About the Secure Firewall Threat Defense REST API

You can use the Secure Firewall Threat Defense REpresentational State Transfer (REST) Application Programming Interface (API), over HTTPS, to interact with the threat defense device through a client program. The REST API uses JavaScript Object Notation (JSON) format to represent objects.

Secure Firewall device manager includes an API Explorer that explains all of the resources and JSON objects available for your programmatic use. The Explorer provides detailed information about the attribute-value pairs in each object, and you can experiment with the various HTTP methods to ensure you understand the coding required to use each resource. The API Explorer also provides examples of the URLs required for each resource.

You can also find reference information and examples online at https://developer.cisco.com/site/ftd-api-reference/.

The API is has its own version number. There is no guarantee that a client designed for one version of the API will work for a future version without error or without requiring changes to your program.

Audience for This Programming Guide

This guide is written on the assumption that you have a general knowledge of programming and a specific understanding of REST APIs and JSON. If you are new to these technologies, please first read a general guide on REST APIs.

Supported HTTP Methods

You can use the following HTTP methods only. The other methods are not supported.

- GET—To read data from the system.

- POST—To create new objects.
- PUT—To modify existing objects. When using PUT, you must include the entire JSON object. You cannot selectively update individual attributes within an object.
- DELETE—To remove a user-defined object.

The Base URL for the API

The easiest way to determine the base URL for a given threat defense device is to try out a GET method in the API Explorer, and simply delete the object part of the URL from the result.

For example, you can do a GET /object/networks, and see something similar to the following in the returned output under Request URL:

```
https://ftd.example.com/api/fdm/v1/object/networks
```

The server name part of the URL is the hostname or IP address of the threat defense device, and will be different for your device in place of “ftd.example.com.” In this example, you delete /object/networks from the path to get the base URL:

```
https://ftd.example.com/api/fdm/v1/
```

All resource calls use this URL as the base for the request URL.

If you changed the HTTPS data port, you must include the custom port in the URL. For example, if you changed the port to 4443: https://ftd.example.com:4443/api/fdm/v1/

The "v" element in the URL is the API version, and this will typically change with the software version. For example, the API version for the threat defense version 6.3.0 is v2, so the base URL would be:

```
https://ftd.example.com/api/fdm/v2/
```

Note Starting with the threat defense 6.4, you can avoid the need to update the path in your API calls by using **latest** instead of the v element in the path. For example, https://ftd.example.com/api/fdm/latest/. The **latest** alias resolves to the most recent API version supported by the device.

In the API Explorer, if you scroll to the bottom of the page, you can see information on the base URL (without the server name) and API version.

Securing SSL/TLS Communications for the REST API

The Threat Defense devices come with a self-signed certificate so that you can initiate HTTPS communications with the device. However, because the certificate is not signed by a known Certificate Authority (CA), any SSL/TLS access attempt will consider the connection insecure.

When connecting with a browser, you are prompted to accept the self-signed certificate, but a command such as curl will reject the certificate. In the case of curl, you can bypass the certificate check failure by adding the **--insecure** keyword. For example:

```
curl --insecure -X GET --header 'Accept: application/json'
'https://ftd.example.com/api/versions'
```

One of the first things you should do is obtain a CA-signed device certificate for the threat defense device. Then, using the device manager or the API, assign this certificate as the management certificate. Subsequently, SSL/TLS certificate checking should not fail, and you will not need to use insecure communications in your API calls.

Procedure

Step 1 Upload the CA-signed device certificate using the **POST /object/internalcertificates** resource.

Step 2 Make this certificate the management certificate using the **PUT /devicesettings/default/webuicertificates/{objId}** resource.

Use the **GET /devicesettings/default/webuicertificates** resource to determine the object ID for the web UI certificate.

Step 3 Deploy the changes using the **POST /operational/deploy** resource.

Determining the Supported API Versions

You can determine which API versions are supported on a device using the GET /api/versions (ApiVersions) method. This method does not require authentication, and it also does not include a version element in the path. For example:

```
curl -X GET --header 'Accept: application/json' 'https://ftd.example.com/api/versions'
```

Replace “ftd.example.com” with the hostname or IP address of the threat defense device.

This method returns a list of API versions you can use. For example:

```
{
    "supportedVersions":["v3", "latest"]
}
```

The version strings are the same ones you use in the URL for subsequent API calls. If you use **latest** instead of the specific version identifier, you can avoid the need to update your calls for subsequent releases. However, using this technique does not overcome changes to the object models used in your calls, which might need adjustment from release to release.

Typically, your next step would be to get an access token, as described in Authenticating Your REST API Client Using OAuth.

API Version Backward Compatibility

The threat defense API version changes with each major release of the threat defense software. New features impact the API calls for the features being added or changed.

However, many features do not change from release to release. For example, the APIs related to network and port objects often remain unchanged in a new release.

Starting with the threat defense version 6.7, if an API resource model for a feature does not change between releases, then the threat defense API can accept calls that are based on the older API version. Even if the feature model did change, if there is a logical way to convert the old model to the new model, the older call can work. For example, a v5 call can be accepted on a v6 system. If you use “latest” as the version number in your calls, these “older” calls are interpreted as a v6 call in this scenario, so whether you are taking advantage of backward compatibility depends on how you are structuring your API calls.

If a feature model has changed between API versions in a way that backward compatibility cannot be supported, you will get an error message, and you will need to check for these errors and update your code for these specific calls.