Configuring External Users for the API

Version Requirement: To use external AAA, you must be running the threat defense version 6.3(0) or higher, and the threat defense REST API v2 or higher.

You can configure the device to use an external RADIUS AAA server to authenticate and authorize user access to the threat defense REST API. You can use RADIUS user accounts instead of, or in addition to, the built-in local **admin** user account.

When using external AAA, you can define accounts to have different authorization levels. This lets you limit who can make configuration changes to the device while still providing read-only access to support staff.

The following procedure explains the end-to-end process of setting up the RADIUS accounts and then configuring the device to use external AAA for authentication and authorization.

Before you begin

Keep the following operational factors in mind when using external authorization.

- If the device is configured for high availability, configure external authorization on the active unit. You must then run a deployment job for the authorization settings to allow user access to the standby device.
- Each time a new user accesses the system, a User resource is created for that user. You need to deploy the configuration to save that user object.

  (Versions prior to the threat defense 6.6.) If you are operating in high availability (HA) mode, you must deploy the configuration before that user can log into the standby unit. Because only admin or read-write users can start a deployment job, a first-time read-only user must have someone else deploy the configuration to save the User object.

  Starting with the threat defense 6.6, the HA restrictions are removed. An external user can log into the standby unit without first logging into the active unit and deploying the configuration. The user object will not be created on the standby unit, but user characteristics will be cached and the user will be given access, assuming a valid username/password is provided.

Procedure

Step 1 Define Authorization Rights in the RADIUS User Accounts, on page 2.

Step 2 Define the RADIUS Servers, on page 2.

Step 3 Create a AAA Server Group for the RADIUS Servers, on page 4.

Step 4 Establish the AAA Server Group as an Authentication Source for HTTPS Access, on page 6.

Step 5 Use **POST /operational/deploy** to start a deployment job.

The **curl** command would be similar to the following:

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json'

'https://ftd.example.com/api/fdm/latest/operational/deploy'
```

For detailed information on deploying changes, see Deploying Configuration Changes.

Step 6 Verify External User Access, on page 9.

- Define Authorization Rights in the RADIUS User Accounts, on page 2
- Define the RADIUS Servers, on page 2
- Create a AAA Server Group for the RADIUS Servers, on page 4
- Establish the AAA Server Group as an Authentication Source for HTTPS Access, on page 6
- Verify External User Access, on page 9

Define Authorization Rights in the RADIUS User Accounts

You can provide access to the threat defense REST API from an external RADIUS server. By enabling RADIUS authentication and authorization, you can provide different levels of access rights, and not have every user log in through the local **admin** account.

Note These external users are also authorized for device manager.

To provide role-based access control (RBAC), update the user accounts on your RADIUS server to define the **cisco-av-pair** attribute. This attribute must be defined correctly on a user account, or the user is denied access to the REST API. Following are the supported values for the cisco-av-pair attribute:

- **fdm.userrole.authority.admin** provides full Administrator access. These users can do all actions that the local **admin** user can do.
- **fdm.userrole.authority.rw** provides read-write access. These users can do everything a read-only user can do, and also edit and deploy the configuration. The only restrictions are for system-critical actions, which include installing upgrades, creating and restoring backups, viewing the audit log, and logging off other users.
- **fdm.userrole.authority.ro** provides read-only access. These users can view dashboards and the configuration, but cannot make any changes. If the user tries to make a change, the error message explains that this is due to lack of permission.

Define the RADIUS Servers

After you configure the user accounts in the RADIUS server to define the appropriate authorization rights, you can configure the device to use the server to authenticate and authorize access to the REST API.

Use the **POST /object/radiusidentitysources** resource to create an object for each RADIUS server you want to define.

Procedure

Step 1 Create the JSON object body for the RADIUS server.

Following is an example of the JSON object to use with this call.

```
{
  "name": "aaa-server-1",
  "description": "RADIUS server for API access.",
  "host": "172.16.246.220",
  "timeout": 10,
  "serverAuthenticationPort": 1812,
  "serverSecretKey": "secret123",
  "type": "radiusidentitysource"
}
```

The attributes are:

- **name**—The object name. This does not need to match anything defined on the RADIUS server.
- **description**—(Optional.) A description of the object.
- **host**—The IP address or fully-qualified host name of the RADIUS server.
- **timeout**—(Optional.) The length of time, 1-300 seconds, that the system waits for a response from the server before sending the request to the next server. If you do not include this attribute, the default is 10 seconds.
- **serverAuthenticationPort**—(Optional.) The port on which RADIUS authentication and authorization are performed. If you do not include this attribute, the default is 1812.
- **serverSecretKey**—(Optional.) The shared secret that is used to encrypt data between the threat defense device and the RADIUS server. The key is a case-sensitive, alphanumeric string of up to 64 characters, with no spaces. The key must start with an alphanumeric character or an underscore, and it can contain the special characters: $ & - _ . + @. The string must match the one configured on the RADIUS server. If you do not configure a secret key, the connection is not encrypted.

Step 2 Post the object.

For example, the **curl** command would look like the following:

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json'
 -d '{
    "name": "aaa-server-1",
   "description": "RADIUS server for API access.",
   "host": "172.16.246.220",
   "timeout": 10,
   "serverAuthenticationPort": 1812,
   "serverSecretKey": "secret123",
   "type": "radiusidentitysource"
 }' 'https://ftd.example.com/api/fdm/latest/object/radiusidentitysources'
```

Step 3 Verify the response.

You should get a response code of 200. A successful response body would look something like the following. Note that sensitive information, such as the secret key, is masked in the response.

```
{
  "version": "nfamb3cr2jlyi",
  "name": "aaa-server-1",
  "description": "RADIUS server for API access.",
  "host": "172.16.246.220",
  "timeout": 10,
  "serverAuthenticationPort": 1812,
  "serverSecretKey": "*********",
  "capabilities": [
    "AUTHENTICATION",
    "AUTHORIZATION"
  ],
  "id": "1b962e3b-6e56-11e8-bd65-379fa8aaaba1",
  "type": "radiusidentitysource",
  "links": {
    "self": "https://ftd.example.com/api/fdm/latest/object/
radiusidentitysources/1b962e3b-6e56-11e8-bd65-379fa8aaaba1"
  }
}
```

Create a AAA Server Group for the RADIUS Servers

After you create the RADIUS server objects, use the **POST /object/radiusidentitysourcegroups** resource to create a AAA group to contain the radiusidentitysource objects.

You can add up to 16 RADIUS servers to a RADIUS AAA server group. These servers must be backups of each other: that is, they need to have the same list of user accounts.

Procedure

Step 1 Create the JSON object body for the RADIUS server group.

Following is an example of the JSON object to use with this call.

```
{
  "name": "radius-group",
  "maxFailedAttempts": 3,
  "deadTime": 10,
  "description": "AAA RADIUS server group.",
  "radiusIdentitySources": [
    {
      "id": "1b962e3b-6e56-11e8-bd65-379fa8aaaba1",
      "type": "radiusidentitysource",
      "version": "nfamb3cr2jlyi",
      "name": "aaa-server-1"
    }
  ],
  "type": "radiusidentitysourcegroup"
}
```

The attributes are:

- **name**—The object name. This does not need to match anything defined on the member RADIUS servers.
- **maxFailedAttempts**—(Optional.) Failed servers are reactivated only after all servers have failed. The dead time is how long to wait, from 0 - 1440 minutes, after the last server fails before reactivating all servers. If you do not include this attribute, the default is 10 minutes.
- **deadTime**—(Optional.) The number of failed requests (that is, requests that do not get a response) sent to a RADIUS server in the group before trying the next server. You can specify 1-5, and the default is 3. When the maximum number of failed attempts is exceeded, the system marks the server as Failed.

  For a given feature, if you configured a fallback method using the local database, and all the servers in the group fail to respond, then the group is considered to be unresponsive, and the fallback method is tried. The server group remains marked as unresponsive for the duration of the dead time, so that additional AAA requests within that period do not attempt to contact the server group, and the fallback method is used immediately.
- **description**—(Optional.) A description of the object.
- **radiusIdentitySources**—This is a group of items that define each radiusidentitysource object that defines a RADIUS server to include in the group. Include the items within the [brackets]. Following are the attributes and syntax for each object. You get the value for the **id**, **version**, and **name** attributes from the individual objects; the information is in the response body when you create the objects. You can also get the information from a **GET /object/radiusidentitysources** call. The **type** must be **radiusidentitysource**.

```
{
  "id": "1b962e3b-6e56-11e8-bd65-379fa8aaaba1",
  "type": "radiusidentitysource",
  "version": "nfamb3cr2jlyi",
  "name": "aaa-server-1"
}
```

Step 2 Post the object.

For example, the **curl** command would look like the following:

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json'
 -d '{
   "name": "radius-group",
   "maxFailedAttempts": 3,
   "deadTime": 10,
   "description": "AAA RADIUS server group.",
   "radiusIdentitySources": [
     {
       "id": "1b962e3b-6e56-11e8-bd65-379fa8aaaba1",
       "type": "radiusidentitysource",
       "version": "nfamb3cr2jlyi",
       "name": "aaa-server-1"
     }
   ],
   "type": "radiusidentitysourcegroup"
 }' 'https://ftd.example.com/api/fdm/latest/object/radiusidentitysourcegroups'
```

Step 3 Verify the response.

You should get a response code of 200. A successful response body would look something like the following.

```
{
```

```
  "version": "7r572novdiyy",
  "name": "radius-group",
  "maxFailedAttempts": 3,
  "deadTime": 10,
  "description": "AAA RADIUS server group.",
  "radiusIdentitySources": [
    {
      "version": "nfamb3cr2jlyi",
      "name": "aaa-server-1",
      "id": "1b962e3b-6e56-11e8-bd65-379fa8aaaba1",
      "type": "radiusidentitysource"
    }
  ],
  "activeDirectoryRealm": null,
  "id": "0a7996ae-6e5b-11e8-bd65-dbab801c44b9",
  "type": "radiusidentitysourcegroup",
  "links": {
    "self": "https://ftd.example.com/api/fdm/latest/object/
radiusidentitysourcegroups/0a7996ae-6e5b-11e8-bd65-dbab801c44b9"
  }
}
```


Establish the AAA Server Group as an Authentication Source for HTTPS Access

Use the **PUT /devicesettings/default/aaasettings/{objId}** resource to identify the RADIUS AAA server group as an identity source for user authorization.

There is no POST method: the objects needed for system authentication already exist. You must first do a GET to determine the relevant ID and version values.

Procedure

Step 1 Use **GET /devicesettings/default/aaasettings** to determine the attributes of the aaasettings objects.

The **curl** command would be similar to the following:

```
curl -X GET --header 'Accept: application/json'
'https://ftd.example.com/api/fdm/latest/devicesettings/default/aaasettings'
```

For example, the response body should be similar to the following. This example shows that the local identity source is the one defined for HTTPS access. It is also used for SSH access, which is not relevant for the REST API.


```
{
  "items": [
    {
      "version": "du52clrtmawlt",
      "name": "HTTPS",
      "identitySourceGroup": {
        "version": "cynutari5ffkl",
        "name": "LocalIdentitySource",
        "id": "e3e74c32-3c03-11e8-983b-95c21a1b6da9",
```

```
        "type": "localidentitysource"
      },
      "description": null,
      "protocolType": "HTTPS",
      "useLocal": "NOT_APPLICABLE",
      "id": "00000003-0000-0000-0000-000000000007",
      "type": "aaasetting",
      "links": {
        "self": "https://ftd.example.com/api/fdm/latest/
devicesettings/default/aaasettings/00000003-0000-0000-0000-000000000007"
      }
    },
    {
      "version": "fgkhvu4kwucgv",
      "name": "SSH",
      "identitySourceGroup": {
        "version": "cynutari5ffkl",
        "name": "LocalIdentitySource",
        "id": "e3e74c32-3c03-11e8-983b-95c21a1b6da9",
        "type": "localidentitysource"
      },
      "description": null,
      "protocolType": "SSH",
      "useLocal": "NOT_APPLICABLE",
      "id": "00000003-0000-0000-0000-000000000008",
      "type": "aaasetting",
      "links": {
        "self": "https://ftd.example.com/api/fdm/latest/
devicesettings/default/aaasettings/00000003-0000-0000-0000-000000000008"
      }
    }
  ],
  "paging": {
    "prev": [],
    "next": [],
    "limit": 10,
    "offset": 0,
    "count": 2,
    "pages": 0
  }
}
```


Step 2 (Optional.) Use **GET /devicesettings/default/aaasettings/{objId}** to obtain a copy of the HTTPS AAA setting object to narrow your view.

Your PUT call will update the HTTPS object only. You do not need to update the SSH object.

In this example, the ID of the HTTPS object is 00000003-0000-0000-0000-000000000007, so a **curl** command would be similar to the following:

```
curl -X GET --header 'Accept: application/json'
'https://ftd.example.com/api/fdm/latest/devicesettings/
default/aaasettings/00000003-0000-0000-0000-000000000007'
```

The response body would be similar to the following.


```
{
  "version": "ha4653ootep7z",
  "name": "HTTPS",
  "identitySourceGroup": {
    "version": "cynutari5ffkl",
    "name": "LocalIdentitySource",
    "id": "e3e74c32-3c03-11e8-983b-95c21a1b6da9",
```

```
    "type": "localidentitysource"
  },
  "description": null,
  "protocolType": "HTTPS",
  "useLocal": "NOT_APPLICABLE",
  "id": "00000003-0000-0000-0000-000000000007",
  "type": "aaasetting",
  "links": {
    "self": "https://ftd.example.com/api/fdm/latest/
devicesettings/default/aaasettings/00000003-0000-0000-0000-000000000007"
  }
```

Step 3 Create the JSON object body for AAA management access.

Following is an example of the JSON object to use with this call.

```
{
  "version": "ha4653ootep7z",
  "name": "HTTPS",
  "identitySourceGroup": {
    "id": "0a7996ae-6e5b-11e8-bd65-dbab801c44b9",
    "type": "radiusidentitysourcegroup",
    "version": "7r572novdiyy",
    "name": "radius-group"
  },
  "description": null,
  "protocolType": "HTTPS",
  "useLocal": "BEFORE",
  "id": "00000003-0000-0000-0000-000000000007",
  "type": "aaasetting"
}
```

The attributes are:

- **version**—The version of the HTTPS object. Find this value in the response body for the GET call.
- **name**—The object name, **HTTPS**. Find this value in the response body for the GET call.
- **identitySourceGroup**—This identifies the RADIUS server group. Obtain the **id**, **version**, and **name** values from the response body when you created the server group (or a **GET /object/radiusidentitysourcegroups** call). The type must be **radiusidentitysourcegroup**.
- **description**—(Optional.) A description of the object.
- **protocolType**—The protocol to which this source applies, **HTTPS**.
- **useLocal**—How to use the local identity source, which contains the local admin user account. Enter one of the following options:
  - **Before**—The system checks the username and password against the local source first.
  - **After**—The local source is checked only if the external source is unavailable or if the user account was not found in the external source.
  - **Never**—(Not recommended.) The local source is never used, so you cannot log in as the **admin** user.

    Caution
    If you select **Never**, you will not be able to log into the device manager or use the API using the **admin** account. You will be locked out of the system if the RADIUS server becomes unavailable, or if you miss-configure the accounts in the RADIUS server.

- **id**—The ID value for the HTTPS object. Find this value in the response body for the GET call.
- **type**—The object type, **aaasetting**.

Step 4 Put the object.

For example, the **curl** command would look like the following. Note that the {objId} in the URL and the id for the aaasettings object in the JSON object are the same.

```
curl -X PUT --header 'Content-Type: application/json' --header 'Accept: application/json'
-d '{
   "version": "ha4653ootep7z",
   "name": "HTTPS",
   "identitySourceGroup": {
     "id": "0a7996ae-6e5b-11e8-bd65-dbab801c44b9",
     "type": "radiusidentitysourcegroup",
     "version": "7r572novdiyy",
     "name": "radius-group"
   },
   "description": null,
   "protocolType": "HTTPS",
   "useLocal": "BEFORE",
   "id": "00000003-0000-0000-0000-000000000007",
   "type": "aaasetting"
 }' 'https://ftd.example.com/api/fdm/latest/devicesettings/
default/aaasettings/00000003-0000-0000-0000-000000000007'
```

Step 5 Verify the response.

You should get a response code of 200. A successful response body would look something like the following.

```
{
  "version": "ehxcytq4iccb3",
  "name": "HTTPS",
  "identitySourceGroup": {
    "version": "7r572novdiyy",
    "name": "radius-group",
    "id": "0a7996ae-6e5b-11e8-bd65-dbab801c44b9",
    "type": "radiusidentitysourcegroup"
  },
  "description": null,
  "protocolType": "HTTPS",
  "useLocal": "BEFORE",
  "id": "00000003-0000-0000-0000-000000000007",
  "type": "aaasetting",
  "links": {
    "self": "https://ftd.example.com/api/fdm/latest/devicesettings/
default/aaasettings/00000003-0000-0000-0000-000000000007"
  }
}
```

Verify External User Access

After the deployment job completes, you can test external user access to both the device manager and the REST API.

Procedure

Step 1 Log into the device manager using an external username that has a valid cisco-av-pair attribute.

The login should be successful, and the upper right of the page should show the username and privilege level.

Step 2 Obtain a REST API token for an external user.

If the user can obtain a token, the user can use the resources and methods allowed for the assigned privilege level.

a) Create the JSON object body for a simple password-granted token.

```
{
  "grant_type": "password",
  "username": "radiusreadwriteuser1",
  "password": "Readwrite123!"
}
```

b) Use **POST /fdm/token** to obtain the token.

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json'
 -d '{
   "grant_type": "password",
   "username": "radiusreadwriteuser1",
   "password": "Readwrite123!"
 }' 'https://ftd.example.com/api/fdm/latest/fdm/token'
```

c) Evaluate the response to verify that a token was granted.

You should get a response code of 200. Getting a token means that the system was able to authenticate the user.

```
{
  "access_token": "eyJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE1Mjg4MjM
3MTAsInN1YiI6InJhZGl1c3JlYWR3cml0ZXVzZXIxIiwianRpIjoiMjk5Zj
Q5YjYtNmU2NC0xMWU4LWJkNjUtNmY0ZmVmYjY1MzI5IiwibmJmIjoxNTI4O
DIzNzEwLCJleHAiOjE1Mjg4MjU1MTAsInJlZnJlc2hUb2tlbkV4cGlyZXNB
dCI6MTUyODgyNjExMDg4OSwidG9rZW5UeXBlIjoiSldUX0FjY2VzcyIsInV
zZXJVdWlkIjoiMjliMjBlNjctNmU2NC0xMWU4LWJkNjUtMzU4MmUwZjU5Yj
Q4IiwidXNlclJvbGUiOiJST0xFX1JFQURfV1JVEUiLCJvcmlnaW4iOiJwY
XNzd29yZCJ9.dtKsl9IB4ds3RAktEeaSuQy_Zs2SrzLr976UtblBt28",
  "expires_in": 1800,
  "token_type": "Bearer",
  "refresh_token": "eyJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE1Mjg4Mj
M3MTAsInN1YiI6InJhZGl1c3JlYWR3cml0ZXVzZXIxIiwianRpIjoiMjk5Z
jQ5YjYtNmU2NC0xMWU4LWJkNjUtNmY0ZmVmYjY1MzI5IiwibmJmIjoxNTI4
ODIzNzEwLCJleHAiOjE1Mjg4MjYxMTAsImFjY2Vzc1Rva2VuRXhwaXJlc0F
0IjoxNTI4ODI1NTEwODg5LCJyZWZyZXNoQ291bnQiOi0xLCJ0b2tlblR5cG
UiOiJKV1RfUmVmcmVzaCIsInVzZXJVdWlkIjoiMjliMjBlNjctNmU2NC0xM
WU4LWJkNjUtMzU4MmUwZjU5YjQ4IiwidXNlclJvbGUiOiJST0xFX1JFQURf
V1JVEUiLCJvcmlnaW4iOiJwYXNzd29yZCJ9.Lc7MYmieNMMrjx7XoTiW-x
8Z8qFCnzfNM1apgbwLQvo",
  "refresh_expires_in": 2400
}
```

Step 3 Use **GET /object/users** to verify that user objects were created for each user.

User objects are automatically created for each new user who logs into the device manager or who obtains an access token. You must run a deployment job to save these user objects. In high availability mode, the deployment job is required before the user can log into the standby unit.

For example, the **curl** command would look like the following:

```
curl -X GET --header 'Accept: application/json'
'https://ftd.example.com/api/fdm/latest/object/users'
```

The following response body shows that two external users logged in. Note that **userRole** shows the rights obtained from the cisco-av-pair configured in the RADIUS server for those user accounts. Use this information to verify you configured the RADIUS user accounts correctly. The **admin** user is the locally-defined user.


```
{
  "items": [
    {
      "version": "h2vom4wckm2js",
      "name": "radiusadminuser1",
      "password": null,
      "newPassword": null,
      "userPreferences": {
        "preferredTimeZone": "(UTC+00:00) UTC",
        "colorTheme": "NORMAL_CISCO_IDENTITY",
        "type": "userpreferences"
      },
      "userRole": "ROLE_ADMIN",
      "identitySourceId": "0a7996ae-6e5b-11e8-bd65-dbab801c44b9",
      "userServiceTypes": [
        "MGMT"
      ],
      "id": "150d9754-6e63-11e8-bd65-ed9b20f62114",
      "type": "user",
      "links": {
        "self": "https://ftd.example.com/api/fdm/latest/
object/users/150d9754-6e63-11e8-bd65-ed9b20f62114"
      }
    },
    {
      "version": "p4rgwcjr5colj",
      "name": "admin",
      "password": null,
      "newPassword": null,
      "userPreferences": {
        "preferredTimeZone": "(UTC-07:00) America/Los_Angeles",
        "colorTheme": "NORMAL_CISCO_IDENTITY",
        "type": "userpreferences"
      },
      "userRole": "ROLE_ADMIN",
      "identitySourceId": "e3e74c32-3c03-11e8-983b-95c21a1b6da9",
      "userServiceTypes": [
        "MGMT"
      ],
      "id": "5023d3ab-6dc5-11e8-b9ed-db6dba9bf94c",
      "type": "user",
      "links": {
        "self": "https://ftd.example.com/api/fdm/latest/
object/users/5023d3ab-6dc5-11e8-b9ed-db6dba9bf94c"
      }
    },
    {
      "version": "ngx7a2dixngoq",
      "name": "radiusreadwriteuser1",
```

```
      "password": null,
      "newPassword": null,
      "userPreferences": {
        "preferredTimeZone": "(UTC+00:00) UTC",
        "colorTheme": "NORMAL_CISCO_IDENTITY",
        "type": "userpreferences"
      },
      "userRole": "ROLE_READ_WRITE",
      "identitySourceId": "0a7996ae-6e5b-11e8-bd65-dbab801c44b9",
      "userServiceTypes": [
        "MGMT"
      ],
      "id": "29b20e67-6e64-11e8-bd65-3582e0f59b48",
      "type": "user",
      "links": {
        "self": "https://ftd.example.com/api/fdm/latest/
object/users/29b20e67-6e64-11e8-bd65-3582e0f59b48"
      }
    }
  ],
  "paging": {
    "prev": [],
    "next": [],
    "limit": 10,
    "offset": 0,
    "count": 3,
    "pages": 0
  }
}
```