FORUM SYSTEMS SENTRY™ VERSION 8.3
REST API MANAGEMENT

Forum Systems Sentry™ Version 8.3 REST API Management, published October, 2014.

D-ASF-SE-02633

Table of Contents

INTRODUCTION TO THE REST API MANAGEMENT GUIDE

Audience for the REST API Management Guide

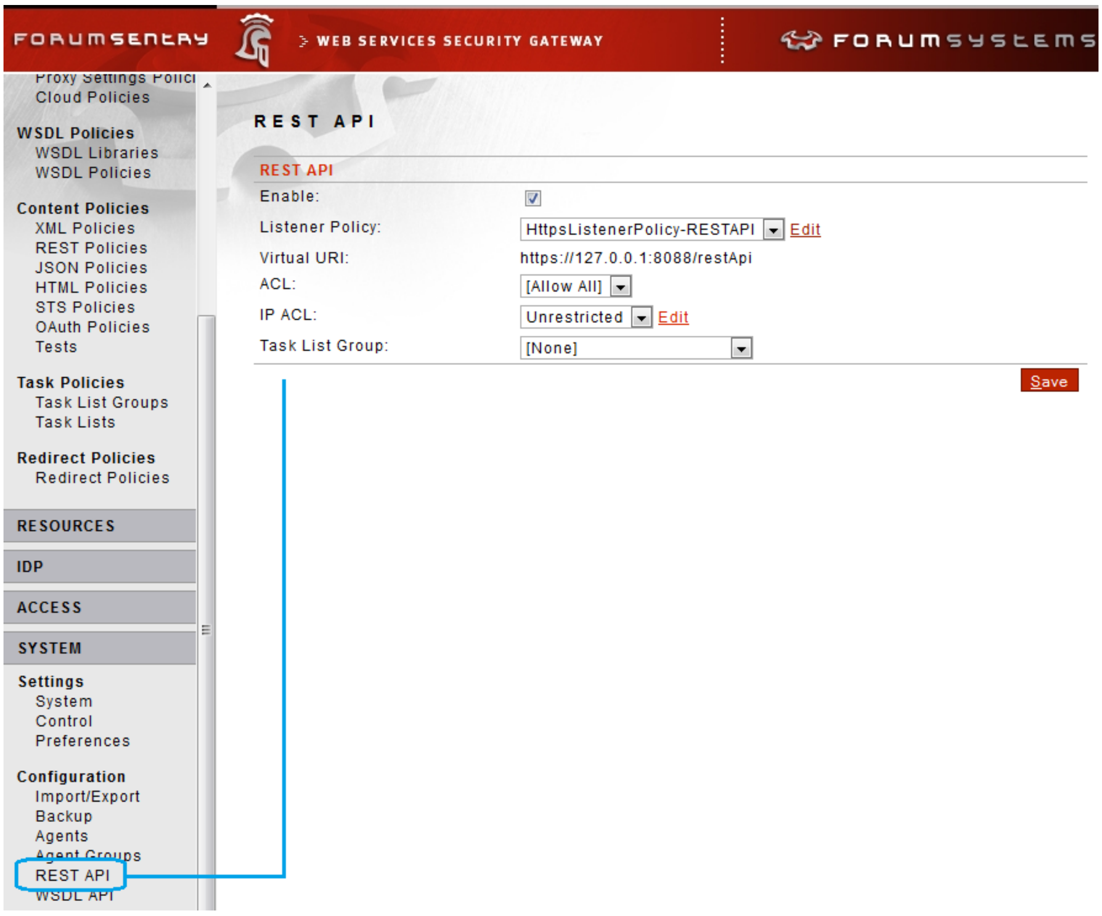
The *Forum Systems Sentry™ Version 8.3 REST API Management Guide* is for System Administrators who will choose to create, update, or delete policies using the CRUD REST API.

CRUD REST API

The Forum Sentry REST API is designed with the CRUD (Create, Read, Update, Delete) design methodology of REST services. Thus, 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.



REST API

REST API

Enable:	✓
Listener Policy:	HttpsListenerPolicy-RESTAPI Edit
Virtual URI:	https://127.0.0.1:8088/restApi
ACL:	[Allow All]
IP ACL:	Unrestricted Edit
Task List Group:	[None]

REST API Policy Options

Listener Policy:
The HTTP 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:

http(s)://[server]:[port]/restAPI/v1.0

ACL:
Enables extending the access control policy for authentication to the REST API beyond the standard required Web Administration user credentials that are required by default

IP ACL:
Enables locking down access to the REST API to only the defined set of IP addresses and ranges.

Task List Group:
Provides fully extensible processing on the inbound calls to the REST API to enable multi-factor authentication or other role based rules that can define the API governance.

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.

REST API Methods

Get Help on the REST API format via the Swagger interface:

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 enter the URI to the REST API as configured 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 which will allow you to see the various operations and examples on how to invoke them.

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","des
cription":"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"},
...]
}
```

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 only.

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, 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, 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"
```