FORUM SYSTEMS SENTRY™ VERSION 9.1 REST API MANAGEMENT

Legal Marks

Forum Sentry™ Version 9.1 REST API Management, published July 2020.

D-ASF-SE-02838

Table of Contents

INTRODUCTION TO THE REST API MANAGEMENT GUIDE

Audience for the REST API Management Guide

The *Forum Systems Sentry™ Version 9.1 REST API Management Guide* is for System Administrators who will choose to create, update, or delete policies using the Forum Sentry REST API.

Create Read Update Delete (CRUD) REST API

The Forum Sentry REST API is designed for the CRUD methodology of REST services. The HTTP methods that are used align with the operations being performed.

Operation	Method
Create	HTTP POST
Copy	HTTP POST
Modify	HTTP PUT
Remove	HTTP DELETE
View	HTTP GET

The REST API is enabled from the System->Configuration->REST API menu option.

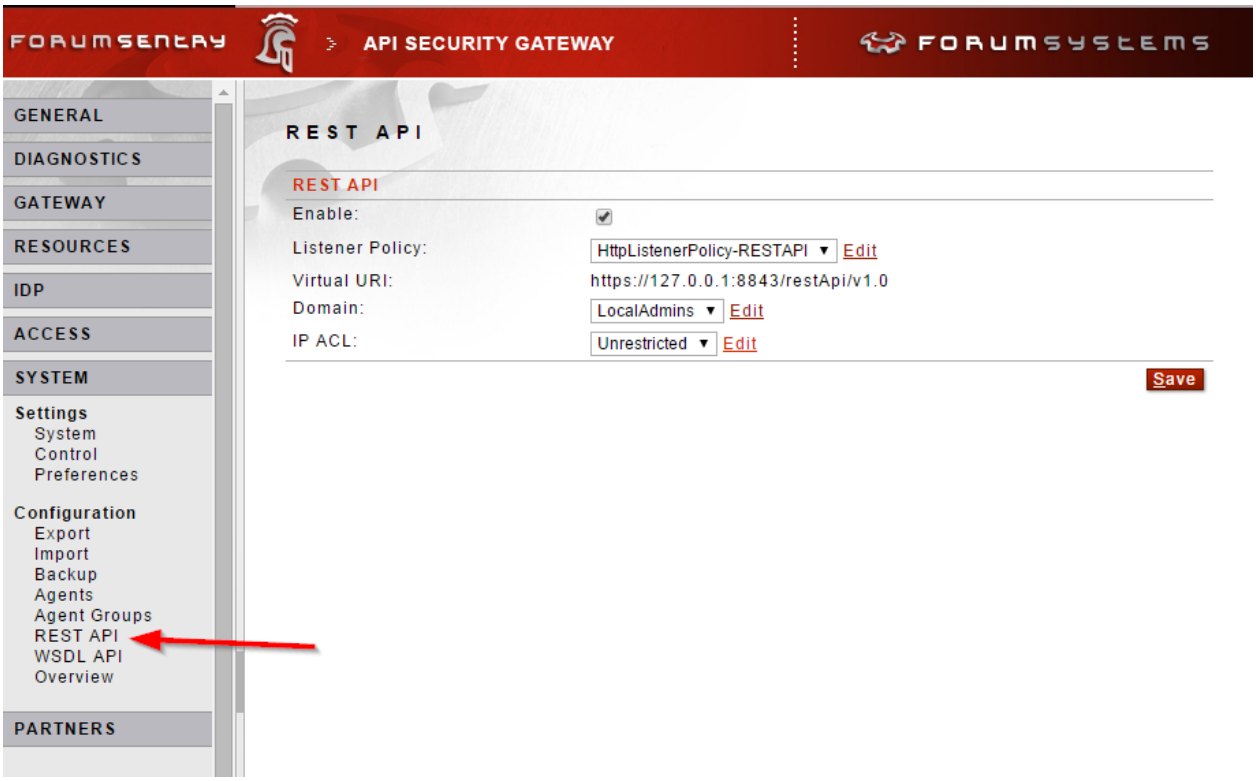

Figure 1: REST API Settings

REST API Policy Options

TERM	DEFINITION
Listener Policy	The HTTPS Listener Policy to be used to handle the inbound requests. This enables security and identity to be set up on the HTTP protocol channel such as 1-way SSL, 2-way SSL.
Virtual URI	This is the API base URI, which is enabled for each version of the REST API (v1.0, etc). The default root is: **https://[server]:[port]/restAPI/v1.0**
Domain	The admin Domain that the REST API admin user(s) belong to. An admin account with privileged access has to reside in this Domain with both Read and Write privileges.
IP ACL	Enables locking down access to the REST API to only the defined set of IP addresses and ranges.

Enabling the Sentry REST API

To enable the Forum Sentry REST API follow the steps outlined below.

1. On the REST API page, click Enable.
2. Select an HTTPS Listener policy already configured on the system. If there are no HTTPS listener policies, create one on the Network Policies page first.
3. Select the Domain that contains a Group that contains an admin account with both Read and Write privileges on the Domain.
4. Optionally specify an IP ACL.

Note: The Default Domain is selected by default. A local admin account with privileged access will not be able to access the REST API by default. The admin account will need to be added to a User Group which will then need Read and Write access to the Default Domain.

REST API Swagger Interface for Help and Examples

The Forum Sentry REST API is self-documenting via a REST Swagger interface. To see the full list of features, and how to invoke the REST API, open a web browser and click the URI to the REST API from the Sentry REST API screen.

URL: http(s)://[server]:[port]**/restApi/v1.0/**

Once you enter a valid Administrator credential, this call returns a web page that contains the various operations and examples on how to invoke them.

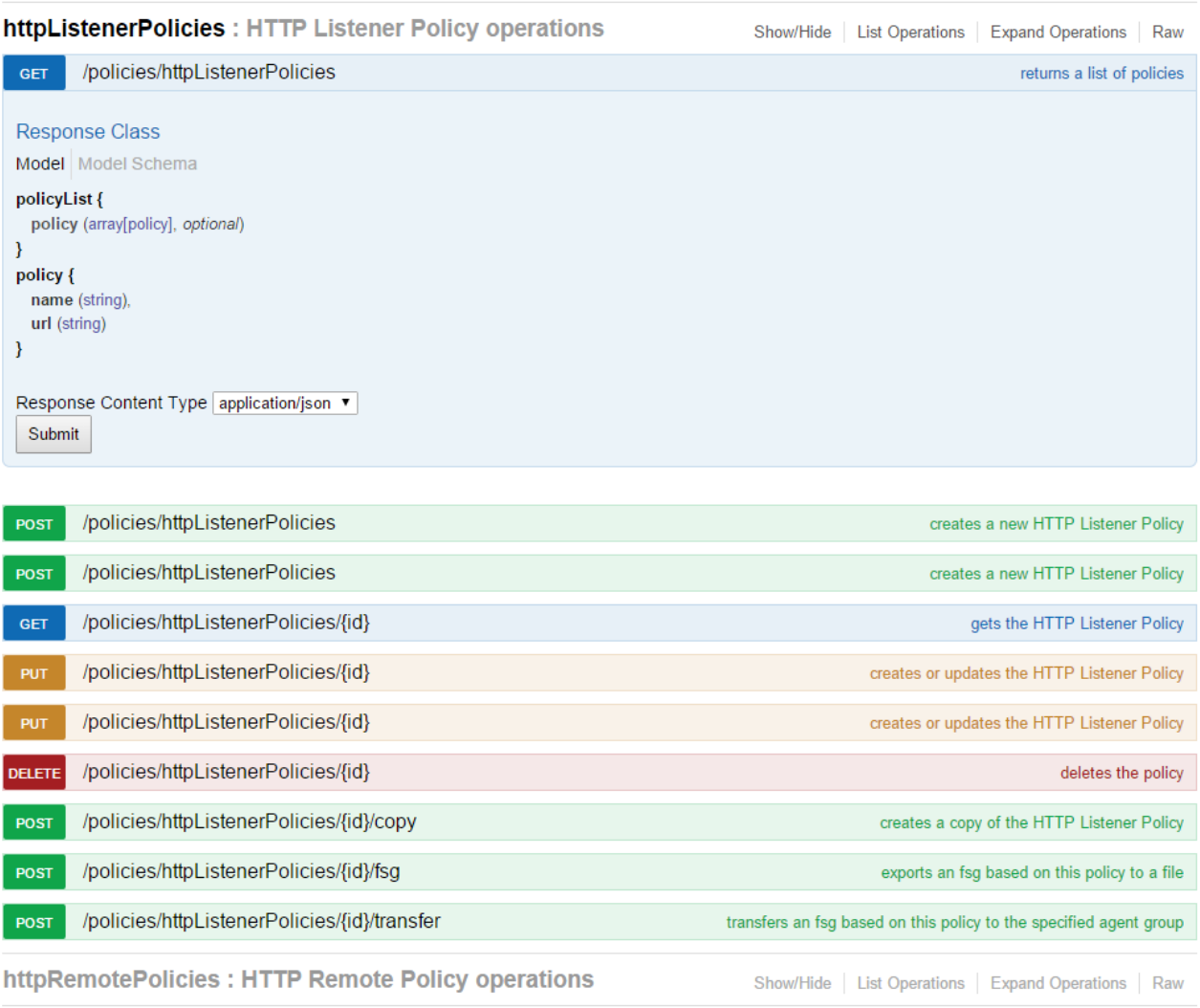

Figure 2: Swagger Example

Retrieve API Documentation

URL: http(s)://[server]:[port]**/restApi/v1.0/api-docs**
Method: HTTP GET

{"apiVersion":"1.0","swaggerVersion":"1.2","apis":[{"path":"/policies/activeMqListenerPolicies","description":"ActiveMQ listener policy

operations"},{"path":"/policies/activeMqRemotePolicies","description":"ActiveMQ remote policy operations"},{"path":"/policies/agentGroups","description":"agent group operations"},{"path":"/policies/agents","description":"agent operations"},
...]
}

REST API cURL Examples

cURL examples are provided in each of the APIs listed below. The cURL utility is widely available and is used in these examples for guidance only.

Note that the '--insecure' flag used in these examples is not recommended in production. This flag disables server certificate checking and should be enabled to prevent man-in-the middle attacks.

Helpful Tip

Using the browser developer tools to trace the REST API invocations made via the Swagger interface will often provide a cURL representation of the request.

The following is an example from Google Chrome. Right click on the transaction and select Copy as cURL.

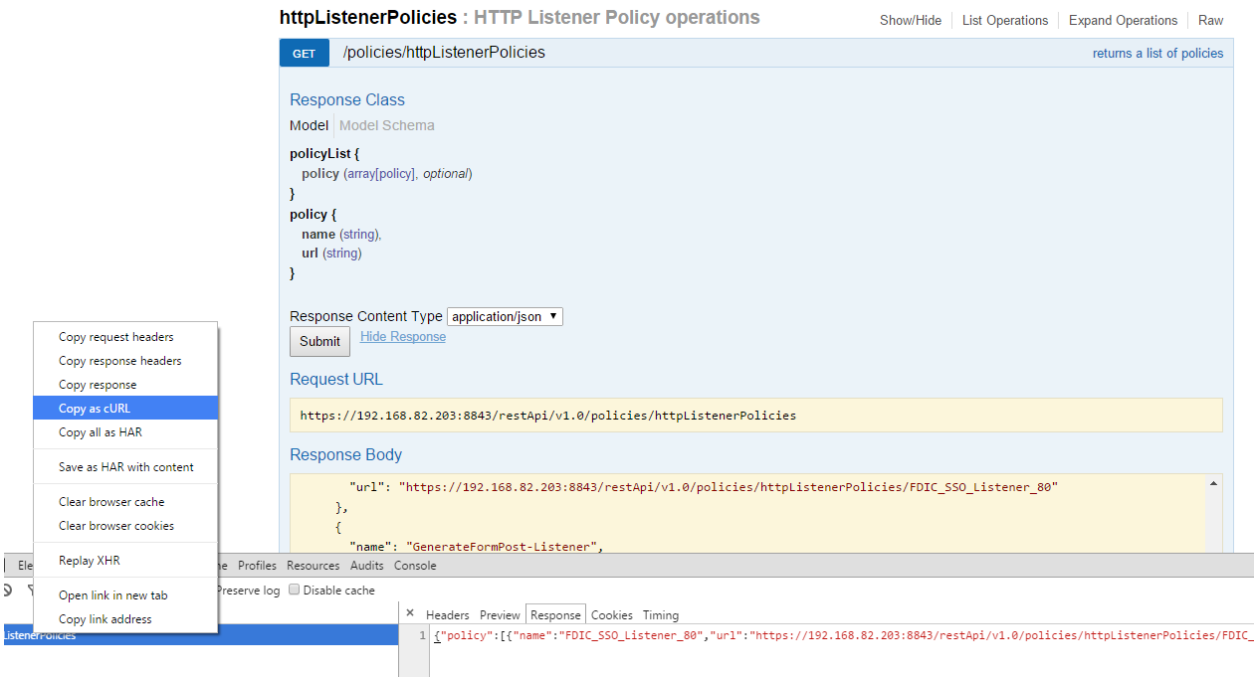

Figure 3: cURL Example

REST API Methods

Sample cURL Scripts

You should always refer to the Swagger interface for up-to-date instruction and self-documenting aspects of each REST API command. The examples below are meant only as a reference.

Sample cURL Script command:

```
cURL -# -D headers.out -o response.out --insecure -u "$USER:$PASSWORD"
"https://$SERVER/restApi/v1.0/api-docs"
```

Controlling Response Type

The response content-type of most commands can be switched between JSON and XML using the accept header.

Sample cURL Script command to list HTTP Listener policies as XML:

```
curl -# -D headers.out -o response.out --insecure -u "$USER:$PASSWORD" -X GET \
    -H "Accept: application/xml" \
   "https://$SERVER/restApi/v1.0/policies/httpListenerPolicies/"
```

Sample cURL Script command to list HTTP Listener policies as JSON:

```
curl -# -D headers.out -o response.out --insecure -u "$USER:$PASSWORD" -X GET \
    -H "Accept: application/json" \
   "https://$SERVER/restApi/v1.0/policies/httpListenerPolicies/"
```

Create a Policy:

URL: http(s)://[server]:[port]**/restApi/v1.0/policies/[Policy Type]**
Query Parameters:
    See Swagger interface for specific details.
Form Parameters:
    See Swagger interface for specific details.
Method: HTTP POST

You can use the Swagger interface or review the api-docs response to obtain the REST API URI to use for the specific policy, and then you can create the policy via the HTTP POST method.

Sample cURL Script command to create an HTTP Remote Policy:

```
curl -# -D headers.out -o response.out --insecure -u "$USER:$PASSWORD" \
   -d "name=ApiApplication_1" \
   -d "remoteServer=10.20.30.40" \
   -d "remotePort=80" \
   -d "useChunking=true" \
```

```
-d "enableSSL=false" \
-d "tcpConnectionTimeout=60000" \
-d "tcpReadTimeout=60000" \
-d "useBasicAuth=false" \
-d "enabled=true" \
-d "processResponse=false" \
"https://$SERVER/restApi/v1.0/policies/httpRemotePolicies/"
```

Modify a Policy:

URL: http(s)://[server]:[port]**/restApi/v1.0/policies/[Policy Type]/[Policy Name]**
Query Parameters:
See Swagger interface for specific details.
Form Parameters:
See Swagger interface for specific details.

Method: HTTP PUT

The PUT method is used for policy modification

Sample cURL Script command to modify an XMLContentPolicy:

```
curl -# -D headers.out -o response.out --insecure -u "$USER:$PASSWORD" -X PUT \
 -d "name=testXmlPolicyMyNew" \
 -d "description=abc" \
 -d "idpGroup=TestIdpGroup2" \
 -d "requestTaskGroup=testRequestTaskGroup" \
 -d "responseTaskGroup=testResponseTaskGroup" \
 "https://$SERVER/restApi/v1.0/policies/xmlPolicies/testXmlPolicyMyNew"
```

Export an FSG Policy

URL: http(s)://[server]:[port]**/restApi/v1.0/policies/[Policy Type]/[Policy Name]/fsg**
Method: HTTP POST

You can use the Swagger interface or review the api-docs response to obtain the REST API URI to use for the specific policy, and then you can export an FSG of the policy via the HTTP POST method. The POST is required for these methods so that you can securely pass the password parameter into the method via URL encoding or multipart Form Post. A password is required to encrypt the FSG response.

Sample CURL Script command:

```
curl -# -D headers.out -o response.out --insecure -u "$USER:$PASSWORD" -d
"password=password1" "https://$SERVER/restApi/v1.0/policies/xmlPolicies/XML-BASE-1/fsg"; cp
response.out test.fsg
```

Import an FSG Policy

URL: http(s)://[server]:[port]**/restApi/v1.0/configuration?format=fsg**
Query Parameters:
format = The format of the configuration file. Should be set to fsg.

Form Parameters:
password = The password to be used for the fsg.
Method: HTTP POST

The import of an FSG policy uses the method PUT since it is considered a policy modification. The configuration URI is used to consume the inbound FSG. The POST is required for these methods so that you can securely pass the password parameter into the method via URL encoding or multipart Form Post. A password is required to decrypt the uploaded FSG.

Sample cURL Script command:
curl -# -D headers.out -o response.out --insecure -u "$USER:$PASSWORD" -F "password=password1" -F "file=@test.fsg" "https://$SERVER/restApi/v1.0/configuration?format=fsg"

Export an FSX (Full System Policy)

URL: http(s)://[server]:[port]**/restApi/v1.0/configuration/fsx**
Query Parameters:
agent = The agent filter to use when generating the fsx file (optional).
Form Parameters:
password = The password to be used for the fsx.
Method: HTTP POST

The POST is required for this method so that you can securely pass the password parameter into the method via the http form body. A password is required to generate the encrypted FSX output file.

Sample cURL Script command:
curl -# -D headers.out -o response.out --insecure -u "$USER:$PASSWORD" -d "password=password1" "https://$SERVER/restApi/v1.0/configuration/fsx"; cp response.out test.fsx

Import an FSX (Full System Policy)

URL: http(s)://[server]:[port]**/restApi/v1.0/configuration**
Query Parameters:
format = The format of the configuration file. Should be set to fsx
Form Parameters:
password = The password to be used for the fsx.
Method: HTTP POST

The POST is required for this method so that you can securely pass the password parameter into the method via the http form body. A password is required to decrypt the encrypted FSX input file.

Sample cURL Script command:
curl -# -D headers.out -o response.out --insecure -u "$USER:$PASSWORD" -F "password=password1" -F "file=@test.fsx" "https://$SERVER/restApi/v1.0/configuration?format=fsx"