managing OTP secret keys" on page 210
TOTP One-Time Password	`reauthenticate`	`true`	An authentication value that indicates whether the user must authenticate even if the user previously authenticated.
TOTP One-Time Password	`username`	No default value	The user name in TOTP authentication. If the **Pass** check box is not checked, TOTP authentication mechanism retrieves the user name from the current authentication service credential.
TOTP One-Time Password	`secretKey`	No default value	The secret key in TOTP authentication. If the **Pass** check box is not checked, the TOTP authentication mechanism retrieves the secret key of the user from its internal database. Users can configure their own secret key on the **OTP Secret Keys management** page. See "Managing OTP secret keys" on page 210
MAC Email One-time Password **MAC One-time Password** **MAC SMS One-time password**	`mobileNumber`	No default value	The phone number that delivers the one-time password value.

Table 6. Authentication mechanism runtime parameters (continued)

Table 6. Authentication mechanism runtime parameters (continued)			
Authentication Mechanism	**Parameter Name**	**Default Value**	**Description**
MAC Email One-time Password **MAC One-time Password** **MAC SMS One-time password**	`emailAddress`	No default value	The email address that delivers the one-time password value.
MAC Email One-time Password **MAC One-time Password** **MAC SMS One-time password**	`reauthenticate`	`true`	An authentication value that indicates whether the user must authenticate even if the user previously authenticated. **Note:** If you create a policy that uses both the SMS and Email delivery types with reauthenticate set to `false`, only the first delivery type is executed.
MAC Email One-time Password **MAC One-time Password** **MAC SMS One-time password**	`username`	No default value	The user name in MAC OTP authentication. If the **Pass** check box is not checked, MAC OTP authentication mechanism retrieves the user name from the current authentication service credential.
MAC Email One-time Password **MAC One-time Password** **MAC SMS One-time password**	`deliveryType`	• `Email` • `SMS`	The type of delivery mechanisms to use for delivering the one-time password value. When specified, the MAC One-Time password bypasses the OTPMethods mapping rule. **Note:** If you create a policy and have both the SMS and Email delivery types defined and `reauthenticate` is set to `false`, only the first delivery type is executed.
RSA One-Time Password	`reauthenticate`	`true`	The authentication value that indicates whether the user must authenticate even if the user previously authenticated.

Table 6. Authentication mechanism runtime parameters (continued)			
Authentication Mechanism	**Parameter Name**	**Default Value**	**Description**
RSA One-Time Password	`username`	No default value	The user name in RSA authentication. If the **Pass** check box is not checked, RSA authentication mechanism retrieves the user name from the current authentication service credential.
HTTP Redirect Authentication	`redirectURL`	No default value	The URL that contacts the custom authentication implementation. The HTTP Redirect authentication mechanism redirects the user's browser to the specified URL.
HTTP Redirect Authentication	`reauthenticate`	`true`	An authentication value that indicates whether the user must authenticate even if the user previously authenticated.
HTTP Redirect Authentication	`returnCredAttrName`	No default value	The credential attribute name that determines whether the HTTP Redirect authentication is successful.
HTTP Redirect Authentication	`returnCredAttrValue`	No default value	The credential attribute value that is compared against to determine whether the HTTP Redirect authentication is successful.
End-User License Agreement	`alwaysShowLicense`	`False`	The prompt for the license file. Set this option to `true` to always prompt the user to accept the license file.

Table 6. Authentication mechanism runtime parameters (continued)			
Authentication Mechanism	**Parameter Name**	**Default Value**	**Description**
End-User License Agreement	`licenseRenewalTerm`	0	The number of days until the user must accept the license again. When you specify a value that is less than 1, there is not a renewal term. **Note:** This parameter compares the date that the user last accepted the license to the current date to determine the number of days since the user last accepted the license.
End-User License Agreement	`licenseFile`	No default value	The path to the license template file to display for the End-User License Agreement. For more information about how to update the license and add more license files, see Chapter 15, "Template files," on page 159 and "Template file macros" on page 193. **Note:** The path to the license file is relative to the locale in the template tree.
End-User License Agreement	`acceptIfLastAccepted Before`	No default value	The date that the license was last accepted. If the date the user last accepted the license is before this date, this parameter requires the user to accept the license again. Use the date format of `YYYY-MM-DD`.
End-User License Agreement	`username`	No default value	The user name of the user who is prompted to accept the license. If the **Pass** check box is not checked, the End-User License Agreement authentication mechanism retrieves the user name from the current authentication service credential.

Authentication Mechanism	Parameter Name	Default Value	Description
End-User License Agreement	`reauthenticate`	`True`	An authentication value that indicates whether the user must authenticate even if the user previously authenticated. **Note:** The mechanism displays the license once per authenticated session under the following conditions: • `alwaysShowLicense=true` • `reauthenticate=false`
Knowledge Questions	`questionPresentationMode`	`Group`	Use one of the following values: **Individual** Presents each question one at a time. **Group** Presents all questions to the user in the same form.
Knowledge Questions	`questionPresentationOrder`	`Random`	Use one of the following values: **Random** Presents the questions in random order. **Sequential** Presents the questions in the order in which they are stored.
Knowledge Questions	`amountOfCorrectAnswersRequired`	`1`	The number of correct answers that is required for successful authentication. Specify any positive integer value that is not higher than the number of questions that is stored for each user.

Table 6. Authentication mechanism runtime parameters (continued)

Table 6. Authentication mechanism runtime parameters (continued)			
Authentication Mechanism	**Parameter Name**	**Default Value**	**Description**
Knowledge Questions	`username`	No default value	The user name of the user who is prompted to answer the knowledge questions. If you do not specify the user name, the user must log in before the Knowledge Questions authentication mechanism starts. The value must be a string.
Knowledge Questions	`reauthenticate`	`True`	An authentication value that indicates whether the user must authenticate with the Knowledge Questions authentication mechanism even if the user previously authenticated. The value is Boolean.
Knowledge Questions	`maxGracePeriodAuthenticationCount`	`0`	The maximum number of user authentications during the grace period. The mechanism does not require the user to configure knowledge questions during the grace period. The value is any positive integer.
FIDO Universal 2nd Factor	`username`	No default value	The user name for the FIDO Universal 2nd Factor authentication. If the Pass check box is not checked, the FIDO Universal 2nd Factor authentication mechanism retrieves the user name from the current authentication service credential.
FIDO Universal 2nd Factor	`appId`	`https:/ / webseal.com`	The protocol, hostname, and port that the user will use to attempt authentication.

Table 6. Authentication mechanism runtime parameters (continued)			
Authentication Mechanism	**Parameter Name**	**Default Value**	**Description**
FIDO Universal 2nd Factor	`mode`	`Authenticate`	The mode the FIDO Universal 2nd Factor authentication mechanism operates in. Use one of the following values: **Authenticate** Performs FIDO U2F Authentication with already registered tokens. **Register** Performs FIDO U2F Registration to add tokens.
FIDO Universal 2nd Factor	`attestationType`	`None`	The type of certificate attestation validation to perform. Use one of the following values: **None** No certificate attestation validation is performed. **Keystore** Certificate attestation validation is performed using the keystore configured in `attestationSource`. **JWKS** Certificate attestation validation is performed using the JSON Web Key Set configured in `attestationSource`.
FIDO Universal 2nd Factor	`attestationSource`	No default value	The keystore or key set to use for certificate attestation validation. Either the name of the keystore on the appliance, or the URL for a JSON Web Key Set.

Authentication Mechanism	Parameter Name	Default Value	Description
FIDO Universal 2nd Factor	`attestationEnforcement`	`Required`	The level of enforcement of certificate attestation validation. Use one of the following values: **Required** Certificate attestation validation is required, and requests that fail validation will return a validation error. **Optional** Certificate attestation validation is performed, but requests that fail validation will not return an error.
FIDO2/WebAuthn Authentication	`username`	No default value	The user name for the FIDO2/WebAuthn authentication. If the Pass check box is not checked, the FIDO2/WebAuthn authentication mechanism retrieves the user name from the current authentication service credential.
FIDO2/WebAuthn Authentication	`relyingPartyConfigID`	No default value	The relying party configuration ID to use with this mechanism, which identifies the Relying Party and the relying party specific configuration to use. If the Pass check box is not checked, the relyingPartyConfigId set on the mechanism will be used instead.

Table 6. Authentication mechanism runtime parameters (continued)

Table 6. Authentication mechanism runtime parameters (continued)			
Authentication Mechanism	**Parameter Name**	**Default Value**	**Description**
FIDO2/WebAuthn Authentication	`userVerification`	No default value	Whether user verification is required, preferred, or discouraged. User verification is authenticator dependent, but could be a PIN code, password entry, biometric or other method. If the Pass check box is not checked, the user verification default that is configured against the relying party ID is used instead.
FIDO2/WebAuthn Authentication	`abortOnError`	No default value	Whether the policy should abort completely in an error case, or return a state ID such that a user can re-attempt authentication. If the Pass check box is not checked, the abortOnError set on the mechanism will be used instead.
FIDO2/WebAuthn Authentication	`timeout`	300	The timeout (in seconds) that is applied to the FIDO assertion request.
MMFA Authenticator	`contextMessage`	No default value	A message that is associated with a transaction, which can contain the detail of the transaction. This message may be displayed on the user's device when prompted for verification.
MMFA Authenticator	`pushMessage`	No default value. If not defined, the `contextMessage` value is used.	Defines a message that is sent as a push notification when a transaction is awaiting verification.

Table 6. Authentication mechanism runtime parameters (continued)			
Authentication Mechanism	**Parameter Name**	**Default Value**	**Description**
MMFA Authenticator	`signingAttributeList`	If not set, the value set for the property **`Signing Attributes`** in the MMFA Authenticator mechanism is used. See Configuring a Mobile Multi-Factor Authentication (MMFA) Authenticator Mechanism.	A comma separated list of context attributes that is added to a new JSON value attribute that gets passed as a new pending attribute to the target mobile device. If supported by the device, this JSON value is used to extract the various messages that is displayed to the end user. The MMFA server also uses this JSON value during signature validation. **Note:** The value that is set here overrides the **`Signing Attributes`** property set in the MMFA authenticator mechanism.
MMFA Authenticator	`username`	No default value	The name of the user for which the challenge is generated.
MMFA Authenticator	`reauthenticate`	`True`	An authentication value that indicates whether the user must authenticate even if the user is previously authenticated.
MMFA Authenticator	`policyURI`	No default value	The policy ID of the authentication policy that handles the challenge response from the Authenticator Client.

Table 6. Authentication mechanism runtime parameters (continued)

Authentication Mechanism	Parameter Name	Default Value	Description
MMFA Authenticator	`mode`	`Initiate`	The mode the MMFA Authenticator authentication mechanism operate in. Use one of the following values: **Initiate** Informs the mechanism that it is initiating the challenge to the Authenticator Client. **Response** Informs the mechanism that it is responsible for completing the MMFA process that was started in the **Initiate** policy.
FIDO2/WebAuthn Registration	`username`	No default value	The user name for the FIDO2/WebAuthn registration. If the Pass check box is not checked, the FIDO2/WebAuthn authentication mechanism retrieves the user name from the current authentication service credential.
FIDO2/WebAuthn Registration	`relyingPartyConfigID`	No default value	The relying party configuration ID to use with this mechanism, which identifies the Relying Party and the relying party specific configuration to use. If the Pass check box is not checked, the relyingPartyConfigId set on the mechanism will be used instead.

Table 6. Authentication mechanism runtime parameters (continued)			
Authentication Mechanism	**Parameter Name**	**Default Value**	**Description**
FIDO2/WebAuthn Registration	`templatePage`	No default value	The template page to be displayed to the end user as part of this mechanism. Allows for the page branding or user experience to be customized depending on the policy. If the Pass check box is not checked, the templatePage set on the mechanism will be used instead.
FIDO2/WebAuthn Registration	`optionsTemplate`	No default value	In the attestation flow an options request is usually sent from the browser to the server, and the response is passed into the navigator.credentials.create call. This registration mechanism will populate the options request from the Options JSON template file, instead of a request payload. Allows for the user experience to be customized depending on the policy. If the Pass check box is not checked, the optionsTemplate set on the mechanism will be used instead.
FIDO2/WebAuthn Registration	`abortOnError`	No default value	Whether the policy should abort completely in an error case, or return a state ID such that a user can re-attempt registration. If the Pass check box is not checked, the abortOnError set on the mechanism will be used instead.
FIDO2/WebAuthn Registration	`timeout`	300	The timeout (in seconds) that is applied to the FIDO attestation request.

Pass

A check in the **Pass** check box passes the parameter to the authenticator. The value for a passed parameter is either specified in the **Value** field or with the session or request information. If the **Pass** check box is not checked, the mechanism takes one of the following actions:

- Uses the default value.
- Uses the default method to get the default value.
- Reports an error, depending on the mechanism and the parameter.

Credentials

When the user completes the authentication process, the Authentication Service creates a credential for that user. It uses the credential to log in the user. The user credential contains information such as the name of the user, the groups that the user belongs to, and attributes that further describe the user. You might want to modify the information that is included in the credential depending on the information required in your policies.

The Authentication Service automatically includes the following attributes:

username
: The name of the user who is making the access request.

authenticationTypes
: A list of URIs of all authentication policies that the user completed.

authenticationMechanismTypes
: A list of URIs of all the authentication mechanisms that the user completed.

authenticationTransactionId
: An identifier of the latest authentication transaction that the user completed.

Use **Credentials** to restrict the attributes in the credential by explicitly including each attribute. These attributes can be:

- A literal string that you provide in the credential settings.
- A context attribute reference

A context attribute consists of an attribute source, attribute namespace, and attribute ID. See Table 7 on page 82 for a list of context attributes that you can use.

Credential attribute
: The name of an attribute to use as an authentication credential.

  - ASCII letters
  - ASCII digits
  - Period (.)
  - Underscore (_)
  - Hyphen (-)

  Note: Do not use any other special characters or non-ASCII Unicode characters.

Source
: The source specifies the provider of the value for the credential:

  - **Value**

    The value for the credential. Use any characters.
  - **Session**

    A context attribute with a lifetime throughout the authentication process.
  - **Request**

    A context attribute with a lifetime of the HTTP Request.

Value
: The value of the credential attribute. The value that you specify depends on the source you select in the previous field.

  - If you select **Value** as a source, type a literal value in this field.

- If you select **Session** or **Request**, type an attribute ID and namespace.

Context attributes

The following table lists of types of values you can retrieve from a session or a request.

Table 7. Context attributes

Type	Description	Attribute Source	Attribute Namespace	Attribute ID
Policy ID	The ID of the authentication policy in the current authentication process.	Session	`urn:ibm:security:asf:policy`	`policyID`
Transaction ID	The ID that triggers the current authentication process.	Session	`urn:ibm:security:asf:transaction`	`transactionID`
HTTP request body	The raw body of the current HTTP request	Request	`urn:ibm:security:asf:request`	`requestBody`
HTTP request method	The HTTP request method of the current HTTP request.	Request	`urn:ibm:security:asf:request`	`method`
HTTP request parameters	The HTTP request parameters of the current HTTP request.	Request	Each attribute can contain multiple values. **`urn:ibm:security:asf:request:parameter`** Retrieves the first value. **`urn:ibm:security:asf:request:parameters`** Retrieves all the values.	The name of the parameter.
HTTP request headers	The HTTP request headers of the current HTTP request.	Request	Each attribute can contain multiple values. You can retrieve the first value or all of the values: **`urn:ibm:security:asf:request:header`** Retrieves the first value. **`urn:ibm:security:asf:request:headers`** Retrieves all the values.	The name of the header.

Type	Description	Attribute Source	Attribute Namespace	Attribute ID
Request credential	The credential of the user in the current request.	Request	Each attribute can contain multiple values. You can retrieve the first value or all of the values: **urn:ibm:security:asf:request:token:attribute** Retrieves the first value. **urn:ibm:security:asf:request:token:attributes** Retrieves all the values.	The name of the Request credential attribute. Use `username` to retrieve the name of the user. Use `group` to retrieve the groups of the user.
Authentication Service credential	The credential of the user that the Authentication Service began constructing at the beginning of the authentication process.	Session	Each attribute can contain multiple values. You can retrieve the first value or all of the values: **urn:ibm:security:asf:response:token:attribute** Retrieves the first value. **urn:ibm:security:asf:response:token:attributes** Retrieves all the values.	The name of the Authentication Service credential attribute. Use `username` to retrieve the name of the user. User `group` to retrieve the groups of the user.
Context-based access attributes	The attributes that specify the context of the request that is evaluated as part of an access control decision.	Session	**Attention:** Before you can use context attributes, you must add the attributes to the `attributeCollection.authenticationContextAttributes` property in the Advanced Configuration settings. See Managing advanced configuration. Each attribute can contain multiple values. You can retrieve the first value or all of the values: **urn:ibm:security:asf:cba:attribute** Retrieves the first value. **urn:ibm:security:asf:cba:attributes** Retrieves all the values.	The name of the attribute.
Request attribute names	A list of attribute names that are present in the request token.	Request	`urn:ibm:security:asf:request`	`attributes`
Request header names	A list of header names in the incoming request.	Request	`urn:ibm:security:asf:request`	`headers`

Table 7. Context attributes (continued)

Table 7. Context attributes (continued)				
Type	**Description**	**Attribute Source**	**Attribute Namespace**	**Attribute ID**
Request parameter names	A list of parameter names in the incoming request.	Request	`urn:ibm:security:asf:request`	`parameters`
Response attribute names	A list of attribute names in the response token.	Session	`urn:ibm:security:asf:response`	`attributes`
Target URL	The target URL that was included when the current authentication process was triggered.	Request	`urn:ibm:security:asf:request`	`targetURL`

Predefined authentication policies

Authentication policies are workflows. They specify the authentication mechanisms that are required so the user can access a resource.

Each step in the workflow consists of an authentication mechanism. Each mechanism has requirements with which the user must comply to successfully authenticate. Most authentication policies require that the user present some credentials, but some requirements can be completed without any user action. The following table describes the predefined authentication policies:

Policy	The user authenticates
Consent Register Device	When prompted for consent to register a device. Optionally, the user can assign a name to the device to be registered.
Email Message	Email Message authentication policy.
Email One-Time Password	With a one-time password that is delivered by email. The one-time password value is generated and verified with the MAC one-time password.
End-User License Agreement	With the End-User License Agreement.
FAPI Cert Authentication	With FAPI Certificate based Client Authentication.
HOTP One-Time Password	With a counter-based, one-time password. No one-time password delivery is required. The one-time password value is verified with the HOTP one-time password provider.
HTTP Redirect	With the HTTP redirect authentication policy.
Knowledge Questions	With knowledge questions.
MAC One-Time Password	With a MAC one-time password. The user is prompted for a password delivery method.
MMFA Fingerprint Authentication	With a MMFA Transaction. This policy initiates a MMFA transaction.

Policy	The user authenticates
MMFA Fingerprint Authentication Response	With fingerprint authentication through a registered MMFA authenticator.
MMFA User Presence Authentication	With a MMFA Transaction. This policy initiates a MMFA transaction.
MMFA User Presence Authentication Response	With user presence authentication through a registered MMFA authenticator.
One-Time Password	With a one-time password. The user is prompted for the type of one-time password to use.
QR Code Login Initiate	With a QR Code scanned by an authenticator. This policy initiates a QR Code login.
QR Code Login Response	With login through a registered QR Code authenticator.
reCAPTCHA	With reCAPTCHA verification policy.
RSA SecurID	With a user name and password and RSA SecurID authentication.
SMS One-Time Password	With a one-time password that is delivered by SMS. The one-time password value is generated and verified with the MAC one-time password provider.
TOTP One-Time Password	With a time-based one-time password. No one-time password delivery is required. The one-time password value is verified with the TOTP one-time password provider.
Two-factor - Username Password and HOTP	With a user name and password and an HOTP one-time password.
Two-factor - Username Password and MAC	With a user name and password and a MAC one-time password.
Two-factor - Username Password and RSA	With a user name and password and an RSA one-time password.
Two-factor - Username Password and TOTP	With a user name and password and a TOTP one-time password.
Two-factor - Username Password and OTP	With a user name and password and a MAC one-time password. The user is prompted to select the type of one-time password to use.
Two-factor - Username Password and Email	With a user name and password and a MAC one-time password. The one-time password is delivered through email.
Two-factor - Username Password and SMS	With a user name and password and a MAC one-time password. The one-time password is delivered through SMS.
Two factor - Username Password and End-User License Agreement	With a user name and password and the End-User License Agreement.
Two-factor - Username Password and Knowledge Questions	With both of the following: • User name and password • Knowledge questions

Policy	The user authenticates
Username Password	With a user name and password.

Several predefined authentication policies do not end in authentication and are instead used for user self-care (USC) flows. The following table describes the USC predefined authentication policies:

Policy	Description
USC Account Create	New user enrollment policy that requires reCAPTCHA and email one-time password.
USC Lost ID	Lost ID policy, requiring reCAPTCHA.
USC Passkey Account Create	New user enrollment policy that requires reCAPTCHA, email one-time password, and prompts the user to enroll a passkey.
USC Password Reset	Password reset policy, that requires reCAPTCHA and email one-time password.

Managing authentication mechanisms

Authentication mechanisms determine conditions that successfully authenticate a user. You can view, add, modify, and delete authentication mechanisms.

Procedure

1. Log in to the local management interface.
2. Click **AAC**.
3. Under **Policy**, click **Authentication**.
4. Click **Mechanisms**.
5. Perform any of the following actions:

   Add an authentication mechanism:

   Note: The button is not enabled if no custom authentication mechanism is imported in the appliance.

   a. Click the drop-down button.
   b. Select your authentication mechanism type.
   c. Complete the fields in the **General** tab.
   d. Click the **Properties** tab.
      i) Select a property that you want to configure.
      ii) Click .
      iii) Enter the value for that property.
      iv) Click **OK**.
   e. Save and deploy the changes.

   Modify an authentication mechanism:

   a. Select the authentication mechanism that you want to modify.
   b. Click .
   c. Modify the authentication mechanism properties.
      i) Select a property that you want to configure.

ii) Click .

iii) Enter the value for that property.

iv) Click **OK**.

d. Save and deploy the changes.

Delete a custom authentication mechanism:

Note: The button is not enabled when you select a predefined authentication mechanism. You cannot delete predefined authentication mechanisms.

a. Select an authentication mechanism from the list. To select multiple authentication mechanisms, press and hold the Ctrl key and select several authentication mechanisms.

b. Click . A message prompts you to confirm the deletion.

c. Click **Delete**.

d. Save and deploy the changes.

What to do next

You can use your authentication mechanism in your authentication policy. See "Managing authentication policies" on page 65.

Chapter 8. Risk profiles

The risk engine uses the active risk profile to calculate risk scores for incoming requests. The administrator can create risk profiles or use predefined risk profiles that are provided with Advanced Access Control.

There are two types of risk profiles:

Predefined risk profiles
: Default risk profiles that are pre-configured on an appliance that has Advanced Access Control.

Each predefined risk profile is tailored to a specific scenario.

Risk profiles
: Custom risk profiles that the administrator creates.

Risk profiles include:

- Attributes that the administrator specifies. The risk engine collects all of the attributes that the administrator specifies in the risk profile from each incoming request device.
- Weight values for each attribute that the administrator specifies in the risk profile. The weight that an administrator gives to each attribute in the risk profile depends on the importance of each attribute. As the attributes increase in importance, their weight values also increase.

  Note: There is not a limit on the number of attributes that the administrator can include in a risk profile. However, the risk engine uses one risk profile at a time.

Related concepts

Risk score calculation
Risk score calculation is the process by which the risk engine determines a risk score. The *risk score* demonstrates the level of risk that is associated with permitting a request to access the resource. This risk score is compared to a *threshold score* that is set in a policy. A decision is made based on the result of this comparison.

Related reference

Predefined risk profiles
Predefined risk profiles are pre-configured on an appliance with Advanced Access Control. The risk engine uses the active risk profile to calculate risk scores for incoming requests.

Managing risk profiles

A risk profile is a collection of attributes with assigned weights. You can view a list of profiles or to add, delete, or clone profiles or set a profile to active. Only one risk profile can be active at a time.

About this task

Several risk profiles are predefined based on risk assessment for a collection of attributes. Predefined risk profiles are read only and you cannot modify their weights or add attributes to them. You can add custom risk profiles by creating your own or by cloning an existing profile.

By default, a risk profile named Default is set to active. The Default profile includes all the risk profile attributes with weights set to 0. With this profile active, if a user logs in with no devices registered, the risk score of that user is 100. If a user logs in with a device registered, the risk score of that user is 0. The Default profile is a sample profile. It is not intended to be used in a production environment. Before you use Security Verify Access, choose another risk profile or create your own.

Note: Use single-value attributes when you create a risk profile. Multivalue attributes are not supported in risk profiles.

There are two sections on the **Risk Profile** page:

Risk Profiles
The Risk Profiles section lists all the available risk profiles. An icon indicates which profile is active. Click a profile to select it.

Selected Profile Contents
The Selected Profile Contents section displays the Attribute and Weight for a selected risk profile. If you select a custom risk profile, you can modify the weight by typing a number or selecting a number.

Procedure

1. Log in to the local management interface.
2. Click **AAC**.
3. Under **Policy**, click **Risk Profiles**.
4. Perform one or more of the following actions:

   Rename a risk profile

   a. Right-click the profile name in the table and click **Rename** to change the name.

      Note: The risk profile name must begin with an alphabetic character. Do not use control characters, leading and trailing blanks, and the following special characters ~ ! @ # $ % ^ & * ( ) + | ` = \ ; : " ' < > ? , [ ] { } / anywhere in the name.

   b. Press Enter to apply the change.

   Modify a custom profile

   Note: You can modify only custom profiles.

   a. Select a custom profile.
   b. Modify the attributes and associated weights in the Selected Profile Contents section.

   Create a custom profile

   a. Click .
   b. Type a name for the profile. Click **Save**.
   c. In the Profile Contents section, click to add one or more attributes.
   d. In the **Add Profile Attributes** window, select an attribute to use. Click the **Attribute** column to sort the list in ascending or descending order. To filter the list of available attributes, type one or more characters in the **Filter** field. For example, if you type `current` in the **Filter** field, all attributes that start with `current` are shown in the attributes list. The attributes that match those characters are displayed.
   e. Select one or more attributes to add.
   f. Click **Add**.
   g. Click **Close** when you are done with the **Add Profile Attributes** window.
   h. Change the weights by typing or selecting a number.
   i. Click **Save** to apply the changes.
   j. If you want to make this new profile active, click Set Active.

   Create a copy of a profile and use it to make a custom profile

   a. Select the profile that you want to clone and give the profile a unique name.
   b. Click .
   c. Type a new name for the cloned profile.
   d. Click **Save**.
   e. Select the clone.
   f. In the Profile Contents section, select an attribute and take one of the following actions:

- Type a new value in the **Weight** field.
- Click to remove both the attribute and weight.
- Click to add one or more attributes. In the **Add Profile Attribute** window, select the attributes to use. To filter the list of available attributes, type one or more characters in the **Filter** field. The attributes that match those characters are displayed. For example, if you type `header` in the **Filter** field, all attributes that start with `header` are shown in the attributes list. Select the attribute that you want to use in the list and click **Add**. Click **Close** when you are done with the **Add Profile Attributes** window.

g. Click **Save** after each change to apply the change.

h. If you want to make this new profile active, click Set Active.

Delete a risk profile

a. Select a risk profile.

Note: You cannot remove predefined risk profiles.

b. Click .

c. Respond to the confirmation prompt.

The risk profile is removed and cannot be used.

Make a profile active

a. Select a profile.

b. Click Set Active.

Note: Only one profile can be active at a time.

5. When you add, modify or delete a risk profile, a message indicates that there are changes to deploy. If you are finished with the changes, deploy them.

For more information, see Chapter 14, "Deploying pending changes," on page 157.

Predefined risk profiles

Predefined risk profiles are pre-configured on an appliance with Advanced Access Control. The risk engine uses the active risk profile to calculate risk scores for incoming requests.

Predefined risk profile types

Predefined risk profiles:

- Are tailored for specific scenarios.
- Can be cloned.
- Cannot be modified.

To choose the most appropriate risk profile, you must determine your security priority. You can also create your own risk profile.

Depending on your environment, choose one of the following scenarios:

Upgrading to an IBM Security Verify Access appliance with Advanced Access Control

By default, a risk profile that is named **Default** is set to active. The **Default** profile includes all the risk profile attributes with weights set to 0. The risk score for this profile is always 0. The **Default** profile is a sample profile. It is not intended for a production environment. Before you use Security Verify Access, choose another risk profile or create your own.

Performing a new installation of an IBM Security Verify Access appliance with Advanced Access Control
By default, the **Browser** risk profile is set as the default risk profile. If the **Browser** risk profile does not suit the needs of your environment, you must choose another risk profile or create your own.

You can choose one of the following predefined risk profiles for the risk engine to use as a filter when it calculates the risk score:

Behavior
Determines a risk score by comparing the time of the current request with the time that the user usually tries to access the resource.

The following table contains the attributes and corresponding weight values that are included in the Behavior risk profile.

Attribute	Weight
`accessTime`	50
`browserPlugins`	10
`deviceFonts`	10
`http:userAgent`	10

Browser
Determines a risk score by comparing the attributes from the requesting browser with the browsers that the user is known to use.

The following table contains the attributes and corresponding weight values that are included in the Browser risk profile.

Attribute	Weight
`browserPlugins`	50
`deviceFonts`	50
`http:accept`	30
`http:acceptEncoding`	50
`http:acceptLanguage`	50
`http:userAgent`	50

Device
Determines a risk score by comparing the attributes from the requesting device with the devices that are associated with the user.

The following table contains the attributes and corresponding weight values that are included in the Device risk profile.

Attribute	Weight
`browserPlugins`	30
`colorDepth`	50
`deviceFonts`	50
`deviceLanguage`	50
`devicePlatform`	50
`screenAvailableHeight`	50

The following table contains the attributes and corresponding weight values that are included in the Device risk profile. *(continued)*	
Attribute	**Weight**
screenAvailableWidth	50
screenHeight	50
screenWidth	50

Location

Determines a risk score by comparing the location of the incoming request with the locations that the user is known to log in from.

The following table contains the attributes and corresponding weight values that are included in the Location risk profile.	
Attribute	**Weight**
geoLocation	50
geoCity	10
geoCountryCode	10
geoRegionCode	10

Usage scenarios

The following example usage scenarios demonstrate risk score calculation in predefined risk profiles.

Each scenario assumes that the administrator wrote a policy, which specifies that:

- Any risk score at or below 40 is permitted.
- Any risk score above 40 is denied.

Scenario 1: Behavior risk profile

The risk engine uses risk profile information in the following table to calculate the risk score.			
Attributes	**Weight values**	**Incoming device fingerprint values**	**Registered device fingerprint values**
accessTime	50	2013-05-07T03:25:13Z	2013-05-06T04:00:39Z, 2013-05-13T03:05:20Z,2013-05-20T03:15:22,2013-05-27T03:26:05Z, 2013-06-03T03:42:45Z
browserPlugins	10	Shockwave Flash,Chrome Remote Desktop Viewer,Widevine Content Decryption Module,Native Client,Chrome PDF Viewer,Java(TM) Plug-in 1.7.0,Citrix Receiver for Linux	Shockwave Flash,Chrome Remote Desktop Viewer,Native Client, Chrome PDF Viewer, Conference Plugin, AmazonMP3DownloaderPlugin, Google Update

The risk engine uses risk profile information in the following table to calculate the risk score. *(continued)*			
Attributes	**Weight values**	**Incoming device fingerprint values**	**Registered device fingerprint values**
deviceFonts	10	Andale Mono,Arial Black,Arial,Bitstream Charter,Century Schoolbook L,Comic Sans MS,Courier 10 Pitch,Courier New,DejaVu Sans Mono,DejaVu Sans,DejaVu Serif,Dingbats,Georgia, Impact,Khmer OS System,Khmer OS,Liberation Mono,Liberation Sans,Liberation Serif,Lohit Bengali,Lohit Gujarati,Lohit Punjabi,Lohit Tamil,Luxi Mono,Luxi Sans,Luxi Serif,Meera,Nimbus Mono L,Nimbus Roman No9 L,Nimbus Sans L,Standard Symbols L,Tahoma,Times New Roman,Trebuchet MS,URW Bookman L,URW Chancery L,URW Gothic L,URW Palladio L,UnBatang,UnDotum,Verdana, Waree,Webdings	Aharoni,Andalus,Angsana New, AngsanaUPC,Aparajita,Arabic Typesetting,Arial Black,Arial, Batang,BatangChe,Browallia New,BrowalliaUPC,Calibri, Cambria Math,Cambria,Candara, Comic Sans MS,Consolas, Constantia,Corbel,Cordia New,CordiaUPC,Courier 10 Pitch,Courier New,David, DilleniaUPC,DokChampa, Dotum,DotumChe,Ebrima, Estrangelo Edessa,EucrosiaUPC, Euphemia,FangSong,FrankRuehl, Franklin Gothic Medium,LilyUPC, Lucida Bright,Lucida Console, Lucida Sans Typewriter,Tahoma, Times New Roman,Traditional Arabic,Wingdings
http:userAgent	10	Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/29.0.1547.2 Safari/537.36	Mozilla/5.0 (X11; Linux i686 (x86_64)) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/27.0.1453.110 Safari/537.36

Results:

- None of the device fingerprint values match except for the incoming device fingerprint value and existing device fingerprint value for `accessTime`.
- Because all of the attributes except for `accessTime` have mismatched values, the collective weight of the mismatched attributes is `30`.
- The total weight of all of the attributes is `80`. The `accessTime` attribute has a weight value of `50`. The `http:userAgent` attribute, `browserPlugins` attribute, and `deviceFonts` attribute each have weight values of `10`.
- According to the risk score calculation formula: `(30/80)×100=38`. Therefore, the risk score is 38.
- Authentication is permitted because the risk score is below `40`.

Scenario 2: Browser risk profile

The risk engine uses risk profile information in the following table to calculate the risk score.			
Attributes	**Weight values**	**Incoming device fingerprint values**	**Registered device fingerprint values**
browserPlugins	50	Shockwave Flash,Chrome Remote Desktop Viewer,Widevine Content Decryption Module,Native Client,Chrome PDF Viewer,Java(TM) Plug-in 1.7.0,Citrix Receiver for Linux	Shockwave Flash,Chrome Remote Desktop Viewer,Native Client, Chrome PDF Viewer, Conference Plugin, AmazonMP3DownloaderPlugin, Google Update

The risk engine uses risk profile information in the following table to calculate the risk score. *(continued)*			
Attributes	**Weight values**	**Incoming device fingerprint values**	**Registered device fingerprint values**
deviceFonts	50	Andale Mono,Arial Black,Arial,Bitstream Charter,Century Schoolbook L,Comic Sans MS,Courier 10 Pitch,Courier New,DejaVu Sans Mono,DejaVu Sans,DejaVu Serif,Dingbats,Georgia, Impact,Khmer OS System,Khmer OS,Liberation Mono,Liberation Sans,Liberation Serif,Lohit Bengali,Lohit Gujarati,Lohit Punjabi,Lohit Tamil,Luxi Mono,Luxi Sans,Luxi Serif,Meera,Nimbus Mono L,Nimbus Roman No9 L,Nimbus Sans L,Standard Symbols L,Tahoma,Times New Roman,Trebuchet MS,URW Bookman L,URW Chancery L,URW Gothic L,URW Palladio L,UnBatang,UnDotum,Verdana, Waree,Webdings	Aharoni,Andalus,Angsana New, AngsanaUPC,Aparajita,Arabic Typesetting,Arial Black,Arial, Batang,BatangChe,Browallia New,BrowalliaUPC,Calibri, Cambria Math,Cambria,Candara, Comic Sans MS,Consolas, Constantia,Corbel,Cordia New,CordiaUPC,Courier 10 Pitch,Courier New,David, DilleniaUPC,DokChampa, Dotum,DotumChe,Ebrima, Estrangelo Edessa,EucrosiaUPC, Euphemia,FangSong,FrankRuehl, Franklin Gothic Medium,LilyUPC, Lucida Bright,Lucida Console, Lucida Sans Typewriter,Tahoma, Times New Roman,Traditional Arabic,Wingdings
http:accept	30	text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8	text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
http:acceptEncoding	50	gzip,deflate,sdch	gzip,deflate,sdch
http:acceptLanguage	50	en-US,en;q=0.8	en-US,en;q=0.8,es;q=0.6
http:userAgent	50	Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/27.0.1453.116 Safari/537.36	Mozilla/5.0 (X11; Linux i686 (x86_64)) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/27.0.1453.110 Safari/537.36

Results:

- None of the device fingerprint values match except for the incoming device fingerprint value and existing device fingerprint value for `http:accept` and `http:acceptEncoding`.
- Because all of the attributes except for `http:accept` and `http:acceptEncoding` have mismatched values, the collective weight of the mismatched attributes is `200`.
- The total weight of all of the attributes is `280`. The `http:accept` attribute has a weight value of `30`. The `browserPlugins` attribute, `deviceFonts` attribute, `http:acceptEncoding` attribute, `http:acceptLanguage` attribute, and `http:userAgent` attribute each have weight values of `50`.
- According to the risk score calculation formula: `(200/280)×100=71`. Therefore, the risk score is `71`.
- Authentication is denied because the risk score is above `40`.

Scenario 3: Device risk profile

The risk engine uses risk profile information in the following table to calculate the risk score.			
Attribute names	**Weight values**	**Incoming device fingerprint values**	**Registered device fingerprint values**
browserPlugins	30	Shockwave Flash,Chrome Remote Desktop Viewer,Widevine Content Decryption Module,Native Client,Chrome PDF Viewer,Java(TM) Plug-in 1.7.0,Citrix Receiver for Linux	Shockwave Flash,Chrome Remote Desktop Viewer,Native Client, Chrome PDF Viewer, Conference Plugin, AmazonMP3DownloaderPlugin, Google Update
colorDepth	50	24	32

The risk engine uses risk profile information in the following table to calculate the risk score. *(continued)*			
Attribute names	**Weight values**	**Incoming device fingerprint values**	**Registered device fingerprint values**
deviceFonts	50	Andale Mono,Arial Black,Arial,Bitstream Charter,Century Schoolbook L,Comic Sans MS,Courier 10 Pitch,Courier New,DejaVu Sans Mono,DejaVu Sans,DejaVu Serif,Dingbats,Georgia, Impact,Khmer OS System,Khmer OS,Liberation Mono,Liberation Sans,Liberation Serif,Lohit Bengali,Lohit Gujarati,Lohit Punjabi,Lohit Tamil,Luxi Mono,Luxi Sans,Luxi Serif,Meera,Nimbus Mono L,Nimbus Roman No9 L,Nimbus Sans L,Standard Symbols L,Tahoma,Times New Roman,Trebuchet MS,URW Bookman L,URW Chancery L,URW Gothic L,URW Palladio L,UnBatang,UnDotum,Verdana, Waree,Webdings	Aharoni,Andalus,Angsana New, AngsanaUPC,Aparajita,Arabic Typesetting,Arial Black,Arial, Batang,BatangChe,Browallia New,BrowalliaUPC,Calibri, Cambria Math,Cambria,Candara, Comic Sans MS,Consolas, Constantia,Corbel,Cordia New,CordiaUPC,Courier 10 Pitch,Courier New,David, DilleniaUPC,DokChampa, Dotum,DotumChe,Ebrima, Estrangelo Edessa,EucrosiaUPC, Euphemia,FangSong,FrankRuehl, Franklin Gothic Medium,LilyUPC, Lucida Bright,Lucida Console, Lucida Sans Typewriter,Tahoma, Times New Roman,Traditional Arabic,Wingdings
deviceLanguage	50	en-US	en-US
devicePlatform	50	Linux x86_64	Win-32
screenAvailable Height	50	1025	870
screenAvailable Width	50	1920	1600
screenHeight	50	1080	900
screenWidth	50	1920	1600

Results:

- None of the device fingerprint values match except for the incoming device fingerprint value and existing device fingerprint value for `deviceLanguage`.
- Because all of the attributes except for `deviceLanguage` have mismatched values, the collective weight of the mismatched attributes is 380.
- The total weight of all of the attributes is 430. The `browserPlugins` attribute has a weight value of 30. The following attributes have weight values of 50:
  - `colorDepth`
  - `deviceFonts`
  - `deviceLanguage`
  - `devicePlatform`
  - `screenAvailableHeight`
  - `screenAvailableWidth`
  - `screenHeight`
  - `screenWidth`
- According to the risk score calculation formula: `(380/430)×100=88`. Therefore, the risk score is 88.
- Authentication is denied because the risk score is above 40.

Scenario 4: Location risk profile

The risk engine uses risk profile information in the following table to calculate the risk score.			
Attributes	**Weight values**	**Incoming device fingerprint values**	**Registered device fingerprint values**
`geoCity`	10	Austin	Austin
`geoCountryCode`	10	US	US
`geoLocation`	50	30.2861, -97.739321, 10	30.274722, -97.740556, 13
`geoRegionCode`	10	TX	TX

Results:

- All of the device fingerprint values match. The `geoLocation` attribute contains the values that the risk engine uses to calculate the distance between the incoming device fingerprint and the registered device fingerprint. In this instance, the distance between the two device fingerprints is 1.27 km.
- Because all of the device fingerprint values match, the total weight of the mismatched attributes is 0.
- The total weight of all of the attributes is 80. The `geoLocation` attribute has a weight value of 50. The `geoCity` attribute, `geoCountryCode` attribute, and `geoRegionCode` attribute each have weight values of 10.
- According to the risk score calculation formula: (0/80)×100=0. Therefore, the risk score is 0.
- Authentication is permitted because the risk score is below 40.

Risk score calculation
Risk score calculation is the process by which the risk engine determines a risk score. The *risk score* demonstrates the level of risk that is associated with permitting a request to access the resource. This risk score is compared to a *threshold score* that is set in a policy. A decision is made based on the result of this comparison.

Chapter 9. Access control policies

An access control policy is a set of conditions that, after they have been evaluated, determine access decisions.

The conditions are a combination of attributes, obligations, authentication policies, and a risk profile.

Before you author a policy, review the available attributes, obligations, authentication policies, and risk profiles in the local management interface to determine if they meet the needs of your policy.

- “Managing attributes” on page 13
- “Managing obligations” on page 59
- “Managing authentication policies” on page 65
- “Managing risk profiles” on page 89

For more information, see Chapter 2, “Risk management overview,” on page 7.

When all of the attributes, obligations, and authentication policies are available and the risk profile you want to use is set active, use the policy editor to select the conditions for your policy.

Managing access control policies

An access control policy is a set of conditions that define whether a user is permitted or denied access to a protected resource.

About this task

You can view, create, edit, or delete an access control policy.

Procedure

1. Log in to the local management interface.
2. Click **AAC**.
3. Under **Policy**, click **Access Control**.
4. Click **All Policies**.
5. Perform one or more of the following actions:

   Search for and view policies

   Type one or more characters in the **Filter** field. The list displays the policies that start with those characters. Click either view icon:

   The details view lists the name and description of the policy.

   The list view lists only the name of the policy.

   Create a policy

   Click . The Policy Editor opens. See “Creating an access control policy” on page 100 for details.

   Edit a policy

   a. Select a policy in the policy list.
   b. Click . The Policy Editor opens.
   c. Change the policy. See “Creating an access control policy” on page 100 for details.
   d. Click **Save**.

Delete one or more policies

a. Select a policy or press Ctrl and select multiple policies in the policy list.

b. Click . A message prompts you to confirm the deletion.

c. Click **OK**. The policy or policies are deleted.

Add one or more polices to a policy set

A policy set is a group of policies that are used together to protect a resource. To add policies to a policy set, see "Managing access control policy sets" on page 103.

Import policy

You can import a policy from another appliance with Advanced Access Control. To import a policy, you must author it in the local management interface and export it using the **Export** function.

a. Select **All Policies** in the left navigation.

b. Click and select **Import Policy**.

c. In the Import Policy dialog box, click **Browse** to locate the file.

The policy file you import must have a `.json` extension. All custom attributes, obligations, and authentication policies referenced by the policy must exist for import to succeed.

d. Click **Import**. The policy displays in the list of policies.

Export policy

You can export a policy from the current appliance to use on another appliance.

a. Select **All Policies** in the left navigation.

b. From the list of policies, select a policy or press Ctrl and select multiple policies to export.

c. Click and select **Export Policy**.

d. Select the proper location and click **Save**.

Creating an access control policy

Use the **Policy Editor** on the appliance local management interface to create and configure an access control policy.

Before you begin

Each policy is a combination of attributes, obligations or authentications, and a risk profile.

Before you create an access control policy:

1. Ensure that the attributes and obligations you want to use in the policy are defined and available in the local management interface:
   - "Managing attributes" on page 13
   - "Managing obligations" on page 59
   - "Managing authentication policies" on page 65
2. Ensure that the risk profile you want to use is set as active. See "Managing risk profiles" on page 89.

About this task

The **Policy Editor** page has several sections:

Name and description

Specify a unique name for the policy and optionally include a description of the policy.

Subjects

Optionally specify one or more subjects to which the policy applies. Subjects can be anything in the Subject part of an access request. For example, use this field to specify that the policy applies to subjects who are members of the `SystemAdministrators` group. Click **Add Subject** to add subjects to the policy. Click to remove a subject from the policy. By specifying subjects, you can ensure that the policy rules are evaluated only if they match at least one of the specified subjects.

Rules

The Rules section has several settings:

Precedence

Specify an access action to take on the policy.

Deny

If any rule in the policy returns deny, the policy returns deny.

Permit

If any rule in the policy returns permit, the policy returns permit.

First

Access is permitted or denied based on the outcome of first rule in the policy that can be evaluated against the access request. The rules in the policy are evaluated in the same order they are listed and all rules will be processed regardless of which rule returns an applicable decision. The policy returns `Not Applicable` if none of the rules evaluates to true. To ensure that either a Permit or Deny decision is returned, include in the policy a Permit or Deny rule that does not contain a condition.

Attributes

When a policy is evaluated, the runtime will attempt to retrieve the values for all attributes that are specified in the policy. Attributes that are not found in the incoming request are considered missing. The **Attributes** setting controls how missing attributes are handled.

Optional

If **Attributes** is set to **Optional**, then all attributes specified in the Rule section of the policy are considered optional. With this setting, missing attributes are treated as empty sets and evaluated against the expression. In most cases, a missing attribute will cause the rule expression to return false.

Required

If **Attributes** is set to **Required**, then all attributes specified in the Rule section of the policy are considered required. With this setting, missing attributes are considered an error and will return a decision of Indeterminate when the rule is evaluated. Indeterminate results often cause the access request to be denied.

Add Rule

Click the **Add Rule** drop-down arrow and select either:

- **Conditional rule**: This type of rule contains one or more conditions and an action. Rules are boolean expressions that are applied to a set of context attributes that are passed in the context object of the decision request. Each rule has an `If` statement and a Then statement. The `If` statement specifies the conditions that are checked when an access request is received. The Then statement specifies the action to take when the rule conditions are true.
- **Unconditional rule**: This type of rule contains only an action and no conditions.

The rule actions are:

Permit

The request must be permitted to pass.

Permit with Obligation

A specific action must take place before the request is permitted to pass. Specify the action in the adjacent field.

Permit with Authentication

A specific authentication action must take place before the request is permitted to pass. Specify the authentication policy in the adjacent field. For more information about authentication policies, see Chapter 7, "Authentication policies," on page 65.

Deny
The request must be denied and not permitted to pass.

Deny with Obligation
The request is denied and an obligation is processed.

Procedure

1. Log in to the local management interface.
2. Click **AAC**.
3. Under **Policy**, click **Access Control**.
4. In the center panel, click .
5. In the **Name** field, type a unique name for the policy.

   Note: The name must begin with an alphabetic character. Do not use control characters, leading and trailing blanks, and the following special characters ~ ! @ # $ % ^ & * ( ) + | ` = \ ; : " ' < > ? , [ ] { } / anywhere in the name.
6. Optional: In the **Description** field, type a description for the policy.
7. Optional: Specify subjects to which the policy applies.
   a) Click **Add Subject**.
   b) In the first box, select a subject attribute.
      Begin typing the name of the subject to filter the list.
   c) In the second box, select an operator.
   d) In the third box, type a value.

   For example, if you want the rule to be evaluated only if the access requestor belongs to the `SecurityAdministrator` group, specify the following selections:

   Parameter
   `groups`

   Operator
   =

   Value
   `SecurityAdministrator`

   Note: If your LDAP root DN is `secauthority=default`, you can only use the **=** (equal) operator in policies that use X.500 names `userDN` and `groupsDN`.

   To specify more subjects, click **Add Subject**.
8. In the Rule section, add one or more rules.
   a) For **Precedence**, select the access action to take for the policy:
   b) For **Attributes**, select the attribute usage of the policy.
   c) To add a rule to the policy, click the **Add Rule** drop-down arrow and choose one of the following:
      - **Conditional rule**: This type of rule contains one or more conditions and an action.
      - **Unconditional rule**: This type of rule contains no conditions.
   d) If you create an unconditional rule, continue with step "8.h" on page 103.
   e) If you are creating a Conditional rule, select whether the rules apply if **All** conditions are true or if **Any** of the conditions are true.
   f) Create a rule by typing or selecting a parameter, operator, and value.
      To specify a value in the value field, click the drop-down menu on the right and select either **Enter Value** or **Select Attribute**.
   g) Take one of the following actions:

- Click **!** to add a NOT operator to the expression. If the expression already has a NOT operator, clicking **!** removes the operator.
- Click **+** to add another expression. The new expression is added below the preceding expression.
- Click **-** to remove an expression.
- Click **()** to create a parenthetical expression. Select the appropriate attributes, operators, and values for the expression. Or, add more expressions to the group. The new expression is added below the preceding expression.

h) Specify the action to take when the rule evaluation is completed.

i) Click **OK** when the rule is complete.

j) To add another rule to the policy, repeat step "8.c" on page 102.

k) If your policy has more than one rule, you can change the sequence of the rules by selecting a rule and clicking or .

Note: The sequence of the rules is important if you have selected **First** as the action for the policy.

9. Click **Save** when the policy is complete.

What to do next

Attach the policy to a resource. See "Managing access control policy attachments" on page 105.

Related reference

"Predefined attributes" on page 19

An appliance with Advanced Access Control uses attributes to provide information about users and devices that try to access a protected resource. The appliance also includes a set of commonly used attributes called *predefined attributes*.

Managing access control policy sets

A *policy set* is a group of policies that are used together to protect a resource.

Before you begin

You must create access control policies. See "Creating an access control policy" on page 100.

About this task

You can view, create, modify, or delete a policy set.

Procedure

1. Log in to the local management interface.
2. Click **AAC**.
3. Under **Policy**, click **Access Control**.
4. Perform one or more of the following actions:

Create a policy set

a. Click .

b. Type a name for the policy set in the **Name** field.

Note: The name must begin with an alphabetic character. Do not use control characters, leading and trailing blanks, and the following special characters ~ ! @ # $ % ^ & * ( ) + | ` = \ ; : " ' < > ? , [ ] { } / anywhere in the name.

c. Optionally, type a description in the **Description** field.

d. In the Policy Combining Algorithm dialog, set the combined action for the policy set by choosing one of the following options:

Deny access if any policy in the set returns deny
Choose this option if you want the policy set to deny access if any policy in the set returns a response of `deny`.

Permit access if any policy in the set returns permit
Choose this option if you want the policy set to permit access if any policy in the set returns a response of `permit`.

Return the decision of the first policy in the set that returns either permit or deny
The policies are evaluated in the order they are listed in the set. Choose this option if you want to use the first policy that returns a response of permit or deny as the result of the policy set. All policies will be evaluated regardless of which policy returns a decision result.

e. Click **Save**.
f. Click **OK**.
g. Next, add one or more policies to the policy set.

Add one or more policies to a policy set

a. Click **All Policies**.
b. Select a policy or press Ctrl and select multiple policies to add to the policy set.
c. Click Add To.
d. Select a policy set.
e. Click **OK**.

Change the order of the policies in a set
If you selected **Return the decision of the first policy in the set that returns either permit or deny**, set the order in which you want the policies to run:

a. Select a policy set.
b. Select a policy.
c. Click or to change the position of the policy in the set.

Modify a policy set

a. Select a policy set in the list of policy sets.
b. Click .
c. Change to the name, description, or policy combining algorithm.

Note: The name must begin with an alphabetic character. Do not use control characters, leading and trailing blanks, and the following special characters ~ ! @ # $ % ^ & * ( ) + | ` = \ ; : " ' < > ? , [ ] { } / anywhere in the name.

d. Click **Save**.

Remove one or more policies from a policy set

a. Select a policy set in the list of policy sets.
b. Select a policy or press Ctrl and select multiple policies to remove from the policy set.
c. Click Remove. Confirm the removal.

Note: When you remove a policy from a set, the policy is not deleted from the policy list. To delete a policy from the list, see "Managing access control policies" on page 99.

d. Click **OK**. The policy is removed.

Delete a policy set

a. Select a policy set in the list of policy sets.
b. Click . Confirm the deletion.

c. Click **OK**. The policy set is deleted.

Managing access control policy attachments

Attach policies or API protection definitions to resources so that the policies and definitions can be enforced.

Before you begin

You must create policies, policy sets, or API protection definitions. You cannot use them until you publish them to resources. After publication, they are enforced during the evaluation of access requests.

About this task

You can perform the following tasks:

- Add a resource
- Add a policy or API protection definition attachment to a resource
- Remove a policy or API protection definition attachment from a resource
- Delete a resource
- Publish a policy or API protection definition attachment

When a deployment is fully configured, the **Resources** panel displays three levels of entries. The top-level entry is the web container for the protected object space for a server instance. The second level shows the resources in the protected object space. The third level lists the policies and API protection definitions that are attached to each resource.

Tip: The user interface provides a quick filter feature for the top-level entry. Use the quick filter to search for a specific top-level entry. Enter the first few characters of the web container, and the list displays only the entries that contain the specified characters.

Procedure

1. Log in to the local management interface.
2. Click **AAC**.
3. Under **Policy**, click **Access Control**.
4. Click **Resources**.
5. Perform one or more of the following actions:

   Add a resource

   If you need to add a resource from a custom domain, see Defining a custom domain for policy attachments.

   a. Click .

   Note: When you add a resource for the first time, the system prompts you to enter the user name, password, and domain for the Security Verify Access policy server. The entered information is cached and used by default when you add a resource again. If you want to change this domain, click **Change Domain** and then enter the new user name, password, and domain information. This new information replaces the old cached values.

   b. Select the resource type in the **Type** field.

   - If you select the **Reverse Proxy** type:

     i) In the **Proxy Instance** field, click the down arrow icon to display a list of proxy instances. Select an entry.

     For example, the list of proxy instances is the WebSEAL protected object space that is defined directly under `/WebSEAL`.

ii) Specify a resource by entering its name or browsing for it. When you browse, you can expand the list of resources. The list hierarchy is based on the structure of the WebSEAL protected object space.

  - In some cases, not all resources are displayed because the WebSEAL protected object space is a sparse tree. For example, you might see only the resource `/myserver-jct/benefits`. You can select this resource and click **OK** to add it to the **Protected Path**. You can then add `/myserver-jct/benefits/medical`.
  - In some cases, you cannot view the object space for the web server junction. For example, if the administrator did not install the IBM Security Verify Access **querycontents** script on the application server, you cannot see the junction contents. In these cases, you can enter the resource path manually.

- If you select the **Application** type:

  i) Select an application ID from the list or click **Add New** to add an application ID.

  ii) Enter the resource ID.

c. Select an option to set the decision cache timeout period for any authorization decisions of any policy that is attached to this resource.

   Note: The decision cache setting takes effect only after the policy is attached to the resource.

d. Click **Save**.

e. Attach a policy to the resource.

Attach a policy or API protection definition to a resource

a. Select a resource node and click Attach.

b. In the **Attach Policies** panel, select **Policies** or **Policy Sets** or **API Protection**.

c. From the list, select one or more items.

   Tip: You can type the name in the quick filter.

   Notes:

   - You can attach both individual policies, policy sets, or API protection definitions.
   - You cannot attach policies or policy sets to a resource where that resource already has API protection definitions attached.
   - You cannot attach API protection definitions to a resource where that resource already has policies and policy sets attached.

d. Click **OK** to save your changes.

   Note: The policy or API protection definition remains inactive until you publish it.

Remove a policy or API protection definition attachment

a. To remove a policy or API protection definition attachment from a resource, select the policy node and click .

b. When prompted, confirm the deletion.

   Note: You must publish the change.

Delete a resource

Note: When you delete a resource:

- Be aware of the status of the reverse proxy server that the resource is attached to before deleting it:
  - If the reverse proxy server is defined but not available for use, restart the server first. Then, follow the instructions below to delete the resource from the local management interface.

– If the reverse proxy server has been deleted, force the delete of the resource to remove it from the local management interface.

- You cannot delete the server node.

a. To delete a resource and all attached policies or API protection definitions, select the resource node and click .

b. When prompted, confirm the deletion.

You do not have to manually publish the change. The deletion is automatically published.

Publish policies or API protection definitions

Publish a specific policy or API protection definition, or publish all of them at once:

- **Publish**: Select a resource in the resource hierarchy and click Publish. When the publication completes, the status column for the resource indicates the status and time of the publication.
- **Publish All**: Click Publish All and then respond to the confirmation. This action publishes only those policies or API protection definitions that have a status of "Publish required".

  When the publication completes, the status column for the resources indicates the status and time of the publication.

Note: Activation of a single published policy or API protection definition might take up to a minute to complete.

Modify Resource

Note: You can use this function only if policy or policy sets are attached to the resource.

a. Select a resource node and click .

b. In the **Modify Resource** panel, you can modify the **Policy Combining Algorithm**. Choose the preferred algorithm:

- **Deny access if any attached policy returns deny**

  If both of the following statements are true, then the access request is denied.

  – Multiple policies or API protection definitions are attached to a resource.

  – Any one of the policies or API protection definitions returns `Deny`.

- **Permit access if any attached policy returns permit**

  If any one of the following statements is true, then the access request is permitted.

  – Multiple policies or API protection definitions are attached to a resource.

  – Any one of the policies or API protection definitions returns `Permit`.

c. Modify the setting for the decision cache timeout period for any authorization decisions of any policy that is attached to this resource.

Note: The decision cache setting takes effect only after the policy is attached to the resource.

Policy scenarios

Several commonly used policy scenarios are provided as examples to help you author policies.

Denying access based on a set of conditions

A common policy scenario is to deny access based on a set of conditions.

About this task

Use the steps in this scenario task to create a policy that denies access if any of the following conditions are true.

- The calculated risk score is higher than a value of 40.
- The reputation of the `ipAddress` in the request is considered malware.

Procedure

1. Log in to the local management interface.
2. Click **AAC**.
3. Under **Policy**, click **Access Control**.
4. In the center panel, click .
5. Enter a name for the policy.
6. In the Rules section, set the Precedence property to **Deny**.

   As a result, access is denied if any rule returns deny.
7. Click **Add Rule**.
8. Select **riskScore** from the attribute list.
9. Select **>** as the operator.
10. Type `40` as the value.
11. In the Decision list, select **Deny**.
12. Click **OK** to complete the rule.
13. Click **Add Rule** to add another rule.
14. Select **ipReputation** from the attribute list.
15. Select **has member** as the operator.
16. Type `Malware` as the value.
17. In the Decision list, select **Deny**.
18. Click **OK** to complete the rule.
19. Click the arrow next to **Add Rule**.
20. Click **Unconditional rule**.
21. In the Decision list, select **Permit**.

    The unconditional Permit rule causes the policy to permit access if none of the deny access rules evaluate to true.
22. Click **OK**.

Results

This scenario uses the following settings in the policy editor.

- Precedence: **Deny**
- Attributes: **Optional**
- Rule 1: If **riskScore >40** Then **Deny**
- Rule 2: If **ipReputation has member Malware** Then **Deny**
- Rule 3: **Permit**

Denying access based on a set of conditions with an OR clause

A common policy scenario is to use multiple conditions in a single rule and to join those conditions with And or `Or`. In this scenario, access is denied if either of the policy conditions that are joined by `Or` are true.

About this task

Use the steps in this scenario task to create a policy that denies access if either of the following conditions are true:

- The calculated risk score is higher than a value of 40.

- The reputation of the `ipAddress` in the request is considered malware.

Procedure

1. Log in to the local management interface.
2. Click **AAC**.
3. Under **Policy**, click **Access Control**.
4. In the center panel, click .
5. Enter a name for the policy.
6. In the Rules section, set the Precedence property to **Deny**.

   As a result, access is denied if any rule returns deny.
7. Click **Add Rule**.
8. Click **If Any are true**.

   The rule evaluates to true if any of the conditions in the rule are true.
9. Select **riskScore** from the attribute list.
10. Select **>** as the operator.
11. Type `40` as the value.
12. Click to add another condition to the rule.
13. Select **ipReputation** from the attribute list.
14. Select **has member** as the operator.
15. Type `Malware` as the value.
16. In the Decision list, select **Deny**.
17. Click **OK** to complete the rule.
18. Click the arrow next to **Add Rule**.
19. Click **Unconditional rule**.
20. In the Decision list, select **Permit**.

    The unconditional Permit rule causes the policy to permit access if none of the deny access rules evaluate to true.
21. Click **OK**.

Results

This scenario uses the following settings in the policy editor.

- Precedence: **Deny**
- Attributes: **Optional**
- Rule 1: If **riskScore >40** or **ipReputation has member Malware** Then **Deny**
- Rule 2: **Permit**

Permitting access based on a set of conditions with an AND clause

A common policy scenario is to use multiple conditions in a single rule and to join those conditions with And or `Or`. In this scenario, access is permitted if both of the policy conditions that are joined by And are true.

About this task

Use the steps in this scenario task to create a policy that permits access if both of the following conditions are true:

- The calculated risk score is less than or equal to a value of 40.
- The reputation of the `ipAddress` in the request is not considered malware.

Procedure

1. Log in to the local management interface.
2. Click **AAC**.
3. Under **Policy**, click **Access Control**.
4. In the center panel, click .
5. Enter a name for the policy.
6. In the Rules section, set the Precedence property to **Permit**.

   As a result, access is permitted if any rule returns permit.
7. Click **Add Rule**.
8. Click **If all are true**.

   The rule evaluates to true if all of the conditions in the rule are true.
9. Select **riskScore** from the attribute list.
10. Select **<=** as the operator.
11. Type `40` as the value.
12. Click to add another condition to the rule.
13. Select **ipReputation** from the attribute list.
14. Select **has member** as the operator.
15. Type `Malware` as the value.
16. Click to convert the condition to a `Not` condition.
17. In the Decision list, select **Permit**.
18. Click **OK** to complete the rule.
19. Click the arrow next to **Add Rule**.
20. Click **Unconditional rule**.
21. In the Decision list, select **Deny**.

    The unconditional Deny rule causes the policy to deny access if none of the permit access rules evaluate to true.
22. Click **OK**.

Results

This scenario uses the following settings in the policy editor.

- Precedence: **Permit**
- Attributes: **Optional**
- Rule 1: If **riskScore <=40** and **not (ipReputation has member Malware**) Then **Permit**
- Rule 2: **Deny**

Permitting access after one-time password authentication

Security Verify Access can prompt users for one-time passwords when they request access to protected resources. You can use a policy to permit access to users who authenticated with a one-time password. Or, you can prompt them for the password and then permit access when they provide it.

Before you begin

Configure the TOTP one-time password mechanism. See Configuring a TOTP one-time password mechanism.

About this task

Use the steps in this scenario task to create a policy that permits access after the user authenticates with a one-time password.

Procedure

1. Log in to the local management interface.
2. Click **AAC**.
3. Under **Policy**, click **Access Control**.
4. In the center panel, click .
5. Enter a name for the policy.
6. In the Rules section, set the Precedence property to **First**.

   As a result, the policy returns a decision for the first rule in the policy that evaluates to true.
7. Click **Add Rule**.
8. Select **authenticationTypes** from the attribute list.
9. Select **has member** as the operator.
10. Type `urn:ibm:security:authentication:asf:totp` as the value.

    If this value is present, the request was already authenticated with a one-time password.
11. In the Decision list, select **Permit**.
12. Click **OK** to complete the rule.
13. Click the arrow next to **Add Rule**.
14. Click **Unconditional rule**.
15. In the Decision list, select **Permit with authentication**.
16. In the Authentication list, select **TOTP One-time Password**.

    This selection results in a request for a one-time password from the user.
17. Click **OK**.

Results

This scenario uses the following settings in the policy editor.

- Precedence: **First**
- Attributes: **Optional**
- Rule 1: If **authenticationTypes has member "urn:ibm:security:authentication:asf:totp"** Then **Permit**
- Rule 2: **Permit with Authentication TOTP One-time Password**

Enforcing an authentication policy for every access per session

You can enforce an authentication policy once per session or every time a user accesses a protected resource. In this scenario, the authentication service relies on the `authenticationTypes` credential attribute to determine which authentication policies the user successfully completed during the authentication session.

About this task

Use this task to enforce a particular authentication policy every time the user accesses a protected resource during a session.

Procedure

1. Log in to the local management interface.
2. Click **AAC**.
3. Under **Policy**, click **Access Control**.

4. In the center panel, click .
5. Enter a name for the policy.
6. In the Rules section, set the Precedence property to **First**.

   As a result, the policy returns a decision for the first rule in the policy that evaluates to true.
7. Click **Add Rule**.
8. Click **authenticationTypes** from the attribute list.
9. Select **has member** as the operator.
10. Enter the unique identifier for the policy.
    For example, to enforce only the user name and password policy add the value: `urn:ibm:security:authentication:asf:password`.
11. In the Decision list, select **Permit**.
12. Click **OK** to complete the rule.
13. Click the arrow next to **Add Rule**.
14. Click **Unconditional rule**.
15. In the Decision list, select **Permit with authentication**.
16. In the Authentication list, select **Username Password**.

    This selection results in request for a user name password from the user.
17. Click **OK**.

Results

This scenario uses the following settings in the policy editor:

- Precedence: **First**
- Attributes: **Optional**
- Rule 1: If **authenicationTypes has member** `urn:ibm:security:authentication:asf:password` Then **Permit**
- Rule 2: If **Permit with Authentication Username Password**

Enforcing an authentication mechanism once per session

You can enforce an authentication mechanism once per session or every time a user accesses a protected resource. In this scenario, the authentication service relies on the `authenticationMechanismTypes` credential attributes to determine which authentication mechanisms the user successfully completed during the authentication session.

About this task

Use this task to enforce a particular authentication mechanism only once during the user's authenticated session. This scenario uses the **Username Password** and **MAC One-time Password** mechanisms. However, you can use any authentication mechanisms.

Procedure

1. Log in to the local management interface.
2. Click **AAC**.
3. Under **Policy**, click **Authentication**.
4. In the center panel, click .
5. Enter the name of the custom authentication policy.
6. Enter the **Identifier**.
7. In the Workflow Steps section, click **Add Step**.
8. Select **Username Password** and click **OK**.

9. Click **Parameter**.
10. Select the **Pass** check box for the **reauthenticate** parameter.
11. Select **Value** in the **Source** field.
12. Select **False** in the **Value** field.
13. Click **OK**.
14. In the Workflow Steps section, click **Add Step**.
15. Select **MAC One-time Password**.
16. Click **OK**.
17. Click **Save**.
18. Under **Policy**, click **Access Control**.
19. In the center panel, click.
20. Enter a name for the policy.
21. In the Rules section, set the **Precedence** property to **First**. As a result, the policy returns a decision for the first rule in the policy that evaluates to true.
22. Click **Add Unconditional Rule**.
23. In the Decision list, select **Permit with authentication**.
24. In the Authentication list, select the name of the custom authentication policy that is created in step “5” on page 112.
25. Click **OK**.
26. Click **Save**.

Results

This scenario uses the following settings in the policy editor:

- Precedence: **First**
- Attributes: **Optional**
- Rule 1: An **Unconditional Rule** that enforces a **Permit with Authentication** with a custom Two-Factor authentication policy.

  Note: If the **Username Password** mechanism was completed by the user during the authenticated session, the user is required to provide only the MAC one-time password as indicated by the second workflow step.

Registering a device after user consent

Device registration is the process that stores the device fingerprint of the user in the risk-based access database. The rules that you specify in a policy determine whether a device is registered silently or only after the user consents to the registration.

About this task

Use the steps in this scenario task to create a policy that registers a device after the user gives consent. This scenario contains the following rules:

- If the calculated risk score is lower than a specified value, then Permit access.
- If the user consents to device registration, then register the device. This action causes the `riskScore` for the request to be lowered when the policy is reevaluated.
- If the user was not yet prompted to register the device, then display the consent registration form to the user.
- Deny access if none of the above rules evaluates to true.

When you use a **Consent Register Device** authentication or a **Register Device** obligation in your access policy, consider the following actions:

- If you include a consent to register rule in your access policy, the user is prompted to give consent. If the user responds to the prompt, by either giving consent or not, the authentication flow continues. If the user does not give consent, the authentication flow does not end. In the following scenario, the "userConsent=true" rule controls whether to display the consent registration form to the user.
- Even if the user gives consent to register the device, the device is not registered unless you include a Device Registration obligation in your policy. In the following scenario, the "Permit with Obligation Register Device" rule controls the registration of the device.

Procedure

1. Log in to the local management interface.
2. Click **AAC**.
3. Under **Policy**, click **Access Control**.
4. In the center panel, click .
5. Enter a name for the policy.
6. In the Rules section, set the Precedence property to **First**.

   As a result, the policy returns a decision for the first rule in the policy that evaluates to true.
7. Click **Add Rule**.
8. Click **If all are true**.

   The rule evaluates to true if all of the conditions in the rule are true.
9. Select **riskScore** from the attribute list.
10. Select **<** as the operator.
11. Type `40` as the value.
12. In the Decision list, select **Permit**.

    As a result, the policy permits access if the risk score is below the specified value.
13. Click **OK** to complete the rule.
14. Add a second rule that checks whether the **userConsent** attribute is present and that the value of **userConsent** is `true`.

    The **userConsent** attribute is present only if the user was previously prompted with the device registration consent form. If the **userConsent** attribute is true, the user granted consent to register the device.

    a) Click **Add Rule**.

    b) Select **userConsent** from the attribute list.

    c) Select **is present** as the operator.

    d) Click to add another condition to the rule.

    e) Select **userConsent** from the attribute list.

    f) Select **=** as the operator.

    g) Type `true` as the value.

    h) In the Decision list, select **Permit with Obligation**.

    i) In the Obligation list, select **Register Device**.

    j) Click **OK**.
15. Add a third rule that checks whether the **userConsent** attribute is missing.

    If the **userConsent** attribute is missing, the user was not yet prompted with the device registration consent form.

    a) Click **Add Rule**.

    b) Select **userConsent** from the attribute list.

    c) Select **is missing** as the operator.

    d) In the Decision list, select **Permit with Authentication**.
    e) In the Authentication list, select **Consent Register Device**.
    f) Click **OK**.
16. Click the arrow next to **Add Rule**.
17. Click **Unconditional rule**.
18. In the Decision list, select **Deny**.

    This rule is evaluated only if none of the previous rules evaluates to true. This situation can occur if:

    - The risk score is higher than the specified value in the first rule (40 in this example).
    - The user decided not to give consent to register the device.
19. Click **OK**.
20. Click **Save**.

Results

This scenario uses the following settings in the policy editor.

- Precedence: **First**
- Attributes: **Optional**
- Rule 1: If **riskScore <40** Then **Permit**
- Rule 2: If **userConsent is present** and **userConsent=true** Then **Permit with Obligation Register Device**
- Rule 3: If **userConsent is missing** Then **Permit with Authentication Consent Register Device**
- Rule 4: **Deny**

Chapter 10. Device fingerprints

Device registration is the process that stores the device fingerprint of the user in the context-based access database.

The rules that you specify in the context-based access policy determine whether a device is registered silently or only after the user consents to the registration.

Device fingerprints can be managed by the administrator and also by the user. The administrator can list the device fingerprints that are registered to a user, and can deregister device fingerprints. The administrator can also set the authentication level for consent-based registration, and modify template pages for use with consent-based registration.

For information on user self-administration of registered devices, see "Managing your registered devices" on page 209

Managing device fingerprints

The device fingerprint contains information that is required for risk score calculation. You can list fingerprints or deregister fingerprints.

About this task

To list registered devices, you can search on a user ID or use an asterisk as a wildcard to search on partial matches.

Note: User names are not stored with the fingerprints and you cannot search on user names.

You can also deregister a selected device fingerprint or deregister all of the device fingerprints that belong to a specific user. When you deregister a device fingerprint, it is removed from the database.

Procedure

1. Log in to the local management interface.
2. Click **AAC**.
3. Under **Manage**, click **Devices**.
4. Take one of the following actions:

   List device fingerprints:

   a. In the **Search for the user ID** field, type:
      - One or more characters and an asterisk * in the first or last position to search on partial matches. For example, enter `g*` to search for all user IDs that begin with g. The search is case-sensitive. Specifying an uppercase letter returns only uppercase results and specifying a lowercase letter returns only lowercase results.
      - A specific user ID.

   b. In the **Maximum results** field, the default value is `100`. Use this value or change it to another value. The minimum value is `1`. The maximum value is `1000`.

   c. Click **Search**. A list of user IDs is displayed.

   d. Click a user ID in the User Results list. The Registered Device Fingerprints table is displayed.

   Deregister a registered device fingerprint:

   a. List the device fingerprints for a specific user.

   b. Click **Deregister All** to remove all of the device fingerprints for the user, or select a device fingerprint and click **Deregister**.

   c. Confirm that you want to deregister the selected device fingerprints.

What to do next

If you want to configure device fingerprint expiration, see "Configuring device fingerprint expiration" on page 121.

Silent device registration

Silent device registration is the process of registering the device after the user responds successfully to a secondary challenge.

Silent registration does not require any further interaction or consent from the user. The context-based access policy that you configure determines whether a device must be registered silently.

You can optionally configure your system to allow incomplete device fingerprints to be registered. Use the Advanced Configuration menu to specify the necessary properties. For more information, see Managing advanced configuration.

Consent-based device registration

Consent-based device registration is the process of registering the device fingerprint only after the user consents to the device registration.

A typical scenario that requires consent-based device registration is when a user attempts to access a protected resource from a public access environment. For example, a user might log in from an internet café or airport kiosk. After the user logs in and successfully responds to the secondary challenge, a consent form is presented. The consent form can be an HTML page where users can specify that they consent to the device registration.

- If the user consents to the registration of the device, the device is registered and access is permitted. The next time that the user logs in from the same device, the consent form is not presented because the device is already registered.
- If the user does not consent to the registration of the device, the device is not registered and the access is permitted. If the user logs in from the same device again, the secondary challenge and the consent form are presented again. The process is repeated because the risk score is high when a user logs in from a device that is not registered.

When a user consents to registration of the device, two attributes are automatically set. You can use these attributes when creating policy:

userConsent
: Sets the boolean value to `true`. Specifies that the user has consented to device registration

authenticationLevel
: Numeric value that specifies the authentication level of a user. It increases as the levels of authentication that belong to the user increase. For example: A possible authentication level is 2

  When a user is granted access, the authentication level is set by the policy enforcement point. This is the default behavior. Optionally, you can control the authentication level for the user, by setting advanced configuration properties.

Advanced Access Control provides a template page that you can use for the HTML page to display in order to obtain user consent.

Context-based access policy sample settings to support consent-based device registration

Consent-based device registration is typically enabled and supported by combining the “Consent Register Device” authentication policy and the “Register Device” obligation within a CBA policy.

The CBA policy typically also references the “riskScore” session attribute as a policy rule condition to prevent duplicate registrations. For consent-based device registration, the CBA policy might include rules similar to the following examples:

```
If riskScore <= 40
Then Permit

If userConsent = “true”
Then Permit with Obligation Register Device

If riskScore > 40 and
   userConsent != “true”
Then Permit with Authentication Consent Register Device
```

Setting the authentication level for consent-based device registration

You can specify the authentication level to grant to a user who consents to device registration.

About this task

By default, a user who consents to device registration is granted access without a specified authentication level. Optionally, you can set properties to specify the authentication level. You might want to do this action if the user needs access to resources that are restricted to users with a higher access level.

Procedure

1. Select **AAC** > **Global Settings** > **Advanced Configuration**.
2. Under **Filter by Category**, select **deviceRegistration**.

   The advanced configuration properties window lists the properties for device registration.
3. Click for the property **consentDeviceRegisration.authLevelHeaderEnabled**.
4. Select the **Enabled** check box, and click **Save**.
5. Click for the property **consentDeviceRegisration.authLevelHeaderValue**.
6. Specify a value for the authentication level to be granted to the user. Click **Save**.
7. When you change configuration settings, the appliance alerts you that there are undeployed changes. If your changes are complete, deploy them.

   For more information, see Chapter 14, “Deploying pending changes,” on page 157.

Modifying consent template pages

Use the local management interface to manage files and directories in the template files.

About this task

The template page for `consent-form.html` displays the form where the user can indicate consent to device registration and optionally provide a name to the device that is being registered

You can run the following tasks on the template files:

- **Edit**- Use this option if you want to view or modify the template file.
- **Import**- Use this option if you to import a file to the template files root.
- **Export**- Use this option if you want to export a file from the template files root.

- **Import Zip**- Use this option if you want to import the template files from a compressed file.
- **Export Zip**- Use this option if you want to export the template files as a compressed file.

Procedure

1. From the top menu, select **AAC** > **Global Settings** > **Template Files**.
2. Work with all the management files and directories.

   View or update the contents of a file in the template files root

   a. Select the file of interest.
   b. Select **Edit**. You can then view the contents of the file.
   c. Edit the contents of the file.
   d. Click **Save**.

   Export a file from the template files root

   a. Select the file of interest.
   b. Select **Manage** > **Export**.
   c. Confirm the save operation when your browser displays a confirmation window.

   Import a file to the template files root

   a. Select the file of interest.
   b. Select **Manage** > **Import**.
   c. Click **Browse**.
   d. Browse to the file you want to import.
   e. Click **Open**.
   f. Click **Import**.

   Export the template file as a compressed file

   a. Select **Manage** > **Export Zip**.
   b. Confirm the save operation when your browser displays a confirmation window.

   Import the template files as a compressed file

   Make sure that the .zip file contains files that exist in the document root.

   a. Select **Manage** > **Import Zip**.
   b. Click **Browse**.
   c. Browse to the file you want to import.
   d. Click **Open**.
   e. Click **Import**.
3. When you edit or import template files, the appliance displays a message that there are undeployed changes. If you have finished making changes, deploy them.

   For more information, see Chapter 14, “Deploying pending changes,” on page 157.

Device fingerprint template page for consent-based registration

This `consent-form.html` template page displays the form where the user can indicate consent to device registration. The user also has the option of providing a name to the device that is being registered.

The `consent-form.html` template page has the following replacement macro: `@ERROR_MESSAGE@`

This macro is replaced with a message that indicates that there was an error in obtaining consent for device registration. For example, the device name is not valid.

Note: The device name must begin with an alphabetic character. Do not use control characters, leading and trailing blanks, and the following special characters: ~ ! @ # $ % ^ & * ( ) + | ` = \ ; " ' < > ? , [ ] { } / anywhere in the name.

Configuring device fingerprint expiration

The administrator can specify the number of days that a device fingerprint can remain valid based on the number of days since the device was last used.

About this task

The user must re-register a device when the device fingerprint expires. For example, the user must re-register a device under the following circumstances:

- The device has a registered fingerprint that was not used in 120 days.
- The administrator specified that devices must be used at least one time per 90 days.

Note: The risk engine does not consider expired device fingerprints when it calculates risk scores. To use an expired device fingerprint, re-register the device.

Procedure

1. Log in to the local management interface.
2. Click **AAC**.
3. Under **Global Settings**, click **Advanced Configuration**.
4. Configure the following properties:
   - `deviceRegistration.checkForExpiredDevices`
     a. Click **Edit** .
     b. Select **Enabled**. This property enables the risk engine to check whether devices are inactive or expired.
     c. Click **Save**.
   - `deviceRegistration.inactiveExpirationTime`
     a. Click **Edit** .
     b. Specify the number of days that device fingerprints remain valid if they are not used. The default is `90` days.
     c. Click **Save**.

Related tasks

"Managing device fingerprints " on page 117
The device fingerprint contains information that is required for risk score calculation. You can list fingerprints or deregister fingerprints.

Related information

Advanced Configuration Properties

Chapter 11. Runtime database

The runtime database stores user data such as session attributes and device fingerprints.

The runtime database is required for environments that utilize the Advanced Access Control and Federation capabilities of Security Verify Access. Although the appliance contains support for an embedded runtime database, it is recommended that the embedded runtime database is only used in environments that require a small amount of data to be stored (for example, in proof-of-concepts, development and test environments, etc). In production environments, it is recommended that an external database be used to store the runtime data. In this way, the administrator can have much greater control over disk space allocations, high availability, disaster recovery, and database tuning. The appliance also provides the ability to migrate the data that is stored in the embedded database into an external database to help those customers who want to start using an external database.

To manage the disk space usage of this database, you can clear some or all of its stored data.

To optimize performance or increase capacity, you can optionally deploy an external runtime database for the appliance.

Managing the runtime database

To manage the disk space usage of this database, you can clear some or all of its stored data.

About this task

When you remove data, the Security Verify Access runtime is stopped.

Procedure

1. Log in to the local management interface.
2. Select **AAC** > **Manage** > **Database Maintenance**
3. Take one of the following actions:

   Remove device fingerprints:

   a. Select **Remove device fingerprints**.
   b. Select a time period in days, months, or years. For example, you might select **1 Year**. When you click **Remove**, device fingerprints that have not been used in more than 1 year are removed from the database. The default is 1 year.
   c. Click **Remove**.

   Remove user session data:

   a. Select **Remove user session data**.
   b. Select a time period in hours or days. For example, you might select **12 Hours**. When you click **Remove**, data older than 12 hours is removed from the database. The default is 1 day.
   c. Click **Remove**.

   Remove all data from the runtime database:

   Attention: Use this selection only if you want to remove *all* data from the runtime database.

   a. Select **Remove all data from the runtime database**.
   b. Click **Remove**.
   c. Respond to the confirmation message. Click **Yes** to remove all data or **No** to cancel the removal.

   Note: If the runtime was not stopped before you selected an action, a message is displayed that asks if the runtime can be stopped. The runtime must be stopped before any removal actions can occur.

4. After the selected action is completed, click **Refresh Status** to see the time that processing was completed and the number of records removed.

   If the status indicates that an error occurred, correct the error and try the action again.
5. When the database record has been removed, restart the runtime by clicking **Go to Runtime Tuning Parameters to restart the runtime**.

Deploying an external runtime database

To optimize performance or increase storage capacity for the appliance, you can deploy an external runtime database. You can configure the appliance to connect to DB2®, PostgreSQL, or Oracle database on an external server.

About this task

Note: If Oracle is set as the external runtime database and either the local management interface or runtime server trace specification includes Oracle trace points (for example, `oracle.*`) the underlying Oracle JDBC jar file is changed to a debugging jar file. This might have adverse effects on performance and as such Oracle tracing should only be enabled for debugging purposes and disabled once complete.

Warning: Enabling trace for Oracle components “oracle.*” might result in the Oracle database administrator password being logged in clear text.

A Security Verify Access appliance with Advanced Access Control includes an internal database to store user data such as session attributes and device fingerprints. This embedded database is suitable for small environments. In a production environment, use an external runtime database that can handle the required volume of data.

The appliance provides scripts to deploy the runtime database on an external DB2, PostgreSQL, or Oracle server. You can then configure the appliance to use the external database.

Procedure

1. Use the **File Downloads** management page in the local management interface to access the runtime database deployment files for your environment.

   Table 8. Runtime database deployment scripts

Database type	Deployment scripts
DB2	`/access_control/database/db2/runtime/isam_access_control_db2.sql`
PostgreSQL	`/access_control/database/postgresql/runtime/isam_access_control_postgresql.sql`
Oracle	`/access_control/database/oracle/runtime/isam_access_control_oracle.sql`

2. Save the deployment script on the database server.
3. Run the DB2, PostgreSQL, or Oracle script to create the external database.

   PostgreSQL script

   Run the following command:

   ```
   psql --echo-all --variable ON_ERROR_STOP=1 --file <sql file name>
   --username <username> --host <host> --port <port> <database name>
   ```

   Oracle script

   a. Copy the downloaded `isam_access_control_oracle.sql` file into the Oracle home directory. For example, `ORACLE_HOME=/opt/oracle/app/oracle/product/11.2.0/dbhome_1`

b. Log in to SQL*Plus.
c. At the SQL prompt, run **`START isam_access_control_oracle.sql`**.

DB2 script

a. Create a DB2 instance to contain the runtime database. For information about creating the DB2 instance, see the DB2 documentation.
b. Open the `isam_access_control_db2.sql` file in an editor on the DB2 server.
c. Replace the following macros with the values specific to your environment:

&DBINSTANCE
The name of the DB2 instance.

&DBUSER
The name of the DB2 administrator.

&DBPASSWORD
The password for the DB2 administrator.

d. Save the changes.
e. Log in to the DB2 Command utility (Windows) or DB2 host (UNIX) as the DB2 administrator.
f. Run the following command:

```
db2 -tsvf <fully_qualified_path_to_script>
```

The following example shows the fully qualified path to the script:

```
db2 -tsvf /tmp/isam_access_control_db2.sql
```

4. Validate that the tables were successfully created.
5. Ensure that no errors were returned during the creation and log in to the database to manually check that the tables exist.
6. From the top menu of the local management interface, select **System** > **Cluster Configuration** to open the **Cluster Configuration** management page.
7. Select the **Database** tab.
8. You must enter the following JDBC connection information:

Type
The database type, which is either DB2, `PostgreSQL`, or `Oracle`.

Address
The IP address of the external database server.

Port
The port on which the external database server is listening.

Username
The name of the database administrator.

Password
The password for the database administrator.

DB2 also requires the following information:

Secure
Select this check box to create a secure connection with the server.

Note: Before a secure connection can be established, you must first import the certificate that the appliance uses to communicate with the server into the **`lmi_trust_store`** and **`rt_profile_keys`** key files. Use the **SSL Certificates** page to import the appropriate certificate.

Database name

The name of the database instance on the external DB2 server.

Complete the following steps to identify and specify the DB2 database name when your DB2 database is remote to the cluster that you are configuring.

a. Open the `isam_access_control_db2.sql` file that was used to create the database and tables.
b. In the **CREATE DATABASE** entry, get the name that is specified. In the following entry, HVDB is the string that identifies the default database name:

```
CREATE DATABASE HVDB ALIAS HVDB using codeset UTF-8 territory us
PAGESIZE 8192 WITH "HVDB Tables";
```

Note: *PAGESIZE 8192* is an example. Adjust according to your requirements.

PostgreSQL also requires the following information:

Secure
Select this check box to create a secure connection with the server.

Note: Before a secure connection can be established, you must first import the certificate that the appliance uses to communicate with the server into the **`lmi_trust_store`** and **`rt_profile_keys`** key files. Use the **SSL Certificates** page to import the appropriate certificate.

Database name
The name of the database instance on the external PostgreSQL server.

Oracle also requires the following information:

Secure
Select this check box to create a secure connection with the server.

Note: Before a secure connection can be established, you must first import the certificate that the appliance uses to communicate with the server into the **`lmi_trust_store`** and also a keystore which only contains public keys that needs to be created. Use the **SSL Certificates** page to create this keystore and to import the appropriate certificate.

Certificate Store
Choose the keystore which contains the certificate that will be used to communicate with the server.

Service name
Specify the name of the Oracle instance on the external server. Contact your Oracle database administrator for this information.

9. Click **Save**.
10. Deploy the changes.

Results

The appliance is configured to use the runtime database that is deployed on the external system.

General Information

HVDB data is language agnostic. Character support might not be an issue with regards to the installation that is chosen.

Oracle `DB_BLOCK_SIZE` or `PAGE_SIZE` can vary based on deployments but it is suggested to have at least 16384.

The suggested character set is ALUTF8.

User permissions for database must have read-write access and the ability to execute the commands in the SQL script.

Oracle:

```
dbca -createDatabase -templateName sampletemplate.dbc -gdbname hvdb -sid hvdb -responseFile
NO_VALUE -
characterSet AL32UTF8 -memoryPercentage 20 -emConfiguration LOCAL -dbsnmpPassword
```

```
mypassword -sysPassword
mypassword -systemPassword mypassword -silent
```

DB2:

```
CREATE DATABASE HVDB ALIAS HVDB using codeset UTF-8 territory us
PAGESIZE 8192 WITH "HVDB Tables";
```

What to do next

- Tune the external database by setting the configuration parameters. See .
- On Oracle 12.2 and latter, check that the supported login protocol is set on the DBMS. If it is not, set the value `SQLNET.ALLOWED_LOGON_VERSION=11` in the `sqlnet.ora` file. This attribute might be deprecated in later versions. For more information about this attribute and any new attributes, see https://docs.oracle.com/en/database/oracle/oracle-database/12.2/upgrd/required-tasks-complete-upgrading-oracle-database.html#GUID-12B920E9-B2DA-48A0-832C-3E07D172A011

Oracle Runtime database advanced connection methods

Provide more than one database host for external oracle runtime database.

When you are using an Oracle as the database platform for the high volume database, in some HA deployments it might be necessary to provide a custom URL, which contains information beyond what can be configured from the Verify Access LMI.

Set the advanced tuning parameter **`isam_cluster.hvdb.properties.URL`** to the value you are using to connect to the database. When this tuning parameter is set, the values provided for **Address**, **Port**, and **Service Name** in the LMI are ignored. The **Username** and **Password** are still used based on the configuration in the Cluster configuration panel.

For example,

```
jdbc:oracle:thin:@(DESCRIPTION = (ENABLE = BROKEN)(CONNECT_TIMEOUT = 1)
(TRANSPORT_CONNECT_TIMEOUT = 1)(RETRY_COUNT = 3)(ADDRESS_LIST =
(LOAD_BALANCE = ON)(FAILOVER = ON)(ADDRESS = (PROTOCOL = tcp)(HOST = host1-test)(PORT = 1521))
(ADDRESS = (PROTOCOL = tcp)(HOST = host2-test)
(PORT = 1521)))(CONNECT_DATA = (SERVICE_NAME = I4M_HVDB)))
```

Database usage requirements

Advanced Access Control uses the runtime database to store user data such as the attributes with which users are associated.

A range of storage space is available in the runtime database.

The average size per user and the maximum size per user are based on the default settings:

- There are 10 devices per user, and each device has 30 attributes.
- Each user has one session at a time.
- There are 1,000 records of behavior data per user.

Note: The administrator can configure custom database settings or use the default settings.

Space requirements for average-length attributes

100 KB is the average amount of space per user in the runtime database.

Your database can use more than 100 KB of space per user and still run effectively.

100 KB is the suggested amount of space for your database to use per user because databases that use 100 KB per user:

- Allow context-based access to run effectively and efficiently.
- Prevent wasted space in the database.

Do not use less than 100 KB of space per user. Databases that use less than 100 KB of space per user might run out of space. Databases that run out of space experience errors when context-based access runs.

Space requirements for maximum-length attributes

3 MB is the maximum amount of space that is possible per user in the runtime database if every attribute uses 2,000 bytes. 2,000 bytes is the maximum amount of space that each attribute can use.

Runtime database tuning parameters

To improve performance, you can configure the database tuning parameters for Advanced Access Control.

From the top menu, select **AAC** > **Global Settings** > **Advanced Configuration** to access these configuration parameters. Table 9 on page 128 contains the key parameters that you can use to tune the runtime database.

The runtime server creates five threads to clean up tables in the Runtime Database. In cluster environments or during high throughput periods, these threads can have a negative impact on database performance. To reduce this impact, several Advanced Configuration Parameters have been changed to no longer require a restart of the Runtime Server. This allows administrators to start or stop threads as needed without impacting service availability.

Note: There is a delay before the new thread frequency is applied. To use the new thread frequency immediately, a runtime restart is required.

For a full list of the available parameters, see Managing advanced configuration.

Table 9. Runtime database tuning parameters

Parameter	Description
attributeCollection.sessionTimeout	The appliance keeps the collected session data for context-based access in the runtime database tables. In an environment with a high volume of transactions, the tables build quickly. Consider the rate of transactions in your runtime environment to determine an appropriate timeout value. The default value of the **attributeCollection.sessionTimeout** parameter is 1800 seconds.
deviceRegistration.maxRegisteredDevices	The device registration process creates entries across numerous tables in the runtime database. This value limits the maximum number of devices that each user can register. A user can continue to register new devices until this maximum is reached. In a dynamic environment where every user has multiple devices, set this value to a number that represents a reasonable number of devices per user. To limit the volume of data in the database, do not use an excessive number. The default value of the **deviceRegistration.maxRegisteredDevices** parameter is 10.

Table 9. Runtime database tuning parameters (continued)	
Parameter	**Description**
deviceRegistration.maxUsageDataPerUser	The number of records each user can have in the runtime database table that holds usage data. If a new usage transaction is received after a user reaches this limit, the oldest record for the user is removed to accommodate the new data. That is, the system retains the most recent usage records for each user. In a large deployment, set this value to a number that retains the necessary usage records without overloading the table with unnecessary data. The default value of the **deviceRegistration.maxUsageDataPerUser** parameter is `200`.
distributedMap.getRetryDelay	The amount of time, in milliseconds, to wait before the appliance does another retrieval against the distributed map. In a cluster environment with failover support, you can use this value to cater for failover time. For example, **distributedMap.getRetryDelay** = `500`. **Note:** Increasing this value might result in longer response times. The default value of the **distributedMap.getRetryDelay** parameter is `0`.
distributedMap.getRetryLimit	The number of retrievals that are done against the distributed map before the appliance returns that the retrieved data is not in the distributed map. The default value is zero, which means that the retry is disabled. You can use this value with the **distributedMap.getRetryDelay** value to control the behavior of the appliance when it tries to retrieve data from the distributed map. In a cluster environment with failover support, you might want to permit multiple retrievals by setting a value such as 5. If there is network latency in the environment between cluster members, you can increase this number of retries along with the retry delay. **Note:** Increasing this value might result in longer response times. The default value of the **distributedMap.getRetryLimit** parameter is `0`.

Table 9. Runtime database tuning parameters (continued)	
Parameter	**Description**
`distributedMap.cleanupWait`	Specifies the interval, in milliseconds, that the Distributed Map clean-up thread runs to remove expired entries. If this thread is disabled there can be a delay of up to 30 seconds before it resumes. The database clean-up thread removes the following types of data: • Distributed map keys • Distributed map values The default value is 60000. A value of `-1` disables this clean-up thread. **Note:** This parameter does not require a restart of the runtime server before it is applied.
`session.dbCleanupInterval`	Specifies the interval, in seconds, that the database cleanup thread runs to remove expired data in the runtime database. If this thread is disabled, there can be a delay of up to 30 seconds before it resumes. The database cleanup thread removes the following types of expired data: • Session data • Device information • Obligation transaction data The default value of the **`session.dbCleanupInterval`** parameter is `86400`. A value of `-1` disables this clean-up thread. **Note:** This parameter does not require a restart of the runtime server before it is applied.
`authsvc.stateMgmt.HVDB.cleanupThread.batchSize`	The number of delete statements to execute in a single SQL commit of the `AUTH_SVC_SESSION_CACHE` table. Decreasing this value can increase the number of database calls required to remove expired sessions but reduces the time taken to execute each statement. The default value 0. A value of zero disables batching (removes all sessions in a single commit). **Note:** This parameter does not require a restart of the runtime server before it is applied.

Table 9. Runtime database tuning parameters (continued)	
Parameter	**Description**
authsvc.stateMgmt.HVDB.cleanupWait	Specifies the interval, in seconds, that the AuthSvc database clean-up thread runs to remove expired sessions in the runtime database. If this thread has been disabled, there can be a delay of up to 30 seconds before it resumes. The clean-up thread removes the following expired data for the authentication service: • Session data The default value is 120. This thread can be disabled by setting a value of -1. **Note:** This parameter is only valid if **authsvc.stateMgmt.cookieless** is `true`. This parameter does not require a restart of the runtime server before it is applied.
authsvc.stateMgmt.HVDB.lifetime	The length of time, in seconds, that authentication sessions will be cached when stateless operation (cookieless) is enabled. A larger value will cache sessions for longer, allowing users more time to complete an authentication challenge; at the cost of increasing database usage. The default value is 3600 seconds. **Note:** This parameter is only valid if **authsvc.stateMgmt.cookieless** is `true`. This parameter does not require a restart of the runtime server before it is applied.
authsvc.stateMgmt.HVDB.maxSessions	Control the maximum number of authentication service sessions which are cached in the Runtime Database. If the number of sessions exceeds this value, the oldest session in the cache is removed. The default value is 1000. This value should be significantly increased for production environments. **Note:** This parameter is only valid if **authsvc.stateMgmt.cookieless** is `true`. This parameter does not require a restart of the runtime server before it is applied.

Table 9. Runtime database tuning parameters (continued)

Parameter	Description
`mmfa.authenticator.cleanupWait`	Specifies the interval, in seconds, that the MMFA Authenticator clean-up thread runs. The clean-up thread removes MMFA Authenticators if the OAuth grant no longer exists, because it expired and was cleaned up, or was manually deleted. If this thread has been disabled, there can be a delay of up to 30 seconds before it resumes. The clean-up thread removes the following expired data: • OAuth Authenticators The default value is 3600. This thread can be disabled by setting a value of -1. **Note:** This parameter does not require a restart of the runtime server before it is applied.
`oauth20.tokenCache.cleanupWait`	Specifies the interval, in seconds, that the OAuth Token clean-up thread should run. If this thread has been disabled, there can be a delay of up to 30 seconds before it resumes. The clean-up thread removes the following expired data: • OAuth 2.0 Tokens • OAuth 2.0 Token extra attributes The default value is 120. This thread can be disabled by setting a value of -1. **Note:** This parameter does not require a restart of the runtime server before it is applied.
`oauth20.cleanupThread.batchSize`	The number of SQL delete statements to execute in a single commit of the `OAUTH20_TOKEN_CACHE` table. Decreasing this value can increase the number of database calls required to remove expired tokens, but reduces the time taken to execute each statement. The default value is 0. A value of 0 disables batching (removes all expired tokens in a single commit). **Note:** This parameter does not require a restart of the runtime server before it is applied.

Manual database clean-up

When you are using an external high volume database, the out-of-the-box periodic clean-up threads can be disabled in order to have fine control over when clean-up is performed.

For example, running clean-up only between the hours of 1am – 4am when traffic on the services is low.

Warning: Incorrect clean-up can cause data-loss and functional issues. It is **not** recommended unless a database admin (DBA) is involved.

There are five features that supports clean-up threads:

- Distributed Map
- Context-based access session
- OAuth Tokens
- Authentication service
- Mobile Multi-Factor Authenticator
- “Mobile Multi-Factor Transaction clean-up” on page 136

The periodic clean-up threads run on an interval that is set in advanced configuration settings. Use these settings to disable periodic cleanup. See Advanced configuration properties.

When you are performing the clean-up, it is essential that database locks are efficient to stop contention. The recommended pattern is to open a read-only cursor on the statement. It identifies rows to be deleted and reuses the cursor to perform the deletion one row at a time. This minimizes the size and duration of the write lock.

In order to achieve this pattern two queries are provided for each table:

1. A **select** pattern. This pattern returns a list of IDs that needs to be deleted.
2. A **delete** query.

How and when these two patterns are run is environment dependent. The discretion and direction of a database admin is recommended. It is also important to manage the commit frequency as some DBMSs have limits. For example, DB2. See How many concurrently running statements allowed for a DB2 Java application and how to increase it?. The database admin might be required to supplement some values from the built-in functions.

Distributed Map Clean-Up

When you are performing a distributed map clean-up, rows are deleted when their expiry is reached.

Manual clean-up queries

To disable the out-of-the-box clean-up, set *distributedMap.cleanupWait* to 0.

Select the rows to delete with:

```
SELECT DMAP_KEY as v1, DMAP_PARTITION as v2
FROM DMAP_ENTRIES
WHERE DMAP_EXPIRY < CURRENT_TIME_MILLIS
```

Variables that must be populated: *CURRENT_TIME_MILLIS* is an integer representing the current time in milliseconds.

Delete the rows with:

```
DELETE FROM DMAP_ENTRIES
WHERE DMAP_KEY = v1 and DMAP_PARTITION = v2
```

In the DELETE query above, the following variables must be substituted:

- *v1* corresponds to the DMAP_KEY that is selected in the previous query.
- *v2* corresponds to the DMAP_PARTITION that is selected in the previous query.

Context-based access clean-up

When performing context-based access clean-up, rows are deleted when their expiry is reached.

Manual clean-up queries

To disable the out-of-the-box cleanup, set *session.dbCleanupInterval* to 0.

Delete the records by using the following SQL queries:

```
1. DELETE FROM RBA_USER_ATTR_SESSION_DATA WHERE SESSION_ID IN (SELECT SESSION_ID FROM
RBA_USER_ATTR_SESSION WHERE REC_TIME < TIMESTAMP);
2. DELETE FROM RBA_USER_ATTR_SESSION WHERE REC_TIME < TIMESTAMP;
3. DELETE FROM AUTH_TXN_OBL_DATA WHERE REC_TIME < TIMESTAMP;
4. DELETE FROM RBA_DEVICE WHERE LAST_USED_TIME < TIMESTAMP;
```

Note:

1. These SQL statements are "pseudo" code examples only and are not specific to any DB type.
2. *TIMESTAMP* is a variable of the TIMESTAMP data type. Records created / last accessed prior to this timestamp will be removed. This can be a derived value calculated from the current date / time and an offset (e.g. Advanced Configuration properties *attributeCollection.sessionTimeout* for query #1 and #2 (in seconds) or *deviceRegistration.inactiveExpirationTime* for query #4 (in days)).
3. Depending on DB type, the format of *TIMESTAMP* may differ (e.g. date / time or epoch representation).
4. Where value comparisons occur, data types must match.
5. Due to a potentially large number of returned records, it may be necessary to limit the 'SELECT' statement in query #1 to a maximum of 1000 records at a time.

OAuth token clean-up

When performing clean-up on the `OAUTH20_EXTRA_ATTRIBUTE` table, a `STATE_ID` is deleted when it is not found in the `OAUTH20_TOKEN_CACHE` table. Entries are deleted from the `OAUTH20_TOKEN_CACHE` table when they’re expired.

Manual clean-up stories:

To disable the out-of-the-box clean-up, set *oauth20.tokenCache.cleanupWait* to 0.

Select the rows to delete with the following queries:

Query 1

```
SELECT TOKEN_ID as v1
FROM OAUTH20_TOKEN_CACHE
WHERE LIFETIME < (CURRENT_TIME_SECONDS - (DATE_CREATED / 1000))
Delete Statement1
DELETE FROM OAUTH20_TOKEN_CACHE
WHERE TOKEN_ID = v1
```

Query 2

```
SELECT STATE_ID as V1 FROM OAUTH20_TOKEN_CACHE WHERE OAUTH20_TOKEN_CACHE.STATE_ID =
OAUTH20_TOKEN_EXTRA_ATTRIBUTE.STATE_ID
```

Query 3

```
SELECT STATE_ID as V2 FROM OAUTH20_TOKEN_EXTRA_ATTRIBUTE WHERE NOT EXISTS V1
Delete Statement2
DELETE FROM OAUTH20_TOKEN_EXTRA_ATTRIBUTE WHERE STATE_ID = v2
```

Note: **`Delete Statement2`** can only be run after Query 2 and Query 3 are executed.

Example script

```
DELETE FROM OAUTH20_TOKEN_EXTRA_ATTRIBUTE (SELECT STATE_ID FROM OAUTH20_TOKEN_EXTRA_ATTRIBUTE
WHERE NOT EXISTS V1(SELECT * FROM OAUTH20_TOKEN_CACHE WHERE OAUTH20_TOKEN_CACHE.STATE_ID =
OAUTH20_TOKEN_EXTRA_ATTRIBUTE.STATE_ID ) FETCH FIRST batchSize ROWS ONLY)
```

Note: Replace *batchSize* with the maximum number of entries that are to be deleted with this query. For example, 5000.

The `CURRENT_TIME_MILLS` variable must be populated. It is an integer representing the current time in miliseconds.

Note: Query 3 must run inside the cursor that is opened with Query 2. The results of both Query 2 and 3 are used together in a subsequent delete.

v1 corresponds to the `TOKEN_ID` that is selected in Query 1.

v2 corresponds to the `STATE_ID` that is selected in Query 2.

Note: The appliance OAuth clean-up is adjusted in version 9.0.6.0 to be more efficient with database transactions.

Authentication service clean-up

The Authentication Service table is only populated when *authsvc.stateMgmt.store* is set to **HVDB**. When the clean-up is performed, rows are deleted when they expire.

Manual clean-up queries

To disable the out-of-the-box cleanup set *authsvc.stateMgmt.HVDB.cleanupWait* to 0.

Select the rows to delete with:

```
SELECT STATE_ID as v1
FROM AUTH_SVC_SESSION_CACHE
WHERE EXPIRY < CURRENT_TIME_MILLIS
```

The `CURRENT_TIME_MILLS` variable must be populated. It is an integer representing the current time in miliseconds.

Delete rows with:

```
DELETE FROM AUTH_SVC_SESSION_CACHE
WHERE STATE_ID = v1
```

where, *v1* is the `STATE_ID` that is selected in the prior query.

Mobile Multi-Factor Authentication (MMFA) clean-up

When performing cleanup on the OAUTH_AUTHENTICATORS table, entries are deleted when the STATE_ID is not found in the OAUTH20_TOKEN_CACHE.

Manual clean-up queries

To disable the out-of-the-box cleanup set *mmfa.authenticator.cleanupWait* to 0.

Select the rows to delete with the following queries:

Query 1

```
SELECT STATE_ID as v1
FROM OAUTH_AUTHENTICATORS
```

Query 2

```
SELECT STATE_ID as v2
FROM OAUTH20_TOKEN_CACHE
WHERE STATE_ID = v1
```

Delete rows with the following query:

Note: Deletes must only be performed when *v2* is null from the execution of query 2.

```
DELETE FROM OAUTH_AUTHENTICATORS
WHERE STATE_ID = v1
```

Mobile Multi-Factor Transaction clean-up

When performing clean-up on the `MMFA_AUTH_TXN_DATA` table, entries are deleted when the advanced configuration property `mmfa.transactionArchival.maxCompletedPerUser` is reached. The MMFA transactions are also archived to the audit log as they are deleted. However, manual clean-up will not be able to perform the archival to audit log step.

Manual clean-up queries

To disable the out-of-the-box cleanup, set `mmfa.transactionArchival.cleanupInterval` to 0.

Select the rows to delete with the following queries:

Query 1

```
SELECT DISTINCT USERNAME as v1 FROM MMFA_AUTH_TXN_DATA;
```

Query 2

```
SELECT COUNT(TXN_ID) as v2
FROM MMFA_AUTH_TXN_DATA WHERE USERNAME = v1
```

Query 3

```
SELECT TXN_ID as v3
FROM MMFA_AUTH_TXN_DATA
WHERE USERNAME = v1
ORDER BY DATE_LAST_USED ASC
LIMIT (v2–MAX_COMPLETED_PER_USER)
```

The `MAX_COMPLETED_PER_USER` variable must be populated. It is an integer representing the maximum amount of transactions to keep in the database per user.

Delete rows with the following query:

```
DELETE FROM MMFA_AUTH_TXN_DATA
WHERE TXN_ID = v3
```

Chapter 12. Policy information points

Policy information points gather information from the request or other sources, such as databases.

The appliance provides several policy information points that are configured to use data from the request. You can use the predefined attributes from these policy information points in your policy evaluations. For more information about predefined attributes, see Predefined attributes.

Note: You cannot delete or modify these preconfigured PIPs through the local management interface. However, you can modify a few settings for some of them with the advanced configuration properties. See Advanced configuration properties for details.

Session attribute PIP
Returns attributes that are related to session information, such as browser information and device characteristics.

GeoLocation attribute PIP
Returns geographic location attributes, such as the city and country code where the device is located.

Risk Calculator PIP
Returns the RiskScore attribute.

IP Reputation PIP
Returns the IP address reputation score.

User Fingerprint Count PIP
Returns the number of fingerprints that are registered for a user.

The appliance also supports a PIP that uses data from outside of the appliance. You must configure this PIP before you can use it and the attributes it returns. See “Managing policy information points” on page 138.

RESTful Web Service PIP
Returns attributes from data that is obtained from a RESTful web service that is hosted outside of the appliance. You can configure multiple instances of this PIP to access different web services.

JavaScript PIP
Returns attributes from data that is obtained from:

- WebSEAL or Web Reverse Proxy data such as HTTP headers or POST data in the request
- Other PIPs

The JavaScript PIP processes this unstructured data and parses it so that the administrator can use it to write authorization policies and risk policies.

Database PIP
Returns attributes from data that is hosted outside of the appliance by using **SQL SELECT** query statements. You can define information points for the following types of databases:

- DB2
- Oracle

You can configure more than one database policy information point instance so that different data sources can be accessed. Within the configuration, you define a query that can allow multiple attributes to be populated. You can then define a policy that relies on the custom attributes that you created.

LDAP PIP
Obtains attributes from a registry hosted outside of the appliance by using LDAP searches. For example, you might want to determine dynamically the credit limit for a user that triggers higher authentication requirements. To make such a determination, a customer directory or database is consulted. An LDAP PIP provides the following function:

- Multiple instances of a configuration are allowed so that different registries can be accessed.

- Multiple attributes can be populated from a single search.
- Support for Active Directory, IBM Security Director Server, Oracle Directory Server, and any LDAP v3 compliant server.

For SSL connections to the LDAP server, only server authentication is supported.

Fiberlink MaaS360 PIP
Enables the use of device attributes from registered devices in MaaS360 in access policies. Separate PIPs are available for browser-based web applications and MaaS360 SDK-based applications or wrapped apps. You can use either PIP to populate the MaaS360 attributes in access policy. For complete instructions on how to set up your appliance to integrate with Fiberlink MaaS360, see http://www.ibm.com/support/docview.wss?uid=swg24038325. The `.zip` file contains an integration guide PDF file.

Managing policy information points

To use data from sources outside of the appliance in your policies, you must add that source as a policy information point.

Before you begin

To use attributes from an external data source, you must add the attributes to the appliance before you add the policy information point. Use the steps for adding an attribute in “Managing attributes” on page 13.

To create a server connection policy information point, define the database server connection first. See “Managing server connections” on page 143.

Procedure

1. Log in to the local management interface.
2. Click **AAC**.
3. Under **Policy**, click **Information Points**.
4. Take one of the following actions:

   Filter policy information points:
   Complete these steps to filter policy information points:

   a. In the Quick Filter field, type one or more characters. For example, enter g to search for all policy information points names that contain g or G.
   b. Press Enter.

   Add a policy information point:
   Complete these steps to configure a policy information point:

   a. Click the drop-down button.
   b. Select the type:
      - **RESTful Web Service**
      - **JavaScript**
      - **Database**
      - **LDAP**
      - **FiberLink MaaS360**
   c. Complete the properties for the information point. See:
      - “RESTful web service PIP” on page 145
      - “JavaScript PIP ” on page 148
      - “Database PIP” on page 150

- “LDAP PIP” on page 151
- “Fiberlink MaaS360 PIP” on page 152

Modify a policy information point:

Complete these steps to modify a policy information point that you previously configured:

a. Select a policy information point.

b. Click .

c. Complete the properties for the policy information point. See:

- “RESTful web service PIP” on page 145
- “JavaScript PIP ” on page 148
- “Database PIP” on page 150
- “LDAP PIP” on page 151
- “Fiberlink MaaS360 PIP” on page 152

Delete a policy information point:

Complete these steps to remove a policy information point that you previously configured:

Attention: Do not delete a policy information point if it returns attributes that are used in a policy or risk score.

a. Select a policy information point.

b. Click .

View, import, or export a JavaScript file:

Note: These instructions apply only to JavaScript policy information points. You must select a JavaScript policy information point type to see the drop-down button.

Advanced Access Control provides the following JavaScript policy information points:

- Worklight Policy Information Point for Adapters
- Fiberlink MaaS360 Policy Information Point

View a JavaScript file that you previously configured as a policy information point:

a. Select a policy information point that is a JavaScript type.

b. Click the drop-down button.

c. Select **View**. A read-only version of the script displays in a pop-up window.

d. Click **Close** when finished.

Import a JavaScript file to replace an existing file for your policy information point:

a. Select a policy information point that is a JavaScript type.

b. Click the drop-down button.

c. Select **Import**. A pop-up window displays to import a script.

d. Click **Browse** to locate the JavaScript file.

e. Click **OK**. The syntax of the file is checked. If there are errors, you must fix them before you can import the file.

Export an existing JavaScript file:

a. Select a policy information point that is a JavaScript type.

b. Click the drop-down button.

c. Select **Export**.

d. Click **Save File**.

Server connection properties

To access a data source outside of the appliance, define the properties of the server.

The Server Connection properties table describes the properties on the **Server Connections** panel for the Advanced Access Control and Federation module activation levels.

- **Advanced Access Control**: Configure LDAP, database, web service, or Cloud Identity server connections so that you can set up policy information points. You can configure any of the server connection types.
- **Federation**: Configure an LDAP server as an attribute source for attribute mapping. Federation does not configure any of the other database server connection types.

Table 10. Server Connection properties

Property	**Description**
Name	Specifies the name for the server connection. Ensure that the name is unique. Select this name when you define the policy information point. **Note:** The server connection name must begin with an alphabetic character. Do not use control characters, leading and trailing blanks, and the following special characters ~ ! @ # $ % ^ & * ( ) + | ` = \ ; " ' < > ? , [ ] { } / anywhere in the name.
Description	Describes the server connection. This property is optional.
Type	Shows the server connection type. (Read only)
JNDI ID (Oracle, DB2, PostgreSQL only)	Specifies the JNDI ID that the server uses. Ensure that the ID is unique. Use only alphanumeric characters: a-b, A-B, 0-9
Set the connection by using the full URL (Oracle only)	Indicates that the connection to the database is set as the full JDBC URL.
Set the connection by using the server name and port (Oracle only)	Indicates that the connection to the database is set by specifying the server name and port.
Server name (Oracle, DB2, PostgreSQL, SMTP only)	Specifies the name or IP address for the server.
Port (Oracle, DB2, PostgreSQL, LDAP, SMTP, Redis only)	Specifies the port number where the connection to the server can be made.
URL (Oracle only)	Specifies the JDBC URL where the connection to the database can be made.
URL (Web Service only)	Specifies the URL where the connection to the server can be made.
Master Name (Redis-Sentinel only)	
User name (Oracle, DB2, PostgreSQL, SMTP, and Web Service only)	Specifies the user name that has the correct permissions to access the resources.
Password (Oracle, DB2, PostgreSQL, SMTP, and Web Service only)	Specifies the password to access the server.

Table 10. Server Connection properties (continued)	
Property	**Description**
SSL	Specifies whether SSL is used for connecting to the server. Select **True** or **False**. The default value is **True**.
Driver type (Oracle only)	Specifies the driver type. Select **Thin** or **OCI**. The default value is **Thin**.
Service name (Oracle only)	Specifies the name of the service.
Database name (DB2, PostgreSQL only)	Specifies the name of the database.
Host name (LDAP and Redis only)	Specifies the host name or IP address of the LDAP and Redis server. For Redis-Sentinel, select the **Servers** tab to add specify the servers.
Bind DN (LDAP only)	Specifies the LDAP distinguished name (DN) that is used when binding, or signing on, to the LDAP server. **Note:** If this value is set to `"anonymous"`, the appliance uses an anonymous bind to the LDAP directory server. Typically the **`bind-dn`** has significant privileges so that it can be used to modify LDAP registry entries, such as creating users and resetting passwords via pdadmin or the Registry Direct Java API. Using an anonymous connection to LDAP typically comes with very limited access, perhaps at most search and view of entries, at the least no access at all. If anonymous access has sufficient privileges, then it might be usable for the WebSEAL level of access on users and groups. This access includes the permission for a user to change password if `"bind-auth-and-pwdchg = yes"` is set (`"ldap.bind-auth-and-pwdchg = true"` for Registry Direct Java API).
Bind Password (LDAP only)	Specifies the password for the LDAP bind DN. **Note:** If bind DN (`bind-dn`) is set to `anonymous`, you can use any non-empty string as the value of bind password (`bind-pwd`).
Administration hostname (Cloud Identity only)	Specifies the administration hostname of the Cloud Identity subscription.
Client ID (Cloud Identity only)	Specifies the client ID of an API Client on Cloud Identity.
Client Secret (Cloud Identity only)	Specifies the client secret of an API Client on Cloud Identity.
SSL Truststore (LDAP, Web Service, Cloud Identity, and Redis only)	Specifies the truststore that verifies the credentials.
SSL Mutual Authentication Key (LDAP, Web Service, Cloud Identity, Redis only)	Label of the client certificate to be presented when connecting to the LDAP. This property is sourced from SSL Truststore. **Note:** This field is required only if mutual SSL authentication is required by the server.

Note: For information on SSL configuration, see Configuring SSL connections.

The properties in the following table are connection manager properties. The defaults that are listed are the current known defaults. All tuning properties are optional.

Table 11. Tuning properties

Property	Description
Aged timeout (seconds) (Oracle, DB2, PostgreSQL only)	Specifies the amount of time, in seconds, before a physical connection is discarded by pool maintenance. Specify `-1` to disable this timeout. The default is `-1`.
Connection timeout (seconds)	Specifies the amount of time, in seconds, after which a connection times out. For Oracle, DB2, PostgreSQL, and SMTP, specify `-1` to disable this timeout. The default is `30` seconds. For LDAP, specify only integers, `1` or greater. The default is `120` seconds. For Redis, the default is `10` seconds
Min Idle Size (Redis only)	Specifies the minimum number of established connections that must be kept in the pool.
Max Idle Size (Redis only)	Specifies the maximum number of established connections that must be kept in the pool.
Max Idle Time (seconds)	Specifies the maximum amount of time, in seconds, after which an unused or idle connection is discarded during pool maintenance. Specify `-1` to disable this timeout. The default is `1800` seconds.
Max Idle Time (seconds) (LDAP only)	Specifies the amount of time, in seconds, after which an established connection is discarded as idle. Set this to a value lower than the connection idle timeout on the LDAP server. **Note:** This is only applicable for performing Attribute Mapping from an LDAP server.
Reap time (seconds) (Oracle, DB2, PostgreSQL only)	Specifies the amount of time, in seconds, between runs of the pool maintenance thread. Specify `-1` to disable pool maintenance. The default is `180` seconds.
Max pool size (Oracle, DB2, PostgreSQL only)	Specifies the maximum number of physical connections for a pool. Specify `0` for unlimited. The default is `50`.
Max pool size (LDAP and Redis only)	Specifies the maximum number of connections that are pooled. **Note:** This is only applicable for performing Attribute Mapping from an LDAP server.
Min pool size (Oracle, DB2, PostgreSQL only)	Specifies the minimum number of physical connections to maintain in a pool. The aged timeout can override the minimum.

Table 11. Tuning properties (continued)	
Property	**Description**
Purge policy (Oracle, DB2, PostgreSQL only)	Specifies which connections to delete when a stale connection is detected in the pool. Select from the following options: **Entire pool** When a stale connection is detected, all connections in the pool are marked stale, and when no longer in use, are closed. This is the default option. **Failing connection only** When a stale connection is detected, only the connection that was found to be bad is closed. **Validate all connections** When a stale connection is detected, connections are tested and the ones that are found to be bad are closed.
Max connections per thread (Oracle, DB2, PostgreSQL only)	Specifies the limit of open connections on each thread.
Cache connections per thread (Oracle, DB2, PostgreSQL only)	Specifies the number of cache connections for each thread.
Idle Timeout (seconds) (Redis only	Specifies the amount of time, in seconds, after which an established connection is discarded as idle. The default is 1800 seconds.
IO Timeout (seconds) (Redis only)	Specifies the amount of time, in seconds, that the client waits for a response from the server, after an established connection, before it is discarded as idle.

Managing server connections

To use data from outside your appliance in your policies, you must define the server connection to access the data.

Before you begin

Obtain the connection information for the existing database server you want to define for your policy information point.

About this task

You can create server connections to data sources, such as Oracle, DB2, PostgreSQL, LDAP, SMTP, Web Service, Cloud Identity, and Verify Access Runtime. You can have multiple servers for an LDAP connection.

Note:

LDAP anonymous bind credentials:

username : anonymous password: anonymous

Procedure

1. Log in to the local management interface.
2. Click **AAC**.
3. Under **Global Settings**, click **Server Connections**.
4. Take one of the following actions:

Filter server connections:

a. In the Quick Filter field, type one or more characters. For example, enter g to search for all server connection names that contain g or G.

b. Press Enter.

Add a server connection:

a. Click the drop-down button.

b. Select **Oracle**, **DB2**, **PostgreSQL**, **LDAP**, **SMTP**, **Web Service**, **Cloud Identity** or **Verify Access Runtime**.

c. Complete the properties for the new server connection.

Modify an existing server connection:

a. Select a server connection.

b. Click .

c. Complete the properties for the server connection.

Delete a server connection:

Note: Do not delete a server connection if it returns attributes that are used in a policy or risk score.

a. Select a server connection.

b. Click .

5. For LDAP connections, take one of the following actions in the **Servers** tab.

Add a server connection:

a. Click the drop-down button.

b. Complete the properties for the new server connection.

Modify an existing server connection:

a. Click .

b. Complete the properties for the server connection.

Delete a server connection:

a. Select a server connection.

b. Click .

Move a server connection:

a. Select a server connection.

b. Click or .

What to do next

After you define a server connection to a data source, you can create a policy information point to access this data and use it in policies. See "Managing policy information points" on page 138.

RESTful web service PIP

When you add or modify a RESTful web service policy information point (PIP), you must specify its properties.

Connection properties

Name
A unique name for the policy information point. Use this name as the Issuer for the attributes that are returned by this policy information point.

Description
A description of the policy information point. (Optional)

Type
The type is **RESTful Web Service**. This field is read only.

URL
The URL for the RESTful web service that starts with `http` (plain-text) or `https` (secure HTTP). For example:

```
https://example.ibm.com/jaxrs/getApprovedAmount/
```

You can also dynamically create the URL by using the attribute values in a request at run time. The attribute that you use must match the name field of that attribute.

Attention: Do not use confuse the attribute name in the name field with the attribute identifier in the identifier field. Use the name that matches the name in the name field.

In the following example, the user name for the request is substituted in the URL at run time. The name of the attribute is `username`:

```
https://example.ibm.com/jaxrs/getApprovedAmount/{username}
```

In the following example, the user name and IP address for the request are substituted in the URL at run time. The attribute names are `username` and `ipAddress.`

```
https://example.ibm.com/jaxrs/getApprovedAmount/{username}/{ipAddress}
```

Attention: The server name in the URL value must match the `cn=` value in the SSL server certificate for the policy information point server. For the example, if the URL is `https://example.ibm.com/jaxrs/getApprovedAmount/`, then the SSL server certificate value must be `cn=example.ibm.com`.

Also, the `cn=` value in the server certificate must be the host name for the server, not the IP address.

Response Format
The format of the response as requested by the service through the URL. Select **XML**, **JSON**, or **Text**.

Media Type
The Accept header in the request. The default values correspond with the response formats:

- **application/json**
- **application/xml**
- **text/plain**

However, you can use any MIME type that you want to use.

Certificate Database
If `https` is used on the RESTful web service URL, specify the key database for the server SSL certificate. For example, `rt_profile_keys`

Client Authentication
If you require client authentication, select the type of authentication and its appropriate properties.

Basic Authentication

An authentication method that uses a user name and a password.

Attention: This property is valid only if the RESTful web service uses HTTPS.

Client Certificate

An authentication method that requires the client to present an SSL certificate. Specify the database that stores the certificate and the certificate label.

Attention: This property is valid only if the RESTful web service uses HTTPS.

Attributes properties

Attribute

The attributes that are retrieved from a response and that can be used in a policy or risk score. The values are mapped to the associated attributes. You can use one or more attributes, and you can add, modify, or delete attributes.

Attention:

- You must add the attributes to the appliance before you can use the attributes in this property.See the steps for adding an attribute in "Managing attributes" on page 13.
- Do not delete an attribute that is used in a policy or risk score.

Selector

XML

XPath 1.0 expressions are supported for XML selectors. Any valid XPath expression is supported.

Plain Text

The plain text selector is a delimiter. The response from the web service can be a single value or list of values for an attribute. The selector specifies the character that separates the values. For example:

- The selector is a comma (,)
- The response from the web service is `"LabA,LabB,LabC"`
- The returned attribute has three values: LabA, LabB, and LabC

Note: If you specify None for the delimiter, the RESTful web service policy information point returns the entire response as the attribute's value. If you do not specify a delimiter, the appliance defaults to None.

JSON

The JSON selector string. All attribute selectors must return either a primitive type or an array of primitive types. If the selector references complex types, a policy evaluation error occurs, and access to the system is denied.

JSON Selector format	Description

`$.x`	Returns the value for the property that is named x in the JSON object. Example:
	`{"name": "Bill", "loan":`
	`  {"amount":100, "rate":0.15, "duration":60},`
	`   "accounts":[10000, 2000, 500]}`
	`$.name` returns `'Bill'`.

<table>
<tr><th>JSON Selector format</th><th>Description</th></tr>
<tr><td>$.x....z</td><td>Returns the value for the property within a nested JSON object. Example:
<pre>
{"name": "Bill", "loan":
  {"amount":100, "rate":0.15, "duration":60},
   "accounts":[10000, 2000, 500]}
</pre>
$.loan.amount returns 100.</td></tr>
<tr><td>$.x[*]</td><td>Returns the array found at the specified property. Example:
<pre>
{"name": "Bill", "loan":
  {"amount":100, "rate":0.15, "duration":60},
   "accounts":[10000, 2000, 500]}
</pre>
$.accounts[*] returns [10000, 2000, 500]. The attribute is multivalued with each object in the array as a value.</td></tr>
<tr><td>$.[*]</td><td>Returns an array that is contained in the JSON response. The attribute is multivalued with each object in the array as a value.

Example:

If the data is ["joe", "bob", "ted"], $.[*] returns ["joe", "bob", "ted"]</td></tr>
<tr><td>$.[x]</td><td>Returns a value from a JSON array index, where $.[x] represents the index on the array of the value you want to return.

Example:

If the data is ["joe", "bob", "ted"], $.[1] returns ["bob"]</td></tr>
<tr><td>$.[*].x</td><td>Returns values from a property within an array of JSON objects.

Example:

If the data is
<pre>
[
      {"name":"joe", "phone":"555-1212"},
      {"name":"bill", "phone":"555-1213"},
      {"name":"ted", "phone":"555-1214"}
]
</pre>
$.[*].name returns ["joe", "bill", "ted"]</td></tr>
</table>

Cache Properties

Cache size
Specifies the maximum number of entries to keep in the cache

Cache entry lifetime
Specifies the lifetime of cache entries, in seconds.

JavaScript PIP

When you add or modify a JavaScript policy information point (PIP), you must define the properties to customize the PIP according to the needs of your environment.

General

Name
A unique name for the policy information point. Use this name as the Issuer for the attributes that are returned by this policy information point.

Description
A description of the policy information point. (Optional)

Type
The type is **JavaScript**. This field is read only.

Script
A file that contains the JavaScript rule. Type the file name or click **Browse** to select the file from your workstation.

JavaScript policy information point instance properties

The **Name** and **Value** properties are optional. They are passed into the JavaScript code so that the administrator can use them to configure how the code runs.

Fiberlink MaaS360 JavaScript PIP

You can configure a JavaScript PIP to call the Fiberlink MaaS360 rule file. This type of PIP is for MaaS360 SDK-based applications or wrapped applications, and it retrieves attributes from the MaaS360 device inventory for use in access policies.

Follow these general steps to set up the PIP:

1. Update the Fiberlink MaaS360 SDK application code to retrieve the device ID. Then, populate an HTTP request header with the value. For example:

   ```
   //Inside application HTTP connection methods
   ...
   String maas360DeviceCsn = MaaS360SDK.getContext().getDeviceCsn();
   DefaultHttpClient client = new DefaultHttpClient();
   HttpHost host = new HttpHost("isam.ibm.com", 443, "https");
   HttpGet httpget = new HttpGet("<path>");
   httpget.setHeader("x-fl-device-id", maas360DeviceCsn);
   ...
   MaaS360SDK.getEnterpriseGatewayService().proxy(client);
   ...
   //Handle response
   ...
   ```

2. Configure the web reverse proxy to populate the following attribute used by the PIP to retrieve the device attributes from MaaS360:

   ```
   [azn-decision-info]
   urn:ibm:security:fiberlink:maas360:device:ids = header:x-fl-device-id
   ```

3. Configure the appliance with Advanced Access Conrol to call the Fiberlink MaaS360 JavaScript PIP rule file, `maas360_pip_rule.js`. This rule file is provided with the integration download package.

For complete instructions on how to set up your appliance to integrate with Fiberlink MaaS360, see http://www.ibm.com/support/docview.wss?uid=swg24038325. The `.zip` file contains an integration guide PDF file and the rule file for this PIP.

Worklight JavaScript PIP

An appliance with Advanced Access Control can provide a JavaScript policy information point (PIP) for the Worklight product.

The Worklight JavaScript PIP is a prepopulated PIP that completes the following tasks:

- Decodes and parses the POST data that exists in a Worklight adapter request.
- Returns custom attributes that are created from the data that is contained within the POST data from the `parameters` element.

To support context-based authorization for Worklight adapters, you must create a custom PIP to URL-decode and tokenize form POST data.

Worklight uses the following HTTP POST format: `x-www-form-urlencoded`

For example:
`adapter=BankingApprovals&procedure=doTransfer&parameters=%5B%22200%22%2C%22sav%22%2C%22chq%22%5D` represents the following JSON data: `["200","sav","chq"]`

`["200","sav","chq"]` starts when $200.00 is transferred from a savings account to a checking account. In this example:

- The JavaScript PIP maps the POST data into the following corresponding Worklight attribute names: `worklight.adapter.adapter`, `worklight.adapter.procedure`, and `worklight.adapter.parameters`.
- The Worklight client calls the Worklight Adapter, which starts a procedure where the parameters are passed. In the example, the name of the Worklight adapter is `BankingApprovals`.
- The Worklight PIP then determines the attribute context of the `worklight.adapter.parameters`. This context is based on the `worklight.adapter.adapter` attribute and the `worklight.adapter.procedure` attribute.
- The JavaScript PIP maps the following values into their corresponding Worklight attributes:

  `parameters`
  : `worklight.adapter.parameters` is the Worklight attribute for `parameters`.

  `200`
  : `worklight.adapter.transfer.amount` is the Worklight attribute for `200`.

  `sav`
  : `worklight.adapter.transfer.account.from` is the Worklight attribute for `sav`.

  `chq`
  : `worklight.adapter.transfer.account.to` is the Worklight attribute for `chq`.

You can complete the following tasks for these PIPs by using the local management interface:

- Add, modify, and delete PIPs.
- View, import, and export JavaScript PIPs.

The sample Worklight JavaScript PIP and associated attributes on the appliance provide built-in support for the IBM Security Verify Access for Worklight integration sample application. See IBM Security Verify Access for IBM Worklight for more information

See Managing policy information points and JavaScript properties.

Trusteer Javascript PIP

This Extension installs a new JavaScript Policy Information Point which provides integration with IBM Security Trusteer Pinpoint Detect.

For more information, see IBM Security Verify Access Extension for IBM Security Trusteer Pinpoint Detect.

Database PIP

When you add or modify a database policy information point (PIP), you configure a connection to a data source. You also determine what information to use from the data source.

Connection properties

Name
: Identifies the policy information point instance. This name must be unique to the instance. Do not use a predefined Advanced Access Control policy information point issuer name.

The name that you create is the issuer for any attributes that the policy information point instance returns.

Description
: Describes the policy information point. (Optional)

Type
: Specifies the policy information point type, which is **Database**. (Read only)

Server Connection
: Specifies the database from which to retrieve the attributes. Select one of the defined databases from the list. If the database you require is not available to select in the list, you must define it. See "Managing server connections" on page 143.

Attribute properties

SQL Query
: Specifies the SQL **SELECT** statement that queries the database for information. You can use any valid SQL **SELECT** statement. You cannot add an attribute unless you enter a query statement in this field.

The format of the SELECT statement:

`SELECT COLNAME1, COLNAME2, ..., COLNAMEn FROM TABLE WHERE ...`

You can also dynamically create the query by using attribute values in a query at run time. The attribute that you use must match the name field of that attribute. In the following example, the user name for the request is substituted in the query at run time. The name of the attribute is `username`:

```
SELECT COLNAME1, COLNAME2, ..., COLNAMEn FROM TABLE WHERE ACCOUNT_HOLDER =
{username}
```

Note: You can specify only a single select statement when you configure the database policy information point. If you specify multiple SQL statements, an error message is returned. Do not end the statement with a semicolon.

Attribute
: Specifies the attributes that are retrieved from a response and that can be used in a policy or risk score. The database column is mapped to the associated attribute. You can use one or more attributes. You also can add, modify, or delete attributes.

Database Column
: Specifies the database column that maps to the attribute. Select it from the list of column names or type the name. The column names from the SQL **SELECT** query are used as the attribute selectors. For example, if you specify the following query:

```
SELECT ACCOUNT_BALANCE, ACCOUNT_NUMBER FROM ACCOUNTS WHERE
ACCOUNT_HOLDER_NAME = 'Joe Smith'
```

ACCOUNT_BALANCE and ACCOUNT_NUMBER are the column names to select from.

If your **SELECT** statement specifies a wildcard character, type the column name in this field.

Cache Properties

Cache size
Specifies the maximum number of entries to keep in the cache

Cache entry lifetime
Specifies the lifetime of cache entries, in seconds.

LDAP PIP

When you add or modify an LDAP policy information point (PIP), you configure a connection to an LDAP server. You also determine what information to use from the LDAP directory.

Connection properties

Name
Identifies the policy information point instance. This name must be unique to the instance. Do not use a predefined Advanced Access Control policy information point issuer name.

The name that you create is the issuer for any attributes that the policy information point instance returns.

Description
Describes the policy information point. (Optional)

Type
Specifies the policy information point type, which is **LDAP**. (Read only)

Server Connection
Specifies the LDAP server from which to retrieve the attributes. Select one of the defined LDAP servers from the list. If the server you require is not available to select in the list, you must define it. See “Managing server connections” on page 143.

Attribute properties

Base DN
Specifies the base DN of the directory server that determines where to search for attribute values. For example, you can specify `o=Example_Organization,c=us`.

Search filter
Specifies the search filter for the attribute values you require. Any LDAP search filter is supported. For example, specify `(|(objectclass=ePerson)(objectclass=Person))`. You can also dynamically create the search by using attribute values in a search at runtime. The attribute that you use must match the name field of that attribute. For example, `(&(cn={username})(|(objectclass=ePerson)(objectclass=Person)))`.

Search timeout (seconds)
Specifies the amount of time in seconds that is allowed for search operation before the LDAP server is considered to be down. The default is 120 seconds.

Attribute
Specifies the attributes that are retrieved from a response and that can be used in a policy or risk score. Each attribute is mapped to an associated LDAP registry attribute. You can use one or more attributes, and you can add, modify, or delete attributes.

The attributes that you add here must already be defined in the appliance local management interface. See “Managing attributes” on page 13 for information on adding an attribute.

Do not delete an attribute that is used in a policy or risk score.

Selector
Specifies the name of an LDAP registry attribute.

Cache Properties

Cache size
Specifies the maximum number of entries to keep in the cache

Cache entry lifetime
Specifies the lifetime of cache entries, in seconds.

Fiberlink MaaS360 PIP

When you add or modify a Fiberlink MaaS360 policy information point (PIP), you configure a connection to the Fiberlink MaaS360 web service API.

See the following link for complete integration information for this type of PIP: http://www.ibm.com/support/docview.wss?uid=swg24038325

Connection properties

Name
Identifies the policy information point instance. You must use `urn:ibm:security:fiberlink:maas360` to ensure that the MaaS360 attributes used in the access policy cause the PIP to be called.

Description
Describes the policy information point. (Optional)

Type
Specifies the policy information point type, which is **FiberLink MaaS360**. (Read only)

URL
Specifies the URL of the Maas360 host. For example: `https://services.fiberlink.com`

Username
Specifies the MaaS360 user name.

Password
Specifies the MaaS360 password.

App Access Key
Specifies the MaaS360 app access key. For example: `abcdef1234`

App Version
Specifies the MaaS360 app version. For example: `1`

App ID
Specifies the MaaS360 app ID. For example: `com.123456789.maas360`

Billing ID
Specifies the MaaS360 billing ID.

Platform ID
Specifies the MaaS360 platform ID.

Certificate Database
If `https` is used on the URL, specify the key database for the server SSL certificate. For example, specify `maas360_keys`.

Cache Properties

Cache size
Specifies the maximum number of entries to keep in the cache

Cache entry lifetime
Specifies the lifetime of cache entries, in seconds.

QRadar UBA PIP

When you add or modify IBM Security QRadar User Behavior Analytics (UBA) policy information point (PIP), you configure a connection to the QRadar UBA web service API.

Connection properties

Note: QRadar deployments are configured for TLS v1.2 connections.

Name
: Identifies the policy information point instance. You must use **`urn:ibm:security:qradar:uba`** to ensure that the QRadar UBA attribute that is used in the access policy cause the PIP to be called.

Description
: Describes the policy information point. (Optional)

Type
: Specifies the policy information point type, which is **QRadar User Behavior Analytics**. (Read only)

URL
: Specifies the URL of the QRadar Console Hostname/IP address. For example: `https://console.qradar.com`

Application ID
: Specifies the application ID of QRadar User Behavior Analytics application. For example: 1234

SEC Token
: Specifies the QRadar User Behavior Analytics security token as configured by QRadar admin. For example: `912feaf8-fdab-476f-a2a7-a618756c46fd`

Polling Interval
: Specifies the time interval, in minutes, when the QRadar UBA server will be polled for risk scores for the users. Default: `2 minutes`

Certificate Database
: If HTTPS is used on the URL, specify the key database for the server SSL certificate. For example, `rt_profile_keys`. For more information about importing SSL certificates to the IBM Security Verify Access data store, see Managing SSL certificates.

Chapter 13. Extensions

Use extensions so that you can implement your own obligation handler and authentication mechanism.

Obligation handler
The obligation handler extension point allows an application developer to handle obligations at the policy decision point before it returns the result to the policy enforcement point application.

For more information about managing obligations in the local management interface, see "Managing obligations" on page 59.

Authentication mechanism
The authentication mechanism determines the conditions that successfully authenticate a user.

For more information about managing authentication mechanisms in the local management interface, see "Managing authentication mechanisms" on page 86.

For more information about the Software Development Kit, see Software Development Kit.

Managing extensions

Deploy your extensions to the appliance so that you can create an instance of it and use the instance in your policies. Extensions are contained in bundles. You can import, replace, export, and delete bundles that contain extensions.

Before you begin

Consider the following restrictions:

- You cannot delete a bundle that has extensions that are used by a policy.
- If you are replacing a bundle that contains extensions that are used by a policy, make sure that the new bundle:
  - Has the same file name as the old bundle
  - Contains the same exact extensions that are used by a policy
  - Contains the same exact properties as the original extensions
- If you are replacing a bundle that contains extensions that are not used by a policy, make sure that the new bundle has the same file name as the old bundle.

Procedure

1. Log in to the local management interface.
2. Click **AAC**.
3. Under **Policy**, click **Extensions**.
4. Take one of the following actions:

   Search for and view a bundle
   Take any of the following actions to filter your view:

   - Type one or more characters in the **Filter** field. The list displays the bundles that start with those characters.
   - Sort the list by column with the up or down arrow on each column. For example, you can view the list of extension IDs in ascending order by clicking the up arrow.

   Import a bundle

   a. Click **Import**.
   b. Click **Browse** to locate the bundle.

c. Click **Import**. The syntax of the file is checked. If there are errors, you must fix them before you can import the file.

Replace a bundle

a. Click **Import**.

b. Click **Browse** to locate the bundle.

c. Click **OK** when you are prompted that the bundle exists.

Export a bundle

a. Select the bundle that you want to export.

b. Click **Export**.

c. Save the file.

Delete a bundle

a. Select the bundle that you want to delete.

b. Click **Delete**. A message prompts you to confirm the deletion.

c. Click **Delete**.

5. Save and deploy your changes. For more information, see Chapter 14, “Deploying pending changes,” on page 157.

What to do next

- If you imported a custom authentication mechanism, see “Managing authentication mechanisms” on page 86.
- If you imported a custom obligation, see “Managing obligations” on page 59.

Chapter 14. Deploying pending changes

Some configuration and administration changes require an extra deployment step.

About this task

When you use the graphical user interface on the appliance to specify changes, some configuration and administration tasks take effect immediately. Other tasks require a deployment step to take effect. For these tasks, the appliance gives you a choice of deploying immediately or deploying later. When you must make multiple changes, you can wait until all changes are complete, and then deploy all of them at one time.

When a deployment step is required, the user interface presents a message that says that there is an undeployed change. The number of pending changes is displayed in the message, and increments for each change you make.

Note: If any of the changes require the runtime server to be restarted, the restart occurs automatically when you select **Deploy**. The runtime server will then be unavailable for a period of time until the restart completes.

Procedure

1. When you finish making configuration changes, select **Click here to review the changes or apply them to the system**.

   The **Deploy Pending Changes** window is displayed.
2. Select one of the following options:

Option	Description
Cancel	Do not deploy the changes now. Retain the undeployed configuration changes. The appliance user interface returns to the previous panel.
Roll Back	Abandon configuration changes. A message is displayed, stating that the pending changes were reverted. The appliance user interface returns to the previous panel.
Deploy	Deploy all configuration changes. When you select **Deploy**, a system message is displayed, stating that the changes were deployed. If any of the changes require the runtime server to be restarted, the restart occurs automatically when you select **Deploy**. The runtime server will then be unavailable for a period of time until the restart completes.

Chapter 15. Template files

Template files are HTML pages that are presented to your users during the authentication process. The pages prompt users for authentication information, such as user names and passwords, or present information to users, such as one-time passwords, status, or errors.

You can customize any of the HTML pages by exporting, modifying, and importing its corresponding template file. Each template file uses one or more specific macros.

Managing template files

Use the local management interface to manage files and directories in the template files.

About this task

You can run the following tasks on the template files:

- **New**- Use this option if you want to create a new file or directory.
- **Edit**- Use this option if you want to view or modify the template file.
- **Import**- Use this option if you to import a file to the template files root.
- **Export**- Use this option if you want to export a file from the template files root.
- **Rename**- Use this option if you want to rename a file or directory from the template files root.
- **Delete**- Use this option if you want to delete a file or directory from the template files root.
- **Import Zip**- Use this option if you want to import the template files from a compressed file.
- **Export Zip**- Use this option if you want to export the template files as a compressed file.

  Note: When you use this option to export template files as a compressed file, you cannot export an individual folder. The root directory, including all the sub-directories, is exported. To access the contents of a specific directory, export the entire template directory, and then view the directory from the extracted compressed file on your local workstation. Administrators can refer to `metadata.json` under file downloads after upgrades to check if there are new configuration parameters included for AAC related endpoints.

Procedure

1. Select **AAC** > **Global Settings** > **Template Files**
2. Work with all the management files and directories.

   Create a file or directory in the template files root

   a. Select the directory of interest or make no selection to create a directory at the root level.
   b. Select **New**.
   c. Select **File** or **Directory**.
   d. Enter the name of the file or directory.
   e. Click **Save**.

   View or update the contents of a file in the template files root

   a. Select the file of interest.
   b. Select **Edit**. You can then view the contents of the file.
   c. Edit the contents of the file.
   d. Click **Save**.

Export a file from the template files root

a. Select the file of interest.
b. Select **Manage** > **Export**.
c. Confirm the save operation when your browser displays a confirmation window.

Rename file from the template files root

a. Select the file or directory of interest.
b. Select **Manage** > **Rename**.
c. Enter the new resource name.
d. Click **Save**.

Delete file from the template files root

a. Select the file or directory of interest.
b. Select **Manage** > **Delete**.
c. Click **Yes**.

Import a file to the template files root

Note: In order to support a new locale, the top level directory name within the compressed file must correspond to the locale identifier. For example, to add support for Swedish, the top level directory name must be `/sv` or `/sv_SE`.

- Select a file.
  a. Select **Manage** > **Import**.
  b. Click **Browse**.
  c. Browse to the file that you want to import the contents from.
  d. Click **Open**.
  e. Click **Import**.
- Select a folder.
  a. Select **Manage** > **Import**.
  b. Click **Browse**.
  c. Browse to the file that you want to import to the selected folder.
  d. Click **Open**.
  e. Click **Import**.

Export the template file as a compressed file

a. Select **Manage** > **Export Zip**.
b. Confirm the save operation when your browser displays a confirmation window.

Import the template files as a compressed file

Make sure that the files in the compressed file are in the same directory structure as the files in the root directory or appliance.

For example, if a file in the compressed file is in the `/C` directory of the appliance, the compressed file must contain the C folder and the file that you want to import. When you import a compressed file that contains:

- A file that exists in the appliance

  The file is replaced in the appliance.
- A file or directory that does not exist in the appliance

  The file or directory is created in the appliance. You can put these new files and directories in an existing non-root directory or add a new directory in the root.

Note: You cannot delete a top level directory after you create it.

a. Select **Manage** > **Import Zip**.

b. Click **Browse**.

c. Browse to the file you want to import.

d. Click **Open**.

e. Click **Import**.

3. When you edit or import template files, the appliance displays a message that there are undeployed changes. If you finish the changes, deploy them.

   For more information, see Chapter 14, "Deploying pending changes," on page 157.

Template files reference

Template files are HTML pages that are presented to your users during the authentication process. The pages prompt users for authentication information, such as user names and passwords, or present information to users, such as one-time passwords, status, or errors.

Consent to register device template files

These files support consent to registering a device.

Consent to register device template files

These files support consent to registering a device.

Table 12. Default template files in the ac/ directory

Page name	File name and macros	Description
Attribute Collection JavaScript	`ac/info.js`	Detects the location of the device from which the requests are made. Collects the web browser attributes and sends them to the server for storing in the database. When the user logs out or when the current session times out, the script deletes the attributes from the database.
Dynamics Attributes JavaScript	`ac/javascript_rules/dynamic_attributes.js`	Runs after each request is processed by risk engine. Use it to capture attributes that do not get captured automatically. Captured attributes are stored either in the session storage or the behavior storage area of the risk-based component, or both. The risk profile configuration dictates where the attributes are stored.

User self-care template files

These files support user self-care tasks.

User self-care template files

These files support user self-care tasks.

Table 13. Default template files in the mga/ directory

Page name	File name and macros	Description
Common User Profile Management JavaScript	`mga/user/mgmt/common.js`	Used by one-time password pages and by device management pages. Contains functions and properties that are used for making calls to the user self-care REST services.
Device Attributes	`mga/user/mgmt/device/device_attributes.html`	Enables or disable devices. Renames or removes device. Displays all of the attributes for a device. For more information, see "Managing your registered devices" on page 209.
Device Attributes JavaScript	`mga/user/mgmt/device/device_attributes.js`	Processes values that are entered in the `device_attributes.html` template
Device Selection	`mga/user/mgmt/device/device_selection.html`	Displays device name, status (enabled or disabled), and time of last activity. For more information, see "Managing your registered devices" on page 209.
Device Selection JavaScript	`mga/user/mgmt/device/device_selection.js`	Processes data to display in the `device_selections.html` template
Authorization Grant	`mga/user/mgmt/device/grant_attributes.html`	Enables or disables an OAuth 2.0 authorization grant. Removes an OAuth 2.0 authorization grant. Displays the OAuth 2.0 tokens and attributes of an authorization grant. For more information, see Managing OAuth 2.0 Authorization Grants.
Authorization Grants JavaScript	`mga/user/mgmt/device/grant-attributes.js`	Processes data to display in the `grant_attributes.html` template.

Table 13. Default template files in the mga/ directory (continued)		
Page name	**File name and macros**	**Description**
HMAC OTP Shared Key	`mga/user/mgmt/otp/otp.html`	Resets TOTP and HOTP Secret Key. For more information, see "Managing OTP secret keys" on page 210.
HMAC OTP Shared Key JavaScript	`mga/user/mgmt/otp/otp.js`	Resets TOTP and HOTP Secret Key.
Knowledge Questions management	`mga/user/mgmt/questions/user_questions.html` Macros: • **@USERNAME @** • **@MAX_STORED_QUESTIONS@**	Displayed for the user to manage their knowledge questions. The user can select pre-configured questions or write their own questions.
Knowledge Questions JavaScript functions	`mga /user/mgmt/questions/user_questions.js`	Consists of the JavaScript functions that: • Display the knowledge questions. • Allow the user to manage their knowledge questions.

Authentication process

These files support the authentication process

Authentication process template files

These files support the authentication process. For more information, see Authentication.

Table 14. Default template files in the authsvc/ directory		
Page name	**File name and macros**	**Description**
Server Error	`authsvc/server_error.html` Macros: • **@REQ_ADDR@** • **@TIMESTAMP@** • **@EXCEPTION_MSG@** • **@EXCEPTION_STACK@**	Displays general server errors.
User Error	`authsvc/user_error.html` Macros: • **@REQ_ADDR@** • **@TIMESTAMP@** • **@EXCEPTION_MSG@** • **@EXCEPTION_STACK@**	Displays errors during authentication policy execution that are caused by user input.

Authentication mechanisms

These files support the authentication mechanisms.

Authentication mechanisms

These files support the authentication mechanisms. For more information, see Authentication.

Table 15. Default template files in the `otp/` directory

<table>
<tr><th>Page name</th><th>File name and macros</th><th>Description and link to file contents</th></tr>
<tr><td>Change PIN required</td><td>

`otp/change_pin.html`

Macros

- **`@ERROR_MESSAGE@`**
- **`@MAPPING_RULE_DATA@`**
- **`@DISPLAY_RESELECT_BUTTON@`**

</td><td>Enables the user to enter a new PIN.</td></tr>
<tr><td>OTP Email Delivery Message</td><td>

`otp/delivery/email_message.xml`

</td><td>

Used by `EmailOTPDelivery` as the content of the email that it sends to the user.

The template file must be a compliant XML file.

The content can be plain text or HTML. Following is an example that uses HTML in the email template:

```
<?xml version="1.0"
encoding="UTF-8"?>
<root>
<Subject>
  <Value>One-time Password</
Value>
</Subject>
<Message>
  <Value><![CDATA[<html>
<body>
<img src="https://
www.example.com/images/
logo.gif" />
<br />This is your HTML
email one-time password
@OTP_STRING@.<br />
    <p>Thank you,<br />
    The Example Co.</p>
</body>
    </html>]]>
  </Value>
</Message>
</root>
```

For more information about HTML formatting of email messages, see HTML format for OTP email messages.

</td></tr>
</table>

Table 15. Default template files in the otp/ directory (continued)		
Page name	**File name and macros**	**Description and link to file contents**
OTP SMS Delivery Message	otp/delivery/sms_message.xml	Used by SMSOTPDelivery as the content of the SMS that it sends to the user. The template file must be a compliant XML file.
One-Time Password Delivery Selection	otp/delivery_selection.html Macros • **@OTP_METHOD_CHECKED@** • **@OTP_METHOD_LABEL@**	Displays the list of methods for generating, delivering, and verifying the one-time password.
OTP General Error	otp/errors/allerror.html Macros • **@REQ_ADDR@** • **@TIMESTAMP@** • **@DETAIL@** • **@EXCEPTION_STACK@**	Displays general errors that happen during the one-time password flow.
OTP Validation Error	otp/errors/error_could_not_validate_otp.html Macros • **@REQ_ADDR@** • **@TIMESTAMP@** • **@DETAIL@** • **@EXCEPTION_STACK@**	Displays errors during the validation of the one-time password that the user submits.
OTP Generation Error	otp/errors/error_generating_otp.html Macros • **@REQ_ADDR@** • **@TIMESTAMP@** • **@DETAIL@** • **@EXCEPTION_STACK@**	Displays errors during the generation of a one-time password.
OTP Methods Retrieval Error	otp/errors/error_get_delivery_options.html Macros • **@REQ_ADDR@** • **@TIMESTAMP@** • **@DETAIL@** • **@EXCEPTION_STACK@**	Displays errors during the retrieval of the list of methods for delivering one-time password to the user.

Table 15. Default template files in the otp/ directory (continued)

Page name	File name and macros	Description and link to file contents
OTP Delivery Error	otp/errors/error_otp_delivery.html Macros • **@REQ_ADDR@** • **@TIMESTAMP@** • **@DETAIL@** • **@EXCEPTION_STACK@**	Displays errors during the delivery of a one-time password to the user.
OTP STS Invocation Error	otp/errors/error_sts_invoke_failed.html Macros • **@REQ_ADDR@** • **@TIMESTAMP@** • **@DETAIL@** • **@EXCEPTION_STACK@**	Displays errors during the invocation of the Security Token Service.
One-Time Password Login	otp/login.html Macros • **@ERROR_MESSAGE@** • **@MAPPING_RULE_DATA@** • **@DISPLAY_RESELECT_BUTTON@**	Displays the form where the user can enter the one-time password.
Enter Next OTP Form	otp/next_otp.html Macros • **@ERROR_MESSAGE@** • **@MAPPING_RULE_DATA@** • **@DISPLAY_RESELECT_BUTTON@**	Enables the user to enter the next one time password.

Table 16. Default template files in the authsvc/authenticator directory

Page name	File name and macros	Description
Authenticator Error Page	authsvc/authenticator/error.html Macros • **@ERROR_MESSAGE@** • **@JUNCTION@** • **@ACTION@**	A generic authenticator error page.

Table 16. Default template files in the `authsvc/authenticator` directory (continued)

Page name	File name and macros	Description
Authenticator Login Page	`authsvc/authenticator/login.html` Macros • **`@USERNAME@`** • **`@JUNCTION@`** • **`@ACTION@`**	A generic authenticator username and password login page.

Table 17. Default template files in the `authsvc/authenticator/basicldapuser` directory

Page name	File name and macros	Description
Change Password	`authsvc/authenticator/basicldapuser/change_password.html` Macros • **`@ERROR_MESSAGE@`** • **`@JUNCTION@`** • **`@USERNAME@`** • **`@ACTION@`** • **`@OLDPASSWORD@`** • **`@POLICYMESSAGE@`**	Enables the user to change their LDAP password.
Username and Password Login	`authsvc/authenticator/bascildapuser/login.html` Macros • **`@ERROR_MESSAGE@`** • **`@JUNCTION@`** • **`@ACTION@`**	Displays the form where the user can enter their username and password to log in.

Table 18. Default template files in the `authsvc/authenticator/branching` directory

Page name	File name and macros	Description
Generic Decision	`authsvc/authenticator/branching/generic_decision.html` Macros • **`@BRANCHES@`** • **`@JUNCTION@`** • **`@ACTION@`**	Displays a form that allows the user to select an authentication branch to use from a list of available branches.

Page name	File name and macros	Description
Second Factor Decision	authsvc/authenticator/branching/second_factor_decision.html • **@MECHANISMS@** • **@ERROR_MESSAGE@** • **@METHODS@** • **@JUNCTION@** • **@MOBILE_NUM@** • **@EMAIL_ADDR@** • **@ACTION@**	Displays a form that allows the user to select a second factor authentication method to use from a list of available methods.
Identifier First Authentication Page	authsvc/authenticator/branching/identifier_first.html Macros: • **@FIDO_RP_ID@** • **@FIDO_TIMEOUT@** • **@FIDO_CHALLENGE@** • **@FIDO_USER_VERIFICATION@** • **@FIDO_STATUS@** • **@FIDO_ERROR_MESSAGE@** • **@STATE@** • **@ACTION@**	Displays a form that allows the user to submit their username or perform autofill FIDO authentication.
IFA Authentication Method Choice Page	authsvc/authenticator/branching/ifa_choice.html • **@ERROR_MESSAGE@** • **@IS_FIDO@** • **@IS_MMFA@** • **@FINGERPRINT_PREFERRED@** • **@STATE@** • **@ACTION@** • **@USERNAME@**	If enrolled, the user is prompted to choose between FIDO2/WebAuthn authentication, MMFA authentication, or standard username/password authentication.

Table 18. Default template files in the authsvc/authenticator/branching directory (continued)

Table 18. Default template files in the *authsvc/authenticator/branching* directory (continued)		
Page name	**File name and macros**	**Description**
IFA FIDO2 PAIR Registration Page	`authsvc/authenticator/branching/ifa_fido2pair_reg.html` Macros: • **@FIDO_ERROR_MESSAGE@** • **@STATE@** • **@FIDO_USER_NAME@** • **@ACTION@** • **@ERROR_MESSAGE@** • **@FIDO_INFOMAP_PARAM@** • **@CANCEL_ACTION@**	Displays a page that allows a user to register a FIDO2 device for authentication.
IFA Redirect Page	`authsvc/authenticator/branching/ifa_redirect.html` • **@ERROR_MESSAGE@** • **@IFA_REDIRECT_URL@** • **@USERNAME@**	Redirects the user to the configured Redirect URL, if the username matched a set regular expression.

Table 19. Default template files in the *authsvc/authenticator/ci* directory		
Page name	**File name and macros**	**Description**
Username and Password Login	`authsvc/authenticator/ci/login.html` Macros • **@USERNAME@** • **@JUNCTION@** • **@ACTION@**	Displays the form where the user can enter their username and password to log in.
Authenticate Page	`authsvc/authenticator/ci/authenticate_dialog.html` Macros • **@HIDE_TRANSIENT_IF_ENROLL@** • **@EXPAND_VERIFY_METHODS@** • **@JIT_ENROLLMENT@** • **@AUTH_METHODS@** • **@SIGNATURE_METHODS@** • **@TRANSIENT_METHODS@** • **@JUNCTION@** • **@ACTION@**	Displays a form that allows the user to select a method to use for two-step verification.

Table 19. Default template files in the `authsvc/authenticator/ci` directory (continued)

Page name	File name and macros	Description
Choose Method Page	`authsvc/authenticator/ci/choose_method.html` Macros • **@AUTH_METHODS@** • **@SIGNATURE_METHODS@** • **@TRANSIENT_METHODS@** • **@ACTION@**	Displays a form that allows the user to select a method to use for two-step verification.
Verify Page	`authsvc/authenticator/ci/verify.html` Macros • **@TYPE@** • **@ERROR_MESSAGE@** • **@CORRELATION@** • **@JUNCTION@** • **@ACTION@** • **@ID@**	Displays a form that allows the user to enter the access code that is used to authenticate the two-step verification.
Device Connected Page	`authsvc/authenticator/ci/device_connected.html` Macros • **@TYPE@** • **@DEVICE_NAME@** • **@JUNCTION@** • **@ACTION@**	Displays a confirmation page that a device was successfully connected and can now be selected as a method for two-step verification.
Enrollment Page	`authsvc/authenticator/ci/enrollment.html` Macros • **@TYPE@** • **@CORRELATION@** • **@JUNCTION@** • **@ACTION@**	Displays a form that is used to validate an OTP during an enrollment flow.
Error Page	`authsvc/authenticator/ci/error.html` Macros • **@ERROR_MESSAGE@** • **@JUNCTION@** • **@ACTION@**	Displays errors during the CI authentication.

Page name	File name and macros	Description
JIT Enroll Page	`authsvc/authenticator/ci/jit_enroll.html` Macros • **@ERROR_MESSAGE@** • **@JIT_TYPE@** • **@JUNCTION@** • **@ACTION@**	Displays a form that allows a user with no existing device registrations to enroll a device during a CI authentication.
TOTP Enrollment Page	`authsvc/authenticator/ci/totp_enrollment.html` Macros • **@ERROR_MESSAGE@** • **@QRCODE@** • **@JUNCTION@** • **@ACTION@**	Displays a page that allows a user to enroll a device for TOTP authentication.
Push Notification Page	`authsvc/authenticator/ci/try_push.html` Macros • **@DEVICE_NAME@** • **@JUNCTION@** • **@ACTION@**	Displays a page that allows a device to be configured and tested to receive push notifications to authenticate the two-step verification.
User Self-Care Page	`authsvc/authenticator/ci/usc.html` Macros • **@ENABLED_METHODS@** • **@NAME@** • **@AUTH_METHODS@** • **@AUTHENTICATORS@** • **@EMAIL@** • **@USERNAME@** • **@DEVICE_COUNT@** • **@METHOD_COUNT@** • **@JUNCTION@** • **@ACTION@**	Displays a page that allows a user to manage their registered authentication devices.
Username Login Page	`authsvc/authenticator/ci/username.html` Macros • **@USERNAME@** • **@JUNCTION@** • **@ACTION@**	Displays the form where the user can enter their username to log in without a password.

Table 19. Default template files in the `authsvc/authenticator/ci` directory (continued)

Page name	File name and macros	Description
Verify Registration Page	`authsvc/authenticator/ci/verify_registration.html` Macros • **@QRCODE@** • **@ERROR_MESSAGE@** • **@JUNCTION@** • **@ACTION**	Displays a page that allows a user to enroll a device to use IBM Verify for authentication.
Wait Page	`authsvc/authenticator/ci/wait.html` Macros • **@ACTION**	Displays a page while it waits between polls during an IBM Verify authentication.

Table 19. Default template files in the `authsvc/authenticator/ci` directory (continued)

Table 20. Default template files in the `authsvc/authenticator/email_message` directory

Page name	File name and macros	Description
Email Delivery Message	`authsvc/authenticator/email_message/email_message.xml` Macros • @ATTRIBUTE@	The email content that is sent to the SMTP server.
Error Page	`authsvc/authenticator/email_message/error.html` Macros • **@ERROR_MESSAGE@** • **@JUNCTION@** • **@REQ_ADDR@** • **@TIMESTAMP@** • **@EXCEPTION_MSG@** • **@EXCEPTION_STACK@**	The email content that is sent to the SMTP server.

Table 21. Default template files in the `authsvc/authenticator/fido` directory

Page name	File name and macros	Description
Assertion Page	`authsvc/authenticator/fido/assertion.html` Macros • **@FIDO_RP_ID@** • **@FIDO_TIMEOUT@** • **@FIDO_CHALLENGE@** • **@FIDO_USER_VERIFICATION@** • **@FIDO_USER_ID@** • **@FIDO_STATUS@** • **@FIDO_ERROR_MESSAGE@** • **@STATE@** • **@ACTION@** • **@FIDO_ALLOW_CREDENTIALS@** • **@FIDO_EXTENSIONS@** • **@ERROR_MESSAGE@** • **@FIDO_INFOMAP_PARAM@** • **@CANCEL_ACTION@**	Displays a form that allows a user to perform a FIDO authentication.

Table 21. Default template files in the `authsvc/authenticator/fido` directory (continued)

Page name	File name and macros	Description
Attestation Page	`authsvc/authenticator/fido/attestation.html` Macros • **@FIDO_RP_ID@** • **@FIDO_RP_NAME@** • **@FIDO_USER_ID@** • **@FIDO_USER_NAME@** • **@FIDO_DISPLAY_NAME@** • **@FIDO_TIMEOUT@** • **@FIDO_CHALLENGE@** • **@FIDO_REQUIRE_RESIDENT_KEY@** • **@FIDO_RESIDENT_KEY@** • **@FIDO_USER_VERIFICATION@** • **@FIDO_AUTHENTICATOR_ATTACHMENT@** • **@FIDO_ATTESTATION@** • **@FIDO_STATUS@** • **@FIDO_ERROR_MESSAGE@** • **@ACTION@** • **@FIDO_ALLOW_CREDENTIALS@** • **@FIDO_EXTENSIONS@** • **@FIDO_PUBKEY_CRED_PARAMS@** • **@ERROR_MESSAGE@** • **@FIDO_INFOMAP_PARAM@** • **@STATE@** • **@CANCEL_ACTION@**	Displays a form that allows a user to register a FIDO device for authentication.
Error Page	`authsvc/authenticator/fido/error.html` Macros • **@ERROR_MESSAGE@** • **@JUNCTION@**	Displays errors during the FIDO authentication.

Table 22. Default template files in the `authsvc/authenticator/fido2pair` directory

Page name	File name and macros	Description
FIDO2 PAIR Authentication Decision Page	`authsvc/authenticator/fido2pair/fido2pair_authn_decision.html` Macros • **@FIDO_RP_ID@** • **@FIDO_TIMEOUT@** • **@FIDO_CHALLENGE@** • **@FIDO_USER_VERIFICATION@** • **@FIDO_USER_ID@** • **@FIDO_STATUS@** • **@FIDO_ERROR_MESSAGE@** • **@STATE@** • **@ACTION@** • **@PERSISTENT_USERNAME@** • **@FIDO_ALLOW_CREDENTIALS@** • **@FIDO_EXTENSIONS@** • **@JUNCTION@**	Displays a page that allows a user to either login by using a username and password or by using an already registered FIDO device.
FIDO2 PAIR Login Success Page	`authsvc/authenticator/fido2pair/fido2pair_login_success.html`	Displays a page that indicates that the login was successful. The page is imported into the reverse proxy management root when FIDO2 PAIR is configured and the administrator chooses to overwrite the default login success page. It saves the persistent token from the reverse proxy into the browser for subsequent FIDO2 authentication.
FIDO2 PAIR Device Registration Complete Page	`authsvc/authenticator/fido2pair/fido2pair_reg_complete.html` Macros • **@JUNCTION@** • **@ACTION@**	Displays a page that indicates that a FIDO2 device was registered successfully and can now be used for subsequent authentications.

Page name	File name and macros	Description
FIDO2 PAIR Device Registration Decision Page	authsvc/authenticator/fido2pair/fido2pair_reg_decision.html Macros • **@USERNAME@** • **@JUNCTION@** • **@ACTION@**	Displays a page that allows a user to decide whether they would like to register a FIDO2 device for authentication after the initial username and password login.
FIDO2 PAIR Device Registration Page	authsvc/authenticator/fido2pair/fido2pair_reg_mechanism.html Macros • **@FIDO_ERROR_MESSAGE@** • **@STATE@** • **@FIDO_USER_NAME@** • **@ACTION@** • **@ERROR_MESSAGE@** • **@FIDO_INFOMAP_PARAM@** • **@CANCEL_ACTION@**	Displays a page that allows a user to register a FIDO2 device for authentication.

Table 22. Default template files in the authsvc/authenticator/fido2pair directory (continued)

Table 23. Default template files in the authsvc/authenticator/infomap directory

Page name	File name and macros	Description
Error Page	authsvc/authenticator/infomap/error.html Macros • **@JUNCTION@** • **@ERROR_MESSAGE@** • **@REQ_ADDR@** • **@TIMESTAMP@** • **@EXCEPTION_MSG@** • **@EXCEPTION_STACK@**	Displays a page that shows the details of an error that was encountered.

Table 23. Default template files in the `authsvc/authenticator/infomap` directory (continued)		
Page name	**File name and macros**	**Description**
FIDO Assertion Page	`authsvc/authenticator/infomap/fido_assertion.html` Macros • **@FIDO_RP_ID@** • **@FIDO_TIMEOUT@** • **@FIDO_CHALLENGE@** • **@FIDO_USER_VERIFICATION@** • **@FIDO_USER_ID@** • **@FIDO_STATUS@** • **@FIDO_ERROR_MESSAGE@** • **@ACTION@** • **@FIDO_ALLOW_CREDENTIALS@** • **@FIDO_EXTENSIONS@** • **@ERROR_MESSAGE@** • **@FIDO_INFOMAP_PARAM@** • **@STATE@** • **@CANCEL_ACTION@**	Displays a form that allows a user to perform a FIDO authentication.

Page name	File name and macros	Description
FIDO Attestation Page	`authsvc/authenticator/infomap/fido_attestation.html` Macros • **@FIDO_RP_ID@** • **@FIDO_RP_NAME@** • **@FIDO_TIMEOUT@** • **@FIDO_CHALLENGE@** • **@FIDO_USER_ID@** • **@FIDO_USER_NAME@** • **@FIDO_USER_DISPLAY_NAME@** • **@FIDO_STATUS@** • **@FIDO_ERROR_MESSAGE@** • **@ACTION@** • **@FIDO_EXCLUDED_CREDENTIALS@** • **@FIDO_PUBKEY_CRED_PARAMS@** • **@FIDO_AUTHENTICATOR_SELECTION@** • **@FIDO_EXTENSIONS@** • **@ERROR_MESSAGE@** • **@FIDO_INFOMAP_PARAM@** • **@STATE@** • **@CANCEL_ACTION@** • **@FIDO_ATTESTATION@**	Displays a form that allows a user to register a FIDO device for authentication.
Username Login Page	`authsvc/authenticator/infomap/login.html` Macros • **@ERROR_MESSAGE@** • **@JUNCTION@** • **@ACTION@**	Displays the form where the user can enter their username to log in.

Table 23. Default template files in the authsvc/authenticator/infomap directory (continued)

Table 24. Default template files in the authsvc/authenticator/mmfa directory		
Page name	**File name and macros**	**Description**
Mobile Multi-Factor Authentication Device Selection Page	authsvc/authenticator/mfa/device_selection.html Macros • **@JUNCTION@** • **@ACTION@** • **@MMFA_DEVICE_CHECKED@** • **@MMFA_DEVICE_ID@** • **@MMFA_DEVICE_LABEL@**	Displays a form that allows a user to select which registered device to send a mobile multi factor authentication notification.
Mobile Multi-Factor Authentication Error Page	authsvc/authenticator/mmfa/error.html Macros • **@JUNCTION@** • **@ERROR_MESSAGE@** • **@REQ_ADDR@** • **@TIMESTAMP@** • **@EXCEPTION_MSG@**	Displays a page that shows the details of an error that was encountered.
Mobile Multi-Factor Authentication Login Wait Page	authsvc/authenticator/mmfa/login_wait.html • **@ACTION@** • **@MMFA_TRANSACTION_STATUS@** • **@ERROR_MESSAGE@** • **@MMFA_TRANSACTION_ID@** • **@MMFA_CONTEXT_MESSAGE@** • **@RETURN_ENABLED@** • **@TOTP_ENROLLED@** • **@MMFA_DEVICE_NAME@** • **@JUNCTION@**	Displays a page that shows that a notification was sent to the selected device and waits for a response.

Table 25. Default template files in the *authsvc/authenticator/mobileuserapproval* directory		
Page name	**File name and macros**	**Description**
Mobile User Approval Challenge Page	`authsvc/authenticator/mobileuserapproval/challenge.html` Macros • **@ERROR_MESSAGE@** • **@JUNCTION@** • **@ACTION@** • **@SERVER_CHALLENGE@** • **@HANDLE@**	Displays a form that allows a user to sign the challenge data with the key that is associated with a given key handle.
Mobile User Approval Error Page	`authsvc/authenticator/mobileuserapproval/error.html` Macros • **@JUNCTION@** • **@ERROR_MESSAGE@** • **@REQ_ADDR@** • **@TIMESTAMP@** • **@EXCEPTION_MSG@** • **@EXCEPTION_STACK@**	Displays a page that shows the details of an error that was encountered.

Table 26. Default template files in the *authsvc/authenticator/qrlogin* directory		
Page name	**File name and macros**	**Description**
QR Login Error Page	`authsvc/authenticator/qrlogin/error.html` Macros • **@JUNCTION@** • **@ERROR@**	Displays errors during the QR code authentication.
QR Login Page	`authsvc/authenticator/qrlogin/qrlogin.html` Macros • **@DSI@** • **@IN_BRANCH@** • **@ACTION@** • **@LSI@** • **@JUNCTION@**	Displays a page that allows a user to authenticate by scanning a QR code with a registered device.

Table 26. Default template files in the `authsvc/authenticator/qrlogin` directory (continued)

Page name	File name and macros	Description
QR Login Response Page	`authsvc/authenticator/qrlogin/qrresponse.html` Macros • **@USERNAME@** • **@ACTION@**	Sends a response to the device to indicate that the QR code authentication was successful.

Table 27. Default template files in the `authsvc/authenticator/recaptcha` directory

Page name	File name and macros	Description
ReCAPTCHA Challenge Page	`authsvc/authenticator/recaptcha/standalone.html` Macros • **@JUNCTION@** • **@ERROR_MESSAGE@** • **@ACTION@** • **@SITE_KEY@**	Displays a form that prompts a user to fulfill a specific re-CAPTCHA challenge.

Table 28. Default template files in the `authsvc/authenticator/rsa_securid` directory

Page name	File name and macros	Description
RSA SecurID One-Time Password Error	`authsvc/authenticator/rsa_securid/error.html` Macros • **@JUNCTION@** • **@ERROR_MESSAGE@** • **@REQ_ADDR@** • **@TIMESTAMP@** • **@EXCEPTION_MSG@** • **@EXCEPTION_STACK@**	Displays errors during the RSA SecurID one-time password authentication.
RSA SecurID One-Time Password Login	`authsvc/authenticator/rsa_securid/code.html` Macros • **@JUNCTION@** • **@ERROR_MESSAGE@** • **@ACTION@**	Displays a form where the user can enter the RSA SecurID one-time password to log in.
RSA SecurID One-Time Password Login (New PIN)	`authsvc/authenticator/rsa_securid/new_pin.html` Macros • **@JUNCTION@** • **@ERROR_MESSAGE@** • **@ACTION@**	Enables a user to enter a new RSA SecurID pin.

Table 28. Default template files in the *authsvc/authenticator/rsa_securid* directory (continued)		
Page name	**File name and macros**	**Description**
RSA SecurID One-Time Password Login (Next OTP)	`authsvc/authenticator/rsa_securid/next_code.html` Macros • **@JUNCTION@** • **@ERROR_MESSAGE@** • **@ACTION@**	Enables a user to enter the next RSA SecurID one-time password.

Table 29. Default template files in the *authsvc/authenticator/u2f* directory		
Page name	**File name and macros**	**Description**
U2F Token Error Page	`authsvc/authenticator/u2f/error.html` Macros • **@JUNCTION@** • **@ERROR_MESSAGE@** • **@REQ_ADDR@** • **@TIMESTAMP@** • **@EXCEPTION_MSG@**	Displays errors during the U2F token authentication.
U2F Token Registration Page	`authsvc/authenticator/u2f/register.html` Macros • **@U2F_APP_ID@** • **@U2F_CHALLENGE@** • **@U2F_VERSION@** • **@ACTION@** • **@CANCEL_ACTION@** • **@U2F_TOKENS@** • **@JUNCTION@** • **@UNREGISTER_ACTION** • **@UPDATE_ACTION@**	Displays a form to allow a user to register a U2F token for authentication.

Table 30. Default template files in the *authsvc/authenticator/verify_gateway* directory		
Page name	**File name and macros**	**Description**
IBM Verify Gateway Response Page	`authsvc/authenticator/verify_gateway/response.html`	This file is not currently used.

Table 31. Default template files in the *authsvc/authenticator/password/* directory		
Page name	**File name and macros**	**Description**
Change Password	authsvc/authenticator/password/change_password.html Macros • **@USERNAME@** • **@ERROR_MESSAGE@** • **@JUNCTION@** • **@ACTION@**	Enables the users to change their registry password.
Username and Password Error	authsvc/authenticator/password/error.html Macros • **@ERROR_MESSAGE@** • **@JUNCTION@** • **@REQ_ADDR@** • **@TIMESTAMP@** • **@EXCEPTION_MSG@** • **@EXCEPTION_STACK@**	Displays errors during the username and password authentication or when the users modify their password.
Username and Password Login	authsvc/authenticator/password/login.html Macros • **@ERROR_MESSAGE@** • **@JUNCTION@** • **@ACTION@**	Displays the form where the users can enter their username and password to log in.

Table 32. Default template files in the *authsvc/authenticator/http_redirect/* directory		
Page name	**File name and macros**	**Description**
HTTP Redirect Authentication Error	authsvc/authenticator/http_redirect/allerror.html Macros • **@JUNCTION@** • **@ERROR_MESSAGE@** • **@REQ_ADDR@** • **@TIMESTAMP@** • **@EXCEPTION_MSG@** • **@EXCEPTION_STACK@**	Displays general errors during for HTTP redirect authentication mechanism.

Table 32. Default template files in the `authsvc/authenticator/http_redirect/` directory (continued)

Page name	File name and macros	Description
HTTP Redirect Authentication Failed	`authsvc/authenticator/http_redirect/error_authenticate.html` Macros • MAC • **@JUNCTION@** • **@ERROR_MESSAGE@** • **@REQ_ADDR@** • **@TIMESTAMP@** • **@EXCEPTION_MSG@** • **@EXCEPTION_STACK@**	Displays errors during the HTTP redirect authentication flow.

Table 33. Default template files in the `authsvc/authenticator/macotp/` directory

Page name	File name and macros	Description
MAC One-Time Password Delivery Selection	`authsvc/authenticator/macotp/delivery_selection.html` Macros • **@JUNCTION@** • **@ACTION@** • **@OTP_METHOD_ID@** • **@OTP_METHOD_CHECKED@** • **@OTP_METHOD_LABEL@**	Displays the list of methods for generating, delivering, and verifying the one-time password.
MAC OTP One-Time Password Error	`authsvc/authenticator/macotp/error.html` Macros • **@JUNCTION@** • **@ERROR_MESSAGE@** • **@REQ_ADDR@** • **@TIMESTAMP@** • **@EXCEPTION_MSG@** • **@EXCEPTION_STACK@**	Displays errors during the MAC one-time password authentication.

Table 33. Default template files in the *authsvc/authenticator/macotp/* directory (continued)		
Page name	**File name and macros**	**Description**
MAC One-Time Password Login	`authsvc/authenticator/macotp/login.html` Macros • **@MAPPING_RULE_DATA@** • **@DISPLAY_RESELECT_BUTTON@** • **@OTP_DELIVERY_ATTR@** • **@ERROR_MESSAGE@** • **@ACTION@** • **@JUNCTION@** • **@OTP_HINT@** • **@OTP_LOGIN_DISABLED@** • **@OTP_REGENERATE_DISABLED@**	Displays the form where the user can enter the MAC one-time password.

Table 34. Default template files in the *authsvc/authenticator/rsa/* directory		
Page name	**File name and macros**	**Description**
RSA One-Time Password Error	`authsvc/authenticator/rsa/error.html` Macros • **@JUNCTION@** • **@ERROR_MESSAGE@** • **@REQ_ADDR@** • **@TIMESTAMP@** • **@EXCEPTION_MSG@** • **@EXCEPTION_STACK@**	Displays errors during the RSA one-time password authentication.
RSA One-Time Password Login	`authsvc/authenticator/rsa/code.html` Macros • **@ERROR_MESSAGE@** • **@JUNCTION@** • **@ACTION@**	Displays the form where the users can enter the RSA one-time password to log in.

Table 34. Default template files in the *authsvc/authenticator/rsa/* directory (continued)		
Page name	**File name and macros**	**Description**
RSA One-Time Password Login (New PIN)	`authsvc/authenticator/rsa/new_pin.html` Macros • **`@ERROR_MESSAGE@`** • **`@JUNCTION@`** • **`@ACTION@`**	Enables users to enter a new RSA pin.
RSA One-Time Password Login (Next OTP)	`authsvc/authenticator/rsa/next_code.html` Macros • **`@ERROR_MESSAGE@`** • **`@JUNCTION@`** • **`@ACTION@`**	Enables users to enter the next RSA one-time password.

Table 35. Default template files in the *authsvc/authenticator/totp/* directory		
Page name	**File name and macros**	**Description**
TOTP One-Time Password Error	`authsvc/authenticator/totp/error.html` Macros • **`@JUNCTION@`** • **`@ERROR_MESSAGE@`** • **`@REQ_ADDR@`** • **`@TIMESTAMP@`** • **`@EXCEPTION_MSG@`** • **`@EXCEPTION_STACK@`**	Displays errors during the TOTP one-time password authentication.
TOTP One-Time Password Login	`authsvc/authenticator/totp/login.html` Macros • **`@JUNCTION@`** • **`@ERROR_MESSAGE@`** • **`@ACTION@`**	Displays the form where the users can enter the TOTP password to log in.

Table 36. Default template files in the authsvc/authenticator/hotp/ directory

Page name	File name and macros	Description
HOTP One-Time Password Error	`authsvc/authenticator/hotp/error.html` Macros • **@JUNCTION@** • **@ERROR_MESSAGE@** • **@REQ_ADDR@** • **@TIMESTAMP@** • **@EXCEPTION_MSG@** • **@EXCEPTION_STACK@**	Displays errors during the HOTP one-time password authentication.
HOTP One-Time Password Login	`authsvc/authenticator/hotp/login.html` Macros • **@JUNCTION@** • **@ERROR_MESSAGE@** • **@ACTION@**	Displays the form where the users can enter the HOTP password to log in.

Table 37. Default template files in the authsvc/authenticator/consent_register_device/ directory

Page name	File name and macros	Description
Consent page	`authsvc/authenticator/consent_register_device/consent-form.html` Macros • **@JUNCTION@** • **@ERROR_MESSAGE@** • **@ACTION@**	Prompts the user to provide consent for registering a device.
Consent to Device Registration Error	`authsvc/authenticator/consent_register_device/error.html` Macros • **@ERROR_MESSAGE@** • **@JUNCTION@** • **@REQ_ADDR@** • **@TIMESTAMP@** • **@EXCEPTION_MSG@** • **@EXCEPTION_STACK@**	Displays errors during the consent to device registration flow.

Table 38. Default template files in the `authsvc/authenticator/eula/` directory		
Page name	**File name and macros**	**Description**
End-User License Agreement license file display	`authsvc/authenticator/eula/license.txt`	Contains the license agreement to display to the user. The template does not use replacement macros. **Note:** You can add more license files to the template tree. Specify the metadata in the End-User License Agreement for the following purposes: • Auditing • Forensic The End-User License Agreement authentication mechanism removes the metadata before it displays the license agreement to the user. The metadata must be on the first line of the license agreement. For example, `Metadata:   Version: 1.0          Identifier: 135223434343` When the user accepts the license agreement or declines the license agreement, the mechanism audits: • The user action. • The license file name. • The corresponding metadata.
End-User License Agreement license agreement display	`authsvc/authenticator/eula/eula.html` Macros • **@USERNAME@** • **@LICENSE@** • **@JUNCTION@** • **@ACTION@**	Displays the page where the user views the license and accepts the license agreement.

Table 38. Default template files in the *authsvc/authenticator/eula/* directory (continued)		
Page name	**File name and macros**	**Description**
End-User License Agreement license agreement decline	`authsvc/authenticator/eula/error_license_declined.html` Macros • **@USERNAME@** • **@ERROR_MESSAGE@** • **@REQ_ADDR@** • **@TIMESTAMP@** • **@ERROR_MESSAGE@** • **@EXCEPTION_MSG@** • **@EXCEPTION_STACK@** • **@LICENSE_FILE@** • **@LICENSE_METADATA@** • **@JUNCTION@**	Displays the page where the user declines the license agreement.

Table 39. Default template files in the *authsvc/authenticator/knowledge_questions/* directory		
Page name	**File name and macros**	**Description**
Knowledge Questions authentication mechanism knowledge login form	`authsvc/authenticator/knowledge_questions/login.html` Macros • **@ERROR_MESSAGE@** • **@JUNCTION@** • **@ACTION@** • **@QUESTION_INDEX@** • **@QUESTION_TEXT@** • **@QUESTION_UNIQUE_ID@** • **@QUESTION_COUNT@**	Displays the form where the user enters the answers to the required knowledge questions.
Knowledge Questions authentication mechanism knowledge question authentication errors	`authsvc/authenticator/knowledge_questions/error.html` Macros • **@JUNCTION@** • **@REQ_ADDR@** • **@TIMESTAMP@** • **@ERROR_MESSAGE@** • **@EXCEPTION_MSG@** • **@EXCEPTION_STACK@**	Displays errors during knowledge-question authentication.

Table 39. Default template files in the `authsvc/authenticator/knowledge_questions/` directory (continued)

Page name	File name and macros	Description
Knowledge Questions authentication mechanism missing knowledge questions with grace period	`authsvc/authenticator/knowledge_questions/not_enough_questions_found_continue.html` Macros • **`@JUNCTION@`** • **`@USERNAME@`** • **`@NUM_REQUIRED_ANSWERS@`** • **`@NUM_REGISTERED_QUESTIONS@`** • **`@GRACE_PERIOD_AUTH_COUNT@`** • **`@MAX_GRACE_PERIOD_AUTH_COUNT@`** • **`@ACTION@`**	Displayed when the user did not register the required number of knowledge questions and answers that are required for successful authentication. The following conditions must also be true: • The administrator configured the environment to allow grace-period authentication. • The user did not reach the limit of grace-period logins.
Knowledge Questions authentication mechanism missing knowledge questions without grace period	`authsvc/authenticator/knowledge_questions/not_enough_questions_found_error.html` Macros • **`@JUNCTION@`** • **`@USERNAME@`** • **`@NUM_REQUIRED_ANSWERS@`** • **`@NUM_REGISTERED_QUESTIONS@`** • **`@REQ_ADDR@`** • **`@TIMESTAMP@`**	Displayed when the user did not register the required number of knowledge questions and answers that are required for successful authentication. One of the following conditions must also be true: • The administrator did not configure the environment to allow grace-period authentication. • The user reached the limit of grace-period logins.

Authentication error template files

These files display errors that occur during authentication.

Authentication error template files

These files display errors that occur during authentication.

Table 40. Default files in the `proper/` directory

Page name	File name and macros	Description
Access Denied	`proper/errors/access_denied.html` Macros: • **`@REQ_ADDR@`** • **`@TIMESTAMP@`**	Displays a message that the user cannot access the requested resource.

Page name	File name and macros	Description
General Error	`proper/errors/allerror.html` Macros: • **@REQ_ADDR@** • **@TIMESTAMP@** • **@DETAIL@** • **@EXCEPTION_STACK@**	Displays general errors that are not displayed in other template files.
Missing Component Error	`proper/errors/missingcomponent.html` Macros: • **@REQ_ADDR@** • **@TIMESTAMP@** • **@DETAIL@** • **@EXCEPTION_MSG@** • **@EXCEPTION_STACK@**	Displays an error that the component required to process the request was not correctly configured or was not initialized.
Authentication Required	`proper/errors/need_authentication.html` Macros: • **@REQ_ADDR@** • **@TIMESTAMP@**	Displays an error that authentication is required to access the requested resource.
Protocol Determination Error	`proper/errors/noprotdet.html` Macros: • **@REQ_ADDR@** • **@TIMESTAMP@** • **@EXCEPTION_MSG@** • **@EXCEPTION_STACK@**	Displays an error that the access request is to an unknown address. The error might occur because no configured endpoint or protocol exists that is mapped to this endpoint.
Protocol Runtime Error	`proper/errors/protocol_error.html` Macros: • **@REQ_ADDR@** • **@TIMESTAMP@** • **@EXCEPTION_MSG@** • **@EXCEPTION_STACK@**	Displays errors that an error occurred fulfilling a request to a specified address, and the error was caused by an unexpected error on the protocol module.

Table 40. Default files in the `proper/` directory (continued)

OAuth template files

These files support OAuth.

OAuth template files

These files support OAuth. For more information, see OAuth 2.0 and OIDC Support.

Table 41. Default files in the oauth20/ directory

Page name	File name and macros	Description
OAuth 2.0 Trusted Clients Manager	oauth20/clients_manager.html Macros: • **@USERNAME@** • **@OAUTH_CLIENT_COMPANY_NAME@** • **@PERMITTED_SCOPES@** • **@OAUTH_CUSTOM_MACRO@**	Used by resource owners to show and manage trusted clients information.
OAuth 2.0 - Consent to Authorize	oauth20/user_consent.html Macros: • **@USERNAME@** • **@OAUTH_CLIENT_COMPANY_NAME@** • **@PERMITTED_SCOPES@** • **@OAUTH_CUSTOM_MACRO@**	Used by the authorization server to determine and store user consent information about which OAuth clients are authorized to access the protected resource. The page also lists of scopes that the OAuth client requests. These lists are shown in the consent page and can be of indeterminate length. The template supports multiple copies of stanzas that are repeated once for each scope in the lists.
OAuth 2.0 - Error	oauth20/user_error.html Macros: • **@OAUTH_CLIENT_COMPANY_NAME@** • **@CLIENT_TYPE@** • **@CLIENT_ID@** • **@REDIRECT_URI@** • **@STATE@** • **@RESPONSE_TYPE@** • **@USERNAME@** • **@OAUTH_TOKEN_SCOPE_REPEAT@** • **@OAUTH_OTHER_PARAM_REPEAT@** • **@OAUTH_OTHER_PARAM_VALUE_REPEAT@**	Shows detailed text information when an error occurs in an OAuth 2.0 flow.

Table 41. Default files in the oauth20/ directory (continued)		
Page name	**File name and macros**	**Description**
OAuth - Response	`oauth20/user_response.html` Macros: • **@OAUTH_CODE@**	Displays the authorization code of an OAuth client that did not specify a client redirection URI upon registration. When the OAuth client does not specify a client redirection URI or cannot receive redirects, the authorization server does not know where to send the resource owner after authorization. As a result, the OAuth client does not receive the authorization code that is required to exchange for an access token or refresh token. The page includes several codes: • The authorization code that the resource owner can provide to the trusted OAuth client. • The authorization code as machine-readable Quick Response (QR) code. **Note:** The encoder that creates the QR code follows the ISO/IEC 18004:2006 specification. Scanners that support this specification can read the generated QR code.

Template file macros

Most template pages contain one or more macros. The macros are replaced by values that are specific to the action that is requested on the page.

Macro	Value that replaces the macro
@CLIENT_ID@	The `client_id` parameter that is specified in the authorization request.
@CONSENT_FORM_VERIFIER@	A unique identifier for the consent_form_verifier parameter value. The value is automatically generated by the authorization server. Do not modify the parameter name or value.
@DETAIL@	The error message.
@ERROR_CODE@	Characters that uniquely identify the error.
@ERROR_DESCRIPTION@	The native language support (NLS) text of the error message that is associated with the error.
@ERROR_MESSAGE@	An error message that is specific to the action in the page. For example, on the One-time password template page for login, the error message indicates that the password submitted

Macro	Value that replaces the macro
	contains errors, such as the password is not valid or has expired.
@EXCEPTION_MSG@	The exception message.
@EXCEPTION_STACK@	The stack trace of the error.
@GRACE_PERIOD_AUTH_COUNT@	The amount of grace-period authentication.
@KICKOFF_METHOD@	The value that is set for the advanced configuration entry `sps.authService.policyKickoffMethod`. Available only through the use of an Infomap authenticator.
@LICENSE@	The contents of the license file.
@LICENSE_FILE@	The name of the license file.
@LICENSE_METADATA@	The metadata that is either: • Defined in the license file. • `Not Available` if it is not defined.
@MAPPING_RULE_DATA@	If the submitted one-time password contains an error, this value is the STS Universal User context attribute with the name `@MAPPING_RULE_DATA@` and is type `otp.sts.macro.type`. This context attribute can be set in the OTPVerify mapping rule.
@MAX_GRACE_PERIOD_AUTH_COUNT@	The maximum count of grace-period authentication that is allotted to a policy.
@MAX_STORED_QUESTIONS@	The maximum number of answers that can be stored per user.
@NUM_REQUIRED_ANSWERS@	The number of valid answers that is required for successful authentication.
@NUM_REGISTERED_QUESTIONS@	The number of questions that the user registered.
@OAUTH_AUTHORIZE_URI@	The URI for the authorization endpoint.
@OAUTH_CLIENT_COMPANY_NAME@	A multi-valued macro that belongs inside an `[RPT trustedClients]` repeatable replacement list. The values are replaced with the name of the company that requests access to the protected resource.
@OAUTH_CLIENTMANAGERURL@	A multi-valued macro that belongs inside an `[RPT trustedClients]` repeatable replacement list. The values are replaced with the endpoint of the trusted clients manager.
@OAUTH_CLIENT_NAME@	A macro that represents the name of the client that requests access to the protected resource.
@OAUTH_CODE@	The `oauth_code` parameter that is specified in the authorization response.
@OAUTH_CUSTOM_MACRO@	A multi-valued macro that belongs inside an `[RPT trustedClients]` repeatable replacement list. The values are replaced with trusted client information that contains additional information about an authorized OAuth client.
@OAUTH_OTHER_PARAM_REPEAT@	A multi-valued macro that belongs inside an `[RPT oauthOtherParamsRepeatable]` repeatable replacement list. The values show the list of extra parameter names.

Macro	Value that replaces the macro
@OAUTH_OTHER_PARAM_VALUE_REPEAT@	A multi-valued macro that belongs inside an `[RPT oauthOtherParamsRepeatable]` repeatable replacement list. The values show the list of extra parameter values.
@OAUTH_TOKEN_SCOPE_REPEAT@	A multi-valued macro that belongs inside an `[RPT oauthTokenScopePreapprovedRepeatable]` or `[RPT oauthTokenScopeNewApprovalRepeatable]`repeatable replacement lists. The values inside the `[RPT oauthTokenScopePreapprovedRepeatable]` show the list of token scopes that have been previously approved by the resource owner. Alternatively, the values inside the `[RPT oauthTokenScopeNewApprovalRepeatable]` show the list of token scopes that have not yet been approved by the resource owner.
@OTP_HINT@	The one-time password hint. The hint is a sequence of characters that is associated with the one-time password.
@OTP_METHOD_CHECKED@	For the first method, this macro is replaced with an HTML radio button attribute that causes that radio button to be selected. For the remaining methods that generate, deliver, and verify one-time passwords, this macro is replaced with an empty string.
@OTP_METHOD_ID@	The ID of the method for generating, delivering, and verifying the one-time password. This ID is generated by the `OTPGetMethods` mapping rule.
@OTP_METHOD_LABEL@	The label of the method for generating, delivering, and verifying the one-time password. This label is generated by the `OTPGetMethods` mapping rule.
@OTP_METHOD_TYPE@	The type of the currently selected method for generating, delivering, and verifying the one-time password. This type is generated by the `OTPGetMethods` mapping rule and was selected by the user.
@OTP_STRING@	The one-time password that is generated by the one-time password provider.
@PERMITTED_SCOPES@	A multi-valued macro that belongs inside an `[RPT trustedClients]` repeatable list. The values are replaced with the token scopes to which the OAuth client has access.
@QUESTION_COUNT@	The number of questions that are presented on the login page.
@QUESTION_TEXT@	The question text. This macro is only populated when the question is a user-provided question.
@QUESTION_INDEX@	The question index. This index corresponds to the array of questions that are presented on the page when questions are presented as a group.
@QUESTION_UNIQUE_ID@	The question unique identifier.
@REDIRECT_URI@	The redirect URI that the authorization server uses to send the authorization code to. The value depends on the following items: • Redirect URI that is entered during partner registration.

Macro	Value that replaces the macro
	• `oauth_redirect` parameter that is specified in the authorization request
@REGENERATE_ACTION@	The URl where the Generate button posts the form to regenerate and deliver the new one-time password value.
@RESPONSE_TYPE@	The `response_type` parameter specified in the authorization request.
@REQ_ADDR@	The URL into which the request from the user is sent.
@RESELECT_ACTION@	The URl where the Reselect button posts the form to reselect the method for generating, delivering, and verifying the one-time password value.
@STATE@	The `state` parameter that is specified in the authorization request.

@STS_USER_MESSAGE_EXCEPTION@	This macro is populated with the customized user exception that is set in a JavaScript mapping rule by using the following utility:
	`IDMappingExtUtils.throwSTSUserMessageException(String message);`
	A template page can then use this macro to display a customized exception.
	For example:
	`<h1 class="pageTitle error">An error has occurred</h1> <div class="instructions">@STS_USER_MESSAGE_EXCEPTION@</div>`

@TIMESTAMP@	The time stamp when the error occurred.
@UNIQUE_ID@	A multi-valued macro that belongs inside an `[RPT trustedClients]` repeatable replacement list. The values are replaced with a unique identifier that identifies the trusted client information for each entry in the list.
@USERNAME@	The Security Verify Access user name.

Chapter 16. SCIM configuration

Security Verify Access provides support for selected parts of the Cross-domain Identity Management (SCIM) protocol.

SCIM is an HTTP-based protocol that makes managing identities in multi-domain scenarios easier to support through a standardized service. Security Verify Access provides SCIM-based web services to facilitate user self care capabilities such as social login association, account enablement, and password reset. You can use the **SCIM Configuration** page in the local management interface to configure the SCIM capabilities.

Note: If a reverse proxy is used in front of the SCIM components, the IV credential headers (also known as **`iv-users`**/**`iv-groups`**/**`iv-creds`**), if provided, will be used to obtain the authenticated user identity for the request.

A demo application is provided for authenticated users to view and modify their own data. This application can be accessed at the following URL:

```
https://<runtime_server>/scim/demo.html
```

By default, the demo application is disabled on the appliance. To enable it, go to **System** > **Advanced Tuning Parameters**, add an advanced tuning parameter named **`runtime_profile.scim_demo_enabled`**, and set its value to **`true`**.

General SCIM settings

The general SCIM settings include common configuration for the SCIM Web Service.

Procedure

1. From the top menu, go to **AAC** > **Manage** > **SCIM Configuration**.
2. On the **General** page, modify the following options as needed.

   Enable Verify Access Header Authentication
   Controls whether Verify Access Header Authentication is enabled. Verify Access Header Authentication is used to add the Security Verify Access credential attributes to the session so they can be used by SCIM.

   Enable Authorization Filter
   The authorization filter is responsible for authorizing the request. It has some pre-defined rules for each of the supported SCIM end-points. These rules are:

   For the user profile functionality

   - Only authenticated users with administrator authority are allowed to do a search of users (`GET /Users`).
   - Unauthenticated access is allowed for creating a new user (`POST /Users`).
   - Only authenticated users with administrator authority or authenticated users who are accessing their own data are allowed to perform create, retrieve, update, and delete operations on a specific user's data (`GET/PUT/DELETE/PATCH /Me` or `/Users/<id>`).

   For other functionalities
   Any authenticated user is allowed to retrieve information about the SCIM service (`GET /ServiceProviderConfig`, `/ResourceTypes`, or `/Schemas`).

   If more advanced or different authorization is required, disable this filter and use a Web Reverse Proxy or the Advanced Access Control component in front of the SCIM application to handle the authorization.

Administration Group
: This group is used by the authorization filter for authorization checks where the user must be a member of the administration group.

Max User Responses
: Sets the maximum number of users that can be returned from a web service query to list users.

Attribute Mode
: Each SCIM attribute has an associated mutability mode. The value can be **ReadOnly**, **ReadWrite**, **AdminWrite**, **UserWrite**, **WriteOnly**, or **Immutable**.

  The value of the default column shows if the mode is default (**true**) or user defined (**false**). A mode can be reset to default by setting this mode to an empty string.

  You can expand an attribute to see its subattributes.

3. Click **Save** to save the changes.

   Note: Due to the caching of configuration data within the runtime, it might take up to 30 seconds before any deployed configuration changes become active.

User profile

The user profile configuration contains the settings that are required to manage the user data that is stored in the user registry.

Procedure

1. From the top menu, go to **AAC** > **Manage** > **SCIM Configuration**.
2. Click **User Profile**.
3. Modify the following settings as needed.

LDAP Server
: This server connection is a pointer to an LDAP server connection that has been defined in the Advanced Access Control server connections page. This field contains a list of the available LDAP server connections and Verify Access Runtime server connections.

  If an LDAP type is selected, it is used directly as the SCIM LDAP server.

  If an Verify Access Runtime type is selected, the bind details in the server connection are used along with the configured Verify Access Runtime LDAP server.

  Important: The selected server connection **must** contain the bind details for the Runtime Component LDAP server. Ensure that you configure the Runtime Component before you attempt to do this.

  This field is required.

Type
: This field shows the server connection type for the selected LDAP server.

  If the server connection type is LDAP, the server connection is used as is. If the server connection type is Verify Access Runtime, the bind details in the server connection are used along with the configured Verify Access Runtime LDAP server.

  Note: If a specific federated directory is selected by using the **Attribute Lookup Directory** field it is used in each of the following lookup operations, otherwise the Verify Access primary user registry is used.

  - The list of available LDAP group related object classes only includes the values that are obtained from the lookup LDAP server.
  - The Group DN attribute selection on this page only includes the values obtained from the lookup LDAP server.

- If an Verify Access Runtime server connection is selected, the list of available LDAP user related object classes only includes the values that are obtained from the lookup LDAP server.
- If an Verify Access Runtime server connection is selected, the available LDAP attributes that are used in SCIM attribute mappings only includes the values that are obtained from the lookup LDAP server.
- If an Verify Access Runtime server connection is selected, the User DN Attribute selection on this page only includes the values that are obtained from the lookup LDAP server.

LDAP User Related Object Classes
The LDAP object classes that are used to reference a user object. These values are the object classes that will be looked for when parsing the response to an LDAP subschema query. This is how the list of LDAP user attributes are determined and made available to the administrator for mapping SCIM attributes to LDAP attributes.

This field is optional. If this field is not set, then no LDAP attributes will be available.

Attribute Mappings
The list of SCIM attributes and the mapped source for the attribute, either an LDAP or session attribute. You can expand an attribute to see its subattributes.

Note: The LDAP server connection and object classes settings must be set in the respective fields before any LDAP attributes are made available.

Enforce Password Policy
This checkbox controls whether password updates that are using the standard **`password`** SCIM attribute takes place as the administrative user or the end user. Password policy is typically only enforced in the user registry when the password is updated by the end user. Select this checkbox only if users have the necessary permissions to change their own passwords in the user registry and the user registry does not enforce password policy when a user password is changed by an administrative user.

Note: If there is an update that includes both **`password`** and **`passwordNoPolicy`** attributes, the **`passwordNoPolicy`** takes precedence and the password is ignored.

Search Suffix

This field contains the user suffix from which LDAP search operations commences.

Note: This field is not required if an Verify Access runtime connection is selected. In this case each of the supported suffixes from the configured directories are searched. The exception to this is that if Verify Access integration is enabled then the search suffix is required.

User Suffix

This field contains the suffix that houses any users that are created through the SCIM interface.

User DN Attribute
This field contains the DN attribute that is used to create users.

Note: The User Profile LDAP server connection and object classes settings must be set in the respective fields before any LDAP attributes are made available.

Attribute Lookup Directory
This field shows the federated directory that is used to retrieve the list of supported LDAP object classes and attributes that are associated with those object classes. The field is only visible if an Verify Access Runtime server connection is selected. The drop-down will then be populated with the list of configured federated directories. An empty selection results in the primary LDAP server being used.

4. Click **Save** to save the changes.

Note: Due to the caching of configuration data within the runtime, it might take up to 30 seconds before any deployed configuration changes become active.

Common attribute mappings

The User DN Attribute, User ID Attribute, and Attribute Mappings must be set appropriately for the configured LDAP server schema. This is especially important in deployments with existing user registries.

Default attribute mappings

The default mapping that ships out-of-the-box makes several assumptions:

1. User DNs are set using the **cn** attribute.
2. There exists a user ID attribute called **uid**.
3. The username attribute is populated in the LDAP attribute **cn**, and the password in **userPassword**.

These assumptions may not be correct depending on the LDAP server that is used. The following sections describe some common deployments that require modifications to the default configuration.

User Registries with users created by pdadmin

pdadmin may populate the **cn** attribute with extra entries which can cause issues when the default SCIM attribute mappings are configured. The following configuration settings must be updated so that the SCIM mapping is correct:

- User DN Attribute: **cn**
- User ID Attribute: **uid**
- Attribute Mappings:

  userName: **uid**

The crucial variation from the default SCIM Attribute Mappings is changing the userName to map to the **uid** LDAP attribute. If this mapping is not changed, multiple entries may be returned via the SCIM API for a single user.

Active Directory User Registry

Active Directory does not have an attributed named **uid** and uses different LDAP attributes to store the username and password. The username and password attributes (**userPrincipalName** and **unicodePwd**) do not exist in the default object class schema, so the object class configuration must also be modified.

- User DN Attribute: **cn**
- User ID Attribute: **userPrincipalName**
- Object classes = **top, organizationalPerson, person, user**
- Attribute Mappings:

  userName: **userPrincipalName**

  password: **unicodePwd**

Groups

The groups configuration contains the settings that are required to manage the group data that is stored in the user registry.

Procedure

1. From the top menu, go to **AAC** > **Manage** > **SCIM Configuration**.
2. Click **Groups**.
3. Modify the following settings as needed.

Note: If an Verify Access Runtime server connection is selected in "User profile" on page 198,

- It also takes effect for groups.
- The list of available LDAP group related object classes is only the values from the primary LDAP server.
- The Group DN Attribute selection on this page is only the values from the primary LDAP server.

LDAP Group Related Object Classes
The LDAP object classes that are used to reference a group object. These values are the object classes that will be looked for when parsing the response to an LDAP sub-schema query. By default, the list is populated with **`groupOfNames`**.

Group DN Attribute
This field contains the DN attribute which will be used to create groups.

Note: The User Profile LDAP server connection and Group object classes settings must be set in the respective fields before any LDAP attributes are made available.

4. Click **Save** to save the changes.

Note: Due to the caching of configuration data within the runtime, it might take up to 30 seconds before any deployed configuration changes become active.

Verify Access user

The Verify Access user configurations enable the SCIM web service to manage the Security Verify Access security entities.

About this task

When Verify Access integration is enabled, the SCIM web service can perform the following operations to manage Security Verify Access identity:

- Import a user to the **`secAuthority=<Domain>`** suffix
- Delete a user from the **`secAuthority=<Domain>`** suffix
- Enable or disable a user account
- Change a users password
- Mark a user password as invalid

This function is implemented through the **`urn:ietf:params:scim:schemas:extension:isam:1.0:User`** schema. The data that is available as a part of this schema can be obtained from the SCIM schema web service.

The Verify Access user configuration only works in conjunction with the user profile configuration if the LDAP registry and suffix used by the user profile configuration is known to Security Verify Access (either as the Security Verify Access user registry or a federated user registry).

Procedure

1. From the top menu, go to **AAC** > **Manage** > **SCIM Configuration**.
2. Click Verify Access User.
3. Modify the following settings as needed.

Enable Verify Access Integration
Select this check box to enable the integration with Security Verify Access and the management of Security Verify Access users.

Verify Access User Registry
The name of an LDAP server connection. This LDAP server connection should reference the Security Verify Access user registry.

This server connection is a pointer to an LDAP server connection that has been defined in the Advanced Access Control server connections page.

This field contains a list of the available LDAP server connections and Verify Access Runtime server connections.

If an LDAP type is selected, it is used directly as the SCIM LDAP server.

If an Verify Access Runtime type is selected, the bind details in the server connection are used along with the configured Verify Access Runtime LDAP server.

Important: The selected server connection **must** contain the bind details for the Runtime Component LDAP server. Ensure that you configure the Runtime Component before you attempt to do this.

This field is required.

Type
This field shows the server connection type for the selected LDAP server.

If the server connection type is LDAP, the server connection is used as is. If the server connection type is Verify Access Runtime, the bind details in the server connection are used along with the configured Verify Access Runtime LDAP server.

Verify Access Domain
The Security Verify Access domain name. The default value for this field is `Default`.

Update Native Users
This option defines whether the uid attribute of the native user entry is updated with the Security Verify Access user identity when a Security Verify Access user is created. Enabling this option allows Security Verify Access to authenticate users with their Security Verify Access user identity.

4. Click **Save** to save the changes.

Note: Due to the caching of configuration data within the runtime, it might take up to 30 seconds before any deployed configuration changes become active.

Custom Schema Extensions

Custom attributes can now be added to SCIM User payloads with Custom Schema Extensions.

Procedure

1. From the top menu, go to **AAC** > **Manage** > **SCIM Configuration**.
2. Click **Custom Schema Extensions**.
3. Click **Add** and specify the following information to create a custom schema extension.

Name
Specify the name of the extension.

ID
Specify the ID of the extension. This is typically a URN.

Description
Provide a description for the extension

Attributes
Click **Add** and specify the following attributes as needed.

Type
Specifies the attribute type.

Source Type
Sets one of the following source types:

- `LDAP`
- `Session`

- `Fixed`

Source Attribute
Specifies the source attribute, depending on which `sourceType` is specified.

If `sourceType` is set to `ldap`, the `sourceAttribute` field must contain the name of the LDAP attribute that is to be mapped. The LDAP attribute is retrieved using the server connection, suffix, dn, ID, and object class configuration defined for the User Profile.

If `sourceType` is set to `session`, the `sourceAttribute` field must contain the name of the Session attribute that is to be mapped.

If `sourceType` is set to `fixed`, the data in the `sourceAttribute` field is inserted as the value.

Multi-valued
Select this check-box to control whether the source attribute is interpreted as an array, instead of a single value.

Required
Select this check-box if the source attribute is required.

Mutability
Specifies the attribute mutability option.

Returned
Specifies the returned option.

Canonical Values
Click **Add** and specify the canonical value as needed.

4. Click **Save** to save the changes.

Chapter 17. MMFA configuration

Use the Mobile Multi-factor Authentication (MMFA) Configuration page to configure the required endpoints for mobile multi-factor authentication, as well as optional discovery mechanisms and QR code options.

General settings

The general settings include common configuration for the multi-factor authentication web service.

Procedure

1. From the top menu, go to **AAC** > **Manage** > **MMFA Configuration**.
2. On the **General** tab, you can perform the following operations.

   Using the wizard to provide configuration settings

   a. Click **Wizard**.
   b. Follow the wizard to complete settings for your new multi-factor authentication mechanism.

   Modifying existing settings

   a. Select the row to modify.
   b. Click **Edit**.
   c. Make changes as needed.
   d. Click **Save**.

   Removing all existing settings

   a. Click **Clear Configuration**.
   b. Click **Remove** to confirm the operation.

Discovery mechanisms

Use this tab to add or remove discovery mechanisms.

Procedure

1. From the top menu, go to **AAC** > **Manage** > **MMFA Configuration**.
2. Click **Discovery Mechanisms**.
3. On the **Discovery Mechanisms** tab, you can perform the following operations.

   Adding a discovery mechanism

   a. Click **Add**.
   b. Select the identifier of the discovery mechanism from the list.
   c. Click **Save** to save the settings.

   Removing a discovery mechanism

   a. Select the mechanism to delete.
   b. Click **Remove**.
   c. Click **Remove** to confirm the operation.

Custom QR code options

Use this tab to add, edit, or remove custom QR code options.

Procedure

1. From the top menu, go to **AAC** > **Manage** > **MMFA Configuration**.
2. Click **Custom QR Code Options**.
3. On the **Custom QR Code Options** tab, you can perform the following operations.

   Adding a QR code option

   a. Click **Add**.
   b. Define the QR code settings in the **Key** and **Value** fields.
   c. Click **Save**.

   Modifying a QR code option

   a. Select the key and value row to modify.
   b. Click **Edit**.
   c. Make changes as needed.
   d. Click **Save**.

   Deleting a QR code option

   a. Select the key and value row to delete.
   b. Click **Delete**.
   c. Click **Delete** to confirm the operation.

MMFA Advanced Configuration

Use this tab to edit advanced configuration properties under the MMFA category.

Procedure

1. From the top menu, go to **AAC** > **Manage** > **MMFA Configuration**.
2. Click **Advanced Configuration**.
3. On the **Advanced Configuration** tab, you can perform the following operations.

   Modify existing settings on the **General** tab.

   Device Prompt

   a. If the user has only one device registration, check the **Skip device selection** checkbox to skip the **device selection** page in an MMFA flow.
   b. Check the **Validate chosen device** checkbox to enable validation that the device that verifies the transaction is the same device that the user chooses at the device selection step.

   Silent Push notification

   Check the **Enable silent push** checkbox to allow the IBM Verify app to prompt users to complete user presence transactions in the notifications tray instead of fully opening the app.

   Policy Response on Successful Verification

   Check the **Return legacy response on successful verification** checkbox to make the MMFA browser policy return the success response as a JSON payload. The configuration applies when the **login wait** page wasn't returned before the transaction is verified.

   View the **Advanced Configuration Properties** grid to see how changing the various configuration options are reflected in the matching properties.

4. Click **Save** to save any modifications.
5. Click **Archive and Cleanup** to view more configuration properties.

   Modify existing settings on the **Archive and Cleanup** tab.

   Authenticator Cleanup

   a. Check the **Enable cleanup thread** checkbox to modify whether expired authenticators are automatically removed from the database.
   b. Modify the **Cleanup interval** field to change the amount of time between each run of the expired authenticators cleanup thread.

   Transaction Cleanup

   a. Check the **Enable transaction archival** checkbox to remove transactions from the database and log them in the archive log when the number of historical transactions reaches a certain number.
   b. Modify the **Cleanup interval** field to change the amount of time between each run of the transaction archival cleanup thread.
   c. Modify the **Maximum completed per user** field to change the number of historical transactions to keep in the database.

   Pending Cleanup

   a. Check the **Enforce number of pending transactions** checkbox to cancel pending transactions automatically when the number of pending transactions reaches a certain number.
   b. Modify the **Maximum pending per user** field to change the number of transactions that are allowed in a pending state.
   c. Check the **Enable cleanup thread** checkbox to automatically fail pending transactions after a certain amount of time.
   d. Modify the **Cleanup interval** field to change the amount of time between each run of the pending transaction cleanup thread.
   e. Modify the **Minimum age before cancellation** field to change the minimum number of seconds a transaction is in the pending state before the cleanup thread cancels it. Because of the cleanup thread interval, the total time a transaction can be in the pending state can be between minimum age and (minimum age + cleanup interval) - 1.

   Cleanup Cluster Nodes

   Cleanup is performed on all nodes by default. Check the **Cleanup only on primary node** checkbox to run transaction cleanup only on the primary node.

   View the **Advanced Configuration Properties** grid to see how changing the various configuration options are reflected in the matching properties.
6. Click **Save** to save any modifications.

Chapter 18. User self-administration tasks

Administrators can configure context-based access to enable users to perform certain self-management tasks.

A common user task is to manage registered devices. For example, users can view the attributes for a device. Also, a user can rename a device.

Users can also view, specify and configure secret keys for use with one-time password.

Managing your registered devices

You can modify the registered device fingerprints that the risk engine compares with incoming request device fingerprints in risk score calculation scenarios.

About this task

You can complete the following tasks:

- View a list of your registered devices and the list of attributes that each device has.
- Rename your registered devices.
- Remove your registered devices.
- Disable your registered devices. When you disable a registered device, the device is still registered. However, when a login request comes from the disabled device, the risk engine gives that session the highest level of risk possible, which is `100`.
- Enable your registered devices.

Procedure

Take one of the following actions:

View your registered devices and the attributes that each device has

a. Log in to `https://<WebSEAL host>:<port>/<junction name>/sps/mga/user/mgmt/html/device/device_selection.html`.

b. Click the name of a device from the table to view the attributes of that device.

Note: You can also use the following URL to go directly to the attributes page of a specific device: `https://<WebSEAL host>:<port>/<junction name>/sps/mga/user/mgmt/html/device/device_attributes.html?id=`*x*.

The query string, `?id=x` indicates the device that you are trying to access. The *x* represents the ID of the device.

Rename your registered devices

a. Log in to `https://<WebSEAL host>:<port>/<junction name>/sps/mga/user/mgmt/html/device/device_selection.html`.

   Note: If you select a specific device name, `https://<WebSEAL host>:<port>/<junction name>/sps/mga/user/mgmt/html/device/device_attributes.html` loads with an attributes page specific to that device.

b. Click the device that you want to rename.

c. Type the name into the device name field, and click **Rename Device**.

   Note: The device name must begin with an alphabetic character. Do not use control characters, leading and trailing blanks, and the following special characters: ~ `! @ # $ % ^ & * ( ) + | `` ` `` = \ ; " ' < > ? , [ ] { }` / anywhere in the name.

Remove a registered device

a. Log in to `https://<WebSEAL host>:<port>/<junction name>/sps/mga/user/mgmt/html/device/device_selection.html`.

   Note: If you select a specific device name, `https://<WebSEAL host>:<port>/<junction name>/sps/mga/user/mgmt/html/device/device_attributes.html` loads with an attributes page specific to that device.

b. Click **Remove** next to the device that you want to remove.

   Note: You can also click the device that you want to remove and view the attributes that belong to the device before you remove the device. Click **Remove Device** underneath the attributes when you are ready to remove the device.

Disable a registered device

a. Log in to `https://<WebSEAL host>:<port>/<junction name>/sps/mga/user/mgmt/html/device/device_selection.html`.

b. Clear the **Enabled** box next to the device that you want to disable.

Enable a registered device

a. Log in to `https://<WebSEAL host>:<port>/<junction name>/sps/mga/user/mgmt/html/device/device_selection.html`.

b. Select the **Enabled** box next to the device that you want to enable.

Managing OTP secret keys

To help manage your mobile data, such as your HOTP secret keys and TOTP secret keys, you can use the REST services capability.

About this task

You can use this capability to complete the following tasks:

- View your OTP secret keys.
- Configure your OTP secret keys.
- Reset your OTP secret keys.

Procedure

1. Log in to `https://<WebSEAL host>:<port>/<junction name>/sps/mga/user/mgmt/html/otp/otp.html`.
2. Perform one or more of the following actions:
   - View your current OTP secret key. When you log in, you see one or both of the following in clear text and as quick response (QR) code:
     - HOTP secret key
     - TOTP secret key
   - Configure your secret key in your OTP generator:
     a. Use the clear text to manually enter either the HOTP secret key or the TOTP secret key into your OTP provider.
     b. Scan the QR code. When you scan the QR code, the OTP generator:
        i) Analyzes the QR code.
        ii) Acquires the following information from the QR code:
           - Secret key.
           - Account with which the secret key is associated.

    - Type of OTP with which the secret key is associated.

   iii) Enter a secret key into your OTP generator.

- Reset your OTP secret key by clicking **reset**.

Configuring knowledge questions

Use the Knowledge Questions management page to manage the knowledge questions that the Knowledge Questions authentication mechanism will present as a step-up authentication measure.

About this task

The user can complete the following self-care management operations that are related to their knowledge questions:

- Create
- Update
- Delete

Procedure

1. Specify and log in to the following URL: `https://<WebSEAL host>:<port>/<junction name>/sps/mga/user/mgmt/html/questions/user_questions.html`
2. Perform one of the following tasks:
   - Create knowledge questions
     a. Select a pre-configured knowledge question, or write your own question.
     b. Specify the answer to each question.
     c. Click **Set User Questions**.
   - Update knowledge questions
     a. Choose the necessary questions to update and specify the answers to the updated questions.
     b. Click **Set User Questions.**
   - Delete knowledge questions
     a. Click **Remove All Questions** to delete you current knowledge questions.

SCIM account management

Security Verify Access provides support for selected parts of the Cross-domain Identity Management (SCIM) protocol. The SCIM interface can be used as a consolidated API for accessing user data from various data sources.

User Self-Care with the SCIM API

Security Verify Access SCIM support provides an API that can consolidate the management of user data from various sources.

Supported data sources include:

- User registry
- Security Verify Access user profile
- Mobile Multi Factor Authentication (MMFA) data
- User knowledge questions

SCIM support provided by the Security Verify Access is based on the SCIM 2.0 standard. The SCIM 2.0 standard comprises of the SCIM Core Schema defined in RFC 7643 and the SCIM Protocol defined in RFC 7644.

The SCIM application is provided on all Advanced Access Control interfaces under the path `/scim`. For example:

```
https://<runtime_listening_address>:<port>/scim
https://<runtime_listening_address>:<port>/scim/Schemas
https://<runtime_listening_address>:<port>/scim/Users
https://<runtime_listening_address>:<port>/scim/Groups
```

Setup

The SCIM application can be accessed internally via the Advanced Access Control endpoints or securely exposed externally using the Web Reverse Proxy with some additional setup.

Access to the SCIM application is controlled by the Advanced Access Control user registry. Administration tasks can be performed by users in the Administration Group that is specified on the **SCIM Configuration** page. By default, the account **`easuser`** is present in the default Administration Group.

Only use the administration accounts internally or as service accounts for other points of contact (such as the Web Reverse Proxy) to authenticate to the SCIM application.

Authenticating as a Security Verify Access user to the SCIM Application

The SCIM application supports authentication as Security Verify Access users via the Web Reverse Proxy. To set up the Web Reverse Proxy as a point of contact for the SCIM application, create a junction to the Advanced Access Control listening interface with the following settings:

Identity parameters

HTTP header identity information:

- IV-USER
- IV-GROUPS
- IV-CREDS

Authentication

- The junction authenticates by using the credentials for a service account that is part of the Administration Group.
- Depending on the configuration of the Advanced Access Control Runtime, this can be basic authentication or mutual SSL authentication.

The SCIM application can interpret the IV-USER/IV-GROUPS/IV-CREDS headers and determine which Security Verify Access user is authenticated. Specifically, the SCIM application determines the user based on the **`AZN_CRED_PRINCIPAL_NAME`** attribute. Using this information, the SCIM application resolves the **`/Users/Me`** endpoint to the current user and grants the following access:

- Read/write access to only the user's own profile
- No access to other user profiles or listing of all users

Note: As the user is determined by the SCIM application based on the value of **`AZN_CRED_PRINCIPAL_NAME`**, this value must be a normalized and globally unique value for any entity that can authenticate in your Security Verify Access environment. This includes users in local or federated user registries, users from federated single sign-on, and users from EAI applications.

Authenticating as SCIM users to the Web Reverse Proxy

It is possible to use the SCIM users as basic users to authenticate to the Web Reverse Proxy. This is useful in scenarios where you do not want to create all of your SCIM users within the Security Verify Access registry.

SCIM users can authenticate if the Security Verify Access Runtime is configured with basic user support. For further information, see Configuring the runtime to authenticate basic users.

Ensure that the LDAP server and suffix containing the SCIM users is configured and the principal attribute (**basic-user-principal-attribute**) is set to the LDAP attribute that the SCIM **userName** attribute is mapped to. By default, the SCIM **userName** attribute is mapped to the LDAP attribute uid.

URL Filtering

Resource responses include URLs that will not be filtered or rewritten by the Web Reverse Proxy by default. To rewrite URLs within SCIM JSON responses, make the following changes to the Web Reverse Proxy configuration file:

```
[filter-content-types]
type = application/scim+json
[script-filtering]
script-filter = yes
rewrite-absolute-with-absolute = yes
```

Endpoints

Security Verify Access supports these SCIM endpoints.

Table 42. Supported SCIM endpoints

URL	Method	Description
/Schemas	GET	Returns a list of all schemas.
/Schemas/{id}	GET	Returns a particular schema.
/ServiceProviderConfig	GET	Returns the SCIM service provider configuration.
/ResourceTypes	GET	Returns a list of resource types that are serviceable by this SCIM service provider.
/ResourceTypes/{id}	GET	Returns a particular resource type.
/Users	GET	Returns a list of all users.
/Users	POST	Creates a new user.
/Users/{id} or /Me	GET	Returns a particular user.
/Users/{id} or /Me	PUT	Updates an existing user.
/Users/{id} or /Me	PATCH	Patches an existing user.
/Users/{id} or /Me	DELETE	Deletes an existing user.
/Groups	GET	Returns a list of all groups.
/Groups	POST	Creates a new group.
/Groups/{id}	GET	Returns a particular group.
/Groups/{id}	PUT	Updates an existing group.
/Groups/{id}	PATCH	Patches an existing group.
/Groups/{id}	DELETE	Deletes an existing group.

Note: The URL /Me is an alias for /Users/{id} where {id} is the ID of the currently authenticated user.

For a list of standard SCIM endpoints that are not supported by the application, see "Unsupported endpoints" on page 220.

Detailed web services API documentation can be downloaded from the **File Downloads** page of the appliance LMI. The Security Verify Access SCIM documentation can be found in the following path:

```
/access_control/doc/ISAM-Access-Control-scim-rest-api.zip
```

User password change and recovery

The SCIM service provides an API that allows a user to update their own password.

Three separate methods can be used to update a user password by using the SCIM API.

- Update the password with no password policy validation. This update is done by using either a PUT or PATCH operation while the new`password` is passed by way of the `passwordNoPolicy` SCIM Attribute.
- Update the password as the actual user without providing the current password. This update is done by using either a PUT or PATCH operation while the new password is passed by way of the `password SCIMattribute`. It updates the password in two phases.

  1. Sets the user's password as a randomly generated password while binding as the LDAP administrator
  2. Binds as the actual user by using the new random password and sets the password as the passed value.

  Note: This method does the following actions.

  – If the `SCIMconfiguration` has enable password policy set to `true`, it enforces the LDAP password policy validation.

  – Creates two separate password updates that add two new passwords to the LDAP password history.

  – If the new password does not meet password requirements, it sets the random password without validating the actual new password against the password policy. This action might result in setting the password to an unknown value.
- Update the password as the actual user by providing the current password with the password schema.

Process to update the password as the actual user

Update the password by using PATCH.

This method provides the current and new passwords a PATCH data by using the patch with no path API. For example,

```
PATCH https://runtime/scim/Me
{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:PatchOp"],
   "Operations":[
     {       "op":"add/replace",
       "value": {
           "urn:ietf:params:scim:schemas:extension:isam:1.0:Password": {
               "newPassword": "password1",
               "currentPassword": "password"
            }
         }
      }
   ]
}
```

Note:

- The PATCH operation can be set as either add or replace. Both values result in the same update process.
- The PATCH with path API is not supported for this password update.

Update the password by using PUT.

This method provides the current and new password as PUT data by using the update API. For example,

```
PUT https://runtime/scim/Users/dGVzdHVzZXI
{
    "schemas":[
      "urn:ietf:params:scim:schemas:core:2.0:User",
      "urn:ietf:params:scim:schemas:extension:isam:1.0:Password"
    ],
   "urn:ietf:params:scim:schemas:extension:isam:1.0:Password": {
       "currentPassword":"password",
       "newPassword":"password1"
   },
   "userName":"testuser",
   "registrySuffix":"dc=test,dc=org",
   "name":{
       "familyName":"User",
       "givenName":"Test"
    }
}
```

Note: The update operation is a full replace operation. Ensure that all the user data is provided or else the existing data that is not specified is erased.

User profile schema LDAP attribute mapping

Security Verify Access provides pre-defined mapping of SCIM attributes to commonly available LDAP attributes.

This default mapping can be customized on the **User Profile** tab of the **SCIM Configuration** page. See "User profile" on page 198.

SCIM attributes that are not mapped to an LDAP attribute are not shown when the user profile schema is queried.

The following table shows the default user schema attribute mapping.

Table 43. User schema attribute mapping

SCIM attribute	LDAP attribute
addresses[0].type == home	
addresses[0].formatted	homePostalAddress
addresses[1].type == work	
addresses[1].formatted	postalAddress
addresses[1].streetAddress	street
addresses[1].postalCode	postalCode
addresses[1].locality	l
addresses[1].region	st
displayName	displayName
emails[0].type == work	
emails[0].primary == true	
emails[0].value	mail
id	Base64URLEncoded version of uid
name.familyName	sn
name.givenName	givenName

Table 43. User schema attribute mapping (continued)	
SCIM attribute	**LDAP attribute**
password	userPassword
phoneNumbers[0].type == work	
phoneNumbers[0].primary == true	
phoneNumbers[0].value	telephoneNumber
phoneNumbers[1].type == home	
phoneNumbers[1].value	homePhone
phoneNumbers[2].type == mobile	
phoneNumbers[2].value	mobile
phoneNumbers[3].type == pager	
phoneNumbers[3].value	pager
preferredLanguage	preferredLanguage
title	title
userName	cn, uid

Note:

- The multi-valued SCIM attributes (addresses, emails, and phone numbers) are not order-dependent and are shown here with array indices for illustrative purposes only.
- **id** is generated by the server based on the **userName** attribute when an account is created. If you are connecting the SCIM application to a user registry that is already populated with users, the **id** field is a Base64URLEncoded version of the **uid** field.

The following table shows the enterprise extension attribute mapping.

Table 44. Enterprise extension attribute mapping	
SCIM attribute	**LDAP attribute**
department	departmentNumber
employeeNumber	employeeNumber
manager.value	manager
organization	o

The following attributes are not mapped by default:

User schema

```
active
entitlements
externalId
groups
ims[]
locale
name.formatted
name.middleName
name.honorificPrefix
name.honorificSuffix
nickName
photos
profileUrl
roles
```

```
timezone
userType
x509Certificates
```

Enterprise user schema

```
costCenter
division
```

Handling of multi-valued LDAP attributes

If an attribute in the SCIM schema is mapped to a multi-valued LDAP attribute, only the first of the multiple values that are provided by the LDAP server is returned.

Handling of multi-valued SCIM attributes

Some SCIM attributes, such as addresses, emails, and phone numbers contain multiple complex values. For these attributes, the returned value is an array where each array element is a sub attribute with a different type string. The type strings are mapped to fixed strings and as such the entire sub attribute is always returned, regardless of whether other attributes such as value or primary are present.

Consider the following LDAP entry and corresponding SCIM JSON representation of an example user.

LDAP representation


```
dn: cn=bjensen,dc=scim-users
o: Universal Studios
givenName: Barbara
sn: Jensen
street: 100 Universal City Plaza
userPassword:: cGFzc3dvcmQ=
departmentNumber: Tour Operations
displayName: Bab Jensen
mail: bjensen@example.com
uid: bjensen
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
postalAddress:: MTAwIFVuaXZlcnNhbCBDaXR5IFBsYXphDQpIb2xseXdvb2QsIENBIDkxNjA4IF
 VTQQ==
postalCode: 91608
title: Tour Guide
cn: bjensen
employeeNumber: 701984
l: Hollywood
st: CA
homePostalAddress:: NDU2IEhvbGx5d29vZCBCbHZkCkhvbGx5d29vZCwgQ0EgOTE2MDggVVNB
telephoneNumber: 555-555-5555
mobile: 555-555-4444
homePhone: 555-555-3333
pager: 555-555-2222
preferredLanguage: en-US
manager: cn=jsmith
```


SCIM JSON representation


```
{
  "addresses": [
    {
      "formatted": "100 Universal City Plaza\r\nHollywood, CA 91608 USA",
      "locality": "Hollywood",
      "postalCode": "91608",
      "region": "CA",
      "streetAddress": "100 Universal City Plaza",
      "type": "work"
    },
    {
 "formatted": "456 Hollywood Blvd\nHollywood, CA 91608 USA",
      "type": "home"
    }
```

```
    ],
    "displayName": "Bab Jensen",
    "emails": [
      {
        "primary": true,
        "type": "work",
        "value": "bjensen@example.com"
      }
    ],
    "id": "YmplbnNlbg",
    "meta": {
      "location": "https://isam-demo.ibm.com/scim/Users/YmplbnNlbg",
      "resourceType": "User"
    },
    "name": {
      "familyName": "Jensen",
      "givenName": "Barbara"
    },
    "phoneNumbers": [
      {
        "primary": true,
        "type": "work",
        "value": "555-555-5555"
      },
      {
        "primary": false,
        "type": "home",
        "value": "555-555-3333"
      },
      {
        "primary": false,
        "type": "mobile",
        "value": "555-555-4444"
      },
      {
        "primary": false,
        "type": "pager",
        "value": "555-555-2222"
      }
    ],
    "preferredLanguage": "en-US",
    "schemas": [
      "urn:ietf:params:scim:schemas:core:2.0:User",
      "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User"
    ],
    "title": "Tour Guide",
    "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User": {
      "department": "Tour Operations",
      "employeeNumber": "701984",
      "manager": {
        "value": "cn=jsmith"
      },
      "organization": "Universal Studios"
    },
    "userName": "bjensen"
  }
```

Resource schemas

Security Verify Access supports the following resource schemas from RFC 7643.

```
“User” Resource Schema
urn:ietf:params:scim:schemas:core:2.0:User

Enterprise User Schema Extension
urn:ietf:params:scim:schemas:extension:enterprise:2.0:User

“Group” Resource Schema
urn:ietf:params:scim:schemas:core:2.0:Group
```

Security Verify Access also provides the following extensions to the “User” Resource Schema:

```
MMFA Authenticators
urn:ietf:params:scim:schemas:extension:isam:1.0:MMFA:Authenticator

MMFA Transactions
```

```
urn:ietf:params:scim:schemas:extension:isam:1.0:MMFA:Transaction

MMFA EAS
urn:ietf:params:scim:schemas:extension:isam:1.0:MMFA:EAS

User Knowledge Questions
urn:ietf:params:scim:schemas:extension:isam:1.0:UserKnowledgeQuestions

ISVA User
urn:ietf:params:scim:schemas:extension:isam:1.0:User

ISVA Group
urn:ietf:params:scim:schemas:extension:isam:1.0:Group

FIDO U2F
urn:ietf:params:scim:schemas:extension:isam:1.0:U2F

EULA
urn:ietf:params:scim:schemas:extension:isam:1.0:EULA

OTP
urn:ietf:params:scim:schemas:extension:isam:1.0:OTP

FIDO2 Authenticators
urn:ietf:params:scim:schemas:extension:isam:1.0:FIDO2Authenticators

Password
urn:ietf:params:scim:schemas:extension:isam:1.0:Password
```

Data in the Security Verify Access schemas can be managed for users that do not necessarily exist in the LDAP user registry. For instance, scenarios where a user logged in with their identity from another provider.

Consider a user logging in with an identity from `social.ibm.com`. Their `AZN_CRED_PRINCIPAL_NAME` is `https://social.ibm.com/myTestUser`. The SCIM interface can be used to manage data on the Security Verify Access extension schemas if the correct SCIM user ID is provided.

The SCIM user ID expected by the SCIM application is the Base64 and URL encoded version of the username, which in this case is “`aHR0cHM6Ly9zb2NpYWwuaWJtLmNvbS9teVRlc3RVc2Vy`”. Even though the user does not exist in the LDAP user registry and has no attributes in the defined User Resource Schema, it is still possible to manage their data in the Security Verify Access specific schemas.

In the following example, a user is not in the user registry but still has MMFA Authenticators data.

```
GET https://scim.ibm.com/scim/Users/aHR0cHM6Ly9zb2NpYWwuaWJtLmNvbS9teVRlc3RVc2Vy

{
  "meta": {
    "location": "https://scim.ibm.com/scim/Users/aHR0cHM6Ly9zb2NpYWwuaWJtLmNvbS9teVRlc3RVc2Vy ",
    "resourceType": "User"
  },
  "schemas": [
    "urn:ietf:params:scim:schemas:extension:isam:1.0:MMFA:Authenticator"
  ],
  "id": "dGVzdHVzZXI1NTU",
  "urn:ietf:params:scim:schemas:extension:isam:1.0:MMFA:Authenticator": {
    "userPresenceMethods": [],
    "authenticators": [
      {
        "osVersion": "2.b",
        "id": "uuid1c689142-be74-4262-9e33-8813b532599b",
        "oauthGrant": "uuid9d06ddc1-0157-16e7-87b9-e593c7ab6dfc",
        "deviceName": "IBM Phone",
        "enabled": true
      }
    ],
    "fingerprintMethods": [
      {
        "id": "uuid4e6e91fe-0956-41be-a933-c01ed4466c05",
        "keyHandle": " SVNBTSBTQ0lNIEVhc3RlciBFZ2cu",
        "authenticator": "uuid1c689142-be74-4262-9e33-8813b532599b",
        "enabled": true,
        "algorithm": "SHA512withRSA"
      }
    ]
```

```
  }
}
```

Limitations

The Security Verify Access SCIM application has some limitations.

General

HTTP ETags support

Security Verify Access does not provide support for HTTP ETags. Resource responses will not contain the ETag HTTP header, and response bodies will not contain the meta attributes: **`created`**, **`lastModified`**, and **`version`**.

Attribute case sensitivity

Attribute names are always handled with case sensitivity. This rule also applies to attributes that are named in JSON bodies and in URL query string parameters.

Resource filtering and sorting

Security Verify Access supports basic filtering operations using attribute operators. Filtering with logical operators or grouping operators is not supported. Sorting operations are not supported.

Unsupported endpoints

Security Verify Access does not provide the following endpoints defined is RFC 7643 and RFC 7644.

Bulk operations and the `/Bulk` endpoint

Security Verify Access does not support the bulk resource update endpoint.

Search operations using the `./search` endpoint

Security Verify Access does not support query resources using HTTP POST functionality. This includes the endpoint **`/.search`** and any endpoint of the format **`<prefix>/.search`**.

Custom Schema Extensions

The SCIM service provides an API that can consolidate the management of user data from various sources.

However, the core schemas and built-in schema extensions provided out of the box have strict payload structure and fixed attribute names.

Custom Schema Extensions can be used to add custom attributes to the User Resource with administrator-defined schema extensions. The custom attributes can be mapped to LDAP attributes, session attributes, or fixed values.

Configuration can be performed on Custom Schema Extensions tab of the SCIM Configuration page. See "Custom Schema Extensions" on page 202. Configuration can also be performed by using the Schema Extensions API.

Schema Extension

A schema extension consists of the following parameters:

- The ID, which is typically a URN
- A human-readable name
- An optional description
- A list of custom attributes

For example:

```
{
    "id": "urn:UserAttributes",
    "name": "Custom User Attributes",
```

```
    "description": "",
    "attributes": [
        ...
    ]
}
```

Custom schemas can be created, modified or removed on the “Custom Schema Extensions” on page 202 page.

The schema extension ID is used as the JSON container when inserting the custom attributes into the User Resource payload, to distinguish attributes belonging to the extension namespace from base schema attributes.

```
{
    "meta": {
        "location": "https://localhost/scim/Users/dGVzdHVzZXI",
        "resourceType": "User"
    },
    "schemas": [
        "urn:ietf:params:scim:schemas:core:2.0:User",
        "urn:ietf:params:scim:schemas:extension:isam:1.0:User",
        "urn:UserAttributes",
    ],
    "id": "dGVzdHVzZXI",
    "userName": "testuser",
    ...
    "urn:ietf:params:scim:schemas:extension:isam:1.0:User": {
        "passwordValid": true,
        "identity": "testuser",
        "accountValid": true
    },
    "urn:UserAttributes": {
        "pwdAccountLocked": true
    }
}
```

Custom Attributes

The configuration of an attribute in a custom schema extension follows a similar format to the attribute definitions from RFC 7643. The key differences are the inclusion of the mapping parameter, and the exclusion of the caseExact and uniqueness parameters.

For example, an attribute in a custom schema extension may be defined as follows:

```
{
    "name": "pwdAccountLocked",
    "description": "Whether this account is locked.",
    "type": "boolean",
    "multiValued": false,
    "required": true,
    "mutability": "adminWrite",
    "returned": "default",
    "mapping": {
        "sourceAttribute": "ibm-pwdAccountLocked",
        "sourceType": "ldap"
    }
}
```

Similar to the User Profile Attribute Mapping, the mapping parameter controls where the attribute value is sourced from (for example, mapped from). The `sourceType` can be set to one of: `ldap`, `session`, `fixed`.

If `sourceType` is set to `ldap`, the `sourceAttribute` field must contain the name of the LDAP attribute that is to be mapped. The LDAP attribute is retrieved using the server connection, suffix, dn, ID, and object class configuration defined for the User Profile.

If `sourceType` is set to `session`, the `sourceAttribute` field must contain the name of the Session attribute that is to be mapped.

If `sourceType` is set to `fixed`, the data in the `sourceAttribute` field is inserted as the value.

The following attribute types are supported:

- String
- Boolean
- Decimal
- Decimal
- Integer
- DateTime
- Binary
- Reference
- Complex (Object)

The following attribute mutability options are supported:

- ReadOnly
- ReadWrite
- AdminWrite
- UserWrite
- WriteOnly
- Immutable

Limitations

- Custom Schema Extensions are only applied to User resource requests, not Group resource requests.
- The Custom Schema Extensions functionality cannot be used to modify the "User" Resource Schema, Enterprise User Schema Extension, or any built-in extension schemas. For example, MMFA transactions, FIDO2 Authenticators.
- The schema attributes "caseExact" and "uniqueness" are not configurable, and default to false and "none" respectively.

User Self-Care operations

Security Verify Access provides some pre-defined authentication policies that can be used to enable certain User Self-Care operations, such as account creation, password reset, and lost ID retrieval.

You can modify and customize these pre-defined authentication policies to suit your particular needs, for example:

- The OTP mechanism can be removed from the Account Create flow or the Passkey Account Create flow so that an OTP is no longer required when a user creates an account.
- The secondary attribute that is used in the Password Reset and Lost ID flows can be changed from **`surname`** to another attribute.
- The required attributes that are collected when an account is created during the Account Create flow can be modified.

You can also use these pre-defined authentication policies as building blocks to compose new User Self-Care scenarios. The logic and server-side processing is almost entirely performed in JavaScript mapping rules. These existing rules can be extended or transformed. New rules can also be written to implement different features.

The pre-defined authentication policies all use HTML templates, which can be customized to match your application. Each policy includes a list of the HTML templates that are used.

Form pre-population and handling of responses is performed by JavaScript mapping rules. Each policy includes a list of the JavaScript mapping rules that are used.

Prerequisites

Before you use any of the authentication policies to achieve SCIM account management, you must complete the following prerequisite setup:

- Create a server connection to the SCIM application endpoint.
  1. Log in to the local management interface.
  2. Click **AAC**
  3. Under **Global Settings**, click **Server Connections**.
  4. Click **New** and select **Web Service**.
  5. Specify the details of your SCIM application endpoint.

     If you are using the SCIM application included with Security Verify Access, the URL is `<runtime_endpoint>/scim`. The user name and password can be that of any user who is part of the Administration Group that is defined in the SCIM Configuration.
  6. Save and deploy the changes.
- Configure the authentication policies to make use of the SCIM application endpoint.
  1. Log in to the local management interface.
  2. Click **AAC**
  3. Under **Policy**, click **Authentication**.
  4. Click **Mechanisms**.
  5. Select the **SCIM Endpoint Configuration** mechanism and click **Edit**.
  6. On the **Properties** tab in the **Modify Authentication Mechanism** window, select the **Server Connection** entry and click **Edit**.
  7. In the **Modify Property** window, select from the list the server connection that was created in the previous step and click **OK**.
  8. Save and deploy the changes.

Account Create policy

The Account Create policy enables users to create new accounts for themselves. This policy uses the ReCAPTCHA mechanism to verify that the requests originate from a human and an Email OTP to ensure that a valid email address is being used.

Account Create setup

- Ensure that the "Prerequisites" on page 223 steps are completed.
- Configure the reCAPTCHA Verification mechanism. See Configuring the reCAPTCHA Verification authentication mechanism.
- Configure the Email OTP delivery mechanism to be used in the Account Create authentication policy.
  1. Log in to the local management interface.
  2. Click **AAC**.
  3. Under **Policy**, click **Authentication**.
  4. Click **Mechanisms**.
  5. Click **Email One-time Password**.
  6. Click **Edit**.
  7. Select the **Properties** tab and configure the connection to the SMTP server.

HTML templates

- `authsvc/usc/account-create/collectEmail.html`

- authsvc/authentictor/macotp/login.html
- authsvc/usc/account-create/collectProfile.html
- authsvc/usc/account-create/success.html

JavaScript Mapping Rules

- USC_CreateAccount_CollectEmail
- USC_CreateAccount_CollectProfile
- USC_CreateAccount_Success

Creating a new account workflow

The Account Create authentication policy enables users to create new accounts with the following workflow.

Note: The new accounts that are created under this workflow are of the type "basic users".

1. The user accesses `https://<WebSEAL host>:<port>/mga/sps/authsvc?PolicyId=urn:ibm:security:authentication:asf:uscAccountCreate`
2. On this screen, the user is prompted to enter an email address and CAPTCHA.
   - The template page that is presented is `authsvc/usc/account-create/collectEmail.html`
   - The JavaScript that pre-populates the form and validates responses is `USC_CreateAccount_CollectEmail`
3. On the next screen, the user is prompted to enter an OTP.
   - The OTP is delivered via Email.
   - The template page that is presented is `authsvc/authentictor/macotp/login.html`
4. On the next screen, the user is presented with the enrollment form.
   - The template page that is presented is `authsvc/usc/account-create/collectProfile.html`
   - The JavaScript that pre-populates the form and validates responses is `USC_CreateAccount_CollectProfile`
5. On the next screen, the account success page is presented.
   - The template page that is presented is `authsvc/usc/account-create/success.html`
   - The JavaScript that pre-populates the template and ends the policy is `USC_CreateAccount_Success.`

Password Reset policy

The Password Reset authentication policy enables users to reset their passwords. This policy uses the ReCAPTCHA mechanism to verify that the request originates from a human. It also uses the Email OTP mechanism and a secondary attribute to ensure that only the account owner can reset the password.

Password Reset Setup

- Ensure that the "Prerequisites" on page 223 steps are completed.
- Configure the reCAPTCHA Verification mechanism. See Configuring the reCAPTCHA Verification authentication mechanism.
- Configure the Email OTP delivery mechanism to be used in the Password Reset authentication policy.
  1. Log in to the local management interface.
  2. Click **AAC**
  3. Under **Policy**, click **Authentication**.
  4. Click **Mechanisms**.

5. Click **Email One-time Password**.
6. Click **Edit**.
7. Select the **Properties** tab and configure the connection to the SMTP server.

HTML templates

- `authsvc/usc/password-reset/collectEmail.html`
- `authsvc/authentictor/macotp/login.html`
- `authsvc/usc/password-reset/collectPassword.html`
- `authsvc/usc/password-reset/success.html`

JavaScript Mapping Rules

- `USC_PasswordReset_CollectEmail`
- `USC_PasswordReset_CollectPassword`
- `USC_PasswordReset_Success`

Password reset workflow

The Password Reset authentication policy enables users to reset their lost or forgotten passwords with the following workflow.

1. The user accesses `https://<WebSEAL host>:<port>/mga/sps/authsvc?PolicyId=urn:ibm:security:authentication:asf:uscPasswordReset`
2. On this screen, the user is prompted to enter an email address, surname, and CAPTCHA.
   - The template page that is presented is `authsvc/usc/password-reset/collectEmail.html`
   - The JavaScript that pre-populates the form and validates responses is `USC_PasswordReset_CollectEmail`
   - If the email address and surname do not match any existing profile, a generic error is returned.
3. On the next screen, the user is prompted to enter an OTP.
   - The OTP is delivered through an email.
   - The template page that is presented is `authsvc/authentictor/macotp/login.html`
4. On the next screen, the user is presented with the password reset form.
   - The template page that is presented is `authsvc/usc/password-reset/collectPassword.html`
   - The JavaScript that pre-populates the form and validates responses is `USC_PasswordReset_CollectPassword`
5. On the next screen, the account success page is presented.
   - The template page that is presented is `authsvc/usc/password-reset/success.html`
   - The JavaScript that pre-populates the page is `USC_PasswordReset_Success`

Lost ID policy

The Lost ID authentication policy enables users to retrieve their lost or forgotten user IDs. This policy uses the reCAPTCHA mechanism to verify that the request originates from a human and a secondary attribute to ensure that only the account owner can start the process. The lost ID is emailed to the user. None of the user's account information is displayed in the browser.

Lost ID Setup

- Ensure that the “Prerequisites” on page 223 steps are completed.

- Configure the reCAPTCHA Verification mechanism. See Configuring the reCAPTCHA Verification authentication mechanism.
- Create an SMTP Server Connection
  1. Log in to the local management interface.
  2. Click **AAC**
  3. Under **Global Settings**, click **Server Connections**.
  4. Click the **New Server Connection** icon.
  5. Select **SMTP**.
  6. Complete the connection details of your SMTP server.
  7. Click **Save**.
  8. Deploy these changes before you continue to the next step.
- Configure the Session Attribute Response mechanism to be used in the Lost ID authentication policy.
  1. Log in to the local management interface.
  2. Click **AAC**
  3. Under **Policy**, click **Authentication**.
  4. Click **Mechanisms**.
  5. Select **USC Lost ID - Send ID**.
  6. Click **Edit**.
  7. In the dialog window, select the **Properties** tab.
  8. Select the **Server Connection** property and click **Edit**.
  9. In the dialog window, select the SMTP server connection that is created in the previous step.
  10. Click **Save**.
  11. Deploy the changes.

HTML templates

- `authsvc/usc/lost-id/collectEmail.html`
- `authsvc/authenticator/email_message/error.html`
- `authsvc/usc/lost-id/success.html`

Email templates

- `authsvc/usc/lost-id/email.xml`

JavaScript Mapping Rules

- `USC_LostId_CollectEmail`
- `USC_LostId_Success`

Lost ID workflow

Upon completion of this flow, the user receives an email message that contains the lost or forgotten user ID.

1. The user accesses `https://<WebSEAL host>:<port>/mga/sps/authsvc?PolicyId=urn:ibm:security:authentication:asf:uscLostId`
2. On this screen, the user is prompted to enter an email address, surname, and complete the CAPTCHA.
   - The template page that is presented is `authsvc/usc/lost-id/collectEmail.html`

- The JavaScript that pre-populates the form and validates responses is `USC_LostId_CollectEmail`
- If the email address and surname do not match any existing profile, a generic error is returned.

3. An email that contains the lost or forgotten user ID is sent to the user.
4. On the next screen, the operation success page is presented.
   - The template page that is presented is `authsvc/usc/lost-id/success.html`
   - The JavaScript that pre-populates the page is `USC_LostId_Success`

Passkey Account Create policy

The Passkey Account Create policy enables users to create new accounts for themselves. This policy uses the ReCAPTCHA mechanism to verify that the requests originate from a human and an email OTP to ensure that a valid email address is being used. Where possible, passkey enrollment is offered to users as the first factor authentication method instead of the traditional password.

Account Create setup

- Ensure that the "Prerequisites" on page 223 steps are completed.
- Enable SCIM Verify Access User.
  1. Log in to the local management interface.
  2. Click **AAC**.
  3. Under **Manage**, click **SCIM Configuration**.
  4. Click **Verify Access User**.
  5. Check **Enable Verify Access Integration**.
  6. Select the SCIM server connection from the **Verify Access User Registry** dropdown.
  7. Check **Update Native Users**.
- Configure the reCAPTCHA Verification mechanism. See Configuring the reCAPTCHA Verification authentication mechanism.
- Configure the email OTP delivery mechanism to be used in the Account Create authentication policy.
  1. Log in to the local management interface.
  2. Click **AAC**.
  3. Under **Policy**, click **Authentication**.
  4. Click **Mechanisms**.
  5. Click **Email One-time Password**.
  6. Click **Edit**.
  7. Select the **Properties** tab and configure the connection to the SMTP server.
- Configure a FIDO2 Relying Party.
  1. Log in to the local management interface.
  2. Click **AAC**.
  3. Under **Manage**, click **FIDO2 Configuration**.
  4. Enter a **Display Name** and **Relying Party ID**.
  5. Click **Next**.
  6. Click **Save**.
  7. Click **AAC**.
  8. Under **Policy**, click **Authentication**.
  9. Click **Mechanisms**.

10. Click **FIDO2 WebAuthn Registration**.
11. Click **Edit**.
12. Select the **Properties** tab and configure the **Relying Party Config ID**.

HTML templates

- `authsvc/usc/passkey/collect_email.html`
- `authsvc/usc/passkey/collect_password.html`
- `authsvc/usc/passkey/collect_profile.html`
- `authsvc/usc/passkey/email_otp.html`
- `authsvc/usc/passkey/passkey_reg_decision.html`
- `authsvc/usc/passkey/success.html`
- `static/css/usc/passkey.css`
- `static/scripts/usc/passkey/collect_email.js`
- `static/scripts/usc/passkey/collect_password.js`
- `static/scripts/usc/passkey/collect_profile.js`
- `static/scripts/usc/passkey/email_otp.js`
- `static/scripts/usc/passkey/passkey_reg_decision.js`
- `static/scripts/usc/passkey/success.js`

JavaScript Mapping Rules

- `USC_Common`
- `USC_Passkey_CollectEmail`
- `USC_Passkey_CollectPassword`
- `USC_Passkey_CollectProfile`
- `USC_Passkey_Reg_Decision`
- `USC_Passkey_Success`

Creating a new account workflow

The Passkey Account Create authentication policy enables users to create new accounts with the following workflow.

1. The user accesses the Passkey Account Create policy.
2. On this screen, the user is prompted to enter an email address and CAPTCHA.
   - The template page that is presented is `authsvc/usc/passkey/collect_email.html.`
   - The JavaScript that pre-populates the form and validates responses is `USC_Passkey_CollectEmail.`
3. On the next screen, the user is prompted to enter an OTP.
   - The OTP is delivered by an email.
   - The template page that is presented is `authsvc/usc/passkey/email_otp.html.`
4. On the next screen, the user is presented with the enrollment form.
   - The template page that is presented is `authsvc/usc/passkey/collect_profile.html.`
   - The JavaScript that pre-populates the form and validates responses is `USC_Passkey_CollectProfile.`
5. On the next screen, if the browser has user-verifying platform authenticator (UVPA) capability, the user is prompted to enroll a passkey.

- The template page that is presented is `authsvc/usc/passkey/passkey_reg_decision.html`.
- The JavaScript that prompts the user to enroll is `USC_Passkey_Reg_Decision`.

6. If the user chooses to enroll a passkey, the user is directed to the FIDO2 WebAuthn Registration mechanism.
7. If the user chooses to skip passkey enrolment, and the user did not enter a password on the collect password page, the user is prompted to enter a password.
   - The template page that is presented is `authsvc/usc/passkey/collect_password.html`
   - The JavaScript that pre-populates the form and validates responses is `USC_Passkey_CollectPassword`
8. On the next screen, the account success page is presented.
   - The template page that is presented is `authsvc/usc/passkey/success.html`
   - The JavaScript that pre-populates the template and ends the policy is `USC_Passkey_Success`

Disabling and re-enabling a predefined User Self-Care policy

The appliance SCIM component provides some predefined User Self Care policies. You can use some of them and disable others.

Disabling a User Self-Care policy

Follow these steps to disable a specific User Self Care policy.

Procedure

1. Replace the mapping rule.

   Replace the first mapping rule in the policy you want to disable with the following snippet .

```
/*
* Disable this policy.
*/
success.setValue(false);
page.setValue("/authsvc/usc/disabled.html");
```

   A list of policies and their first mapping rule is shown as follows:

Table 45. Policies and their first mapping rule

Policy	Mapping rule name
Account Create	`USC_CreateAccount_CollectEmail`
Lost ID	`USC_LostId_CollectEmail`
Lost Password	`USC_PasswordReset_CollectEmail`

   This snippet will cause the policy to stop and return the template page `disabled.html` when it is first accessed.

   You can modify the mapping rules using the Mapping Rules page in the management UI at `<appliance>/mga/mapping_rules`.

2. Add a new HTML template for the disabled page.

   Add a new HTML template at `<locale>/authsvc/usc/disabled.html`. The following page is an example of such template.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/
loose.dtd">
<!--
  **********************************************************************
  *   Licensed Materials - Property of IBM
```

```
  *   (C) Copyright IBM Corp. 2016. All Rights Reserved
  *
  *   US Government Users Restricted Rights - Use, duplication, or
  *   disclosure restricted by GSA ADP Schedule Contract with
  *   IBM Corp.
  **********************************************************************
  -->
<HTML>
<HEAD>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <TITLE>Policy Disabled</TITLE>
  <!--
    /authsvc/usc/disabled.html
  -->
  <LINK REL="stylesheet" TYPE="text/css" HREF="/sps/static/styles.css">
  <LINK REL="stylesheet" TYPE="text/css" HREF="/sps/static/usc.css">
  <SCRIPT TYPE="text/javascript">
  function goHome() {
    window.location.assign("/");
  }
  </SCRIPT>
</HEAD>
<BODY>
  <DIV CLASS="header">
    <DIV CLASS="prodname">IBM Security Verify Access - Policy Disabled</DIV>
    <SPAN CLASS="headerLogo"><DIV></DIV></SPAN>
  </DIV>
  <DIV CLASS="content">
    <DIV CLASS="contentHeader">
      <h1 CLASS="pageTitle">Policy Disabled</h1>
      <DIV CLASS="instructions">The administrator has disbled this policy.</DIV>
    </DIV>
    <DIV CLASS="pageContent">
      <P>Return to the <A HREF="#" ONCLICK="goHome()">home page</A> to log in.</P>
    </DIV>
  </DIV>
</BODY>
</HTML>
```

You can add new template files using the **Template Files** page in the management UI at `<appliance>/mga/template_files`.

Re-enabling a User Self-Care policy

You can re-enable a User Self-Care policy by restoring the original mapping rule contents.

The original mapping rule files can be retrieved from the **File Downloads** page in the management UI at `<appliance>/isam/downloads`.

The mapping rules are stored under the following path:

```
/access_control/examples/mapping_rules/
```

A list of the mapping rules and their corresponding file name is shown as follows:

Table 46. Mapping rules and file names

Mapping rule name	**File name**
USC_CreateAccount_CollectEmail	usc_ac_collect_email.js
USC_LostId_CollectEmail	usc_li_collect_email.js
USC_PasswordReset_CollectEmail	usc_pr_collect_email.js

Chapter 19. Configuring Password Vault

Use the password vault configuration page to enable and configure the Password Vault API. The configuration page provides an interface for storing username and password credentials in an encrypted format for each configured resource.

About this task

When the password vault is configured, stored credentials can be accessed or modified by an authenticated user with the URL:

```
https://<runtime server>/sps/pwdvault/{resource}/user/{url_b64encode_username}
```

When a WebSEAL instance is configured with AAC, a user is able to manage their stored credentials with the URL:

```
https://<webseal>/<junction>/sps/pwdvault/{resource}/user/{url_b64encode_username}
```

The request and response format conforms to the API that is documented by IBM Security Verify. For more information on Verify API, see https://docs.verify.ibm.com/verify/reference/updatepwdvaultenrollment_10.

When the password vault is configured, stored credentials can be accessed or modified by an authenticated user with the URL:

```
https://<runtime server>/sps/pwdvault/{resource}/user/{url_b64encode_username}
```

When a WebSEAL instance is configured with AAC, a user is able to manage their stored credentials with the URL:

```
https://<webseal>/<junction>/sps/pwdvault/{resource}/user/{url_b64encode_username}
```

The request and response format conforms to the API that is documented by IBM Security Verify. For more information on Verify API, see https://docs.verify.ibm.com/verify/reference/updatepwdvaultenrollment_10.

IBM Security Verify Access provides a password vault API that provides username and password single sign-on (SSO) capabilities for a specified list of resources. The following configuration options allow the functionality to be managed:

Enable password vault
: Turn the password vault API on or off.

Resources
: Specify the list of resources for which the password vault API will be invoked.

Data Location
: Specify the storage location for the username and password information. This can be either the runtime database or the traditional GSO lock-box.

Public Key
: Optionally specify a certificate from the `rt_profile_keys` key database that will be used to generate a JWE from the password.

Admin Group
: Specify the group that is used to allow admin privileges for a user. For example, "adminGroup".

Procedure

1. Log in to the local management interface.
2. From the top menu, select **AAC** > **Manage** > **Password Vault Configuration**.

The current password vault configuration is shown.

3. To update the configuration, modify the values as required.
4. Click **Save** to save any configuration changes.

Chapter 20. Managing LTPA keys

You can create, import, export, and delete LTPA key files that are used by the LTPA token conversion module.

Before you begin

Ensure that your browser allows pop-up windows to be displayed.

Procedure

1. Log in to the local management interface.
2. Click **Federation** or **AAC**.
3. Under **Global Keys**, click **LTPA Keys**.
4. Perform any of the following actions:

   Importing an LTPA key:

   a. Click **Manage** > **Import**.
   b. Click **Browse**.
   c. Select the file that you want to import.
   d. Click **Import**.

   Exporting an LTPA key:

   a. Click **Browse**.
   b. Select the file that you want to export.
   c. Click **Manage** > **Export**.
   d. Confirm that you want to save the file to your local workstation.

   Deleting an LTPA key:

   a. Select the file that you want to delete.
   b. Click **Delete**.
   c. Click **Yes** when you are prompted to confirm the deletion.
5. Deploy the changes as described in Configuration changes commit process.

Accessibility features for Security Verify Access

Accessibility features assist users who have a disability, such as restricted mobility or limited vision, to use information technology content successfully.

Accessibility features

Security Verify Access includes the following major accessibility features:

Accessibility features
Supports interfaces commonly used by screen readers. This feature applies to applications on Windows operating systems only.
Can be operated by using only the keyboard.
Allows the user to request more time to complete timed responses.
Supports customization of display attributes such as color, contrast, and font size.
Communicates all information independently of color.
Supports interfaces commonly used by screen magnifiers. This feature applies to applications on Windows operating systems only.
Allows the user to access the interfaces without inducing seizures due to photosensitivity.

Security Verify Access uses the latest W3C Standard, WAI-ARIA 1.0 (http://www.w3.org/TR/wai-aria/), to ensure compliance to US Section 508 (http://www.access-board.gov/guidelines-and-standards/communications-and-it/about-the-section-508-standards/section-508-standards), and Web Content Accessibility Guidelines (WCAG) 2.0 (http://www.w3.org/TR/WCAG20/). To take advantage of accessibility features, use the latest release of your screen reader in combination with the latest web browser that is supported by this product.

The Security Verify Access online product documentation in IBM Knowledge Center is enabled for accessibility. The accessibility features of IBM Knowledge Center are described at https://www.ibm.com/support/knowledgecenter/help?view=kc#accessibility.

Keyboard navigation

This product uses standard navigation keys.

Interface information

The Security Verify Access user interfaces do not have content that flashes 2 - 55 times per second.

The Security Verify Access web user interfaces and the IBM Knowledge Center rely on cascading style sheets to render content properly and to provide a usable experience. The application provides an equivalent way for low-vision users to use a user's system display settings, including high-contrast mode. You can control font size by using the device or web browser settings.

The Security Verify Access web user interface includes WAI-ARIA navigational landmarks that you can use to quickly navigate to functional areas in the application.

Related accessibility information

In addition to standard IBM help desk and support websites, IBM has established a TTY telephone service for use by deaf or hard of hearing customers to access sales and support services:

TTY service 800-IBM-3383 (800-426-3383) (within North America)

IBM and accessibility

For more information about the commitment that IBM has to accessibility, see IBM Accessibility (www.ibm.com/able).

Index

E

F

H

I

J

K

L

O

P

R

S

T

U

W