# IBM Multi-Cloud Data Encryption Powered by SPx®
# Version 2.3

## *REST API Specification*

GI13-4922-00

**Note**

Before you use this information and the product it supports, read the information in “Notices” on page 189.

This edition applies to Version 2.3 of IBM Multi-Cloud Data Encryption (product number 5737-C67) and to all subsequent releases and modifications until otherwise indicated in new editions.

# Contents


**Chapter 1. Overview** ..... **1**

**Chapter 2. Basic Requirements** ..... **3**
- Application Layer Protocol ..... 3
  - Version ..... 3
  - Supported methods ..... 3
- Connection security ..... 3
  - Encryption ..... 3
  - User authentication ..... 3
  - Cross Site Request Forgery (CSRF) protection ..... 4
- Payload format ..... 4
- URLs, URL patterns, resource orientation, and other topics ..... 4
  - Resource Orientation ..... 4
  - Job handling ..... 5

**Chapter 3. Base Resource Types** ..... **7**
- Primitives ..... 7
- Collections ..... 7
  - Schema ..... 7
  - Examples ..... 7

**Chapter 4. Response Status Codes** ..... **9**

**Chapter 5. Log Resources** ..... **11**
- /logs ..... 11
  - GET ..... 11
- /logs/events/ ..... 12
  - GET ..... 12
- /logs/events/{sequence_number} ..... 14
  - GET ..... 14
- /logs/debug/ ..... 15
  - GET ..... 15
- /logs/debug/{log_id} ..... 16
  - GET ..... 16
- /logs/recipients ..... 17
  - GET ..... 17
  - POST ..... 18
- /logs/recipients/{id} ..... 19
  - GET ..... 19
  - DELETE ..... 19
  - PATCH ..... 20
  - PUT ..... 21

**Chapter 6. Jobs** ..... **23**
- /jobs ..... 23
  - GET ..... 23
- /jobs/schedules ..... 24
  - GET ..... 24
  - POST ..... 25
- /jobs/schedules/{id} ..... 31

GET....31
PATCH....32
PUT....33
/jobs/schedulers....34
GET....34
/jobs/schedulers/{id}....37
GET....37
PATCH....38
PUT....39
/jobs/schedulers/{id}/actions....41
GET....41
POST....42
/jobs/history[r]....42
GET....42
/jobs/history/{id}....44
GET....44

**Chapter 7. Policy Resources....47**
/pathsets....47
GET....47
POST....48
/pathsets/{id}....49
GET....49
DELETE....50
PATCH....51
PUT....52
/processes....53
GET....53
POST....55
/processes/{id}....56
GET....56
DELETE....57
PATCH....58
PUT....59
/selectors....60
GET....60
POST....62
/selectors/{id}....63
GET....63
DELETE....64
PATCH....65
PUT....66
/keys....67
GET....67
POST....69
/keys/{id}....69
GET....70
PATCH....70
PUT....71
/groupscopes....72
GET....72
/groupscopes/{name}....73
GET....74
PATCH....74
PUT....75
/groupsets....76
GET....76

POST ... 78
/groupsets/{id} ... 78
GET ... 78
DELETE ... 79
/datatypes ... 80
GET ... 80
POST ... 83
/datatypes/{id} ... 84
GET ... 84
DELETE ... 85
PATCH ... 86
PUT ... 87

**Chapter 8. Managed Agents ... 89**
/agents ... 89
GET ... 89
POST ... 94
/agents/{id} ... 96
GET ... 96
DELETE ... 97
PATCH ... 98
/agents/{id}/certificates ... 99
GET ... 99
POST ... 100
/agents/{id}/certificates/{certid} ... 101
GET ... 101
DELETE ... 102
PATCH ... 102
PUT ... 103
/agents/{id}/install_bundle ... 104
GET ... 104
/agents/{id}/install_bundle/authorized_users ... 105
GET ... 105
PUT ... 106
/agents/{id}/policy_snapshots ... 107
GET ... 107
POST ... 108
/agents/{id}/policy_snapshots/{snapshot_id} ... 109
GET ... 109
PUT ... 110
DELETE ... 111
/agents/{id}/policy ... 112
GET ... 112
POST ... 113
/agents/{id}/policy/{binding_id} ... 114
GET ... 114
PUT ... 115
PATCH ... 117
DELETE ... 118
/agents/{id}/policy/{binding_id}/storage ... 118
GET ... 119
PUT ... 119
PATCH ... 120
/agents/{id}/features ... 121
GET ... 121
/agents/{id}/features/{feature} ... 122
GET ... 122

PUT ... 123
PATCH ... 124
/agents/{id}/tools ... 125
GET ... 125
POST ... 126
**/agents/{id}/tools/{tool_id}** ... 127
GET ... 127
PUT ... 128
PATCH ... 130
DELETE ... 131
**/agents/{id}/object_store** ... 132
GET ... 132
PUT ... 134
PATCH ... 136
/agents/{id}/object_store/credentials ... 136
/agents/{id}/object_store/buckets ... 138
/agents/{id}/object_store/buckets/<bid>/policy ... 141
/agents/{id}/policy/<pid> ... 142

**Chapter 9. User Accounts ... 145**
/users ... 145
GET ... 145
POST ... 146
/users/{username} ... 147
GET ... 147
DELETE ... 148
PATCH ... 148
PUT ... 149
/roles ... 150
GET ... 150
/roles/{id} ... 152
GET ... 152
/sessions ... 153
GET ... 153
POST ... 154
/sessions/{id} ... 155
GET ... 155
DELETE ... 156

**Chapter 10. Server Settings ... 157**
/settings ... 157
GET ... 157
/settings/directories ... 158
GET ... 158
POST ... 159
/settings/directories/{id} ... 160
GET ... 160
DELETE ... 160
PUT ... 161
/settings/keystores ... 162
GET ... 162
POST ... 164
/settings/keystores/{id} ... 165
GET ... 165
PUT ... 166
PATCH ... 167
DELETE ... 168

**Chapter 11. Archive and Upload Files..... 169**
/files..... 169
GET..... 169
/files/upload..... 170
GET..... 170
POST..... 171
/files/upload/{id}..... 171
GET..... 172
DELETE..... 172

**Chapter 12. Advanced Properties..... 175**
/properties..... 175
GET..... 175
/properties/{key}..... 176
GET..... 176
PATCH..... 177

**Chapter 13. Setup and Configuration Issues..... 179**
Setup and Configuration Issues..... 179
/issues..... 179
/issues/{id}..... 180

**Chapter 14. Globalization Resources..... 183**
/locales..... 183
GET..... 183
/locales/{id}..... 184
GET..... 184
/strings..... 185
GET..... 185
/strings/{id}..... 186
GET..... 186

**Notices..... 189**
Trademarks..... 191
Terms and conditions for product documentation..... 191
Privacy policy considerations..... 192

# Chapter 1. Overview

This specification presents requirements and architecture for a REST API for the MDE product.

This document assumes familiarity with general REST API concepts, and seeks to clarify the architectural choices made for MDE.

# Chapter 2. Basic Requirements

## Application Layer Protocol

### Version

The MDE REST API will be designed to be carried with HTTP 1.1, in keeping with overwhelming industry convention.

### Supported methods

The REST API will support the following request methods:

- GET
- PUT
- PATCH *(not HTTP defined, see https://tools.ietf.org/html/rfc5789)*
- POST
- DELETE

All other request methods will be responded to with the HTTP 405 (Not Implemented) status code.

## Connection security

### Encryption

The REST API will be exposed on a TLS 1.2+ secured connection only. Cipher suites, algorithms, and other parameters will be restricted to those approved under NIST sp800-131a and FIPS 140-2 guidelines.

### User authentication

The REST API will require login. Two authentication methods will be supported

1. **Session authentication** leveraging the form authentication methods exposed for the web UI and referenced on following requests using the session cookie and other headers (such as CSRF tokens) will be allowed. This will be done primarily to allow the web UI to leverage the REST API for its own function and to enable browser-based testing during development. There are no known reasons to try and disable this for non-browser clients in production, but it is not expected that customer scripting will use this method.

Starting an authenticated session (logging in) is done by sending a POST request (with valid CSRF token) to the sessions resource. Likewise, ending that session (logging out) is done by sending a DELETE request (with valid CSRF token) to the specific session resource for the currently logged in session. For more information, see the sessions resource session.

1. **HTTP Basic Authentication**

The REST API partially supports the Basic access authentication scheme as detailed in IETF RFC 1945 section 11.1 with the following provisos:

- Responses to unauthorized requests will not offer an authentication challenge (WWW-Authenticate header).
- Credentials are not cached for any period of time

- An HTTP session is not stored or started

That is to say that the REST API will accept an Authorization header of the prescribed form that will allow authenticated access for the duration of that single request. Follow-on requests must always provide the header and clients expecting to begin using the header only after having first received an authentication challenge will not function properly.

*Cross Site Request Forgery protection (as detailed in the next section) is not active for Basic authenticated requests.*

### Cross Site Request Forgery (CSRF) protection

REST API requests made using *session authentication* must be protected from CSRF attacks. Specifically the following will be implemented by the MDE and will apply to session authenticated requests:

- PATCH, PUT, POST, and DELETE requests are considered to be CSRF protected requests and must be accompanied by a valid CSRF token
- Two tokens will be retained in the active session. Either may be used but the most recent should be sent by the client whenever possible (the older token is only kept to avoid failure in certain race conditions)
- The currently valid token will be given to the client on every supported request as a cookie named Csrf-Token
- The valid token must be passed on CSRF protected requests in the X-Csrf-Token header
- Every CSRF protected request made using a given session will rotate the session CSRF tokens regardless of whether the request passed validation or not. The newest token will be set in the server response headers.
- A request passing an invalid token will be refused with HTTP status code 400 and the Warning header populated with text indicating the cause as CSRF token failure.
- CSRF tokens will 256 bit BASE64 encoded values randomly generated by the most secure available random number generator.

## Payload format

Unless otherwise indicated by a defined sub-resource in this specification, all resources are represented with the application/json content type. For those resources that support or require a different content type, the REST API will expect the HTTP Accept header to be set. Any other requested content types will respond with an empty response body and a 415 Unsupported Media Type status code

## URLs, URL patterns, resource orientation, and other topics

### Resource Orientation

The REST API will be highly resource oriented. The following principles will be adhered to unless stated otherwise:

- HTTP methods GET, PUT, POST, and DELETE are correlated with reading, updating, creating, and deleting resources
- The bodies of GET and PUT requests made to the same URL should be identical in syntax and structure except for any field updates requested in the PUT request
- POST requests most often are only valid against array resources and the body of such should be of the same type (whether language defined or API defined) as the other elements of the array resource unless otherwise defined by the specific API
- All changes in product configuration should follow a GET-Update-PUT pattern, meaning that the client should be able to perform a GET request on the URL with the resource / setting, update desired values

in the GET response text, and then use the updated response text as the body of a corresponding PUT request to the original URL.

- URLs and hierarchy within returned objects should follow each other, meaning:
  - Object fields should be directly accessible by URL by appending a slash (/) and the name of the field to the URL.

    For example: if /rest/resource returns {"id": 1, "name": "myResourceNumber1"} then /rest/resource/name should return "myResourceNumber1"
  - Array elements should be directly accessible by URL by appending a slash (/) and the value of the id field of the desired element to the URL.

    For example: if /rest/resources returns [{"id": 1}, {"id": 2}] then /rest/resources/1 should return {"id": 1}.

    **Note:** id fields are defined and may not be named "id"
- Array resources should use plural naming in URLs.

  For example, an array resource returning users may respond to /rest/user
- URL categorization should be avoided to prevent module implementation leak into the REST API resource hierarchy

  For example, /rest/users is preferred over /rest/user/list even if other user-related resources may be found under /rest/user

## Job handling

REST API requests may be long or short running. Any PUT, POST, or DELETE to a REST API that is expected to run long may optionally spawn a server side task / thread and then respond as follows:

- Location header set to a URL that will accept GET requests for status checks
- Status code 202 (Accepted)
- Response body optional

The job resource defined at the URL indicated by the location header will not accept PUT or POST requests. DELETE requests will be interpreted to either cancel the job or delete the post-run job record depending on the job state. All job resources are volatile with respect to the application and will not persist across application restarts.

# Chapter 3. Base Resource Types

## Primitives

The set of primitive types will be aligned with the set of primitive types available in the JSON language specification.

## Collections

### Schema

Collections are lists, sets, and arrays of other types. The items in collections will be represented in responses as arrays, with ordering (lists) or uniqueness (sets) constraints being enforced by the REST server rather than explicitly indicated by syntax.

```
{
    "type": "array",
    "minItems": 0,
    "items": { ... }
    "uniqueItems": true
}
```

### Examples

Collections will be presented in array structures such as this:

```
[
    {
        "type": "user",
        "id": "4"
    },
    {
        "type": "user",
        "id": "5"
    },
    {
        "type": "user",
        "id": "8"
    }
]
```

# Chapter 4. Response Status Codes

| Code | Message | Description |
|---|---|---|
| 200 | OK | The request successful. Reported content appears in the response body. |
| 201 | Created | A new resource was created. The **Location** header will be set to the URL of the created resource. |
| 202 | Accepted | The response would be unacceptably delayed if processed synchronously. A job was created to process the request. The **Location** header will be set to the URL of the job. |
| 204 | No Content | The request was successful, but no content is required in the response. This is most often returned for DELETE requests, but may also be used for degenerate PUT and PATCH requests. |
| 205 | Reset Content | The request was successful. Client side models of the target resource should be reset due to property change. This is most often returned for PUT and PATCH requests. |
| 400 | Bad Request | The request could not be processed due to errors in the request. |
| 401 | Unauthorized | The request is not authenticated. |
| 403 | Forbidden | The request is authenticated, but the user is not allowed to perform the requested operation. |
| 404 | Not Found | The requested resource does not exist, nor does the server retain any record of it ever having existed. |
| 405 | Method Not Allowed | The HTTP method is not valid for the target resource. |
| 406 | Not Acceptable | The content type requested in the **Accept** header is not supported. Currently application/json is the only accepted type. |

| | | |
|---|---|---|
| 409 | Conflict | Varies based on request method:<br>• POST – A resource with the same identity already exists. The parent resource is locked for changes.<br>• PATCH / PUT – An identity field is being changed and the new identity field is the same as another already used identity. The resource being updated was updated by another request in a way that invalidates this request. The resource is locked for changes.<br>• DELETE – The resource is locked for changes |
| 410 | Gone | A resource bearing the requested identity is known to have existed at one time, but has been deleted. |
| 424 | Failed Dependency | The requested change depends on another resource state that is not currently met. |
| 428 | Precondition Required | The request cannot be fulfilled until one or more other requests are sent and the state of another related is changed. |
| 500 | Internal Server Error | Fallback for internal server errors with no more specific failure code. The **Warning** header will contain information that should be reported when filing an error report. |
| 503 | Service Unavailable | An internal service needed to perform the request was unavailable. The **Warning** header will contain information that should be reported when filing an error report. |

# Chapter 5. Log Resources

## /logs

The collection of all known logs

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain a collection of references to child logs. The references will not contain any log entries themselves. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

```
{
    "type": "array",
    "minItems": 2,
    "uniqueItems": true,
    "items": [
        {
            "type": "object",
            "required": ["type","id", "href" ],
            "additionalProperties": true,
            "properties": {
                "type": { "enum": [ "ref" ] },
                "id": { "enum": [ "events", "debug", "recipients" ] },
                "href": "string"
            }
        }
    ]
}
```

#### Example

```
Request: GET /logs

Response: 200 OK

[
```

```
    {
        "type": "ref",
        "id": "event",
        "href": "/rest/logs/events",
    },
    {
        "type": "ref",
        "id": "debug",
        "href": "/rest/logs/debug",
    },
    {
        "type": "ref",
        "id": "recipients",
        "href": "/rest/logs/recipients",
    }
]
```

## /logs/events/

The system event and audit log

### GET

**Query parameters**

| Parameter | Description |
|---|---|
| **event_type** | Set to one of the values that an event object type field may take on to show only the log entries matching that type. Unrecognized values will be ignored. **Example: event_type=audit&event_type=system** |
| **order** | Order value that puts most recent events first if set to "descending". **Example: order=descending** |

**Supported content types**

| Content type | Description |
|---|---|
| **application/json** | Objects as described by schema below |
| **text/csv** | A comma separated value (CSV) representation of the log resources described below |

**Status codes**

| Code | Description |
|---|---|
| **200** | The response body will contain the system event and audit log entries as a collection. If a query filter is applied and no logs match, an empty collection is sent. |
| **400** | The current request is improperly formatted. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |

| | |
|---|---|
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

```
{
    "type": "array",
    "minItems": 0,
    "uniqueItems": true,
    "items": [
        {
            "type": "object",
            "required": ["type", "sequence_number", "event_id", "source",
                  "event_type", "severity", "timestamp", "message_args", "redacted" ],
            "additionalProperties": false,
            "properties": {
                "type": { "enum": [ "event" ] },
                "sequence_number": {
                    "type": "number",
                    "multipleOf": 1,
                    "minimum": 1
                },
                "event_id": "string",
                "source": "string",
                "event_type": { "enum": [ "AUDIT", "SYSTEM" ] },
                "severity": { "enum": [ "DEBUG", "INFO",
                              "WARN", "ERROR", "CRITICAL" ] },
                "timestamp": {
                    "type": "number",
                    "multipleOf": 1,
                    "minimum": 0
                },
                "message": "string",
                "message_args": {
                    "type": [ "array" ],
                    "minItems": 0,
                    "uniqueItems": false,
                    "items": [ { "type": [ "number", "string",
                             "boolean", "null" ] } ]
                },
                "redacted": "boolean"
            }
        }
    ]
}
```

**Example**

```
Request: GET /rest/logs/events?event_type=audit&event_type=system

Response: 200 OK

[
    {
        "type": "event",
        "sequence_number": 12,
        "event_id": "0X00020003",
        "source": "localhost",
        "event_type": "AUDIT",
        "severity": "INFO",
        "timestamp": 1442947787281,
        "message_args": [ "admin", "policy255" ],
        "redacted ": false
    },
    {
        "type": "event",
        "sequence_number": 14,
        "event_id": "0x00220334",
        "source": "192.168.4.234",
        "event_type": "SYSTEM",
        "severity": "CRITICAL",
        "timestamp": 1442947787282,
        "severity": "CRITICAL",
        "message": "Policy %s could not be applied: RC %s",
        "message_args": [ "policy255", 23 ],
        "redacted ": false
```

```
    }
]
```

## /logs/events/{sequence_number}

An individual event

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain the event object matching the {sequence_number} part of the URL. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | An event with the specified {sequence_number} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **410** | An event with the specified {sequence_number} was not found but has existed before |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

*See event log collection schema*

#### Example

```
Request: GET /rest/logs/events/12

Response: 200 OK

{
        "type": "event",
        "sequence_number": 12,
        "event_id": "0X00020003",
        "source": "localhost",
        "event_type": "AUDIT",
        "severity": "INFO",
        "timestamp": 1442947787281,
        "message_args": [ "admin", "policy255" ],
        "redacted ": false
}
```

## /logs/debug/

The system debug log

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain a collection of references to child logs. The references will not contain any log entries themselves. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

```
{
    "type": "array",
    "minItems": 0,
    "uniqueItems": true,
    "items": [
        {
            "type": "object",
            "required": ["type", "id", "href" ],
            "additionalProperties": true,
            "properties": {
                "type": { "enum": [ "ref" ] },
                "id": "string",
                "href": "string"
            }
        }
    ]
}
```

#### Example

```
Request: GET /rest/logs/debug

Response: 200 OK

[
    {
        "type": "ref",
        "id": "bundleAll.log",
        "href": "/rest/logs/debug/bundleAll.log",
    },
    {
        "type": "ref",
        "id": "bundleWarnPlus.log",
```

```
        "href": "/rest/logs/debug/bundleWarnPlus.log",
    }
]
```

## /logs/debug/{log_id}

An individual debug log

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain the content of the specified debug log as an array of strings, one for each line in the file. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | A log with the specified {log_id} was not found |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

```
{
    "type": "array",
    "minItems": 0,
    "uniqueItems": false,
    "items": [
        {
            "type": "string"
        }
    ]
}
```

#### Example

```
Request: GET /rest/logs/debug/bundleAll.log

Response: 200 OK

[
  "r22 Sep 2015 13:09:38 [INFO ] c.s.a.f.l.LogConfigurator(LogConfigurator.java:42)
 Logs Configured",
  "22 Sep 2015 13:09:38 [INFO ] c.s.a.f.b.BundleValidator(BundleValidator.java:37)
 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ",
  "22 Sep 2015 13:09:38 [INFO ] c.s.a.f.l.Activator(Activator.java:35)
Starting Loggeref"
]
```

## /logs/recipients

Access event log recipients

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain all configured event log recipients |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

```
{
    "type": "array",
    "minItems": 0,
    "uniqueItems": true,
    "items": [
        {
            "type": "object",
            "required": [ "type", "id", "transport", "user",
 "password", "host", "port", "email", "security", "actions", "format" ],
            "additionalProperties": false,
            "properties": {
                "type": { "enum": [ "event-recipient" ] },
                "id": "number",
                "transport": { "enum": ["syslog"] },
                "user": "string",
                "password": "string",
                "host": "string",
                "port": "number",
                "email": "string",
                "format": { "enum": [ "LEEF", "CADF-CSV", "CADF-JSON", "CEF"] },
                "security": { "enum": [ "none", "SSL", "TLS"] },
            }
        }
    ]
}
```

#### Example

```
Request: GET /rest/logs/recipients

Response: 200 OK

[
    {
```

```
        "type": "event-recpient",
        "id": 4,
        "transport": "syslog",
        "host": "sfc.ibm.com",
        "port": 5687,
        "security": "none"
    }
]
```

## POST

### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

### Status codes

| Code | Description |
|---|---|
| **201** | The recipient has been created. The **Location** header gives the URL of the new recipient. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to create policies. |
| **406** | The **Accept** header must be set to a supported content type. |
| **409** | The new recipient conflicts with an existing recipient email or id. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

### Schema

*See event recipients collection schema, event-recipient object.* ***Changes:*** *id's are neither required nor should they be set.*

### Example

```
Request: POST /rest/logs/recipients

{
    "type": "event-recpient",
    "transport": "syslog",
    "host": "sfc.ibm.com",
    "port": 5687,
    "format": " LEEF",
    "security": "none"
}

Response: 201 Created | Location: /logs/recipients/4
```

## /logs/recipients/{id}

A specific event recipient

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain the requested event recipient |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | An event recipient with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **410** | An event recipient with the specified {id} was not found and but has existed before |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

*See event recipients collection schema, event-recipient object*

#### Example

```
Request: GET /rest/logs/recipients/4

Response: 200 OK

{
    "type": "event-recpient",
    "id": 4,
    "transport": "syslog",
    "host": "sfc.ibm.com",
    "port": 5687,
    "format": "LEEF",
    "security": "none"
}
```

### DELETE

#### Query parameters

| Parameter | Description |
|---|---|

| | |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **202** | A job was created to process deletion. See the **Location** header for job URL |
| **204** | The delete was successful |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | An event recipient with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*No body sent on DELETE*

**Example**

```
Request: DELETE /rest/logs/recipients/4

Response: 204 No content
```

## PATCH

***Query parameters***

| Parameter | Description |
|---|---|
| **<none>** | |

***Status codes***

| Code | Description |
|---|---|
| **204** | The request was successful, but contained no changes. Client side representation does not need to be refreshed. |
| **205** | The request was successful and the resource was updated. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to create event recipients. |

| | |
|---|---|
| **404** | An event recipient with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **409** | The changes to the recipient conflict with another recipient. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See event recipients collection schema, event-recipient object.* ***Changes:*** *Event recipient objects only require the type and id fields on PATCH, all other fields are optional.*

**Example**

```
Request: PATCH /rest/logs/recipients/4

{
    "id": 4,
    "email": "noreply-2@example.com"
}

Response: 205 Reset content
```

## PUT

***Query parameters***

| Parameter | Description |
|---|---|
| **<none>** | |

***Status codes***

| Code | Description |
|---|---|
| **204** | The request was successful, but contained no changes. Client side representation does not need to be refreshed. |
| **205** | The request was successful and the resource was updated. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to create event recipients. |
| **404** | An event recipient with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |

| | |
|---|---|
| **409** | The changes to the recipient conflict with another recipient. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

### Schema

*See event recipients' collection schema, event-recipient object.*

### Example

```
Request: PUT /rest/logs/recipients/4

{
    "type": "event-recpient",
    "id": 4,
    "transport": "email",
    "user": "admin",
    "password": "",
    "host": "",
    "port": null,
    "email": "noreply-3@example.com",
    "locale": "de",
    "security": "none"
}

Response: 205 Reset content
```

# Chapter 6. Jobs

## /jobs

A list of job sub resources.

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain a collection of references to child resources. The references will not contain any job schedule or job entries themselves. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

```
{
    "type": "array",
    "minItems": 3,
    "uniqueItems": true,
    "items": [
        {
            "type": "object",
            "required": ["type", "id", "href"],
            "additionalProperties": true,
            "properties": {
                "type": { "enum": [ "ref" ] },
                "id": { "enum": [ "schedules", "schedulers", "history" ] },
                "href": "string"
            }
        }
    ]
}
```

#### Example

```
Request: GET /jobs

Response: 200 OK
```

```
[
    {
        "type": "ref",
        "id": "schedules",
        "href": "/jobs/schedules",
    },
    {
        "type": "ref",
        "id": "schedulers",
        "href": "/jobs/schedulers",
    },
    {
        "type": "ref",
        "id": "history",
        "href": "/jobs/history",
    }
]
```

## /jobs/schedules

All job schedules in the system.

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain all known job schedules. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to see jobs. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

```
{
    "type": "array",
    "minItems": 0,
    "uniqueItems": true,
    "items": [
        {
            "type": "object",
         "required": [ "type", "id", "job_id", "when_starts", "when_ends", "starts_unit",
                       "ends_unit", "frequency", "multiplier", "state", "job_type",
                       "job_notes", "job_properties" ],
         "additionalProperties": false,
            "properties": {
                "type": { "enum": [ "job-schedule" ] },
                "id": "number",
                "job_id": "number",
                "when_starts": "number",
                "when_ends": "number",
```

```
                "starts_unit": { "enum": [ "ABS_MS", "REL_MS", "REL_SECONDS", "REL_MINUTES",
                       "REL_HOURS", "REL_DAYS", "REL_WEEKS", "REL_MONTHS", "REL_YEARS" ] },
                "ends_unit": { "enum": [ "ABS_MS", "REL_MS", "REL_SECONDS", "REL_MINUTES",
                       "REL_HOURS", "REL_DAYS", "REL_WEEKS", "REL_MONTHS", "REL_YEARS" ] },
                "frequency": { "enum": [ "ONCE", "MINUTELY", "HOURLY", "DAILY", "WEEKLY",
                       "MONTHLY", "YEARLY" ] },
                "multiplier": "number",
                "state": { "enum": [ "NOT_APPROVED", "WAITING", "RUNNING", "PAUSED",
                      "CANCELLED", "FINISHED" ] },
                "job_type": "string",
                "job_notes": "string",
                "job_properties": {
                    "type": "object",
                    "required": [],
                    "additionalProperties": true,
                    "properties": {}
                }
            }
        }
    ]
}
```

**Example**

```
Request: GET /rest/jobs/schedules

Response: 200 OK

[
    {
    "id": 1,
    "job_id": 1,
    "when_starts": 0,
    "when_ends": 0,
    "starts_unit": "ABS_MS",
    "ends_unit": "ABS_MS",
    "frequency": "ONCE",
    "multiplier": 1,
    "state": "NOT_APPROVED",
    "job_properties": {
        "c.s.a.b.u.j.createUser": "secadm",
        "runnable.passwordSalt64": "oPgSfm+htrj13c73xgytESU2hiWJVfqkQB8CMcu71+4\u003d",
        "job-submitter": "admin",
        "runnable.passwordDigest64": "tPV9p7WDjpkNv7czq1KUh+BCC3LcySTsUnZcNlOUYYQ\u003d",
        "c.s.a.b.u.j.createNotes": ""
    },
    "job_type": "c.s.a.b.u.j.CreateJob",
    "job_notes": "Creating a security administrator",
    "type": "job-schedule"
    }
]
```

## POST

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **201** | The job schedule has been created. The **Location** header gives the URL of the newly created scheduler job that activates the actual job schedule once it is approved. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |

| | |
|---|---|
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to create the job type indicated for this job schedule. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

```
{
    "type": "object",
    "required": [ "type", "when_starts", "when_ends", "frequency", "multiplier",
            "job_type", "job_notes", "job_properties" ],
    "additionalProperties": false,
    "properties": {
        "type": { "enum": [ "job-schedule" ] },
        "id": "number",
        "job_id": "number",
        "when_starts": "number",
        "when_ends": "number",
        "starts_unit": { "enum": [ "ABS_MS", "REL_MS", "REL_SECONDS", "REL_MINUTES",
             "REL_HOURS", "REL_DAYS", "REL_WEEKS", "REL_MONTHS", "REL_YEARS" ] },
        "ends_unit": { "enum": [ "ABS_MS", "REL_MS", "REL_SECONDS", "REL_MINUTES",
            "REL_HOURS", "REL_DAYS", "REL_WEEKS", "REL_MONTHS", "REL_YEARS" ] },
        "frequency": { "enum": [ "ONCE", "MINUTELY", "HOURLY", "DAILY", "WEEKLY",
             "MONTHLY", "YEARLY" ] },
        "multiplier": "number",
        "state": { "enum": [ "NOT_APPROVED", "WAITING", "RUNNING", "PAUSED",
              "CANCELLED", "FINISHED" ] },
        "job_type": "string",
        "job_notes": "string",
        "job_properties": {
            "type": "object",
            "required": [],
            "additionalProperties": true,
            "properties": {}
        }
    }
}
```

**Job Schedule Field Descriptions**

| Field | Type | Allowed Values | Description |
|---|---|---|---|
| **id** | Integer | Autogenerated | The id of the schedule. |
| **job_id** | Integer | Set by system | The id of the scheduler job associated with the schedule. |
| **when_starts** | Long | 0 to MAX_LONG<br>0 means start immediately once approved. | Absolute epoch milliseconds OR number of time units relative to the time the schedule is approved.<br>This value determines when the schedule starts executing. |

| | | | |
|---|---|---|---|
| **when_ends** | Long | 0 to MAX_LONG<br>0 means run forever (for recurring schedules only). | Absolute epoch milliseconds OR number of time units relative to the time the schedule started executing.<br>This value determines for how long the schedule will execute. |
| **starts_unit** | Enum | ABS_MS = absolute epoch milliseconds<br>REL_MS = relative (from approval) milliseconds<br>REL_SECONDS = relative (from approval) seconds<br>REL_MINUTES = relative (from approval) minutes<br>REL_HOURS = relative (from approval) hours<br>REL_DAYS= relative (from approval) days<br>REL_WEEKS = relative (from approval) weeks<br>REL_MONTHS = relative (from approval) months<br>REL_YEARS = = relative (from approval) years | Determines how the system interprets the input values on the when_starts field. If set to ABS_MS, when_starts has epoch milliseconds; otherwise, when_starts has the number of time units relative to approval time. |

| | | | |
|---|---|---|---|
| **ends_unit** | Enum | ABS_MS = absolute epoch milliseconds<br>REL_MS = relative (from schedule start) milliseconds<br>REL_SECONDS = relative (from schedule start) seconds<br>REL_MINUTES = relative (from schedule start) minutes<br>REL_HOURS = relative (from schedule start) hours<br>REL_DAYS= relative (from schedule start) days<br>REL_WEEKS = relative (from schedule start) weeks<br>REL_MONTHS = relative (from schedule start) months<br>REL_YEARS = = relative (from schedule start) years | Determines how the system interprets the input values on the when_ends field. If set to ABS_MS, when_ends has epoch milliseconds; otherwise, when_ends has the number of corresponding time units relative to start time. In other words, the length of the schedule. |
| **frequency** | Enum | ONCE = execute once / do not repeat<br>MINUTELY = repeat every M minutes<br>HOURLY = repeat every M hours<br>DAILY = repeat every M days<br>WEEKLY = repeat every M weeks<br>MONTHLY = repeat every M months<br>YEARLY = repeat every M years<br>**Note:** M is the multiplier field below. | Determines a recurring schedule frequency. If set to ONCE the schedule does not repeat (not recurring) and the when_ends parameter is ignored.<br>Otherwise, the value along with the multiplier field defines the frequency of the recurring job, and the when_ends parameter determines for how long the schedule continues. |

| | | | |
|---|---|---|---|
| **multiplier** | Integer | 0 to MAX_INT | Works along with the frequency to determine the interval to wait before recurring executions. For example, if frequency=HOURLY, and multiplier=3, the job will be executed every 3 hours, and so forth. |
| **state** | Enum | NOT_APPROVED<br>The schedule is not approved, not running, and will not execute jobs. This is the initial state before approval.<br>WAITING<br>The schedule is running, job was approved, but it has not executed anything for the first time yet.<br>RUNNING<br>The schedule has executed one or more jobs already, and additional executions are scheduled.<br>PAUSED<br>The schedule has been temporarily paused, no jobs are executed while it remains in this state.<br>CANCELLED<br>The schedule has been explicitly cancelled. No more job executions will ever happen.<br>FINISHED<br>The schedule completed execution normally. No more job executions will ever happen. | State of the schedule. Managed by the system. |
| **job_type** | String | See section on job types. | Type of job to submit. This determines which type of job runnable (the actual workload) will be executed when the scheduler triggers it. |

| | | | |
|---|---|---|---|
| **job_notes** | String | Any string value. | Notes/comments to add to the job(s) by default. |
| **job_properties** | Object | A JSON object. | Job properties object that defines the job to be run on schedule. The values here vary depending on the type of job. See next section for details. |

**Job Properties Schema**

```
{
    "type": "object",
    "required": [ "type", "job_type", "keys_to_update", "public_key"],
    "additionalProperties": false,
    "properties": {
        "type": { "enum": [ "key-update-job" ] },
        "job_type": { "enum": [ "key-rotation", "key-revocation", "key-shred" ] },
        "keys_to_update": {
            "type": ["array"],
            "minItems": 0,
            "uniqueItems": true,
            "items": [
                {
                    "type": "object",
                    "required": ["type", "id"]
                    "additionalProperties": false,
                    "properties": {
                        "id": "number"
                    }
                }
            ]
        },
        "public_key": {
            "type": "object",
            "required": [ "name", "id", "type", "href"],
            "additionalProperties": false,
            "properties": {
                "type": { "enum": [ "ref" ] },
                "name": "string",
                "id": "number",
                "href": "string"
            }
        }
    }
}
```

**Example**

```
Request: POST /rest/jobs/schedules

{
    "type": "job-schedule",
    "when_starts": 2,
    "when_ends": 0,
    "starts_unit" : "REL_WEEKS",
    "frequency": "WEEKLY",
    "multiplier": 1,
    "job_type": "key-update-job",
    "job_notes" : "",
    "job_properties": {
    "type": "key-update-job",
    "job_type": "key-rotation",
    "keys_to_update": [{
        "id": 3
    },
    {
        "id": 2
    },
    {
        "id": 1
    }],
    "notes": "",
```

```
    "public_key": {
        "name": "server1.cert.pem",
        "id": 1,
        "type": "client-file"
    }
    }
}

Response: 201 Created | Location: /rest/jobs/schedulers/2
```

## /jobs/schedules/{id}

A specific job schedule.

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain the requested job schedule. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to see jobs or the job type associated with this job schedule. |
| **404** | A job schedule with the specified {id} was not found and has not existed before. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

See section "Schema" on page 24, object type *job-schedule* within array.

#### Example

The schedule from the previous POST section. Note that in this example it has not been approved yet. Also, note that the *job_properties* object will change from the external object form to an internal representation.

```
Request: GET /rest/jobs/schedules/2

Response: 200 OK

{
    "id": 2,
    "job_id": 2,
    "when_starts": 2,
    "when_ends": 0,
    "starts_unit": "REL_WEEKS",
```

```
    "ends_unit": "ABS_MS",
    "frequency": "WEEKLY",
    "multiplier": 1,
    "state": "NOT_APPROVED",
    "job_properties": {
        "com.securityfirstcorp.atlantis.bundles.keygen.keys.agentIndex": "",
        "com.securityfirstcorp.atlantis.bundles.keygen.keys.keyUpdateIndexes": "1 2 3",
        "runnable.updateJobType": "ROTATE",
        "runnable.keyIndexes2": "1",
        "runnable.fileBytes64":
       "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUdIakNDQkFhZ0F3SUJBZ0lDRUFFd0RRWUpLb1pJaH
... omitted ...
DNVYWpCclVJbHN0a0Uwbk5MTjIyK2JoRGtOSHJNNmZSU1ZESEd3PT0
KLS0tLS1FTkQgQ0VSVElGSUNBVEUtLS0tLQo\u003d",
        "runnable.keyIndexes0": "3",
        "runnable.keyIndexes1": "2",
        "runnable.user": "admin",
        "runnable.notes": "",
        "job-submitter": "admin"
    },
    "job_type": "c.s.a.b.p.j.KeyRotationJob",
    "job_notes": null,
    "type": "job-schedule"
}
```


## PATCH

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **204** | The request was successful, but contained no changes. Client side representation does not need to be refreshed. |
| **201** | The schedule update job has been created. The **Location** header gives the URL of the newly created job that refreshes the actual job schedule with the new parameters once it is approved. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change jobs or the specific job type associated with this job schedule. |
| **404** | A job schedule with the specified {id} was not found and has not existed before. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

### Schema

See section “Schema” on page 24, object type *job-schedule* within array. **Changes:** Only *type* and *id* are required fields, and only the *when_starts*, *when_ends*, *starts_unit*, and *ends_unit* fields can be changed.

### Example

Patch an active job schedule by overriding its time range to the past, causing it to end immediately once the schedule update job is approved.

```
Request: PATCH /rest/jobs/schedules/2

{
    "type": "job-schedule",
    "id": 2,
    "when_starts” : 1,
    "when_ends": 2,
    "starts_unit": "ABS_MS",
    "ends_unit": "ABS_MS"
}

Response: 201 Created | Location: /rest/jobs/schedulers/3
```

## PUT

### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

### Status codes

| Code | Description |
|---|---|
| **204** | The request was successful, but contained no changes. Client side representation does not need to be refreshed. |
| **201** | The schedule update job has been created. The **Location** header gives the URL of the newly created job that refreshes the job schedule with the new parameters once it is approved. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change jobs or the specific job type associated with this job schedule. |
| **404** | A job schedule with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

### Schema

See section "Schema" on page 24, object type *job-schedule* within array. **Changes:** Only the *when_starts*, *when_ends*, *starts_unit*, and *ends_unit* fields can be changed.

### Example

Like the previous PATCH example except the complete schedule resource needs to be sent.

```
Request: PUT /rest/jobs/schedules/2

{
    "id": 2,
    "job_id": 2,
    "when_starts": 1,
    "when_ends": 2,
    "starts_unit": "ABS_MS",
    "ends_unit": "ABS_MS",
    "frequency": "WEEKLY",
    "multiplier": 1,
    "state": "NOT_APPROVED",
    "job_properties": {
        "com.securityfirstcorp.atlantis.bundles.keygen.keys.agentIndex": "",
        "com.securityfirstcorp.atlantis.bundles.keygen.keys.keyUpdateIndexes": "1 2 3",
        "runnable.updateJobType": "ROTATE",
        "runnable.keyIndexes2": "1",
        "runnable.fileBytes64":
"LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUdIakNDQkFhZ0F3SUJBZ0lDRUFFd0RRWUpLb1pJaH
... omitted ...
DNVYWpCclVJbHN0a0Uwbk5MTjIyK2JoRGtOSHJNNmZSU1ZESEd3PT0KLS0
tLS1FTkQgQ0VSVElGSUNBVEUtLS0tLQo\u003d",
        "runnable.keyIndexes0": "3",
        "runnable.keyIndexes1": "2",
        "runnable.user": "admin",
        "runnable.notes": "",
        "job-submitter": "admin"
    },
    "job_type": "c.s.a.b.p.j.KeyRotationJob",
    "job_notes": null,
    "type": "job-schedule"
}

Response: 201 Created | Location: /rest/jobs/schedulers/4
```

## /jobs/schedulers

All scheduler jobs.

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain all known scheduler jobs |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to see jobs |
| **406** | The **Accept** header must be set to a supported content type. |

| | |
|---|---|
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

```
{
    "type": "array",
    "minItems": 0,
    "uniqueItems": true,
    "items": [
        {
            "type": "object",
            "required": [ "type", "id", "scheduler_id", "job_type",
"state", "substate", "when_created", "when_started", "when_done",
"required_approvals", "required_rejections", "notes", "properties",
"results", "schedules", "actions" ],
            "additionalProperties": false,
            "properties": {
                "type": { "enum": [ "job" ] },
                "id": "number",
                "scheduler_id": "number",
                "job_type": "string",
                "state": { "enum": [ "WAITING", "RUNNING", "DONE",
"REJECTED", "ERROR" ] },
                "substate": { "enum": [ "R_OK", "R_BLOCKED", "D_OK",
 "D_REVIEW", "D_FAILED" ] },
                "when_created": "number",
                "when_started": "number",
                "when_done": "number",
                "required_approvals": "number",
                "required_rejections": "number",
                "notes": "string",
                "properties": {
                    "type": "object",
                    "required": [],
                    "additionalProperties": true,
                    "properties": {}
                },
                "results": {
                    "type": "object",
                    "required": [],
                    "additionalProperties": true,
                    "properties": {}
                },
                "schedules": (array of job-schedule, see 6.2.1.3)
                "actions": {
                    "type": [ "array" ],
                    "minItems": 0,
                    "uniqueItems": true,
                    "items": [
                        {
                            "type": "object",
                            "required": ["type", "id", "job_id", "user",
 "action", "when_taken", "notes" ],
                            "additionalProperties": false,
                            "properties": {
                                "type": { "enum": [ "job-action" ] },
                                "id": "number",
                                "job_id": "number",
                                "user": "string",
                                "action": { "enum": [ "APPROVE",
"REJECT", "ABSTAIN" ] },
                                "when_taken": "number",
                                "notes": "string"
                            }
                        }
                    ]
                }
            }
        }
    ]
}
```

**Example**

```
Request: GET /rest/jobs/schedulers

Response: 200 OK

[
    {
    "id": 1,
    "scheduler_id": 1,
    "job_type": "c.s.a.b.j.j.SchedulingJob",
    "state": "DONE",
    "substate": "D_OK",
    "when_created": 1503619618974,
    "when_started": 1503672302393,
    "when_done": 1503672302402,
    "required_approvals": 1,
    "required_rejections": 1,
    "notes": "Creating a security administrator",
    "properties": {
        "c.s.a.b.u.j.createUser": "secadm",
        "runnable.passwordSalt64":
"oPgSfm+htrj13c73xgytESU2hiWJVfqkQB8CMcu7l+4\u003d",
        "job-submitter": "admin",
        "runnable.passwordDigest64":
"tPV9p7WDjpkNv7czq1KUh+BCC3LcySTsUnZcNlOUYYQ\u003d",
        "c.s.a.b.u.j.createNotes": ""
    },
    "results": null,
    "actions": [{
        "id": 2,
        "job_id": 1,
        "user": "admin",
        "action": "APPROVE",
        "when_taken": 1503672302385,
        "notes": "Approved on 16 April [JTK]",
        "type": "job-action"
    }],
    "schedules": [{
        "id": 1,
        "job_id": 1,
        "when_starts": 0,
        "when_ends": 0,
        "starts_unit": "ABS_MS",
        "ends_unit": "ABS_MS",
        "frequency": "ONCE",
        "multiplier": 1,
        "state": "FINISHED",
        "job_properties": {
            "c.s.a.b.u.j.createUser": "secadm",
            "runnable.passwordSalt64": "oPgSfm+htrj13c73xgytESU2hiWJVfqkQB8CMcu7l+4\u003d",
            "job-submitter": "admin",
            "runnable.passwordDigest64": "tPV9p7WDjpkNv7czq1KUh+BCC3LcySTsUnZcNlOUYYQ\u003d",
            "c.s.a.b.u.j.createNotes": ""
        },
        "job_type": "c.s.a.b.u.j.CreateJob",
        "job_notes": "Creating a security administrator",
        "type": "job-schedule"
    }],
    "type": "job"
    }
]
```

**Substate**

Substate only exists for some state values. Where not applicable, it will be null. Substate values are defined as follows:

| Substate | Allowed in state | Description |
|---|---|---|
| **R_OK** | **RUNNING** | The job has been approved, and it either queued up for thread time to run or is actively running. No problems detected so far. |

| | | |
|---|---|---|
| **R_BLOCKED** | **RUNNING** | The job is either fully or partially blocked from completion and will require intervention. Data from the results field of the job can be used to determine the right course of action. |
| **D_OK** | **DONE** | The job is done running and no problems were found. |
| **D_REVIEW** | **DONE** | The job is done running and success or failure cannot be programmatically determined. Data from the results field and manual validation of the intended effects of the job can be used to determine whether the job was successful or not. *This substate is expected to be rare.* |
| **D_FAILED** | **DONE** | The job is done running but is known to have failed. Data from the result field can be used to identify the failure |

## /jobs/schedulers/{id}

A specific scheduler job.

### GET

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **200** | The response body will contain the requested job |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to see jobs or this job type |
| **404** | A job with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

### Schema

*See section* “Schema” on page 35*, job and job-action objects.*

### Example

```
Request: GET /rest/jobs/schedulers/1

Response: 200 OK

{
    "id": 1,
    "scheduler_id": 1,
    "job_type": "c.s.a.b.j.j.SchedulingJob",
    "state": "DONE",
    "substate": "D_OK",
    "when_created": 1503619618974,
    "when_started": 1503672302393,
    "when_done": 1503672302402,
    "required_approvals": 1,
    "required_rejections": 1,
    "notes": "Creating a security administrator",
    "properties": {
        "c.s.a.b.u.j.createUser": "secadm",
        "runnable.passwordSalt64": "oPgSfm+htrj13c73xgytESU2hiWJVfqkQB8CMcu7l+4\u003d",
        "job-submitter": "admin",
        "runnable.passwordDigest64": "tPV9p7WDjpkNv7czq1KUh+BCC3LcySTsUnZcNlOUYYQ\u003d",
        "c.s.a.b.u.j.createNotes": ""
    },
    "results": null,
    "actions": [
           {
        "id": 2,
        "job_id": 1,
        "user": "admin",
        "action": "APPROVE",
        "when_taken": 1503672302385,
        "notes": "Approved on 16 April [JTK]",
        "type": "job-action"
        }
       ],
    "schedules": [
           {
        "id": 1,
        "job_id": 1,
        "when_starts": 0,
        "when_ends": 0,
        "starts_unit": "ABS_MS",
        "ends_unit": "ABS_MS",
        "frequency": "ONCE",
        "multiplier": 1,
        "state": "FINISHED",
        "job_properties": {
            "c.s.a.b.u.j.createUser": "secadm",
            "runnable.passwordSalt64": "oPgSfm+htrj13c73xgytESU2hiWJVfqkQB8CMcu7l+4\u003d",
            "job-submitter": "admin",
            "runnable.passwordDigest64": "tPV9p7WDjpkNv7czq1KUh+BCC3LcySTsUnZcNlOUYYQ\u003d",
            "c.s.a.b.u.j.createNotes": ""
        },
        "job_type": "c.s.a.b.u.j.CreateJob",
        "job_notes": "Creating a security administrator",
        "type": "job-schedule"
        }
       ],
    "type": "job"
}
```

## PATCH

### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **204** | The request was successful, but contained no changes. Client side representation does not need to be refreshed. |
| **205** | The request was successful and the resource was updated. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change jobs or the specific job type. |
| **404** | A job with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See section* "Schema" on page 35*, job object.* ***Changes:*** *Only type and id are required fields, and only the notes field can be changed.*

**Example**

```
Request: PATCH /rest/jobs/schedulers/1

{
    "type": "job",
    "id": 1,
    "notes": "Creating a security administrator (delete after 1/1/2018)"
}

Response: 205 Reset content
```

## PUT

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **204** | The request was successful, but contained no changes. Client side representation does not need to be refreshed. |
| **205** | The request was successful and the resource was updated. |

| | |
|---|---|
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change jobs or the specific job type |
| **404** | A jobs with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See section* "Schema" on page 35*, job object.* ***Changes:*** *Only the notes field can be changed.*

**Example**

```
Request: PUT /rest/jobs/schedulers/1

{
    "id": 1,
    "scheduler_id": 1,
    "job_type": "c.s.a.b.j.j.SchedulingJob",
    "state": "DONE",
    "substate": "D_OK",
    "when_created": 1503619618974,
    "when_started": 1503672302393,
    "when_done": 1503672302402,
    "required_approvals": 1,
    "required_rejections": 1,
    "notes": "Creating a security administrator (delete after 1/1/2018)",
    "properties": {
        "c.s.a.b.u.j.createUser": "secadm",
        "runnable.passwordSalt64": "oPgSfm+htrj13c73xgytESU2hiWJVfqkQB8CMcu7l+4\u003d",
        "job-submitter": "admin",
        "runnable.passwordDigest64": "tPV9p7WDjpkNv7czq1KUh+BCC3LcySTsUnZcNlOUYYQ\u003d",
        "c.s.a.b.u.j.createNotes": ""
    },
    "results": null,
    "actions": [
          {
        "id": 2,
        "job_id": 1,
        "user": "admin",
        "action": "APPROVE",
        "when_taken": 1503672302385,
        "notes": "Approved on 16 April [JTK]",
        "type": "job-action"
        }
       ],
    "schedules": [
          {
        "id": 1,
        "job_id": 1,
        "when_starts": 0,
        "when_ends": 0,
        "starts_unit": "ABS_MS",
        "ends_unit": "ABS_MS",
        "frequency": "ONCE",
        "multiplier": 1,
        "state": "FINISHED",
        "job_properties": {
            "c.s.a.b.u.j.createUser": "secadm",
            "runnable.passwordSalt64": "oPgSfm+htrj13c73xgytESU2hiWJVfqkQB8CMcu7l+4\u003d",
            "job-submitter": "admin",
            "runnable.passwordDigest64": "tPV9p7WDjpkNv7czq1KUh+BCC3LcySTsUnZcNlOUYYQ\u003d",
```

```
            "c.s.a.b.u.j.createNotes": ""
         },
         "job_type": "c.s.a.b.u.j.CreateJob",
         "job_notes": "Creating a security administrator",
         "type": "job-schedule"
         }
       ],
    "type": "job"
}

Response: 205 Reset content
```

## /jobs/schedulers/{id}/actions

The actions taken for a specific job.

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain he actions array for the job |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to see jobs or this job type |
| **404** | A job with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

*See section* "Schema" on page 35*, job object, job-action object.*

#### Example

```
Request: GET /rest/jobs/schedulers/1/actions

Response: 200 OK

[
    {
    "id": 2,
    "job_id": 1,
    "user": "admin",
    "action": "APPROVE",
    "when_taken": 1503672302385,
    "notes": "Approved on 16 April [JTK]",
    "type": "job-action"
```

```
    }
]
```

### POST

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|
| **201** | The job action has been created. The **Location** header gives the URL of the new job action. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to see jobs, this job type, or take the specified action on this job type. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

*See section* "Schema" on page 35*, job object, job-action object.* ***Changes:*** *Only type, job_id, and action are required fields. The user, id, and when_taken fields are generated on successful POST.*

#### Example

```
Request: POST /rest/jobs/schedulers/1/actions

{
    "type": "job-action",
    "job_id": 1,
    "action": "APPROVE",
    "notes": "Approved on 16 April [JTK]"
}

Response: 201 Created
```

## /jobs/history[r}

All jobs executed by a schedule.

### GET

#### Query parameters

| Parameter | Description |
|---|---|

| | |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **200** | The response body will contain all known scheduler jobs |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to see jobs |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See section* "Schema" on page 35*, job object. This is like a scheduler job except that the id field will normally be different than scheduler_id field. The scheduler_id field correlates the execution job with the corresponding scheduler job.*

**Example**

```
Request: GET /rest/jobs/history

Response: 200 OK

[
   {
      "id":4,
      "scheduler_id":1,
      "job_type":"c.s.a.b.u.j.CreateJob",
      "state":"DONE",
      "substate":"D_OK",
      "when_created":1503672302432,
      "when_started":1503672302455,
      "when_done":1503672302509,
      "required_approvals":1,
      "required_rejections":1,
      "notes":"Creating a security administrator",
      "properties":{
         "c.s.a.b.u.j.createUser":"secadm",
   "runnable.passwordSalt64":"oPgSfm+htrj13c73xgytESU2hiWJVfqkQB8CMcu7l+4\u003d",
  "runnable.passwordDigest64":"tPV9p7WDjpkNv7czq1KUh+BCC3LcySTsUnZcNlOUYYQ\u003d",
         "job-submitter":"admin",
         "c.s.a.b.u.j.createNotes":""
      },
      "results":null,
      "actions":[
         {
            "id":2,
            "job_id":1,
            "user":"admin",
            "action":"APPROVE",
            "when_taken":1503672302385,
            "notes":"Approved on 16 April [JTK]",
            "type":"job-action"
         }
      ],
      "schedules":[
         {
            "id":1,
            "job_id":1,
            "when_starts":0,
            "when_ends":0,
            "starts_unit":"ABS_MS",
```

```
            "ends_unit":"ABS_MS",
            "frequency":"ONCE",
            "multiplier":1,
            "state":"FINISHED",
            "job_properties":{
               "c.s.a.b.u.j.createUser":"secadm",
              "runnable.passwordSalt64":"oPgSfm+htrj13c73xgytESU2hiWJVfqkQB8CMcu7l+4\u003d",
               "job-submitter":"admin",
              "runnable.passwordDigest64":"tPV9p7WDjpkNv7czq1KUh+BCC3LcySTsUnZcNlOUYYQ\u003d",
               "c.s.a.b.u.j.createNotes":""            },
            "job_type":"c.s.a.b.u.j.CreateJob",
            "job_notes":"Creating a security administrator",
            "type":"job-schedule"
         }
      ],
      "type":"job"
   }
]
```


## /jobs/history/{id}

A specific executed job.

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain the requested executed job |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to see jobs or this job type |
| **404** | A job with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

*See section* "Schema" on page 35*, job object. This is like a scheduler job except that the id field will normally be different than scheduler_id field. The scheduler_id field correlates the execution job with the corresponding scheduler job.*

#### Example

```
Request: GET /rest/jobs/history/4

Response: 200 OK
```

```
{
   "id":4,
   "scheduler_id":1,
   "job_type":"c.s.a.b.u.j.CreateJob",
   "state":"DONE",
   "substate":"D_OK",
   "when_created":1503672302432,
   "when_started":1503672302455,
   "when_done":1503672302509,
   "required_approvals":1,
   "required_rejections":1,
   "notes":"Creating a security administrator",
   "properties":{
      "c.s.a.b.u.j.createUser":"secadm",
     "runnable.passwordSalt64":"oPgSfm+htrj13c73xgytESU2hiWJVfqkQB8CMcu7l+4
\u003d",      "runnable.passwordDigest64":"tPV9p7WDjpkNv7c
zq1KUh+BCC3LcySTsUnZcNlOUYYQ\u003d",
      "job-submitter":"admin",
      "c.s.a.b.u.j.createNotes":""
   },
   "results":null,
   "actions":[
      {
         "id":2,
         "job_id":1,
         "user":"admin",
         "action":"APPROVE",
         "when_taken":1503672302385,
         "notes":"Approved on 16 April [JTK]",
         "type":"job-action"
      }
   ],
   "schedules":[
      {
         "id":1,
         "job_id":1,
         "when_starts":0,
         "when_ends":0,
         "starts_unit":"ABS_MS",
         "ends_unit":"ABS_MS",
         "frequency":"ONCE",
         "multiplier":1,
         "state":"FINISHED",
         "job_properties":{
            "c.s.a.b.u.j.createUser":"secadm",
            "runnable.passwordSalt64":"oPgSfm+htrj13c73xg
ytESU2hiWJVfqkQB8CMcu7l+4\u003d",
            "job-submitter":"admin",
            "runnable.passwordDigest64":"tPV9p7WDjpkNv7czq1
KUh+BCC3LcySTsUnZcNlOUYYQ\u003d",
            "c.s.a.b.u.j.createNotes":""
         },
         "job_type":"c.s.a.b.u.j.CreateJob",
         "job_notes":"Creating a security administrator",
         "type":"job-schedule"
      }
   ],
   "type":"job"
}
```

# Chapter 7. Policy Resources

## /pathsets

All available path sets.

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain all configured path sets |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

```
{
    "type": "array",
    "minItems": 0,
    "uniqueItems": true,
    "items": [
        {
            "type": "object",
            "required": [ "type", "id", "name", "notes",
"snapshot", "source", "paths", "href" ],
            "additionalProperties": false,
            "properties": {
                "type": { "enum": [ "pathset" ] },
                "id": "number",
                "name": "string",
                "notes": "string",
                "snapshot": "boolean",
                "source": "number",
                "paths": {
                    "type": [ "array" ],
                    "minItems": 0,
                    "uniqueItems": true,
                    "items": [
                        {
                            "type": "object",
                            "required": ["type", "id", "path_set_id",
 "path", "notes", "source", "snapshot" ],
                            "additionalProperties": false,
                            "properties": {
```

```
                                "type": { "enum": [ "path" ] },
                                "id": "number",
                                "path": "string",
                                "path_set_id": "number",
                                "notes": "string",
                                "source": "number",
                                "snapshot": "boolean"
                            }
                        }
                    ]
                }
            }
        }
    ]
}
```

**Example**

```
Request: GET /rest/pathsets

Response: 200 OK

[
    {
        "id" : 1,
        "name" : "newPathset",
        "notes" : "",
        "source" : 0,
        "snapshot" : false,
        "paths" : [
            {
                "id" : 1,
                "path_set_id" : 1,
                "path" : "/file/path/one",
                "notes" : "",
                "source" : 0,
                "snapshot" : false,
                "type" : "path"
            },
            {
                "id" : 2,
                "path_set_id" : 1,
                "path" : "/file/path/two",
                "notes" : "",
                "source" : 0,
                "snapshot" : false,
                "type" : "path"
            }
        ],
        "href": null,
        "type": "pathset"
    }
]
```

## POST

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **201** | The path set has been created. The **Location** header gives the URL of the new path set. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |

| | |
|---|---|
| **403** | The current user is not authorized to create path sets. |
| **406** | The **Accept** header must be set to a supported content type. |
| **409** | The new path set conflicts with an existing path set name or id. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

*See* path *sets collection schema,* path*set and* path *objects.* ***Changes:*** *id is neither required nor should it be set for either object type.*

#### Example

```
Request: POST /rest/pathsets

{
    "name" : "newPathset",
    "notes" : "",
    "source" : 0,
    "snapshot" : false,
    "paths" : [
        {
            "path" : "/file/path/one",
            "notes" : "",
            "source" : 0,
            "snapshot" : false,
            "type" : "path"
        },
        {
            "path" : "/file/path/two",
            "notes" : "",
            "source" : 0,
            "snapshot" : false,
            "type" : "path"
        }
    ],
    "type": "pathset"
}

Response: 201 Created | Location: /rest/pathsets/1
```

## /pathsets/{id}

A specific path set

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|

| | |
|---|---|
| **200** | The response body will contain the requested path set |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | A path set with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See* path *sets collection schema,* path s*et and* path *objects.*

**Example**

```
Request: GET /rest/pathsets/1

Response: 200 OK

{
    "id" : 1,
    "name" : "newPathset",
    "notes" : "",
    "source" : 0,
    "snapshot" : false,
    "paths" : [
        {
            "id" : 1,
            "path_set_id" : 1,
            "path" : "/file/path/one",
            "notes" : "",
            "source" : 0,
            "snapshot" : false,
            "type" : "path"
        },
        {
            "id" : 2,
            "path_set_id" : 1,
            "path" : "/file/path/two",
            "notes" : "",
            "source" : 0,
            "snapshot" : false,
            "type" : "path"
        }
    ],
    "href": null,
    "type": "pathset"
}
```

## DELETE

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|

| | |
|---|---|
| **204** | The delete was successful |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | A path set with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*No body sent on DELETE*

**Example**

```
Request: DELETE /rest/pathsets/1

Response: 204 No content
```

## PATCH

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **204** | The request was successful, but contained no changes. Client side representation does not need to be refreshed. |
| **205** | The request was successful and the resource was updated. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change process sets. |
| **404** | A path set with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **409** | The changes to the path set conflict with another path set. |

| | |
|---|---|
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See* path *sets collection schema,* paths*et and* path *objects.* ***Changes:*** *Only type and id are required fields for both object types.*

**Example**

```
Request: PATCH /rest/pathsets/1

{
    "type": "pathset",
    "id": 1,
    "name": "newPathSetName"
}

Response: 205 Reset content
```

## PUT

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **204** | The request was successful, but contained no changes. Client side representation does not need to be refreshed. |
| **205** | The request was successful and the resource was updated. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change process sets |
| **404** | A path set with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **409** | The changes to the path set conflict with another path set. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

### Schema

*See path sets collection schema, path set and path objects.*

### Example

```
Request: PUT /rest/pathsets/1

{
    "id" : 1,
    "name" : "newPathsetName",
    "notes" : "Add some notes to the path set",
    "source" : 0,
    "snapshot" : false,
    "paths" : [
        {
            "id" : 1,
            "path_set_id" : 1,
            "path" : "/file/path/one",
            "notes" : "",
            "source" : 0,
            "snapshot" : false,
            "type" : "path"
        },
        {
            "id" : 2,
            "path_set_id" : 1,
            "path" : "/file/path/two",
            "notes" : "",
            "source" : 0,
            "snapshot" : false,
            "type" : "path"
        }
    ],
    "href": null,
    "type": "pathset"
}

Response: 205 Reset content
```

## /processes

All available processes

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain all configured processes |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |

| | |
|---|---|
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

```
{
    "type": "array",
    "minItems": 0,
    "uniqueItems": true,
    "items": [
        {
            "type": "object",
            "required": [ "type", "id", "name", "path",
 "version", "os", "distribution", "snapshot", "source", "rows" ],
            "additionalProperties": false,
            "properties": {
                "type": { "enum": [ "process" ] },
                "id": "number",
                "name": "string",
                "path": "string",
                "version": "number",
                "os": "number",
                "distribution": "number",
                "snapshot": "boolean",
                "source": "number",
                "rows": {
                    "type": [ "array" ],
                    "minItems": 0,
                    "uniqueItems": true,
                    "items": [
                        {
                             "type": "object",
                             "required": ["type", "id", "hash", "source", "snapshot" ],
                             "additionalProperties": false,
                             "properties": {
                                 "type": { "enum": [ "process-hash" ] },
                                 "id": "number",
                                 "hash": "string",
                                 "source": "number",
                                 "snapshot": "boolean"
                             }
                         }
                    ]
                }
            }
        }
    ]
}
```

**OS**

The Operating System values are represented as an integer. The integer values map as follows:

| Value | Mapping |
|---|---|
| **0** | **None** |
| **1** | **Linux** |
| **2** | **Windows** |

**Distribution**

The Distribution values are represented as an integer. The integer values map as follows:

| Value | Mapping |
|---|---|
| **0** | **None** |
| **1** | **Red Hat** |
| **2** | **CentOS** |

| | |
|---|---|
| **3** | **Windows** |

**Example**

```
Request: GET /processes

Response: 200 OK

[
    {
        "id": 1,
        "name": "testProcess",
        "path": "/home/bin/processA",
        "version": "1.1",
        "os": 1,
        "distribution": 2,
        "source": 0,
        "snapshot": false,
        "rows": [
            {
                "id": 1,
                "hash":
"0120010203040506070809101112131415161718192021222324252627282930313 2",
                "source": 0,
                "snapshot": false,
                "type": "process-hash"
            },
            {
                "id": 2,
                "hash":
"0120010203040506070809101112131415161718192021222324252627282930313 2",
                "source": 0,
                "snapshot": false,
                "type": "process-hash"
            }
        ],
        "href": null,
        "type": "process"
    },
    {
        "id": 2,
        "name": "testProcess",
        "path": "/home/bin/processA",
        "version": "1.1",
        "os": 1,
        "distribution": 2,
        "source": 1,
        "snapshot": true,
        "rows": [],
        "href": null,
        "type": "process"
    }
]
```

## POST

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **201** | The process has been created. The **Location** header gives the URL of the new process set. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |

| | |
|---|---|
| **403** | The current user is not authorized to create processes. |
| **406** | The **Accept** header must be set to a supported content type. |
| **409** | The new process conflicts with an existing process set name or id. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

*See processes collection schema, process and process-hash objects.* ***Changes:*** *id is neither required nor should it be set for either object type.*

#### Example

```
Request: POST /processes

{
        "name": "newProcess",
        "path": "/home/bin/newProcess",
        "version": "1.1",
        "os": 1,
        "distribution": 2,
        "source": 0,
        "snapshot": false,
        "rows": [
            {
                "hash":
 "0120010203040506070809101112131415161718192021222324252627282930313 2",
                "source": 0,
                "snapshot": false,
                "type": "process-hash"
            },
            {
                "hash":
 "0120010203040506070809101112131415161718192021222324252627282930313 2",
                "source": 0,
                "snapshot": false,
                "type": "process-hash"
            }
        ],
        "type": "process"
}

Response: 201 Created | Location: /processes/2
```

## /processes/{id}

A specific process

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **200** | The response body will contain the requested process |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | A process with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See processes collection schema, process and process-hash objects.*

**Example**

```
Request: GET /rest/processes/1

Response: 200 OK

    {
        "id": 1,
        "name": "testProcess",
        "path": "/home/bin/processA",
        "version": "1.1",
        "os": 1,
        "distribution": 2,
        "source": 0,
        "snapshot": false,
        "rows": [
            {
                "id": 1,
                "hash": "012001020304050607080910111213141516171819202122232425262728293031 32",
                "source": 0,
                "snapshot": false,
                "type": "process-hash"
            },
            {
                "id": 2,
                "hash": "012001020304050607080910111213141516171819202122232425262728293031 32",
                "source": 0,
                "snapshot": false,
                "type": "process-hash"
            }
        ],
        "href": null,
        "type": "process"
    },
```

## DELETE

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **204** | The delete was successful |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | A process with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*No body sent on DELETE*

**Example**

```
Request: DELETE /rest/processes/1

Response: 204 No content
```

## PATCH

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **204** | The request was successful, but contained no changes. Client side representation does not need to be refreshed. |
| **205** | The request was successful and the resource was updated. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change process sets. |
| **404** | A process with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |

| | |
|---|---|
| **409** | The changes to the process conflict with another process. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See processes collection schema, process and process-hash objects.* ***Changes:*** *Only type and id are required fields for both object types.*

**Example**

```
Request: PATCH /rest/processes/1

{
    "type": "process",
    "id": 1,
    "name": "ProcessA-BC"
}

Response: 205 Reset content
```

## PUT

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **204** | The request was successful, but contained no changes. Client side representation does not need to be refreshed. |
| **205** | The request was successful and the resource was updated. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change process sets |
| **404** | A process with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **409** | The changes to the process conflict with another process. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |

| | |
|---|---|
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See processes collection schema, process and process-hash objects.*

**Example**

```
Request: PUT /rest/processes/2

{
        "id": 2,
        "name": "newProcess",
        "path": "/home/bin/newProcess",
        "version": "1.1",
        "os": 1,
        "distribution": 2,
        "source": 0,
        "snapshot": false,
        "rows": [
            {
                "id": 4,
                "hash":
 "0120010203040506070809101112131415161718192021222324252627282930313 2",
                "source": 0,
                "snapshot": false,
                "type": "process-hash"
            },
            {
                "id": 5,
                "hash":
 "0120010203040506070809101112131415161718192021222324252627282930313 2",
                "source": 0,
                "snapshot": false,
                "type": "process-hash"
            }
        ],
        "type": "process"
}

Response: 205 Reset content
```

## /selectors

All available selectors

### GET

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **200** | The response body will contain all configured selectors |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **406** | The **Accept** header must be set to a supported content type. |

| | |
|---|---|
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

```
{
    "type": "array",
    "minItems": 0,
    "uniqueItems": true,
    "items": [
        {
            "type": "object",
            "required": [ "type", "id", "name", "notes", "snapshot", "rows", "href"],
            "additionalProperties": false,
            "properties": {
                "type": { "enum": [ "selector" ] },
                "id": "number",
                "name": "string",
                "notes": "string",
                "snapshot": "boolean",
                "source": "number",
                "rows": {
                    "type": [ "array" ],
                    "minItems": 0,
                    "uniqueItems": true,
                    "items": [
                        {
                             "type": "object",
                             "required": ["type", "id", "order",
 "user", "group", "external_group", "process_set", "snapshot"],
                             "additionalProperties": false,
                             "properties": {
                                 "type": { "enum": [ "selector-row" ] },
                                 "id": "number",
                                 "order": "number",
                                 "user": "string",
                                 "group": "string",
                                 "external_group": "boolean",
                                 "process_set": {
                                     "type": "object",
                                     "required": ["type", "id", "href" ],
                                     "additionalProperties": true,
                                     "properties": {
                                         "type": { "enum": [ "ref" ] },
                                         "id": "number",
                                         "href": "string"
                                     }
                                 },
                                 "source": "number",
                                 "snapshot": "boolean"
                             }
                        }
                    ]
                }
            }
            "href": "string"
        }
    ]
}
```

**Example**

```
Request: GET /rest/selectors

Response: 200 OK

[
    {
        "id": 2,
        "name": "selector1",
        "notes": "",
        "snapshot": false,
        "source": 0,
        "rows": [
```

```
            {
                "id": 2,
                "order": 1,
                "user": "user1",
                "group": null,
                "external_group": false,
                "process_set": {
                    "id": 0,
                    "href": "/rest/processsets/0",
                    "type": "ref"
                },
                "source": 0,
                "snapshot": false,
                "type": "selector-row"
            },
            {
                "id": 3,
                "type": "selector-row",
                "order": 2,
                "user": "sfc1",
                "group": null,
                "external_group": false,
                "process_set": null,
                "snapshot": false,
                "source": 0
            }
        ],
        "href": null,
        "type": "selector"
    }
]
```


## POST

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **201** | The selector has been created. The **Location** header gives the URL of the new selector. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to create selectors. |
| **406** | The **Accept** header must be set to a supported content type. |
| **409** | The new selector conflicts with an existing selector name or id. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

### Schema

*See selectors collection schema, selector and selector-row objects.* ***Changes:*** *id is neither required nor should it be set for either object type.*

### Example

```
Request: POST /rest/selectors

{
    "type": "selector",
    "name": "sfcUsers2",
    "notes": "Manual copy of selector for internal users",
    "snapshot": false,
    "source": 0,
    "rows": [
        {
            "type": "selector-row",
            "order": 1,
            "user": "sfc0",
            "group": null,
            "external_group": false,
            "process_set": null,
            "snapshot": false,
            "source": 0
        },
        {
            "type": "selector-row",
            "order": 2,
            "user": "sfc1",
            "group": null,
            "external_group": false,
            "process_set": null,
            "snapshot": false,
            "source": 0
        }
    ]
}

Response: 201 Created | Location: /rest/selectors/3
```

## /selectors/{id}

A specific selector

## GET

### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain the requested selector |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | A selector with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |

| | |
|---|---|
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See selectors collection schema, selector and selector-row objects.*

**Example**

```
Request: GET /rest/selectors/3

Response: 200 OK

{
    "type": "selector",
    "id": 3,
    "name": "sfcUsers2",
    "notes": "Manual copy of selector for internal users",
    "snapshot": false,
    "source": 0,
    "rows": [
        {
            "type": "selector-row",
            "id": 5,
            "order": 1,
            "user": "sfc0",
            "group": null,
            "external_group": false,
            "process_set": null,
            "snapshot": false,
            "source": 0
        },
        {
            "type": "selector-row",
            "id": 6,
            "order": 2,
            "user": "sfc1",
            "group": null,
            "external_group": false,
            "process_set": null,
            "snapshot": false,
            "source": 0
        }
    ],
    "href": null
}
```

## DELETE

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **204** | The delete was successful |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | A selector with the specified {id} was not found and has not existed before |

| | |
|---|---|
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*No body sent on DELETE*

**Example**

```
Request: DELETE /rest/selectors/3

Response: 204 No content
```

## PATCH

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **204** | The request was successful, but contained no changes. Client side representation does not need to be refreshed. |
| **205** | The request was successful and the resource was updated. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change selectors. |
| **404** | A selector with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **409** | The changes to the selector conflict with another selector. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

### Schema

*See selectors collection schema, selector and selector-row objects.* ***Changes:*** *Only type and id are required fields for both object types.*

### Example

```
Request: PATCH /rest/selectors/3

{
    "type": "selector",
    "id": 3,
    "notes": "Change to the notes field",
}

Response: 205 Reset content
```

## PUT

### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

### Status codes

| Code | Description |
|---|---|
| **204** | The request was successful, but contained no changes. Client side representation does not need to be refreshed. |
| **205** | The request was successful and the resource was updated. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change selectors |
| **404** | A selector with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **409** | The changes to the selector conflict with another selector. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

### Schema

*See selectors collection schema, selector and selector-row objects.*

### Example

```
Request: PUT /rest/selectors/3

{
    "type": "selector",
    "id": 3,
    "name": "sfcUsers2",
    "notes": "Another change to the notes field",
    "snapshot": false,
    "source": 0,
    "rows": [
        {
            "type": "selector-row",
            "id": 5,
            "order": 1,
            "user": "sfc0",
            "group": null,
            "external_group": false,
            "process_set": null,
            "snapshot": false,
            "source": 0
        },
        {
            "type": "selector-row",
            "id": 6,
            "order": 2,
            "user": "sfc1",
            "group": null,
            "external_group": false,
            "process_set": null,
            "snapshot": false,
            "source": 0
        }
    ]
}

Response: 205 Reset content
```

## /keys

All available named keys

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain all configured named keys |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |

| | |
|---|---|
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

```
{
    "type": "array",
    "minItems": 0,
    "uniqueItems": true,
    "items": [
        {
            "type": "object",
            "required": [ "type", "id", "name", "notes", "scope", "autogen", "versions" ],
            "additionalProperties": true,
            "properties": {
                "type": { "enum": [ "encryption-key" ] },
                "id": "number",
                "name": "string",
                "notes": "string",
                "scope": { "enum": [ "user", "internal" ] },
                "autogen": "boolean",
                "versions": {
                    "type": [ "array" ],
                    "minItems": 1,
                    "uniqueItems": true,
                    "items": [
                        {
                            "type": "object",
                            "required": [ "id", "version", "current", "timestamp" ],
                            "additionalProperties": false,
                            "properties": {
                                "id": "number",
                                "version": "number",
                                "current": "boolean",
                                "timestamp": "number"
                            }
                        }
                    ]
                }
            }
        }
    ]
}
```

**Example**

```
Request: GET /rest/keys

Response: 200 OK

[
    {
        "id": 1,
        "name": "keyA",
        "autogen": false,
        "versions": [
            {
                "id": 1,
                "version": 1,
                "current": true,
                "timestamp": 1478268524124
            }
        ],
        "scope": "user",
        "notes": "",
        "type": "encryption-key"
    },
    {
        "id": 2,
        "name": "keyB",
        "autogen": false,
        "versions": [
            {
                "id": 2,
                "version": 1,
                "current": true,
                "timestamp": 1478268526490
```

```
            }
        ],
        "scope": "user",
        "notes": "",
        "type": "encryption-key"
    }
]
```

## POST

### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

### Status codes

| Code | Description |
|---|---|
| **201** | The key has been created. The **Location** header gives the URL of the new key. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to create keys. |
| **406** | The **Accept** header must be set to a supported content type. |
| **409** | The new key conflicts with an existing key name or id. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

### Schema

*See keys collection schema, encryption-key object.* ***Changes:*** *id is neither required nor should it be set. Keys created by POST may only use the "user" scope.*

### Example

```
Request: POST /rest/keys

{
    "type": "encryption-key",
    "name": "testKey",
    "notes": "Key for users in the test org",
    "scope": "user",
    "autogen": false
}

Response: 201 Created | Location: /rest/keys/1
```

# /keys/{id}

A specific named key

## GET

### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain the requested named key |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | A named key with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

### Schema

*See keys collection schema, encryption-key object.*

### Example

```
Request: GET /rest/keys/1

Response: 200 OK

{
    "id": 1,
    "name": "keyA",
    "autogen": false,
    "versions": [
        {
            "id": 1,
            "version": 1,
            "current": true,
            "timestamp": 1478268524124
        }
    ],
    "scope": "user",
    "notes": "",
    "type": "encryption-key"
}
```

## PATCH

### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **204** | The request was successful, but contained no changes. Client side representation does not need to be refreshed. |
| **205** | The request was successful and the resource was updated. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change keys. |
| **404** | A key with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **409** | The changes to the key conflict with another key. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See keys collection schema, encryption-key object.* ***Changes:*** *Only type and id are required fields.*

**Example**

```
Request: PATCH /rest/keys/1

{
    "type": "encryption-key",
    "id": 1,
    "name": "testKey-orig",
    "notes": "Original key for users in the test org"
}

Response: 205 Reset content
```

## PUT

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **204** | The request was successful, but contained no changes. Client side representation does not need to be refreshed. |

| | |
|---|---|
| **205** | The request was successful and the resource was updated. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change keys |
| **404** | A key with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **409** | The changes to the key conflict with another key. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

*See keys collection schema, encryption-key object.*

#### Example

```
Request: PUT /rest/keys/1

{
    "id": 1,
    "name": "testKey",
    "autogen": false,
    "versions": [
        {
            "id": 3,
            "version": 1,
            "current": true,
            "timestamp": 1478547390464
        }
    ],
    "scope": "user",
    "notes": "Key for users in the test org",
    "type": "encryption-key"
}

Response: 205 Reset content
```

## /groupscopes

External group membership scopes.

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|

| | |
|---|---|
| **200** | The response body will contain all configured group scopes |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

```
{
    "type": "array",
    "minItems": 0,
    "uniqueItems": true,
    "items": [
        {
            "type": "object",
            "required": [ "type", "name", "host", "port",
 "secure", "bind_dn", "bind_password", "groups_dn", "users_dn" ],
            "additionalProperties": false,
            "properties": {
                "type": { "enum": [ "group-scope" ] },
                "name": "string",
                "host": "string",
                "port": "number",
                "secure": "boolean",
                "bind_dn": "string",
                "bind_password": "string",
                "groups_dn": "string",
                "users_dn": "string"
            }
        }
    ]
}
```

**Example**

```
Request: GET /rest/groupscopes

Response: 200 OK

[
    {
        "type": "group-scope",
        "name": "ldap",
        "host": "192.168.4.223",
        "port": 389,
        "secure": false,
        "bind_dn": "cn=admin,ou=Users,dc=example,dc=com",
        "bind_password": null,
        "users_dn": "ou=Users,dc=example,dc=com",
        "groups_dn": "ou=Groups,dc=example,dc=com"
    }
]
```

## /groupscopes/{name}

A specific group scope

## GET

### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain the requested group scope |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | A group scope with the specified {name} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

### Schema

*See group scopes collection schema, group-scope object.*

### Example

```
Request: GET /rest/groupscopes/ldap

Response: 200 OK

{
    "type": "group-scope",
    "name": "ldap",
    "host": "192.168.4.223",
    "port": 389,
    "secure": false,
    "bind_dn": "cn=admin,ou=Users,dc=example,dc=com",
    "bind_password": null,
    "users_dn": "ou=Users,dc=example,dc=com",
    "groups_dn": "ou=Groups,dc=example,dc=com"
}
```

## PATCH

### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

### Status codes

| Code | Description |
|---|---|

| | |
|---|---|
| **204** | The request was successful, but contained no changes. Client side representation does not need to be refreshed. |
| **205** | The request was successful and the resource was updated. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change group scopes. |
| **404** | A group scope with the specified {name} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See group scopes collection schema, group-scope object.* ***Changes:*** *Only type and name are required fields.*

**Example**

```
Request: PATCH /rest/groupscopes/ldap

{
    "type": "group-scope",
    "name": "ldap",
    "port": 636,
    "secure": true,
}

Response: 205 Reset content
```

## PUT

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **204** | The request was successful, but contained no changes. Client side representation does not need to be refreshed. |
| **205** | The request was successful and the resource was updated. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |

| | |
|---|---|
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change group scopes |
| **404** | A group scope with the specified {name} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See group scopes collection schema, group-scope object.*

**Example**

```
Request: PUT /rest/groupscopes/ldap

{
    "type": "group-scope",
    "name": "ldap",
    "host": "192.168.4.223",
    "port": 636,
    "secure": true,
    "bind_dn": "cn=admin,ou=Users,dc=example,dc=com",
    "bind_password": null,
    "users_dn": "ou=Users,dc=example,dc=com",
    "groups_dn": "ou=Groups,dc=example,dc=com"
}

Response: 205 Reset content
```

## /groupsets

All available group sets

•

### GET

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **200** | The response body will contain all configured group sets |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |

| | |
|---|---|
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

```
{
    "type": "array",
    "minItems": 0,
    "uniqueItems": true,
    "items": [
        {
            "type": "object",
            "required": [ "type", "id", "policy_binding_id", "groups" ],
            "additionalProperties": false,
            "properties": {
                "type": { "enum": [ "group-set" ] },
                "id": "number",
                "policy_binding_id": "number",
                "groups": {
                    "type": [ "array" ],
                    "minItems": 0,
                    "uniqueItems": true,
                    "items": [
                        {
                            "type": "object",
                            "required": ["type", "name", "members" ],
                            "additionalProperties": false,
                            "properties": {
                                "type": { "enum": [ "group " ] },
                                "name": "string",
                                "members": {
                                    "type": [ "array" ],
                                    "minItems": 0,
                                    "uniqueItems": true,
                                    "items": [
                                        {
                                            "type":
"string",
                                        }
                                    ]
                                }
                            }
                        }
                    ]
                }
            }
        }
    ]
}
```

**Example**

```
Request: GET /rest/groupsets

Response: 200 OK

[
    {
        "type": "group-set",
        "id": 1,
        "policy_binding_id": 1,
        "groups": [
            {
                "type": "group",
                "name": "users",
                "members": [
                    "user0",
                    "anotheruser",
                    "root",
                    "someone"
```

```
                ]
            }
        ]
    }
]
```

### POST

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|
| **201** | The group set has been created. The **Location** header gives the URL of the new group set. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to create group sets. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

*See group sets collection schema, group-set object.* ***Changes:*** *only type and policy_binding_id should be sent on a POST, all other fields are generated server-side.*

#### Example

```
Request: POST /rest/groupsets

{
    "type": "group-set",
    "policy_binding_id": 1
}

Response: 201 Created | Location: /rest/groupsets/3
```

## /groupsets/{id}

A specific group set

### GET

#### Query parameters

| Parameter | Description |
|---|---|

| | |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **200** | The response body will contain the requested group set |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | A group set with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See group sets collection schema, group-set object.*

**Example**

```
Request: GET /rest/groupsets/3

Response: 200 OK

{
    "type": "group-set",
    "id": 3,
    "policy_binding_id": 1,
    "groups": [
        {
            "type": "group",
            "name": "users",
            "members": [
                "user1",
                "user2",
                "user3"
            ]
        }
    ]
}
```

## DELETE

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **204** | The delete was successful |
| **401** | The current request is not authenticated. |

| | |
|---|---|
| **403** | The current user is not authorized. |
| **404** | A group set with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

*No body sent on DELETE*

#### Example

```
Request: DELETE /groupsets/3

Response: 204 No content
```

## /datatypes

All available datatypes

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain all configured datatypes |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

```
{
    "type": "array",
    "minItems": 0,
    "uniqueItems": true,
    "items": [
```

```
        {
            "type": "object",
            "required": [ "type", "id", "name",
 "notes", "snapshot", "source", "rows" ],
            "additionalProperties": false,
            "properties": {
                "type": { "enum": [ "data-type" ] },
                "id": "number",
                "name": "string",
                "notes": "string",
                "snapshot": "boolean",
                "source": "number",
                "rows": {
                    "type": [ "array" ],
                    "minItems": 0,
                    "uniqueItems": true,
                    "items": [
                        {
                             "type": "object",
                             "required": ["type", "id",
 "datatype_id", "order", "selector", "ops", "actions", "notes",
 "override_flags", "snapshot", "source"],
                             "additionalProperties": true,
                             "properties": {
                                 "type": { "enum": [ "data-type-row" ] },
                                 "id": "number",
                                 "datatype_id": "number",
                                 "order": "number",
                                 "selector": {
                                     "type": "object",
                                     "required": ["type", "id", "href" ],
                                     "additionalProperties": true,
                                     "properties": {
                                         "type": { "enum": [ "ref" ] },
                                         "id": "number",
                                         "href": "string"
                                     }
                                 },
                                 "ops": { "enum": ["read", "write", "read/write"] },
                                 "actions": { "enum": ["permit",
 "permit, log", "deny", "deny, log"] },
                                 "notes": "string",
                                 "override_flags": {
                                     "type": "object",
                                     "required": "required": ["type"],
                                     "additionalProperties": true,
                                     "properties": {
                                         "type": { "enum": [ "override-flag-map" ] }
                                     }
                                 },
                                 "source": "number",
                                 "snapshot": "boolean"
                             }
                         }
                    ]
                }
            }
        }
    ]
}
```

**Override Flags**

Each field in the datatype will have a corresponding override flag value. For the allowed values (defined below) of "may", "may_not", and "must", the values dictate whether an agent policy binding respectively may, may not, or must override a value from the datatype. The flags are correlated with a particular field in the datatype by encoding the datatype ID, row ID, and column ID into the mapping's field name.

If a value is not given for any mapping on datatype row POST or PUT, then the flag will be stored as "may_not". Missing map values are assumed to remain unchanged on PATCH.

***Override flag values and meanings***

| Flag value | Meaning |
|---|---|

| | |
|---|---|
| **may** | If the agent policy binding has set a different value, the binding value will be used. Otherwise the value defined by the datatype row is used. |
| **may_not** | The value defined by the datatype row is used regardless of whether the agent policy binding has set another value. |
| **must** | The agent policy binding must set another value to be valid. Any value stored in the datatype row itself is not used. |

***Override flag field name encoding***

The mapping field name is built by concatenating the datatype ID, datatype row ID, and column ID (defined below), in that order, along with prefixes for each value. That is for x, y, and z as the datatype ID, datatype row ID, and column ID respectively, the field name for the override flag is given by substituting x, y, and z into the string "dxrycz" (without quotes).

On first POST of a datatype row, datatype ID and datatype row ID may be given as 0 to imply that the flag applies to the containing row and the containing datatype.

| Row field | Column ID |
|---|---|
| **selector** | 1 |
| **ops** | 2 |
| | |
| **actions** | 3 |

***Example override flags:***

The following defines override flags for datatype 1, datatype row 1 that

- Prevent the row selector from being overridden
- Require that the row operations are overridden
- Allow for the row actions to be overridden or not

```
"override_flags": {
    "d1r1c1": "may_not",
    "d1r1c2": "must",
    "d1r1c3": "may",
    "type": "override-flag-map"
},
```

**Example**

```
Request: GET /rest/datatypes

Response: 200 OK

[
    {
        "id": 1,
        "name": "testDatatype",
        "notes": "",
        "source": 0,
        "snapshot": false,
        "rows": [
            {
                "id": 1,
                "datatype_id": 1,
                "order": 1,
                "selector": {
                    "id": 3,
                    "href": "/rest/selectors/3",
                    "type": "ref"
                },
```

```
                "ops": "read/write",
                "actions": "permit",
                "override_flags": {
                    "d1r1c1": "may_not",
                    "d1r1c2": "may_not",
                    "d1r1c3": "may_not",
                    "type": "override-flag-map"
                },
                "notes": "",
                "source": 0,
                "snapshot": false,
                "type": "data-type-row"
            }
        ],
        "type": "data-type"
    }
]
```

## POST

### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

### Status codes

| Code | Description |
|---|---|
| **201** | The datatype has been created. The **Location** header gives the URL of the new policy. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to create datatypes. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

### Schema

*See policies collection schema, policy and policy-row objects.* ***Changes:*** *id is neither required nor should it be set for either object type.*

### Example

```
Request: POST /rest/datatypes

{
    "name": "testDatatype",
    "notes": "",
    "source": 0,
    "snapshot": false,
    "rows": [
        {
            "order": 1,
            "selector": {
                "id": 3,
                "href": "/rest/selectors/3",
```

```
                "type": "ref"
            },
            "ops": "read/write",
            "actions": "permit",
            "override_flags": {
                "d0r0c1": "may_not",
                "d0r0c2": "may_not",
                "d0r0c3": "may_not",
                "type": "override-flag-map"
            },
            "notes": "",
            "source": 0,
            "snapshot": false,
            "type": "data-type-row"
        }
    ],
    "type": "data-type"
}

Response: 201 Created | Location: / datatypes/1
```

## /datatypes/{id}

A specific datatype

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain the requested datatype |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | A datatype with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

*See policies collection schema, policy and policy-row objects.*

#### Example

```
Request: GET /rest/datatypes/1

Response: 200 OK

{
```

```
    "id": 1,
    "name": "testDatatype",

    "notes": "",
    "source": 0,
    "snapshot": false,
    "rows": [
        {
            "id": 1,
            "datatype_id": 1,
            "order": 1,
            "selector": {
                "id": 3,
                "href": "/rest/selectors/3",
                "type": "ref"
            },
            "ops": "read/write",
            "actions": "permit",
            "override_flags": {
                "d1r1c1": "may_not",
                "d1r1c2": "may_not",
                "d1r1c3": "may_not",
                "type": "override-flag-map"
            },
            "notes": "",
            "source": 0,
            "snapshot": false,
            "type": "data-type-row"
        }
    ],
    "type": "data-type"
}
```

## DELETE

### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

### Status codes

| Code | Description |
|---|---|
| **204** | The delete was successful |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | A datatype with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

### Schema

*No body sent on DELETE*

### Example

```
Request: DELETE /rest/datatypes/1

Response: 204 No content
```

## PATCH

### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

### Status codes

| Code | Description |
|---|---|
| **204** | The request was successful, but contained no changes. Client side representation does not need to be refreshed. |
| **205** | The request was successful and the resource was updated. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change datatypes. |
| **404** | A datatype with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **409** | The changes to the policy conflict with another policy. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

### Schema

*See datatypes collection schema, data-type and data-type-row objects.* ***Changes:*** *Only type and id are required fields for both object types.*

### Example

```
Request: PATCH /rest/datatypes/1

{
    "id": 1,
    "name": "newDatatypeName",
    "notes": "new datatype notes",
    "type": "data-type"
}

Response: 205 Reset content
```

## PUT

### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

### Status codes

| Code | Description |
|---|---|
| **204** | The request was successful, but contained no changes. Client side representation does not need to be refreshed. |
| **205** | The request was successful and the resource was updated. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change datatypes |
| **404** | A datatype with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **409** | The changes to the datatype conflict with another datatype. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

### Schema

*See datatypes collection schema, data-type and data-type-row objects.*

### Example

```
Request: PUT /rest/datatypes/1

Response: 205 Reset content

{
    "id": 1,
    "name": "testDatatype",
    "notes": "updated with new selector and overrides",
    "source": 0,
    "snapshot": false,
    "rows": [
        {
            "id": 1,
            "datatype_id": 1,
            "order": 1,
            "selector": {
                "id": 4,
                "href": "/rest/selectors/4",
                "type": "ref"
            },
            "ops": "read/write",
```

```
            "actions": "permit",
            "override_flags": {
                "d1r1c1": "may",
                "d1r1c2": "may",
                "d1r1c3": "may",
                "type": "override-flag-map"
            },
            "notes": "",
            "source": 0,
            "snapshot": false,
            "type": "data-type-row"
        }
    ],
    "type": "data-type"
}
```

# Chapter 8. Managed Agents

## /agents

All managed agents

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain all known managed agents |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

```
{
    "type": "array",
    "minItems": 0,
    "uniqueItems": true,
    "items": [
        {
            "type": "object",
            "required": [ "type", "id", "name",
 "agent_type", "cores", "host", "peer_host", "uuid",
            "state", "notes", "certificates",
"policy", "install_bundle", "snapshots"],
            "additionalProperties": false,
            "properties": {
                "type": { "enum": [ "agent" ] },
                "id": "number",
                "name": "string",
                "agent_type": { "enum": [ "volume",
 "file", "hybrid", "objstore" ] },
                "cores": "number",
                "host": "string",
                "peer_host": "string",
                "uuid": "string",
                "state": { "enum": [ "ACTIVE",
 "WAITING", "INACTIVE" ] },
                "notes": "string",
                "certificates": {
                    "type": [ "array" ],
                    "minItems": 0,
```

```
                    "uniqueItems": true,
                    "items": [
                        {
                          "type": "object",
                          "required": ["type", "id",
"agent_id", "fingerprint", "state", "notes" ],
                          "additionalProperties": true,
                          "properties": {
                                "type": { "enum":
[ "agent-certificate" ] },
                                "id": "number",
                                "agent_id": "number",
                                "subject": "string",
                                "fingerprint": "string",
                                "expiry": "number",
                                "has_privkey": "boolean",
                                "state": { "enum":
[ "ACTIVE", "WAITING", "INACTIVE" ] },
                                "notes": "string"
                            }
                        }
                    ]
                },
                "policy": {
                    "type": "array",
                    "minItems": 0,
                    "uniqueItems": true,
                    "items": [
                        {
                            "type": { "enum": [ "volume-policy" ] },
                            "id": "number",
                            "agent_id": "number",
                            "device_label": "string",
                            "mofn": { "enum": [ "1:1" ] },
                            "key": "string"
                        },
                        {
                            "type": { "enum": [ "file-policy" ] },
                            "id": "number",
                            "agent_id": "number",
                            "path": "string",
                            "key": "string",
                            "data_type_id": "number",
                            "path_set_id": "number",
                            "group_set_id": "number",
                            "variables": {
                                "type": "object",
                                "required": [],
                                "additionalProperties": true
                            }
                        }
                    ]
                }
                "install_bundle": {
                    "type": { "enum": [ "install-bundle" ] },
                    "id": "number",
                    "agent_id": "number",
                    "operating_system": { "enum":
[ "centos-6x", "centos-7x", "windows" ] },
                    "authorized_users": {
                        "type": [ "array" ],
                        "minItems": 0,
                        "uniqueItems": true,
                        "items": [
                            { "type": "string" }
                        ]
                    },
                    "supported_formats": {
                        "type": [ "array" ],
                        "minItems": 0,
                        "uniqueItems": true,
                        "items": [
                            { "type": "string" }
                        ]
                    }
                },
                 "object_store": {
                    "mofn": "2:3","1:1","2:4"
                    "certificates": {
                    "type": [ "array" ],
                    "minItems": 0,
                    "uniqueItems": true,
```

```
                    "items": [
                        {
                            "type": "object",
                            "required": ["type", "id",
 "agent_id", "fingerprint",  state", "notes" ],
                            "additionalProperties": true,
                            "properties": {
                                "type": { "enum": [ "agent-certificate" ] },
                                "id": "number",
                                "agent_id": "number",
                                "subject": "string",
                                "fingerprint": "string",
                                "expiry": "number",
                                "has_privkey": "boolean",
                                "state": { "enum":
 [ "ACTIVE", "WAITING", "INACTIVE" ] },
                                "notes": "string"
                            }
                        }
                    ]
                },
            "backends": [{
                        "url": "string",
                        "key_id": "string",
                        "api_key":"string",
                        "auth_type": "S3_IBM4","Se_AMZ4","SWIFT"
                        "share": "number",
                    }
            ],
            "credentials":[{
                        "key_id": "string",
                        "access_key": "string",
                        "auth_type": "S3_IBM4","Se_AMZ4","SWIFT"
                    }
            ],
            "buckets":[{
                        "name": "string",
                        "log_denial": true,
                        "policy": [{
                                    "cred_id": "string",
                                    "access": "string",
                                    "log": "boolean",
                                }]
                    }
           ],

                "policy_snapshots": {
                    "type": [ "array" ],
                    "minItems": 0,
                    "uniqueItems": true,
                    "items": [
                        {
                            "type": "object",
                            "required": ["type", "id",
 "policy", "state", "notes" ],
                            "additionalProperties": true,
                            "properties": {
                                "type": { "enum":
 [ "policy-snapshot" ] },
                                "id": "number",
                                "state": { "enum":
 [ "ACTIVE", "WAITING", "ACTIVATING",
                                     "INACTIVE" ] },
                                "notes": "string",
                                "policy": {
                                    "type": "array",
                                    "minItems": 0,
                                    "uniqueItems": true,
                                    "items": [
                                        {
                                            "type":
{ "enum": [ "volume-policy" ] },
                                            "id": "number",
                                            "agent_id": "number",
                                            "device_label": "string",
                                            "mofn": { "enum": [ "1:1" ] },
                                            "key": "string"
                                        },
                                        {
                                            "type":
{ "enum": [ "file-policy" ] },
                                            "id": "number",
```

```
                                            "agent_id": "number",
                                            "path": "string",
                                            "key": "string",
                                            "data_type_id": "number",
                                            "path_set_id": "number",
                                            "group_set_id": "number"

}
                                    ]
                                }
                            }
                        }
                    ]
                },
                "storage": {
                      "protocol": “smbv1, nfsv3, nfsv4”,
                      "host": "string",
                      "share_path": "string",
                      "share_user": "string",
                      "share_pass": "string",
                      "options": "string",
                      "type": “network-share ",
                "features": {
                    "type": { "enum": [ "agent_feature" ] },
                    "policy": {
                        "type": [ "agent_feature_status" ],
                        "state": "number" {0, 1},
                        "has_feature": "boolean",
                    },
                    "volume": {
                        "type": [ "agent_feature_status" ],
                        "state": "number" {0, 1},
                        "has_feature": "boolean",
                    },
                    "su_block": {
                        "type": [ "agent_feature_status" ],
                        "state": "number" {0, 1},
                        "has_feature": "boolean",
                    },
                    "objstore": {
                        "type": [ "agent_feature_status" ],
                        "state": "number" {0, 1},
                        "has_feature": "boolean",
                    }
                },
                "tools": {
                    "type": [ "array" ],
                    "minItems": 0,
                    "uniqueItems": true,
                    "items": [
                           {
                              "type": { "enum":
 [ "agent_tool" ] },
                              "tool_id": "number",
                            "purpose": { "enum":
[ "BACKUP", "RESTORE", "SEND", "RECEIVE",
                                          "OBJSTORE" ] },
                              "key": "string",
                              "active": "boolean",
                           },
                    ]
                },

            }
        }
    ]
}
```

**Example**

**Request: GET /rest/agents**

```
Response: 200 OK
[
    {
        "id": 1,
        "name": "fileAgent",
        "agent_type": "file",
        "cores": 0,
        "host": "1.1.1.1",
```

```
        "peer_host": "1.1.1.10",
        "uuid": "9faae641-347c-4f05-b47e-202b7430969d",
        "state": "ACTIVE",
        "notes": "",
        "certificates": [
            {
                "id": 1,
                "agent_id": 1,
                "subject": null,
                "fingerprint": "a15b027baaa1182b32e160dff4955bb2064d2d0b",
                "expiry": null,
                "has_privkey": null,
                "state": "ACTIVE",
                "notes": "",
                "type": "agent-certificate"
            }
        ],
        "policy": [
            {
                "id": 1,
                "agent_id": 1,
                "path": "/home/path",
                "key": "homekey",
                "data_type_id": 1,
                "group_set_id": 0,
                "path_set": null,
                "variables": {
                    "d1r1c1": "2",
                    "d1r1c3": "permit"
                },
                "type": "file-policy"
            },
            {
                "id": 2,
                "agent_id": 1,
                "path": "/data/path",
                "key": "datakey",
                "data_type_id": 1,
                "group_set_id": 0,
                "path_set": null,
                "variables": {
                    "d1r1c1": "3",
                    "d1r1c3": "deny, log"
                },
                "type": "file-policy"
            }
        ],
        "install_bundle": {
            "id": 1,
            "agent_id": 1,
            "operating_system": "centos-6x",
            "authorized_users": [
                "admin"
            ],
            "supported_formats": [
                "application/x-tar",
                "application/zip",
                "application/json"
            ],
            "type": "install-bundle"
        },
        "policy_snapshots": [
            {
                "id": 1,
                "notes": "",
                "state": "ACTIVE",
                "policy": [
                    {
                        "id": 3,
                        "agent_id": 1,
                        "path": "/home/path",
                        "key": "homekey",
                        "data_type_id": 2,
                        "group_set_id": 0,
                        "path_set": null,
                        "variables": {},
                        "type": "file-policy"
                    },
                    {
                        "id": 4,
                        "agent_id": 1,
                        "path": "/data/path",
```

```
                        "key": "datakey",
                        "data_type_id": 3,
                        "group_set_id": 0,
                        "path_set": null,
                        "variables": {},
                        "type": "file-policy"
                    }
                ],
                "type": "policy-snapshot"
        }
         ],
         "features": {
             "policy": {
                 "has_feature": true,
                    "state": 1,
                 "type": "agent_feature_status"
             },
              "volume": {
                 "has_feature": false,
               "state": 0,
                     "type": "agent_feature_status"
             },
             "su_block": {
                     "has_feature": true,
                     "state": 1,
                     "type": "agent_feature_status"
             },
             "objstore": {
                     "has_feature": false,
                     "state": 0,
                     "type": "agent_feature_status"
             },
             "type": "agent_feature"
          },
               "type": "agent"
            },
               "tools": [
         {
             "tool_id": 1,
             "purpose": "BACKUP",
             "key": “toolKeyA”,
             "active": true,
             "type": "agent_tool"
         },
         {
             "tool_id": 2,
             "purpose": "RESTORE",
             "key": “toolKeyA”,
             "active": true,
             "type": "agent_tool"
         }
     ]
]
```

## POST

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **201** | The agent has been created. The **Location** header gives the URL of the new user. |
| **202** | The agent will be created in an asynchronous job. The **Location** header gives the URL of the job to check for status. |

| | |
|---|---|
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current agent is not authorized to create new users |
| **406** | The **Accept** header must be set to a supported content type. |
| **409** | The new agent conflicts with an existing agent. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See* agent *collection schema, agent object.*

**Notes**

The **uuid** field must contain only hexadecimal characters and be written in hyphen separated 8-4-4-4-12 form, but more detailed validation for variant and namespace constraints is not done.

**Example**

**Request: POST /rest/agents**

```
{
    "name": "volumeAgent",
    "agent_type": "volume",
    "host": "1.1.1.1",
    "peer_host": "1.1.1.10",
    "uuid": "caca0d57-1f6a-4663-a2c7-e9b9591e48f5",
    "notes": "",
    "certificates": [
        {
            "has_privkey": true,
            "content_type": "application/x-pem-file",
            "subject": "CN=localhost,OU=Development,O=Test.,L=Test,ST=CA,C=US",
            "fingerprint": "a15b027baaa1182b32e160dff4955bb2064d2d0b",
            "name": "http.pem",
            "expiry": 1478733066000,
            "id": 3,
            "type": "client-file",
            "href": "/rest/files/upload/3"
        }
    ],
    "policy": [
        {
            "key": "webkey",
            "type": "volume-policy",
            "device_label": "webroot"
        },
        {
            "key": "nfskey",
            "type": "volume-policy",
            "device_label": "nfsshare"
        }
    ],
    "install_bundle": {
        "operating_system": "centos-7x",
        "authorized_users": ["admin"],
        "type": "install-bundle"
    },
    "policy_snapshots": [],
    "tools": [
        {
            "purpose": "BACKUP",
            "key": “toolKeyA”,
```

```
            "active": true,
            "type": "agent_tool"
        }
    ]
}

Response: 202 Accepted | Location: /jobs/23
```

## /agents/{id}

A specific managed agent

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain the requested agent |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | An agent with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

*See agents collection schema, agent object.*

#### Example

**Request: GET /rest/agents/4**

**Response: 200 OK**

```
{
    "id": 4,
    "name": "volumeAgent",
    "agent_type": "volume",
    "host": "1.1.1.1",
    "peer_host": "1.1.1.10",
    "uuid": "caca0d57-1f6a-4663-a2c7-e9b9591e48f5",
    "notes": "",
"certificates": [
        {
            "subject": "CN=localhost,OU=Development,
O=Test.,L=Test,ST=CA,C=US",
            "fingerprint":
 "a15b027baaa1182b32e160dff4955bb2064d2d0b",
```

```
            "expiry": 1478733066000,
            "id": 6,
            "type": "agent-certificate"
        }
    ],
    "policy": [
        {
            "id": 13,
            "agent_id": 4,
            "key": "webkey",
            "device_label": "webroot",
            "mofn": "1:1",
            "type": "volume-policy"
        },
        {
            "id": 14,
            "agent_id": 4,
            "key": "nfskey",
            "device_label": "nfsshare",
            "mofn": "1:1",
            "type": "volume-policy"
        }
    ],
    "install_bundle": {
        "id": 5,
        "agent_id": 4,
        "operating_system": "centos-7x",
        "authorized_users": ["admin"],
        "supported_formats": [
            "application/x-tar",
            "application/zip",
            "application/json"
        ],
        "type": "install-bundle"
    },
    "policy_snapshots": [],
    "features": {
            "policy": {
                "has_feature": true,
                   "state": 1,
                "type": "agent_feature_status"
            },
             "volume": {
                "has_feature": false,
              "state": 0,
                    "type": "agent_feature_status"
            },
            "su_block": {
                    "has_feature": true,
                    "state": 1,
"type": "agent_feature_status"
            },
            "objstore": {
                    "has_feature": false,
                    "state": 0,
                    "type": "agent_feature_status"
            },
            "type": "agent_feature"
         },
        "type": "agent"
    },
    "tools": []
}
```


## DELETE

### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

### Status codes

| Code | Description |
|---|---|

| | |
|---|---|
| **202** | A job was created to process deletion. See the **Location** header for job URL |
| | |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | An agent with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*No body sent on DELETE*

**Example**

```
Request: DELETE /rest/agents/4

Response: 202 Accepted | Location: /rest/jobs/9
```

## PATCH

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **204** | The request was successful, but contained no changes. Client side representation does not need to be refreshed. |
| **205** | The request was successful and the resource was updated. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change agents. |
| **404** | An agent with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **409** | The changes to the agent conflict with another agent. |

| | |
|---|---|
| **424** | There are not enough license slots open to approve the agent |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See agents collection schema, agent object.* ***Changes:*** *Only type and id are required fields.*

**Example**

```
Request: PATCH /rest/agents/4

{
    "type": "agent",
    "id": 4,
    "state": "ACTIVE"
}

Response: 205 Reset Content
```

## /agents/{id}/certificates

The known certificates for a specific managed agent

### GET

**Query parameters**

| • ***Parameter*** | • ***Description*** |
|---|---|
| • ***<none>*** | • |

**Status codes**

| Code | Description |
|---|---|
| **200** | The response body will contain the requested agent certificates |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | An agent with the specified {id} was not found and has not existed before. The certificates field will always exist for an extant agent, even if the field is empty. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

### Schema

*See agents collection schema, agent-certificate object.*

### Example

```
Request: GET /rest/agents/4/certificates

Response: 200 OK

[
    {
        "subject": "CN=localhost,OU=Development,O=Test.,L=Test,ST=CA,C=US",
        "fingerprint": "a15b027baaa1182b32e160dff4955bb2064d2d0b",
        "expiry": 1478733066000,
        "id": 6,
        "type": "agent-certificate"
    }
]
```

## POST

### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

### Status codes

| Code | Description |
|---|---|
| **201** | The certificate has been created. The **Location** header gives the URL of the new certificate. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to create certificates. |
| **406** | The **Accept** header must be set to a supported content type. |
| **409** | The new certificate conflicts with an existing certificate. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

### Schema

*See agent collection schema, agent-certificate object.* ***Changes:*** *id's are neither required nor should they be set.*

### Example

```
Request: POST /rest/agents/4/certificates

{
    "id": 4,
    "type": "client-file",
    "href": "/rest/files/upload/4"
```

```
    "notes": "After expiration, contact person@example.com for a new one"
}

Response: 201 Created | Location: /rest/agents/4/certificates/7
```

## /agents/{id}/certificates/{certid}

A specific managed agent certificate

•

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain the requested agent certificate |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | An agent with the specified {id} was not found and has not existed before, or a certificate with the specified {certid} was not found and has not existed before for the specified agent. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

*See agents collection schema, agent-certificate object.*

#### Example

```
Request: GET /rest/agents/4/certificates/7

Response: 200 OK

{
    "id": 7,
    "state": "ACTIVE",
    "subject": "CN=localhost2,OU=Development,O=Test.,L=Test,ST=CA,C=US",
    "fingerprint": "b88b327ba3a3182b32e160dff4955bb2064d2d1f",
    "expiry": 1478733066676,
    "has_privkey": true,
    "notes": "After expiration, contact person@example.com for a new one"
    "type": "agent-certificate"
}
```

## DELETE

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **202** | A job was created to process deletion. See the **Location** header for job URL |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | An agent with the specified {id} was not found and has not existed before, or a certificate with the specified {certid} was not found and has not existed before for the specified agent. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*No body sent on DELETE*

**Example**

```
Request: DELETE /rest/agents/4/certificates/7

Response: 202 Accepted | Location: /rest/jobs/33
```

## PATCH

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **204** | The request was successful, but contained no changes. Client side representation does not need to be refreshed. |
| **205** | The request was successful and the resource was updated. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |

| | |
|---|---|
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to create agent certificates. |
| **404** | An agent with the specified {id} was not found and has not existed before, or a certificate with the specified {certid} was not found and has not existed before for the specified agent. |
| **406** | The **Accept** header must be set to a supported content type. |
| **409** | The changes to the agent conflict with another agent. |
| **424** | There are not enough license slots open to approve the agent |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

*See agents collection schema, agent object.* ***Changes:*** *Only type and id are required fields.*

#### Example

```
Request: PATCH /rest/agents/4/certificates/7

{
    "type": "agent-certificate",
    "id": 7,
    "state": "ACTIVE",
    "notes": "Expires 2017-04-28,
 contact jack@example.com for a new one. Activated 2016-04-15."
}

Response: 205 Reset Content
```

## PUT

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|
| **204** | The request was successful, but contained no changes. Client side representation does not need to be refreshed. |
| **205** | The request was successful and the resource was updated. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |

| | |
|---|---|
| **403** | The current user is not authorized to create policies. |
| **404** | An agent with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **409** | The changes to the agent conflict with another agent. |
| **424** | There are not enough license slots open to approve the agent |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See agents collection schema, agent object.*

**Example**

```
Request: PUT /rest/agents/4/certificates/1

{
    "id": 7,
    "state": "ACTIVE",
    "subject": "CN=localhost2,OU=Development,O=Test.,L=Test,ST=CA,C=US",
    "fingerprint": "b88b327ba3a3182b32e160dff4955bb2064d2d1f",
    "expiry": 1478733066676,
    "has_privkey": true,
    "notes": " Expires 2017-04-28,
contact jill@example.com for a new one. Activated 2016-04-15."
    "type": "agent-certificate"
}

Response: 205 Reset Content
```

## /agents/{id}/install_bundle

The install bundle for an agent

The agent install bundle is a user restricted resource. Only users in the authorized_users list of the bundle will be able to read it.

### GET

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|

| | |
|---|---|
| **200** | The response body will contain the requested agent install bundle or install bundle metadata depending on the **Accept** header sent. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | An agent with the specified {id} was not found and has not existed before. The install_bundle field will always exist for an extant agent. |
| **406** | The **Accept** header must be set to a supported content type. Send the **Accept** header with type **application/json** and the returned JSON data will list acceptable type strings. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See agents collection schema, install-bundle object.*

**Example**

```
Request: GET /rest/agents/4/install_bundle

Response: 200 OK

{
    "id": 4,
    "agent_id": 4,
    "operating_system": "centos-7x",
    "authorized_users": ["admin"],
    "supported_formats": [
        "application/x-tar",
        "application/zip",
        "application/json"
    ],
    "expiry": 1478733066000,
    "type": "install-bundle"
}

Request: GET /rest/agents/4/install_bundle | Accept: application/x-tar

Response: 200 OK (binary content - tar file with agent installer and configuration)
```

## /agents/{id}/install_bundle/authorized_users

The install bundle authorized user list

•

### GET

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **200** | The response body will contain the requested agent install bundle authorized users list. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | An agent with the specified {id} was not found and has not existed before. The install_bundle field and the install_bundle/authorized_users field will always exist for an extant agent. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See agents collection schema, install-bundle object, authorized_users field.*

**Example**

```
Request: GET /rest/agents/4/install_bundle/authorized_users

Response: 200 OK

[
    "admin"
]
```

## PUT

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **204** | The request was successful, but contained no changes. Client side representation does not need to be refreshed. |
| **205** | The request was successful and the resource was updated. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change install bundle authorized users. |

| | |
|---|---|
| **404** | An agent with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See agents collection schema, install-bundle object, authorized_users field.*

**Example**

```
Request: PUT /rest/agents/4/install_bundle/authorized_users

[
    "admin",
    "wspoppleton",
    "zriggs"
]

Response: 205 Reset Content
```

## /agents/{id}/policy_snapshots

Snapshots of the agent policy field. Snapshots are immutable and will be identical in effect to the policy at the time of snapshot creation (see POST). The snapshot objects have new IDs and any set variables may have been applied directly to the new objects as well, but despite this should be semantically the same.

### GET

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **200** | The response body will contain the requested agent certificates |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | An agent with the specified {id} was not found and has not existed before. The certificates field will always exist for an extant agent, even if the field is empty. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |

| | |
|---|---|
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See agents collection schema, policy-snapshot object.*

**Example**

```
Request: GET /rest/agents/4/policy_snapshots

Response: 200 OK

[
    {
        "id": 1,
        "notes": "This is a file agent for the CentOS7.x file system",
        "state": "ACTIVE",
        "policy": [
            {
                "id": 11,
                "agent_id": 4,
                "path": "/mrslate/misc",
                "key": "public",
                "data_type_id": 6,
                "group_set_id": 0,
                "path_set": null,
                "variables": {},
                "type": "file-policy"
            },
            {
                "id": 12,
                "agent_id": 4,
                "path": "/mrslate/company",
                "key": "company",
                "data_type_id": 7,
                "group_set_id": 0,
                "path_set": null,
                "variables": {},
                "type": "file-policy"
            }
        ],
        "type": "policy-snapshot"
    }
]
```

## POST

Makes a new policy snapshot

•

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **201** | The snapshot has been created. The **Location** header gives the URL of the new snapshot. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |

| | |
|---|---|
| **403** | The current user is not authorized to create snapshots. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*An empty object.*

**Example**

```
Request: PUT /rest/agents/4/policy_snapshots/

{}

Response: 201 Created | Location: /rest/agents/4/policy_snapshots/3
```

## /agents/{id}/policy_snapshots/{snapshot_id}

Snapshots of the agent policy field. Snapshots are immutable and will be identical in effect to the master policy at the time of snapshot creation (see POST). The snapshot objects have new IDs and any set variables may have been applied directly to the new objects as well, but despite this should be semantically the same.

### GET

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **200** | The response body will contain the requested agent policy snapshot |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | An agent with the specified {id} was not found or a snapshot with the given {snapshot_id} was not found for the agent, and has not existed before. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

### Schema

*See agents collection schema, policy-snapshot object.*

### Example

```
Request: GET /rest/agents/4/policy_snapshots/3

Response: 200 OK

{
    "id": 1,
    "notes": "This is a file agent for the CentOS7.x file system",
    "state": "INACTIVE",
    "policy": [
        {
            "id": 11,
            "agent_id": 4,
            "path": "/mrslate/misc",
            "key": "public",
            "data_type_id": 6,
            "group_set_id": 0,
            "path_set": null,
            "variables": {},
            "type": "file-policy"
        },
        {
            "id": 12,
            "agent_id": 4,
            "path": "/mrslate/company",
            "key": "company",
            "data_type_id": 7,
            "group_set_id": 0,
            "path_set": null,
            "variables": {},
            "type": "file-policy"
        }
    ],
    "type": "policy-snapshot"
}
```

## PUT

Starts a snapshot activation job for the agent. After the job runs, the snapshot active at the time of the PUT will be inactive and the snapshot targeted by the PUT will be active. If the job fails, then the old snapshot will remain active.

•

### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

### Status codes

| Code | Description |
|---|---|
| **204** | The request was successful, but contained no changes. Client side representation does not need to be refreshed. |
| **205** | The request was successful and the resource was updated. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |

| | |
|---|---|
| **403** | The current user is not authorized to activate policy snapshots. |
| **404** | An agent with the specified {id} or a snapshot was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See agents collection schema, install-bundle object, policy_snapshots field.* ***Changes:*** *Only state and notes may be changed.*

**Example**

```
Request: PUT /rest/agents/4/policy_snapshots/1

{
    "id": 1,
    "notes": "This is a file agent for the CentOS7.x file system",
    "state": "ACTIVE",
    "policy": [
        {
            "id": 11,
            "agent_id": 4,
            "path": "/mrslate/misc",
            "key": "public",
            "data_type_id": 6,
            "group_set_id": 0,
            "path_set": null,
            "variables": {},
            "type": "file-policy"
        },
        {
            "id": 12,
            "agent_id": 4,
            "path": "/mrslate/company",
            "key": "company",
            "data_type_id": 7,
            "group_set_id": 0,
            "path_set": null,
            "variables": {},
            "type": "file-policy"
        }
    ],
    "type": "policy-snapshot"
}

Response: 205 Reset Content
```

## DELETE

Deletes an agent policy snapshot

•

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

•

**Status codes**

| Code | Description |
|---|---|
| **204** | The delete was successful |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | An agent with the specified {id} or a snapshot with the given {snapshot_id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

•

**Schema**

*None.*

•

**Example**

```
Request: GET /rest/agents/4/policy_snapshots/1

Response: 204 No Content
```

## /agents/{id}/policy

Agent protection master policy. This is a list of policy bindings - objects that bind together a path or path set, a set of variable field overrides, and the various different policy objects.

•

### GET

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **200** | The response body will contain the requested agent master policy |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | An agent with the specified {id} was not found. |

| | |
|---|---|
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See agents collection schema, policy field.*

**Example**

```
Request: GET /rest/agents/4/policy

Response: 200 OK

[
    {
        "id": 13,
        "agent_id": 4,
        "key": "webkey",
        "device_label": "webroot",
        "mofn": "1:1",
        "type": "volume-policy"
    },
    {
        "id": 14,
        "agent_id": 4,
        "key": "nfskey",
        "device_label": "nfsshare",
        "mofn": "1:1",
        "type": "volume-policy"
    }
]

Request: GET /rest/agents/3/policy

Response: 200 OK

[
    {
        "id": 2,
        "agent_id": 3,
        "path": "/mrslate/misc",
        "key": "public",
        "data_type_id": 1,
        "group_set_id": 0,
        "path_set": null,
        "variables": {},
        "type": "file-policy"
    },
    {
        "id": 3,
        "agent_id": 3,
        "path": "/mrslate/company",
        "key": "company",
        "data_type_id": 7,
        "group_set_id": 3,
        "path_set": null,
        "variables": {
            "d7r2c1": "3",
            "d7r2c2": "permit, log",
            "d7r2c3": "read"
        },
        "type": "file-policy"
    }
]
```

## POST

Makes a new policy binding

•

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **201** | The binding has been created. The **Location** header gives the URL of the new binding. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to create bindings. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See agents collection schema, file-policy and volume-policy objects.*

**Example**

```
Request: POST /rest/agents/3/policy/

{
    "agent_id": 3,
    "path": "/mrslate/company",
    "key": "company",
    "data_type_id": 7,
    "group_set_id": 3,
    "path_set": null,
    "variables": {
        "d7r2c1": "3",
        "d7r2c2": "permit, log",
        "d7r2c3": "read"
    },
    "type": "file-policy"
}

Response: 201 Created | Location: /rest/agents/3/policy/3
```

## /agents/{id}/policy/{binding_id}

### GET

**Query parameters**

| Parameter | Description |
|---|---|

| | |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **200** | The response body will contain the requested agent policy binding |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | An agent with the specified {id} was not found or does not have a binding in the master policy with the given {binding_id}. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See agents collection schema, file-policy and volume-policy objects.*

**Example**

```
Request: GET /rest/agents/3/policy/3

Response: 200 OK

{
    "id": 3,
    "agent_id": 3,
    "path": "/mrslate/company",
    "key": "company",
    "data_type_id": 7,
    "group_set_id": 3,
    "path_set": null,
    "variables": {
        "d7r2c1": "3",
        "d7r2c2": "permit, log",
        "d7r2c3": "read"
    },
    "type": "file-policy"
}
```

## PUT

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **204** | The request was successful, but contained no changes. Client side representation does not need to be refreshed. |

| | |
|---|---|
| **205** | The request was successful and the resource was updated. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change policy bindings. |
| **404** | An agent with the specified {id} or a binding with the given {binding_id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See agents collection schema, file-policy and volume-policy objects.*

**Example**

```
Request: PUT /rest/agents/3/policy/3

{
    "id": 3,
    "agent_id": 3,
    "path": "/mrslate/company",
    "key": "company",
    "data_type_id": 7,
    "group_set_id": 4,
    "path_set": null,
    "variables": {
        "d7r2c1": "4",
        "d7r2c2": "deny, log",
        "d7r2c3": "read"
    },
    "storage":null,
    "type": "file-policy"
}

Response: 205 Reset Content

Request: PUT /rest/agents/3/policy/3 (for adding storage object)
{
    "id": 3,
    "agent_id": 3,
    "path": "/mrslate/company",
    "key": "company",
    "data_type_id": 7,
    "group_set_id": 4,
    "storage": {
    "protocol":"smvb1",
    "host": "nfs.sfc.local",
    "share_path": "/sfc",
    "share_pass": "Passw0rd",
    "options": "port=112, sec=ext,noexec",
    "type": "network-share"
    },
    "type": "file-policy"
}

Response: 205 Reset Content
```

## PATCH

### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

### Status codes

| Code | Description |
|---|---|
| **204** | The request was successful, but contained no changes. Client side representation does not need to be refreshed. |
| **205** | The request was successful and the resource was updated. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change policy bindings. |
| **404** | An agent with the specified {id} or a binding with the given {binding_id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

### Schema

*See agents collection schema, file-policy and volume-policy objects.*

### Example

```
Request: PATCH /rest/agents/3/policy/3

{
    "group_set_id": 4,
    "variables": {
        "d7r2c1": "4",
        "d7r2c2": "deny, log",
        "d7r2c3": "read"
    },

    "type": "file-policy"

}

Response: 205 Reset Content


Request: PATCH /rest/agents/3/policy/3 (to reset the storage )

{
    "storage": {
       "id":1,
       "protocol":"smvb1",
        "host": "nfs.sfc.local",
```

```
        "share_path": "/sfc",
        "share_pass": "Passw0rd",
        "options": "port=112, sec=ext,noexec",
        "type": "network-share"
    },

    "type": "file-policy"

}

Response: 205 Reset Content
```

## DELETE

Deletes an agent policy binding

•

### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

### Status codes

| Code | Description |
|---|---|
| **204** | The delete was successful |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | An agent with the specified {id} or a binding with the given {binding_id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

### Schema

*None.*

### Example

```
Request: GET /rest/agents/3/policy/2

Response: 204 No Content
```

# /agents/{id}/policy/{binding_id}/storage

## GET

### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain the requested agent policy binding. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | An agent with the specified {id} was not found or does not have a binding in the master policy with the given {binding_id}. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

### Schema

See agents collection schema, file-policy, and storage objects.

### Example

```
Request: GET /rest/agents/3/policy/3/storage

Response: 200 OK

{
    "id": 84,
     "protocol":"smvb1",
     "host": "nfs.sfc.local",
     “share_path": "/sfc",
     "share_pass": "Passw0rd",
     "options": "port=112, sec=ext,noexec",
     "type": "network-share"
}
```

## PUT

### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

### Status codes

| Code | Description |
|---|---|

| | |
|---|---|
| **204** | The request was successful, but contained no changes. Client side representation does not need to be refreshed. |
| **205** | The request was successful and the resource was updated. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change agent feature status. |
| **404** | An agent with the specified {id} or a binding with the given {binding_id} was not found or the storage with the given {storage_id} was not found and has not existed before. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

See agents collection schema, file-policy, and storage objects.

**Example**

```
Request: PUT /rest/agents/3/policy/3/storage

{
    "id": 3,
    "path": "/mrslate/company",
    "key": "company",
    "data_type_id": 7,
    "group_set_id": 4,
    "storage": {
    "protocol":"smvb1",
    "host": "nfs.sfc.local",
    "share_path": "/sfc",
    "share_pass": "Passw0rd",
    "options": "port=112, sec=ext,noexec",
    "type": "network-share"
    },
    "type": "file-policy"
}

Response: 205 Reset Content
```

## PATCH

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|

| | |
|---|---|
| **204** | The request was successful, but contained no changes. Client side representation does not need to be refreshed. |
| **205** | The request was successful and the resource was updated. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change agent feature status. |
| **404** | An agent with the specified {id} or a binding with the given {binding_id} or the storage object with given {storage_id} was not found and has not existed before. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

See agents collection schema, file-policy, and storage objects.

**Example**

```
Request: PATCH /rest/agents/3/policy/3/storage

{
    "storage": {
       "id":1,
       "protocol":"smvb1",
        "host": "nfs.sfc.local",
       "share_path": "/sfc",
       "share_pass": "Passw0rd",
       "options": "port=112, sec=ext,noexec",
       "type": "network-share"
    },

    "type": "file-policy"

}

Response: 205 Reset Content
```

## /agents/{id}/features

### GET

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **200** | The response body will contain the requested agent feature. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized |
| **404** | An agent with the specified {id} was not found. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See agents collection schema agent_feature object.*

**Example**

```
Request: GET /rest/agents/1/features

Response: 200 OK
{
    "policy": {
        "has_feature": true,
        "state": 1,
        "type": "agent_feature_status"
    },
    "volume": {
        "has_feature": false,
        "state": 0,
        "type": "agent_feature_status"
    },
    "su_block": {
        "has_feature": true,
        "state": 1,
        "type": "agent_feature_status"
    },
    "objstore": {
        "has_feature": false,
        "state": 0,
        "type": "agent_feature_status"
    },
    "type": "agent_feature"
}
```

## /agents/{id}/features/{feature}

### GET

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **200** | The response body will contain the requested agent feature. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized |
| **404** | An agent with the specified {id} and {feature} was not found. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See agents collection schema agent_feature_status object.*

**Example**

```
Request: GET /rest/agents/1/features/policy

Response: 200 OK
"policy": {
     "has_feature": true,
     "state": 1,
     "type": "agent_feature_status"
 },
```

## PUT

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **202** | The feature will be updated in an asynchronous job. The **Location** header gives the URL of the job to check for status. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change agent feature status. |
| **404** | An agent with the specified {id} and {feature} was not found. |

| | |
|---|---|
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See agents collection schema agent_feature_status object.*

**Example**

```
Request: PUT /rest/agents/3/feature/policy

{
    "has_feature": true,
    "state": 0,
    "type": "agent_feature_status"
}

Response: 202 Accepted | Location: /rest/jobs/schedulers/9
```

## PATCH

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **202** | The feature will be updated in an asynchronous job. The **Location** header gives the URL of the job to check for status. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change policy bindings. |
| **404** | An agent with the specified {id} and {feature} was not found. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See agents collection schema agent_feature_status object.*

### Example

```
Request: PATCH /rest/agents/3/feature/su_block

{
    "state": 0,
}

Response: 202 Accepted | Location: /rest/jobs/schedulers/9
```

## /agents/{id}/tools

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain the requested agent tool |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized |
| **404** | An agent with the specified {id} was not found. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

*See agents collection schema agent_tool object.*

#### Example

```
Request: GET /rest/agents/1/tools

Response: 200 OK
[
    {
        "tool_id": 1,
        "purpose": "BACKUP",
        "key": {
            "id": 3,
            "name": "Normal",
            "autogen": false,
            "scope": "User",
            "notes": "The key used for general encryption",
            "versions": [
                {
                    "id": 3,
                    "version": 1,
                    "current": true,
```

```
                    "timestamp": 1523371339398
                }
            ]
        },
        "active": true,
        "type": "agent_tool"
    },
    {
        "tool_id": 2,
        "purpose": "RESTORE",
        "key": {
            "id": 3,
            "name": "Normal",
            "autogen": false,
            "scope": "User",
            "notes": "The key used for general encryption",
            "versions": [
                {
                    "id": 3,
                    "version": 1,
                    "current": true,
                    "timestamp": 1523371339398
                }
            ]
        },
        "active": true,
        "type": "agent_tool"
    },
    {
        "tool_id": 3,
        "purpose": "OBJSTORE",
        "key": {
            "id": 3,
            "name": "Normal",
            "autogen": false,
            "scope": "User",
            "notes": "The key used for general encryption",
            "versions": [
                {
                    "id": 3,
                    "version": 1,
                    "current": true,
                    "timestamp": 1523371339398
                }
            ]
        },
        "active": true,
        "type": "agent_tool"
    }
]
```


## POST

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **202** | The agent will be created in an asynchronous job. The **Location** header gives the URL of the job to check for status. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current agent is not authorized to create new users |

| | |
|---|---|
| **406** | The **Accept** header must be set to a supported content type. |
| **409** | The new agent conflicts with an existing agent. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See agents collection schema agent_tool object.*

**Example**

```
Request: POST /rest/agents/1/tools

    {
        "tool_id": 1,
        "purpose": "BACKUP",
        "key": “toolKeyB”
        "active": true,
        "type": "agent_tool"
    }

Response: 202 Accepted | Location: /rest/jobs/schedulers/9
```

## /agents/{id}/tools/{tool_id}

### GET

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **200** | The response body will contain the requested agent tool |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized |
| **404** | An agent with the specified {id} and {tool_id} was not found. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

### Schema

*See agents collection schema agent_tool object.*

### Example

```
Request: GET /rest/agents/1/tools/1

Response: 200 OK

    {
        "tool_id": 1,
        "purpose": "BACKUP",
        "key": {
            "id": 3,
            "name": "Normal",
            "autogen": false,
            "scope": "User",
            "notes": "The key used for general encryption",
            "versions": [
                {
                    "id": 3,
                    "version": 1,
                    "current": true,
                    "timestamp": 1523371339398
                }
            ]
        },
        "active": true,
        "type": "agent_tool"
    }
```

## PUT

### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

### Status codes

| Code | Description |
|---|---|
| **202** | The feature will be updated in an asynchronous job. The **Location** header gives the URL of the job to check for status. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change agent tool. |
| **404** | An agent with the specified {id} and {tool_id} was not found. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

### Schema

*See agents collection schema agent_tool object.*

### Example

```
Request: PUT /rest/agents/86/object_store
{
    {
        "id": 11,
        "mofn": "2:3",
        "certificate": {
            "id": 45,
            "agent_id": 90,
            "subject": "CN=192.168.1.26",
            "fingerprint":
"432663eaf2ae18e44be41a9185f1b7cd7e902cdbdc7006b884a2ec678fcedb9b",
            "expiry": 1606341305000,
            "has_privkey": true,
            "state": "ACTIVE",
            "notes": "",
            "type": "agent-certificate"
        },
        "backends": [
            {
                "id": 25,
                "url": "https://s3.us-south.objectstorage.softlayer.net",
                "key_id": "owerjl78937jkjkj8888",
                "api_key": null,
                "auth_type": "S3_IBM4",
                "share": 1,
                "type": "object-store-backend"
            },
            {
                "id": 26,
                "url": "https://s3.us-south.objectstorage.softlayer.net",
                "key_id": "78937jkjkj9owerjl999",
                "api_key": null,
                "auth_type": "S3_IBM4",
                "share": 2,
                "type": "object-store-backend"
            },
            {
                "id": 27,
                "url": "https://s3.us-south.objectstorage.softlayer.net",
                "key_id": "7jkjkj9ower7893jl999",
                "api_key": null,
                "auth_type": "S3_IBM4",
                "share": 3,
                "type": "object-store-backend"
            }
        ],
        "credentials": [
            {
                "id": 17,
                "key_id": "344Q75QUHMTRYUCN4QT7",
                "access_key": null,
                "auth_type": "S3_IBM4",
                "type": "object-store-credential"
            },
            {
                "id": 18,
                "key_id": "411QJK9Y5450MWALTKY5",
                "access_key": null,
                "auth_type": "S3_IBM4",
                "type": "object-store-credential"
            }
        ],
        "buckets": [
            {
                "id": 17,
                "name": "bucket1",
                "log_denial": false,
                "policy": [
                    {
                        "id": 33,
                        "cred_id": "344Q75QUHMTRYUCN4QT7",
                        "access": "RO",
                        "log": false,
                        "type": "object-store-bucket-policy"
```

```
                    },
                    {
                        "id": 34,
                        "cred_id": "411QJK9Y5450MWALTKY5",
                        "access": "RW",
                        "log": true,
                        "type": "object-store-bucket-policy"
                    }
                ],
                "type": "object-store-bucket"
            },
            {
                "id": 18,
                "name": "bucket2",
                "log_denial": false,
                "policy": [
                    {
                        "id": 35,
                        "cred_id": "344Q75QUHMTRYUCN4QT7",
                        "access": "RW",
                        "log": true,
                        "type": "object-store-bucket-policy"
                    },
                    {
                        "id": 36,
                        "cred_id": "411QJK9Y5450MWALTKY5",
                        "access": "WO",
                        "log": false,
                        "type": "object-store-bucket-policy"
                    }
                ],
                "type": "object-store-bucket"
            }
        ],
        "source": 0,
        "snapshot": false,
        "snapshotIndex": 0,
        "type": "object-store"
    }
}

Response: 202 Accepted | Location: /rest/jobs/schedulers/9
```

## PATCH

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **202** | The feature will be updated in an asynchronous job. The **Location** header gives the URL of the job to check for status. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change policy bindings. |
| **404** | An agent with the specified {id} and {tool_id} was not found. |
| **406** | The **Accept** header must be set to a supported content type. |

| | |
|---|---|
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See agents collection schema agent_tool object.*

**Example**

```
Request: PATCH /rest/agents/1/tools/1

    {
        "key": “toolKeyB”
    }

Response: 202 Accepted | Location: /rest/jobs/schedulers/9
```

## DELETE

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **202** | A job was created to process deletion. See the **Location** header for job URL |
| **204** | The delete was successful |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | A user with the specified {id} and {tool_id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

-

**Example**

```
Request: DELETE /rest/agents/1/tools/1

Response: 202 Accepted | Location: /rest/jobs/schedulers/9
```

## /agents/{id}/object_store

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain the requested agent tool |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized |
| **404** | An agent with the specified {id} was not found. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

*See agents collection schema on the object store object.*

#### Example

```
Request: GET /rest/agents/1/object_store

Response: 200 OK

    {
    "id": 9,
    "mofn": "2:3",
    "certificate": {
        "id": 42,
        "agent_id": 86,
        "subject": "CN=192.168.1.27",
        "fingerprint":
 "24d2327171ce69c5080081579f2be4ddc3fc19ca988b8030c430460df17a931b",
        "expiry": 1606341293000,
        "has_privkey": false,
        "state": "ACTIVE",
        "notes": "",
        "type": "agent-certificate"
    },
    "backends": [
        {
            "id": 19,
            "url":
"https://s3.us-south.objectstorage.softlayer.net",
            "key_id": "owerjl78937jkjkj8888",
            "api_key": null,
            "auth_type": "S3_IBM4",
            "share": 1,
            "type": "object-store-backend"
        },
```

```
        {
            "id": 20,
            "url":
 "https://s3.us-south.objectstorage.softlayer.net",
            "key_id": "78937jkjkj9owerjl999",
            "api_key": null,
            "auth_type": "S3_IBM4",
            "share": 2,
            "type": "object-store-backend"
        },
        {
            "id": 21,
            "url":
"https://s3.us-south.objectstorage.softlayer.net",
            "key_id": "7jkjkj9ower7893jl999",
            "api_key": null,
            "auth_type": "S3_IBM4",
            "share": 3,
            "type": "object-store-backend"
        }
    ],
    "credentials": [
        {
            "id": 13,
            "key_id": "344Q75QUHMTRYUCN4QT7",
            "access_key": null,
            "auth_type": "S3_IBM4",
            "type": "object-store-credential"
        },
        {
            "id": 14,
            "key_id": "411QJK9Y5450MWALTKY5",
            "access_key": null,
            "auth_type": "S3_IBM4",
            "type": "object-store-credential"
        }
    ],
    "buckets": [
        {
            "id": 13,
            "name": "bucket1",
            "log_denial": true,
            "policy": [
                {
                    "id": 25,
                    "cred_id": "344Q75QUHMTRYUCN4QT7",
                    "access": "RO",
                    "log": false,
                    "type": "object-store-bucket-policy"
                },
                {
                    "id": 26,
                    "cred_id": "411QJK9Y5450MWALTKY5",
                    "access": "RW",
                    "log": true,
                    "type": "object-store-bucket-policy"
                }
            ],
            "type": "object-store-bucket"
        },
        {
            "id": 14,
            "name": "bucket2",
            "log_denial": false,
            "policy": [
                {
                    "id": 27,
                    "cred_id": "344Q75QUHMTRYUCN4QT7",
                    "access": "RW",
                    "log": true,
                    "type": "object-store-bucket-policy"
                },
                {
                    "id": 28,
                    "cred_id": "411QJK9Y5450MWALTKY5",
                    "access": "WO",
                    "log": false,
                    "type": "object-store-bucket-policy"
                }
            ],
            "type": "object-store-bucket"
        }
```

```
    ],
    "source": 0,
    "snapshot": false,
    "snapshotIndex": 0,
    "type": "object-store"
}
```

## PUT

### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain the requested agent tool |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized |
| **404** | An agent with the specified {id} was not found. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

### Schema

*See agents collection schema on the object store object.*

### Example

```
Request: PUT /rest/agents/86/object_store
{
    {
        "id": 11,
        "mofn": "2:3",
        "certificate": {
            "id": 45,
            "agent_id": 90,
            "subject": "CN=192.168.1.26",
            "fingerprint":
"432663eaf2ae18e44be41a9185f1b7cd7e902cdbdc7006b884a2ec678fcedb9b",
            "expiry": 1606341305000,
            "has_privkey": true,
            "state": "ACTIVE",
            "notes": "",
            "type": "agent-certificate"
        },
        "backends": [
            {
                "id": 25,
                "url": "https://s3.us-south.objectstorage.softlayer.net",
                "key_id": "owerjl78937jkjkj8888",
                "api_key": null,
                "auth_type": "S3_IBM4",
                "share": 1,
                "type": "object-store-backend"
            },
            {
                "id": 26,
```

```
            "url": "https://s3.us-south.objectstorage.softlayer.net",
            "key_id": "78937jkjkj9owerjl999",
            "api_key": null,
            "auth_type": "S3_IBM4",
            "share": 2,
            "type": "object-store-backend"
        },
        {
            "id": 27,
            "url": "https://s3.us-south.objectstorage.softlayer.net",
            "key_id": "7jkjkj9ower7893jl999",
            "api_key": null,
            "auth_type": "S3_IBM4",
            "share": 3,
            "type": "object-store-backend"
        }
    ],
    "credentials": [
        {
            "id": 17,
            "key_id": "344Q75QUHMTRYUCN4QT7",
            "access_key": null,
            "auth_type": "S3_IBM4",
            "type": "object-store-credential"
        },
        {
            "id": 18,
            "key_id": "411QJK9Y5450MWALTKY5",
            "access_key": null,
            "auth_type": "S3_IBM4",
            "type": "object-store-credential"
        }
    ],
    "buckets": [
        {
            "id": 17,
            "name": "bucket1",
            "log_denial": false,
            "policy": [
                {
                    "id": 33,
                    "cred_id": "344Q75QUHMTRYUCN4QT7",
                    "access": "RO",
                    "log": false,
                    "type": "object-store-bucket-policy"
                },
                {
                    "id": 34,
                    "cred_id": "411QJK9Y5450MWALTKY5",
                    "access": "RW",
                    "log": true,
                    "type": "object-store-bucket-policy"
                }
            ],
            "type": "object-store-bucket"
        },
        {
            "id": 18,
            "name": "bucket2",
            "log_denial": false,
            "policy": [
                {
                    "id": 35,
                    "cred_id": "344Q75QUHMTRYUCN4QT7",
                    "access": "RW",
                    "log": true,
                    "type": "object-store-bucket-policy"
                },
                {
                    "id": 36,
                    "cred_id": "411QJK9Y5450MWALTKY5",
                    "access": "WO",
                    "log": false,
                    "type": "object-store-bucket-policy"
                }
            ],
            "type": "object-store-bucket"
        }
    ],
    "source": 0,
    "snapshot": false,
    "snapshotIndex": 0,
```

```
        "type": "object-store"
    }
}

Response: 202 Accepted | Location: /rest/jobs/schedulers/9
```

## PATCH

### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

### Status codes

| Code | Description |
|---|---|
| **202** | The feature will be updated in an asynchronous job. The **Location** header gives the URL of the job to check for status. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change policy bindings. |
| **404** | An agent with the specified {id} was not found. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

### Schema

*See agents collection schema on the object store object.*

### Example

```
Request: PATCH /rest/agents/1/object_store

    {
       “id”: 11,
       “credentials”:[{
               “id”: 17,
               “key_id”: “244Q75QUHMTRYUCN4QT7"
       }],
       “type”: “object-store”
    }

Response: 202 Accepted | Location: /rest/jobs/schedulers/9
```

# /agents/{id}/object_store/credentials

POST

### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **200** | The response body will contain the requested agent tool |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | An agent with the specified {id} was not found. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning**header for more information. |
| **503** | A server service is not available. See the **Warning**header for more information. |

**Schema**

*See agents collection schema on the credentials object under the object_store object*

**Example**

```
Request: POST /rest/agents/1/object_store/credentials

{
    "key_id": "344Q75QUHMTRYUCN4QT7",
    "access_key": "hdclImOt2kQ7xxchmRkGxZyuGfGlOnotActualKey",
    "auth_type": "S3_IBM4"
}

Response: 202 Accepted | Location: /rest/jobs/schedulers/9
```

**DELETE**
**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **202** | The feature will be updated in an asynchronous job. The **Location** header gives the URL of the job to check for status. |
| **400** | The request body was not formed properly. See the **Warning**header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change agent tool. |

| | |
|---|---|
| **404** | An agent with the specified {id} and {cid} was not found. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning**header for more information. |
| **503** | A server service is not available. See the **Warning**header for more information. |

**Schema**

*See agents collection schema credential object under the object_store.*

**Example**

```
Request: DELETE /rest/agents/86/object_store/credentials/<cid>

Response: 202 Accepted | Location: /rest/jobs/schedulers/9
```

## /agents/{id}/object_store/buckets

GET

**GETQuery parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **200** | The response body will contain the requested agent tool |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | An agent with the specified {id} with object store and buckets was not found. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning**header for more information. |
| **503** | A server service is not available. See the **Warning**header for more information. |

**Schema**

*See agents collection schema object store agent object and buckets object.*

**Example**

```
Request: GET /rest/agents/1/object_store/buckets

Response: 200 OK
[
```

```
    {
        "id": 17,
        "name": "bucket1",
        "log_denial": true,
        "policy": [
            {
                "id": 33,
                "cred_id": "344Q75QUHMTRYUCN4QT7",
                "access": "RO",
                "log": false,
                "type": "object-store-bucket-policy"
            },
            {
                "id": 34,
                "cred_id": "411QJK9Y5450MWALTKY5",
                "access": "RW",
                "log": true,
                "type": "object-store-bucket-policy"
            }
        ],
        "type": "object-store-bucket"
    },
    {
        "id": 18,
        "name": "bucket2",
        "log_denial": false,
        "policy": [
            {
                "id": 35,
                "cred_id": "344Q75QUHMTRYUCN4QT7",
                "access": "RW",
                "log": true,
                "type": "object-store-bucket-policy"
            },
            {
                "id": 36,
                "cred_id": "411QJK9Y5450MWALTKY5",
                "access": "WO",
                "log": false,
                "type": "object-store-bucket-policy"
            }
        ],
        "type": "object-store-bucket"
    }
]
```


**POST**
**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **200** | The response body will contain the requested agent tool |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | An agent with the specified {id} } and object store was not found. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning**header for more information. |

| | |
|---|---|
| **503** | A server service is not available. See the **Warning**header for more information. |

**Schema**

*See agents collection schema on the buckets object under the object_store object*

**Example**

```
Request: POST /rest/agents/1/object_store/buckets

 {
        "name": "bucket12",
        "log_denial": true,
        "policy": [
            {
                "cred_id": "344Q75QUHMTRYUCN4QT7",
                "access": "RO",
                "log": false,
                "type": "object-store-bucket-policy"
            },
            {
                "cred_id": "411QJK9Y5450MWALTKY5",
                "access": "RW",
                "log": true,
                "type": "object-store-bucket-policy"
            }
        ],
        "type": "object-store-bucket"
    }
Response: 202 Accepted | Location: /rest/jobs/schedulers/9
```

**DELETE**
**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **202** | The feature will be updated in an asynchronous job. The **Location** header gives the URL of the job to check for status. |
| **400** | The request body was not formed properly. See the **Warning**header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change agent tool. |
| **404** | An agent with the specified {id} and/or {bucket id} was not found. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning**header for more information. |
| **503** | A server service is not available. See the **Warning**header for more information. |

**Schema**

*See agents collection schema on the buckets object under the object_store object*

**Example**

```
Request: DELETE /rest/agents/86/object_store/buckets/<bid>

Response: 202 Accepted | Location: /rest/jobs/schedulers/9
```

## /agents/{id}/object_store/buckets/<bid>/policy

**POST**
**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **200** | The response body will contain the requested agent tool |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | An agent with the specified {id} and/or {bucket_id} was not found. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning**header for more information. |
| **503** | A server service is not available. See the **Warning**header for more information. |

**Schema**

*See agents collection schema on the policy object under the buckets and object_store object*

**Example**

```
Request: POST /rest/agents/1/object_store/buckets/<bid>/policy

  {
        "cred_id": "344Q75QUHMTRYUCN4QT7",
        "access": "RO",
        "log": false,
        "type": "object-store-bucket-policy"
    }
```

**DELETE**
**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **202** | The feature will be updated in an asynchronous job. The **Location** header gives the URL of the job to check for status. |
| **400** | The request body was not formed properly. See the **Warning**header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change agent tool. |
| **404** | An agent with the specified {id} and/or {bucket_id} and/or {policy_id} was not found. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning**header for more information. |
| **503** | A server service is not available. See the **Warning**header for more information. |

**Schema**

*See agents collection schema on the policy object under the buckets and object_store object*

**Example**

```
Request: DELETE /rest/agents/86/object_store/buckets/<bid>/policy/<pid>

Response: 202 Accepted | Location: /rest/jobs/schedulers/9
```

## /agents/{id}/policy/<pid>

The known certificates for a specific managed agent

**GET**
**Query parameters**

| Parameter | Description |
|---|---|
| <none> | |

**Status codes**

| Code | Description |
|---|---|
| **200** | The response body will contain the requested agent certificates |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | An agent with the specified {id} was not found and has not existed before. The certificates field will always exist for an extant agent, even if the field is empty. |

| | |
|---|---|
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning**header for more information. |
| **503** | A server service is not available. See the **Warning**header for more information. |

**Schema**

*See agents collection schema, agent-certificate object.*

**Example**

```
Request: GET /rest/agents/4/certificates

Response: 200 OK

[
    {
        "subject": "CN=localhost,OU=Development,O=Test.,L=Test,ST=CA,C=US",
        "fingerprint": "a15b027baaa1182b32e160dff4955bb2064d2d0b",
        "expiry": 1478733066000,
        "id": 6,
        "type": "agent-certificate"
    }
]
```

**POST**
**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **201** | The certificate has been created. The **Location** header gives the URL of the new certificate. |
| **400** | The request body was not formed properly. See the **Warning**header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to create certificates. |
| **406** | The **Accept** header must be set to a supported content type. |
| **409** | The new certificate conflicts with an existing certificate. |
| **500** | An internal error has occurred. See the **Warning**header for more information. |
| **503** | A server service is not available. See the **Warning**header for more information. |

**Schema**

*See agent collection schema, agent-certificate object.* ***Changes:*** *id's are neither required nor should they be set.*

### Example

```
Request: POST /rest/agents/4/certificates

{
    "id": 4,
    "type": "client-file",
    "href": "/rest/files/upload/4"
    "notes": "After expiration, contact person@example.com for a new one"
}

Response: 201 Created | Location: /rest/agents/4/certificates/7
```

# Chapter 9. User Accounts

## /users

All server level users

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain all known users |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

```
{
    "type": "array",
    "minItems": 1,
    "uniqueItems": true,
    "items": [
        {
            "type": "object",
            "required": [ "type", "username",
 "password", "password_confirm", "state", "roles", "properties"],
            "additionalProperties": false,
            "properties": {
                "type": { "enum": [ "user" ] },
                "username": "number",
                "password": "string",
                "password_confirm": "string",
                "state": { "enum": [ "enabled",
 "disabled", "expired", "locked" ] },
                "roles": {
                    "type": "array",
                    "minItems": 0,
                    "uniqueItems": true,
                    "items": [
                        { "type": "string" }
                    ]
                },
                "properties": {
                    "type": "object",
                    "required": [ "notes", "passwordLastModified"],
                    "additionalProperties": true,
                    "properties": {
```

```
                        "notes": "string",
                        "passwordLastModified": "number",
                    }
                }
            }
        }
    ]
}
```

**Example**

```
Request: GET /rest/users

Response: 200 OK

[
    {
        "type": "user",
        "username": "admin",
        "password": null,
        "password_confirm": null,
        "state": "enabled",
        "roles": ["product-administrator", "admin"],
        "properties": {
            "notes": "Paul's user account",
            "passwordLastModified": 1460911763077
        }
    }
]
```

## POST

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **201** | The user has been created. The **Location** header gives the URL of the new user. |
| **202** | The user will be created in an asynchronous job. The **Location** header gives the URL of the job to check for status. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to create new users |
| **406** | The **Accept** header must be set to a supported content type. |
| **409** | The new user conflicts with an existing user. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

### Schema

*See users collection schema, user object.* ***Changes:*** *properties.passwordLastModified should not be set*

### Example

```
Request: POST /rest/users

{
    "type": "user",
    "username": "admin-joe2",
    "password": "MyPassword12345",
    "password_confirm": "MyPassword12345",
    "state": "enabled",
    "roles": ["policy-server-administrator"],
    "properties": {
        "notes": "By management request, a second user account for Joe.",
    }
}

Response: 202 Accepted | Location: /jobs/67
```

## /users/{username}

A specific server user

## GET

### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain the requested user |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | A user with the specified {username} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

### Schema

*See users collection schema, user object.*

### Example

```
Request: GET /rest/users/admin

Response: 200 OK
```

```
{
    "type": "user",
    "username": "admin",
    "password": null,
    "password_confirm": null,
    "state": "enabled",
    "roles": ["policy-server-administrator", "admin"],
    "properties": {
        "notes": "",
        "passwordLastModified": 1460911763077
    }
}
```

## DELETE

### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

### Status codes

| Code | Description |
|---|---|
| **202** | A job was created to process deletion. See the **Location** header for job URL |
| **204** | The delete was successful |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | A user with the specified {username} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

### Schema

*No body sent on DELETE*

### Example

```
Request: DELETE /rest/users/admin

Response: 202 Accepted | Location: /rest/jobs/5
```

## PATCH

### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **202** | A job was created to process the change. See the **Location** header for job URL |
| **204** | The request was successful, but contained no changes. Client side representation does not need to be refreshed. |
| **205** | The request was successful and the resource was updated. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change users. |
| **404** | A user with the specified {username} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **409** | The changes to the user conflict with another user. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See users collection schema, user object.* ***Changes:*** *Only type and username are required fields.*

**Example**

```
Request: PATCH /rest/users/admin

{
    "properties": {
        "notes": "updated notes for Paul",
    },
    "type": "user"
}

Response: 205 Reset Content
```

## PUT

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **202** | A job was created to process the change. See the **Location** header for job URL |

| | |
|---|---|
| **204** | The request was successful, but contained no changes. Client side representation does not need to be refreshed. |
| **205** | The request was successful and the resource was updated. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change users. |
| **404** | A user with the specified {username} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **409** | The changes to the user conflict with another user. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See users collection schema, user object.*

**Example**

```
Request: PUT /rest/users/admin

{
    "type": "user",
    "username": "admin",
    "password": null,
    "password_confirm": null,
    "state": "disabled",
    "roles": ["product-administrator", "admin"],
    "properties": {
        "notes": " Disabled 2016-04-15 due to temporary reassignment.",
        "passwordLastModified": 1460911763077
    }
}

Response: 205 Reset Content
```

## /roles

All server user roles

### GET

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **200** | The response body will contain all known roles |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

```
{
    "type": "array",
    "minItems": 2,
    "uniqueItems": true,
    "items": [
        {
            "type": "object",
            "required": [ "type", "id", "permissions"],
            "additionalProperties": false,
            "properties": {
                "type": { "enum": [ "rbac-role" ] },
                "id": "string",
                "permissions": {
                    "type": "array",
                    "minItems": 1,
                    "uniqueItems": true,
                    "items": [
                        {
                            "type": "object",
                            "required": [ "type", "id", "actions"],
                            "additionalProperties": false,
                            "properties": {
                                "type": { "enum": [ "rbac-permission" ] },
                                "id": "string",
                                "actions": {
                                    "type": "array",
                                    "minItems": 1,
                                    "uniqueItems": true,
                                    "items": [
                                        { "type": "string" }
                                    ]
                                }
                            }
                        ]
                    }
            }
        }
    ]
}
```

**Example**

```
Request: GET /rest/roles

Response: 200 OK

[
    {
        "type": "rbac-role",
        "id": "product-administrator",
        "permissions": [
            {
                "id": "see-keystore-settings",
                "actions": [
```

```
                    "see-keystore-settings"
                ],
                "type": "rbac-permission"
            },
            {
                "id": "see-licenses",
                "actions": [
                    "see-licenses"
                ],
                "type": "rbac-permission"
            }
        ]
    }
]
```

## /roles/{id}

A specific server user role

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain the requested role |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | A role with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

*See roles collection schema, rbac-role object.*

#### Example

```
Request: GET /rest/roles/product-administrator

Response: 200 OK

{
    "type": "rbac-role",
    "id": "product-administrator",
    "permissions": [
        {
            "id": "see-keystore-settings",
            "actions": [
                "see-keystore-settings"
            ],
```

```
            "type": "rbac-permission"
        },
        {
            "id": "see-licenses",
            "actions": [
                "see-licenses"
            ],
            "type": "rbac-permission"
        }
    ]
}
```

## /sessions

All server user sessions

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain all active user sessions |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

```
{
    "type": "array",
    "minItems": 1,
    "uniqueItems": true,
    "items": [
        {
            "type": "object",
            "required": [ "type", "id",
 "user", "roles", "activated_permissions", "when_started"],
            "additionalProperties": false,
            "properties": {
                "type": { "enum": [ "web-session" ] },
                "id": "string",
                "user": "string",
                "when_started": "number",
                "roles": {
                    "type": "array",
                    "minItems": 0,
                    "uniqueItems": true,
                    "items": [
                        { "type": "string" }
                    ]
                }
```

```
                "activated_permissions": {
                    "type": "array",
                    "minItems": 0,
                    "uniqueItems": true,
                    "items": [
                        { "type": "string" }
                    ]
                }
            }
        }
    ]
}
```

**Example**

```
Request: GET /rest/sessions

Response: 200 OK

[
    {
        "type": "web-session",
        "id": "1d3eldnegio001ksj59gedjln5",
        "user": "admin",
        "when_started": 1460911832224,
        "roles": [
            "admin",
            "product-administrator"
        ],
        "activated_permissions": [
            "add-product-administrator-role",
            "approve-agents",
            "change-agents",
            "change-own-user"
        ]
    }
]
```

**Notes**

*Right now only the currently authenticated user session will be exposed.*

## POST

This request method on this resource is used to authenticate and start a new HTTP session.

•

**Supported content types**

| Content type | Description |
|---|---|
| **application/x-www-form-urlencoded** | Used for both form authentication and programmatic authentication in the first release. |

**Status codes**

| Code | Description |
|---|---|
| **201** | The session has been started. The current session resource URL will be given in the Location header of the response |
| **401** | The credentials are invalid, the body is not properly formed, or other problems - intentionally obfuscated to reduce information exposure when targeted by malicious activity |

### Schema

```
{
    "type": "string",
    "description":
"URL encoded parameter list of j_username,
 j_password, and optionally j_directory with
 values set to the username, password, and
 directory respectively"
}
```

### Example

```
Request: POST /rest/sessions

j_username=admin&j_password=MyPass12345

Response: 201 Created | Location: /rest/sessions/i1et8oc714vekl534fso3uw9


Request: POST /rest/sessions

j_username=admin&j_password=MyPass12345&j_directory=IT_LDAP

Response: 201 Created | Location: /rest/sessions/1a2ascg8qi7dr16pue3nvvv4dz
```

### Notes

*For the POST to be accepted, the CSRF token will have to be valid. Thus the HTTP session must be started (usually by a GET to another resource, /rest/sessions being a particularly useful choice) before the authenticated session can be started with the POST to the sessions collection.*

*A recommend flow is to do a GET on /rest/sessions, check the response code, and then perform the POST to /rest/sessions only if the response code is 401.*

## /sessions/{id}

A specific server session.

## GET

### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain the requested session. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | A session with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |

| | |
|---|---|
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See sessions collection schema, sessions object.*

**Example**

```
Request: GET /rest/sessions/1d3eldnegio001ksj59gedjln5

Response: 200 OK

{
    "type": "web-session",
    "id": "1d3eldnegio001ksj59gedjln5",
    "user": "admin",
    "when_started": 1460911832224,
    "roles": [
        "admin",
        "product-administrator"
    ],
    "activated_permissions": [
        "add-product-administrator-role",
        "approve-agents",
        "change-agents",
        "change-own-user"
    ]
}
```

## DELETE

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **204** | The delete was successful, meaning that the targeted session is no longer valid |
| **401** | The current request is not authenticated. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |

**Schema**

*No body sent on DELETE*

**Example**

```
Request: DELETE /rest/sessions/i1et8oc714vekl534fso3uw9

Response: 204 No content
```

# Chapter 10. Server Settings

## /settings

The collection of all known logs

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain a collection of references to child settings resources. The references will not contain any settings themselves. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

```
{
    "type": "array",
    "minItems": 0,
    "uniqueItems": true,
    "items": [
        {
            "type": "object",
            "required": ["type","id", "href" ],
            "additionalProperties": true,
            "properties": {
                "type": { "enum": [ "ref" ] },
                "id": { "enum": [ "directories", "keystores"] },
                "href": "string"
            }
        }
    ]
}
```

#### Example

```
Request: GET /settings

Response: 200 OK
```

```
[
    {
        "type": "ref",
        "id": "directories",
        "href": "/rest/settings/directories",
    },
    {
        "type": "ref",
        "id": "keystores",
        "href": "/rest/settings/keystores",
    }
]
```

## /settings/directories

Server LDAP directories configuration

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain the product keystore settings |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

```
{
    "type": "array",
    "minItems": 0,
    "uniqueItems": true,
    "items": [
        {
            "type": "object",
            "required": ["type", "id", "host", "port", "binddn", "secure"],
            "additionalProperties": false,
            "properties": {
                "type": { "enum": [ "directory" ] },
                "id": "string",
                "host": "string",
            "port": "number",
            "binddn": "string",
            "directory_type": { "enum": [ "LDAP", "ACTIVE_DIRECTORY" ] },
            "secure": "boolean"
        }
    ]
}
```

### Notes

The **binddn** field is ignored when the **directory_type** is set to **ACTIVE_DIRECTORY**. Callers are advised to set **binddn** to **null** or an empty string in this case to avoid any confusion.

### Example

```
Request: GET /rest/settings/directories

Response: 200 OK

[
    {
        "id": "human-resources",
        "host": "192.168.4.114",
        "port": 389,
        "host": "LDAP",
        "binddn": "uid={$username},cn=humanresources,ou=Groups,dc=example,dc=com",
        "secure": false,
        "type": "directory"
    }
]
```

## POST

### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

### Status codes

| Code | Description |
|---|---|
| **201** | A job to create the new directory has been created. The **Location** header gives the URL of the new job. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to create directories. |
| **406** | The **Accept** header must be set to a supported content type. |
| **409** | The new directory conflicts with an existing directory id. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

### Schema

*See* directories *collection schema, directory objects.* ***Changes:*** *id is neither required nor should it be set for either object type.*

### Example

```
Request: POST /rest/settings/directories

{
```

```
    "id": "human-resources",
    "host": "192.168.4.114",
    "port": 389,
    "host": "LDAP",
    "binddn": "uid={$username},cn=humanresources,ou=Groups,dc=example,dc=com",
    "secure": false,
    "type": "directory"
}

Response: 201 Created | Location: /rest/jobs/1
```

## /settings/directories/{id}

A specific LDAP directory configuration

### GET

#### Query parameters

#### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain directory settings for a specific interface |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

*See directories settings collection schema, directory object.*

#### Example

```
Request: GET /rest/settings/directory/human-resources

Response: 200 OK

{
    "id": "human-resources",
    "host": "192.168.4.114",
    "port": 389,
    "host": "LDAP",
    "binddn": "uid={$username},cn=humanresources,ou=Groups,dc=example,dc=com",
    "secure": false,
    "type": "directory"
}
```

### DELETE

#### Query parameters

| Parameter | Description |
|---|---|

| | |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **204** | The delete was successful |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | A directory with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*No body sent on DELETE*

**Example**

```
Request: DELETE /rest/settings/directories/human-resources

Response: 204 No content
```

## PUT

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **204** | The request was successful, but contained no changes. Client side representation does not need to be refreshed. |
| **205** | The request was successful and the resource was updated. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change process sets |
| **404** | A directory with the specified {id} was not found and has not existed before |

| | |
|---|---|
| **406** | The **Accept** header must be set to a supported content type. |
| **409** | The changes to the directory conflict with another directory. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See directories collection schema, directory objects.*

**Example**

```
Request: PUT /rest/settings/directories/tenant1

{
    "id": "human-resources",
    "host": "192.168.2.134",
    "port": 389,
    "host": "LDAP",
    "binddn": "uid={$username},cn=humanresources,ou=Groups,dc=example,dc=com",
    "secure": false,
    "type": "directory"
}

Response: 205 Reset content
```

## /settings/keystores

Server keystore configuration

### GET

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **200** | The response body will contain the product keystore settings |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

### Schema

```
{
    "type": "array",
    "minItems": 0,
    "uniqueItems": true,
    "items": [
        {
            "type": "object",
            "required": ["type", "id", "state", "name",
 "product", "host", "port", "keystore", "keystore_alias",
 "truststore", "truststore_alias", "truststore_password", "master_keystore_id" ],
            "additionalProperties": false,
            "properties": {
                "type": { "enum": [ "keystore-kmip", “keystore-hsm" ] },
                "id": "number",
                "name ": "string",
                "state": { "enum": [ "ACTIVE", "INACTIVE" ] },
                "product": { "enum": [ "IBM_SKLM", “HSM” ] },
                "host": "string",
                "port": "number",
                "keystore":  {
                    "type": "object",
                    "required": [ "type", "id", "name", "href"],
                    "additionalProperties": false,
                    "properties": {
                        "type": { "enum": [ "ref" ] },
                        "id": "number",
                        "name": "string",
                        "href": "string"
                    },
                "keystore_alias": "string",
                "keystore_password": "string",
                "truststore": {
                    "type": "object",
                    "required": [ "type", "id", "name", "href"],
                    "additionalProperties": false,
                    "properties": {
                        "type": { "enum": [ "ref" ] },
                        "id": "number",
                        "name": "string",
                        "href": "string"
                    },
                "truststore_alias": "string",
                "truststore_password": "string",
        “hsmpassword”: “string”,
              “hsmtoken”: “string”,
              “key_handle”: “string”,
                "master_keystore_id": "number"
            }
        }
    ]
}
```

### Example

```
Request: GET /rest/settings/keystores

Response: 200 OK

[
    {
        "id": 1,
        "name": "builtin",
        "state": "INACTIVE",
        "product": "IBM_SKLM",
        "host": null,
        "port": 0,
        "keystore": null,
        "keystore_alias": null,
        "keystore_password": "*******",
        "truststore": null,
        "truststore_alias": null,
        "truststore_password": "*******",
        "master_keystore_id": 0,
        "type": "keystore-kmip"
    }

    {
```

```
        id": 1,
        "name": "hsm",
        "state": "INACTIVE",
        "product": "HSM ",
        "hsmpassword": null,
        "hsmtoken": “0”,
        "key_handle": “441”,
        "master_keystore_id": 0,
        "type": "keystore-hsm"
    }

]
```

## POST

### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

### Status codes

| Code | Description |
|---|---|
| **201** | A job to create the new keystore has been created. The **Location** header gives the URL of the new job. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to create directories. |
| **406** | The **Accept** header must be set to a supported content type. |
| **409** | The new directory conflicts with an existing directory id. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

### Schema

*See* keystores *collection schema, keystore objects.* ***Changes:*** *id is neither required nor should it be set..*

### Example

```
KMIP Keystores:
Request: POST /rest/settings/keystores

{
    "name": "standby",
    "state": "INACTIVE",
    "product": "IBM_SKLM",
    "host": "192.168.4.133",
    "port": 5696,
    "keystore": {
        "name": "sklm-client-keystore.jks",
        "id": 42,
        "type": "ref",
        "href": "/rest/files/upload/42"
    },
    "keystore_alias": "sklmclient",
```

```
    "keystore_password": "keystorePa55word123",
    "truststore": {
        "name": "sklm-truststore.jks",
        "id": 43,
        "type": "ref",
        "href": "/rest/files/upload/43"
    },
    "keystore_alias": "sklmserver",
    "keystore_password": "trustp@ss123",
    "master_keystore_id": 1,
    "type": "keystore-kmip"
}

Response: 202 Accepted | Location: /rest/jobs/112

HSM keystores:
Request: POST /rest/settings/keystores

{
        "name": "hsm",
        "state": "INACTIVE",
        "product": "HSM",
     "hsmpassword": null,
        "hsmtoken": "0",
        "keystore_handle": "441",
        "master_keystore_id": 0,
        "type": "keystore-hsm"
}

Response: 202 Accepted | Location /rest/jobs/112
```

## /settings/keystores/{id}

A specific keystore configuration

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain keystore settings for a specific interface |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

*See keystore settings collection schema, keystore object.*

### Example

```
Request: GET /rest/settings/keystores/1
KMIP Keystores:
Response: 200 OK

{
    "id": 1,
    "name": null,
    "state": "INACTIVE",
    "product": "IBM_SKLM",
    "host": null,
    "port": 0,
    "keystore": null,
    "keystore_alias": null,
    "keystore_password": "*******",
    "truststore": null,
    "truststore_alias": null,
    "truststore_password": "*******",
    "master_keystore_id": 0,
    "type": "keystore-kmip"
}

HSM Keystores:
Request: GET /rest/settings/keystores/1

{
       "id": 1,
        "name": “hsm”,
        "state": "INACTIVE",
        "product": " HSM ",
       "hsmpassword": null,
       "hsmtoken": "0",
       "key_handle": "448",
       "master_keystore_id": 0,
       "type": "keystore-hsm"
}
```

## PUT

### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

### Status codes

| Code | Description |
|---|---|
| **204** | The request was successful, but contained no changes. Client side representation does not need to be refreshed. |
| **205** | The request was successful and the resource was updated. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change keystore settings |
| **404** | An interface with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |

| | |
|---|---|
| **409** | The changes to the interface conflict with another interface. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See keystore settings collection schema, keystore object.*

**Example**

```
KMIP Keystores:
Request: PUT /rest/settings/keystores/1

{
    "id": 1,
    "name": null,
    "state": "INACTIVE",
    "product": "IBM_SKLM",
    "host": "192.168.2.87",
    "port": 5696,
    "keystore": null,
    "keystore_alias": null,
    "keystore_password": "*******",
    "truststore": null,
    "truststore_alias": null,
    "truststore_password": "*******",
    "master_keystore_id": 0,
    "type": "keystore-kmip"
}

Response: 205 Reset Content

HSM Keystores:
Request: PUT /rest/settings/keystores/1
{
       "id": 1,
        "name": “hsm”,
        "state": "INACTIVE",
        "product": " HSM ",
       "hsmpassword": null,
       "hsmtoken": "0",
       "key_handle": "448",
       "master_keystore_id": 0,
       "type": "keystore-hsm"
}

Response: 205 Reset Content
```

## PATCH

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **204** | The request was successful, but contained no changes. Client side representation does not need to be refreshed. |

| | |
|---|---|
| **205** | The request was successful and the resource was updated. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change keystore settings |
| **404** | An interface with the specified {id} was not found and has not existed before |
| **406** | The **Accept** header must be set to a supported content type. |
| **409** | The changes to the interface conflict with another interface. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

*See keystore settings collection schema, keystore object.*

#### Example

```
KMIP Keystores:
Request: PATCH /rest/settings/keystores/1
{

{
    "id": 1,
    "state": “INACTIVE”,
    "type": "keystore-kmip"
}

Response: 205 Reset Content

HSM Keystores:
Request: PATCH /rest/settings/keystores/1
{

{
    "id": 1,
    "state": “INACTIVE”,
    "type": "keystore-hsm"
}

Response: 205 Reset Content
```

## DELETE

#### Schema

*No body sent on DELETE*

#### Example

```
KMIP or HSM Keystores:
Request: DELETE /rest/settings/keystores/1
Response: 204 No content
```

# Chapter 11. Archive and Upload Files

## /files

The collection of server generated archives (future releases) and customer uploaded files

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain a collection of references to child files or file collections. The references will not contain file content. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

```
{
    "type": "array",
    "minItems": 4,
    "uniqueItems": true,
    "items": [
        {
            "type": "object",
            "required": ["type", "id", "href" ],
            "additionalProperties": true,
            "properties": {
                "type": { "enum": [ "ref" ] },
                "id": { "enum": ["upload"] },
                "href": "string"
            }
        }
    ]
}
```

#### Example

```
Request: GET /rest/files

Response: 200 OK

[
```

```
    {
        "type": "ref",
        "id": "upload",
        "href": "/rest/files/upload",
    }
]
```

## /files/upload

A place for clients to upload files for use with other resources.

This resource enforces ownership visibility, so only files that a user has uploaded will be seen by that user. Once passed to another resource in a request, the file may be removed by the server. Manual deletion of the file is generally not required.

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain a collection of references to customer uploaded files. The references will not contain file content. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

```
{
    "type": "array",
    "minItems": 0,
    "uniqueItems": true,
    "items": [
        {
            "type": "object",
            "required": [ "type", "name", "id", "href" ],
            "additionalProperties": true,
            "properties": {
                "type": { "enum": [ "ref" ] },
                "type": "string",
                "id": "number",
                "href": "string"
            }
        }
    ]
}
```

### Example

```
Request: GET /rest/files/upload

Response: 200 OK

[
    {
        "name": "cert.pem",
        "id": 1,
        "type": "ref",
        "href": "/rest/files/upload/1"
    }
]
```

## POST

Upload a new file to the server

•

### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

### Status codes

| Code | Description |
|---|---|
| **201** | The new file has been uploaded. The **Location** header gives the URL of the file. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to upload files |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

### Schema

*No content should be sent on this POST.*

### Example

```
Request: POST /rest/files/upload

Response: 201 Created | Location: /files/upload/my_saved_backup_2015-10-15.tar.gz
```

# /files/upload/{id}

A specific client uploaded file

## GET

### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

### Status codes

| Code | Description |
|---|---|
| **200** | The response body will be a binary download of the file |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | An upload with the specified {id} was not found |
| **406** | The **Accept** header must be left unset or set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

### Schema

*Customer uploads are considered binary data and have no schema*

## DELETE

### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

### Status codes

| Code | Description |
|---|---|
| **204** | The delete was successful |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **404** | An upload with the specified {id} was not found |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

### Schema

*No body sent on DELETE*

### Example

```
Request: DELETE /rest/files/upload/my_saved_backup_2015-10-15.tar.gz

Response: 204 No Content
```

# Chapter 12. Advanced Properties

## /properties

Resource to configure advanced product properties

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain the properties summary |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

```
{
    "type": "array",
    "minItems": 0,
    "uniqueItems": true,
     "items": [
         {
              "type": "object",
              "required": [ "type", "key",
"scope", "value", "defaultValue", "needsRestart" ],
              "additionalProperties": false,
              "properties": {
                   "type": { "enum": [ "property" ] },
                   "key": "string",
                   "scope": { "enum": [ "LOCAL", "CLUSTER" ] },
                   "value": "string",
                   "defaultValue": "string",
                   "needsRestart": "string"
               }
          }
     ]
}
```

#### Example

```
Request: GET /rest/properties

Response: 200 OK
```

```
[
    {
        "key": "com.securityfirstcorp.atlantis.bundles.dataman.password",
        "alias": "node.db.password",
        "scope": "LOCAL",
        "value": "PASSWORD",
        "defaultValue": "PASSWORD",
        "needsRestart": false,
        "type": "property"
    },
    {
        "key": "com.securityfirstcorp.atlantis.bundles.dataman.initialConnections",
        "alias": "node.db.initConns",
        "scope": "LOCAL",
        "value": "1",
        "defaultValue": "1",
        "needsRestart": false,
        "type": "property"
    }
]
```


## /properties/{key}

A specific property

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain the specific property |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

*See property schema, property object.*

#### Example

```
Request: GET /rest/property/com.securityfirstcorp.atlantis.bundles.dataman.password

Response: 200 OK

{
    "key": "com.securityfirstcorp.atlantis.bundles.dataman.password",
    "alias": "node.db.password",
    "scope": "LOCAL",
```

```
    "value": "PASSWORD",
    "defaultValue": "PASSWORD",
    "needsRestart": false,
    "type": "property"
}
```

## PATCH

### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

### Status codes

| Code | Description |
|---|---|
| **204** | The request was successful, but contained no changes. Client side representation does not need to be refreshed. |
| **205** | The request was successful and the resource was updated. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change time settings |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

### Schema

*See property object schema.*

### Example

```
Request: PATCH /rest/property/com.securityfirstcorp.atlantis.bundles.dataman.password

{
    "value": "newPassword34"
}

Response: 205 Reset Content
```

# Chapter 13. Setup and Configuration Issues

## Setup and Configuration Issues

### /issues

Resource to resolve open product setup or ongoing configuration issues

**GET**

***Query parameters***

| Parameter | Description |
|---|---|
| **<none>** | |

***Status codes***

| Code | Description |
|---|---|
| **200** | The response body will contain the issues summary |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

***Schema***

```
{
    "type": "array",
    "minItems": 0,
    "uniqueItems": true,
    "items": [
        {
            "type": "object",
            "required": [ "type", "id",
 "issue_id", "dismiss_status", "importance", "permisstion_to_dismiss" ],
            "additionalProperties": false,
            "properties": {
                "type": { "enum": [ "issue" ] },
                "id": "number",
                "issue_id": "string",
                "dismiss_status": { "enum":
[ "DISMISSED", "NOT_ALLOWED", "NOT_DISMISSED", "UNKNOWN" ] },
                "importance": { "enum":
[ "HIGH", "MEDIUM", "LOW", "UNKNOWN" ] },
                "permisstion_to_dismiss": "boolean"
            }
        }
    ]
}
```

***Example***

```
Request: GET /rest/properties

Response: 200 OK

[
    {
        "id": 1,
        "issue_id": "US_2_policyAndSecurityRoles",
        "dismiss_status": "NOT_DISMISSED",
        "importance": "LOW",
        "permission_to_dismiss": true,
        "type": "issue"
    },
    {
        "id": 2,
        "issue_id": "US_1_onlyOneUser",
        "dismiss_status": "NOT_ALLOWED",
        "importance": "HIGH",
        "permission_to_dismiss": true,
        "type": "issue"
    }
]
```

## /issues/{id}

A specific property

**GET**

***Query parameters***

| Parameter | Description |
|---|---|
| **<none>** | |

***Status codes***

| Code | Description |
|---|---|
| **200** | The response body will contain the specific issue |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

***Schema***

*See property schema, property object.*

***Example***

```
Request: GET /rest/issues/1

Response: 200 OK

{
    "id": 1,
    "issue_id": "US_2_policyAndSecurityRoles",
    "dismiss_status": "NOT_DISMISSED",
    "importance": "LOW",
```

```
    "permission_to_dismiss": true,
    "type": "issue"
}
```

**PATCH**

***Query parameters***

| Parameter | Description |
|---|---|
| **<none>** | |

***Status codes***

| Code | Description |
|---|---|
| **204** | The request was successful, but contained no changes. Client side representation does not need to be refreshed. |
| **205** | The request was successful and the resource was updated. |
| **400** | The request body was not formed properly. See the **Warning** header for more information. |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized to change time settings |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

***Schema***

*See issues object schema.*

***Example***

```
Request: PATCH /rest/issues/1

{
    "dismiss_status": "DISMISSED"
}

Response: 205 Reset Content
```

# Chapter 14. Globalization Resources

## /locales

Resource to list product supported locales.

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **<none>** | |

#### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain the supported locales |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

```
{
    "type": "array",
    "minItems": 0,
    "uniqueItems": true,
    "items": [
        {
            "type": "object",
            "required": [ "type", "id", "display_name",
 "language", "country", "default" ],
            "additionalProperties": false,
            "properties": {
                "type": { "enum": [ "product-locale" ] },
                "id": "string",
                "display_name": "string",
                "language": "string",
                "country": "string",
                "default": "boolean"
            }
        }
    ]
}
```

#### Example

```
Request: GET /rest/locales

Response: 200 OK
```

```
[
    {
        "id": "en_US",
        "display_name": "english (United States)",
        "language": "en",
        "country": "United States",
        "default": true,
        "type": "product-locale"
    },
    {
        "id": "es_MX",
        "display_name": "español (México)",
        "language": "es",
        "country": "Mexico",
        "default": false,
        "type": "product-locale"
    }
]
```

## /locales/{id}

A specific locale

•

### GET

**Query parameters**

| Parameter | Description |
|---|---|
| **<none>** | |

**Status codes**

| Code | Description |
|---|---|
| **200** | The response body will contain the specific locale |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

**Schema**

*See product-locale schema, product-locale object.*

**Example**

```
Request: GET /rest/locales/en_US

Response: 200 OK

{
    "id": "en_US",
    "display_name": "english (United States)",
    "language": "en",
    "country": "United States",
```

```
    "default": true,
    "type": "product-locale"
}
```

## /strings

Resource to list product strings.

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **locale** | Locale value that indicates the language the strings are returned. If no locale is provided strings are returned as the default language. **Example: locale=en_US** |

#### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain the product strings |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

```
{
    "type": "array",
    "minItems": 0,
    "uniqueItems": false,
    "items": [
        {
            "type": "object",
            "required": [ "id" ],
            "minItems": 0,
            "uniqueItems": true,
            "items": [
               { "type": "string" }
        }
    ]
}
```

#### Example

```
Request: GET /rest/strings

Response: 200 OK

[
    {
        "id": "web",
        "sLoadingRecords": "Loading...",
```

```
        "ButtonLabels_ExportCSV": "Export CSV",
        "Jobs_Status": "Status",
        "PageNames_Processes": "Processes",
        "US_1_ONLY_ONE_SECURITY_APPROVER_category": "User Setup"
    },
    {
        "id": "events",
        "PS00070003": "Backup data creation job {0} has started.",
        "PS00070002": "User {0} has deleted backup file {1}.",
        "PS00070005": "Backup data creation job {0} has failed.",
        "PS00070004": "Backup data creation job {0} has completed successfully."
    }
]
```

## /strings/{id}

A specific set of strings

### GET

#### Query parameters

| Parameter | Description |
|---|---|
| **locale** | Locale value that indicates the language the strings are returned. If no locale is provided strings are returned as the default language. **Example: locale=en_US** |

#### Status codes

| Code | Description |
|---|---|
| **200** | The response body will contain the specific set of strings |
| **401** | The current request is not authenticated. |
| **403** | The current user is not authorized. |
| **406** | The **Accept** header must be set to a supported content type. |
| **500** | An internal error has occurred. See the **Warning** header for more information. |
| **503** | A server service is not available. See the **Warning** header for more information. |

#### Schema

*See strings schema.*

#### Example

```
Request: GET /rest/strings/web

Response: 200 OK

{
    "id": "web",
    "sLoadingRecords": "Loading...",
    "ButtonLabels_ExportCSV": "Export CSV",
    "Jobs_Status": "Status",
    "PageNames_Processes": "Processes",
```

```
    "US_1_ONLY_ONE_SECURITY_APPROVER_category": "User Setup"
}
```

# Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

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

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

*IBM Director of Licensing*
*IBM Corporation*
*North Castle Drive, MD-NC119*
*Armonk, NY 10504-1785*
*US*

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

*Intellectual Property Licensing*
*Legal and Intellectual Property Law*
*IBM Japan Ltd.*
*19-21, Nihonbashi-Hakozakicho, Chuo-ku*
*Tokyo 103-8510, Japan*

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

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

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

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

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

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

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

*IBM Director of Licensing*
*IBM Corporation*
*North Castle Drive, MD-NC119*
*Armonk, NY 10504-1785*
*US*

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

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

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

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

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

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

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

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

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

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

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_.

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

## Trademarks

SPx and Security First Corp are trademarks or registered marks of Security First Corp., registered in many jurisdictions worldwide. Other products and services may be trademarks of Security First Corp. or other companies.

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

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

The Apache Software Foundation (ASF) owns all Apache-related trademarks, service marks, and graphic logos on behalf of our Apache project communities, and the names of all Apache projects are trademarks of the ASF.

Node.JS is a registered trademark of Joyent, Inc. CORPORATION DELAWARE 345 California Street; Suite 2000 San Francisco CALIFORNIA 94104.

Unicode and the Unicode Logo are registered trademarks of Unicode, Inc. in the U.S. and other countries.

The CentOS Marks are trademarks of Red Hat, Inc. (“Red Hat”).

“Red Hat,” Red Hat Linux, the Red Hat “Shadowman” logo, and the products listed are trademarks or registered trademarks of Red Hat, Inc. in the US and other countries.

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

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

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

## Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

**Applicability**

These terms and conditions are in addition to any terms of use for the IBM website.

**Personal use**

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

**Commercial use**

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

**Rights**

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

## Privacy policy considerations

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below. This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM’s Privacy Policy at http://www.ibm.com/privacy and IBM’s Online Privacy Statement at http://www.ibm.com/privacy/details the section entitled “Cookies, Web Beacons and Other Technologies” and the “IBM Software Products and Software-as-a-Service Privacy Statement” at http://www.ibm.com/software/info/product-privacy.

Part Number CC0LUEN


GI13-4922-00