IBM Security Verify Access
Version 10.0.9
January 2025

Development topics

Contents

Part 1. IBM Security Verify Access development.....1

Chapter 1. Web administration Java classes API documentation..... 3

Chapter 2. Web authorization Java classes API documentation.....5

Chapter 3. Setting up a Verify Access Runtime for Java system..... 7

Chapter 4. Installing IBM Security Verify Access Runtime for Java.....9

Part 2. Advanced Access Control Development.....11

Chapter 5. REST services for OTP secret keys 13

Chapter 6. REST services for knowledge questions..... 15

Chapter 7. REST services for device fingerprint registration.....19

Chapter 8. Software Development Kit.....21

Index..... 23

Part 1. IBM Security Verify Access development

Chapter 1. Web administration Java classes API documentation

Use the IBM Security Access Manager administration Java classes API to add administration tasks to new or existing Java applications.

You can access the following documentation from the Support site:

- Administration Java Classes Developer Reference
- Javadoc API reference

Chapter 2. Web authorization Java classes API documentation

Use the IBM Security Access Manager authorization Java classes API to add product authorization and security services to new or existing Java applications.

You can access the following documentation from the Support site:

- Authorization Java Classes Developer Reference
- Javadoc API reference

Chapter 3. Setting up a Verify Access Runtime for Java system

The Verify Access Runtime for Java offers a reliable environment for developing and deploying Java™ applications in a Security Verify Access secure domain. Use it to add Security Verify Access authorization and security services to new or existing Java applications.

Set up this system by following the instructions that are appropriate for your operating system.

IBM Security Verify Access Runtime for Java configures additional security features into the specified JRE.

Note:

1. IBM Security Verify Access Runtime for Java supports the following Java runtime environments (JREs):
   - IBM® Java Runtime
   - The JRE provided with WebSphere® Application Server
   - OpenJDK
2. If you reinstall and reconfigure the Security Verify Access policy server, or install any IBM WebSphere Application Server patches, you must unconfigure and reconfigure IBM Security Verify Access Runtime for Java.

Chapter 4. Installing IBM Security Verify Access Runtime for Java

Extract the contents of the runtime package file to install the IBM Security Verify Access Runtime for Java system and use the **pdjrtecfg** utility to configure it.

Procedure

1. Log on as **root** for UNIX systems or a user with Administrator group privileges for Windows systems.
2. Download the runtime package file from the appliance.
   a. Go to **System** > **File Downloads**.
   b. Download the runtime package file `pdjrte-10.0.0.zip` from the **Verify Access** directory.
3. Extract the contents of the `pdjrte-10.0.0.zip` file.
4. Ensure that either IBM Java Runtime or the JRE provided with WebSphere Application Server is installed.
   IBM Security Verify Access Runtime Runtime for Java configures extra security features into the specified JRE and only these two JREs are supported.
5. To set up IBM Security Verify Access Runtime for Java with a configuration type of **Full**, ensure that both the policy server and registry server are running. If the configuration type is **standalone**, this step is not required.
6. Before you configure the IBM Security Verify Access Runtime for Java component, ensure that either the IBM Java Runtime or the JRE provided with WebSphere Application Server can be located by using the PATH environment variable.
7. To configure the IBM Security Verify Access Runtime for Java component, run the `pdjrtecfg` utility.

   On UNIX systems, use the following command:

   ```
   pdjrtecfg -action config -interactive
   ```

   On Windows systems, use the following command:

   ```
   pdjrtecfg.bat -action config -interactive
   ```

Results

This step completes the setup of the IBM Security Verify Access Runtime for Java component.

Part 2. Advanced Access Control Development

Chapter 5. REST services for OTP secret keys

You can use the REST services capability to help manage your mobile data, such as your HOTP and TOTP secret keys.

To prevent unauthorized users from confiscating and resetting the secrets keys that belong to authorized users, the administrator must complete the following steps:

1. Write a policy that requires a form of two-factor authentication other than the following authentication types:
   - HOTP
   - TOTP
2. Attach the policy to the OTP management URLs.

Note: The user must authenticate to use the REST services capability.

The REST services return a deactivated message instead of the QR code or secret key payload when the advanced configuration property `otp.usc.legacyBehavior` is set to `false`. See Advanced configuration properties.

REST services usage scenarios

Depending on your usage scenario, type the following URLs into the web page that calls the REST services:

Table 1. REST services instructions

Method	URL	Response	Response type
GET	`https://`*`hostname`*`/mga/sps/mga/user/mgmt/otp/{`*`otpType`*`}` **Note:** Valid values for *`otpType`* include the following values: 1. `totp` 2. `hotp`	`{"username":` *`username`*`, "secretKey":` *`secretKey`*`, "secretKeyUrl":` *`secretKeyUrl`*`}` If the request completes successfully, the HTTP response code is `200`. If the request does not complete successfully, the HTTP response is `500`.	`application/json`
GET	`https://`*`hostname`*`/mga/sps/mga/user/mgmt/otp/qr/{`*`otpType`*`}` **Note:** Valid values for *`otpType`* include the following values: 1. `totp` 2. `hotp`	Quick response (QR) code If the request completes successfully, the HTTP response code is `200`. If the request does not complete successfully, the HTTP response is `500`.	`image/gif`

Table 1. REST services instructions (continued)			
Method	**URL**	**Response**	**Response type**
DELETE	https://*hostname*/mga/sps/mga/user/mgmt/otp/{*otpType*} **Note:** Valid values for *otpType* include the following values: 1. totp 2. hotp	{"result": *message*} If the request completes successfully, the HTTP response code is 200. If the request does not complete successfully, the HTTP response is 500. The following message is in the JSON response: FBTRBA168E The HMAC OTP secret key could not be reset.	application / json

Chapter 6. REST services for knowledge questions

You can use REST services to manage knowledge questions.

Users can perform self care management operations to retrieve, create, update, and delete knowledge questions. You can use REST services to support these operations.

The knowledge question attributes are:

- `id`

  The unique ID of the question
- `answer`

  The user-supplied answer to the question
- `question`

  The text of the user defined question

The base URL for all the self-care management REST interfaces for knowledge questions is:

```
https://<WebSEAL host>:<port>/<junction_name>/sps/mga/user/mgmt/
```

The REST interfaces represent a collection of questions. The URI template is `/questions`.

Note:

- For all methods, the response type for Accept Header and Content-Type Header is `application/json`.
- Sorting, filtering, and paging are not supported.
- Users must authenticate in order to use the REST services.

Table 2. REST interface for managing knowledge questions

Method	Operation	URL	Response
GET	Retrieve a list of questions for a user.	`http://<WebSEAL host>:<port>/<junction_name>/sps/mga/user/mgmt/questions`	Response code: `200 OK` The response is a JSON array of questions and JSON objects. To view an example response, see "User questions REST model" on page 16
POST	Create all the questions for a user.		Response code: `201 CREATED`
PUT	Update all the questions for a user		Response code: `204 NO CONTENT`
DELETE	Delete all questions that are registered for a user		

Table 3. Error responses	
Method	**Error Response**
POST	Response code: `400 BAD REQUEST` Response type: `application/json` Explanation: Malformed question found. Possible reasons: • No answer submitted. • Non-Unique Questions ID The ID for each question must be unique for each user. The ID does not have to be globally unique.
PUT	

User questions REST model

The following table lists the attributes for user knowledge questions.

Table 4. Attributes for user questions	
Attribute	**Examples**
Name: username Description: the username Data type: String **Note:** Only provided on the response for a GET	"sec_master"
Name: questions Description: the list of knowledge questions Data Type: JSON array of JSON objects **Note:** • The caller supplies the `id` value. However, for PUT or POST methods if `id` is not provided the server generates a unique `id` for the question. • The parameter `question` is optional • The `answer` attribute will be obfuscated (replaced by the asterisk symbol "*") on responses.	Example questions: `[ { "id": "1", "answer" : "Smith", "question" : "What is your mother's maiden name?" }, { "id": "2", "answer" : "Bandit", "question" : "What is your Pet's name?" } ]`
Name: result Description: An error message that indicates the nature of the failure. Datatype: String	`{ "result": "FBTRBA310E The user knowlege questions answer(s) could not be stored because a duplicate question unique identifier [1] was included on the user questions." }`

Example JSON representation of a collection of knowledge questions

The following example shows the JSON representation of a collection of knowledge questions that belong to a user.

```
{
  "username": "sec_master",
  "questions" : [
                 {
                  "id"    : "1",
                  "answer"  : "******",
                  "question"   : "What is your mother's maiden name?"
                 },
                 {
                  "id"    : "2",
                  "answer"  : ********,
                  "question"   : "What is your Pet's name?"
                 }
                ]
}
```

Chapter 7. REST services for device fingerprint registration

You can use the REST services capability to manage your mobile data such as your registered devices.

Note: The user must authenticate to use the REST services capability.

REST services usage scenarios

Depending on your usage scenario, type the following URLs into the web page that calls the REST services:

Table 5. REST services instructions

Method	URL	Request	Response	Response type
GET	https://*hostname*/mga/sps/mga/user/mgmt/device	There is no JSON payload included in the request.	{"username": *username*, "devices":[{"name":*deviceName*, "id":*deviceId*, "lastUsedTime":*lastUsedTime*, "isEnabled": {true | false}}, ...]} If the request completes successfully, the HTTP response code is 200. If the request does not complete successfully, the HTTP response is 500.	application /json
GET	https://*hostname*/mga/sps/mga/user/mgmt/device/{*deviceId*}	There is no JSON payload included in the request.	{"username": *username*, "name":*deviceName*, "attributes":[{"name": *attributeName1*, "value": *attributeValue1*}, ... , {"name":*attributeNameN*, "value":*attributeValueN*}]} If the request completes successfully, the HTTP response code is 200. If the request does not complete successfully, the HTTP response is 500.	application /json

Table 5. REST services instructions (continued)				
Method	**URL**	**Request**	**Response**	**Response type**
DELETE	https://*hostname*/mga/sps/mga/user/mgmt/device/{*deviceId*}	There is no JSON payload included in the request.	{"result": *message*} If the request completes successfully, the HTTP response code is 200. If the request does not complete successfully, the HTTP response is 500. One of the following messages is in the JSON response: • FBTRBA164E The device *device name* could not be removed. • FBTRBA163I The device *device name* was removed successfully.	application /json
PUT	https://*hostname*/mga/sps/mga/user/mgmt/device/{*deviceId*}	{"name": "*new_name*", "isEnabled": {true|false}}	{"result": *message*} If the request completes successfully, the HTTP response code is 200. The following message is in the JSON response: FBTRBA165I The device <Variable formatSpec="{0}">name</Variable> was updated successfully. If you give the device an invalid name, the HTTP response code is 400. The following message is in the JSON response: FBTRBA182E The value *device name* is invalid. If you give the device a name that belongs to another device, the response code is 400. The following message is in the JSON response: FBTRBA186E A device named '<Variable formatSpec="{0}">device name</Variable>' already exists.	application /json

Chapter 8. Software Development Kit

Use the Software Development Kit so that you can create customized components such as your own obligation handler or authentication mechanism.

The Advanced Access Control Software Development Kit includes extensions that you can use to create an obligation handler or an authentication mechanism.

Use the obligation handler extension point so that obligations are handled at the policy decision point before it returns the result to the policy enforcement point application. Use the authentication mechanism extension point to create conditions that successfully authenticate a user.

Extensions that are created by using the Software Development Kit are run in a restricted environment in to prevent accidental damage to appliance resources. This step is done by denying access to function calls that require certain permissions. If your implementations call to such functions, directly or indirectly, the code execution is stopped and an `AccessControlException` error is displayed.

The following are the restricted permissions that cause exceptions:

- `java.io.FilePermission("<<ALL FILES>>", "read,write"));`
- `java.net.SocketPermission("", "listen"));`
- `java.lang.RuntimePermission("setSecurityManager"));`

For information about managing extensions in the local management interface, see Extensions.

Index

A

API documentation
- Web administration Java classes 3
- Web authorization Java classes 5

J

Java classes
- Web administration
  - API documentation 3
- Web authorization
  - API documentation 5

K

knowledge questions
- rest services 15

R

rest services for device fingerprint registration 19
rest services for otp secret keys
- hotp secret keys 13
- totp secret keys 13

runtime for java
- setting up 7

S

software development kit
- authentication mechanism 21
- server-side obligation handler 21

W

Web administration
- Java classes
  - API documentation 3

Web authorization
- Java classes
  - API documentation 5

IBM®