# Using ApiDeployTool

Table of Contents

[Using ApiDeployTool 1](#_Toc512855179)

[Overview 2](#_Toc512855180)

[The registerApi Command 2](#_Toc512855181)

[Defining Policies 3](#_Toc512855182)

[Defining a Client List 4](#_Toc512855183)

## Overview

The ApiDeployTool was created primarily to provide a coding example for how to use the various Anypoint Platform APIs to automate the API registration aspects for deploying an API.

It can be used in its current form to register APIs, although it makes many assumptions about naming conventions and expected usage of the API that may not fit with a specific set of customer requirements.

This document will explain the current tool and how it uses the command line values to perform the registration.

## The registerApi Command

The ApiDeployTool is a java program that can be executed from a shell java command as follows:

java -jar target/ApiDeployTool.jar ***registerApi*** **myAnypointUser MyAnypointPassword** "**businessGroupName**" **myApi** **v1** "**myEnvironmentName**" **keyId** **my-policies.json** **my‑clients.json**

***registerApi*** is the operation to execute. The document “Automating API Mgmt” explains all the steps that are performed in more detail. The end result is an API instance that is registered in Anypoint API Manager with a specified set of client applications registered to use the API and a set of policies attached to the API to govern its uses.

**myAnypointUser** is the Anypoint user that will be used to perform all the registration steps. Note that this tool will create any client applications that are listed in the my‑clients.json file. In doing so, the user specified here becomes the owner of record for the application and its client credentials...no other users will be able to see these applications except the master org owner. Using a consistent user name here is important in order to have consistent visibility of the credentials for all automated API registrations.

**MyAnypointPassword** is the password for the user specified above.

**businessGroupName** is the Anypoint business group Exchange where the API will be registered.

**myApi** is the name of the API that will be registered in Exchange. In order to register the API into many Exchanges, the tool will append “\_**businessGroupName**” to the name specified here. So after running the registerApi operation for an API named ***MyExample*** in the business org named ***Transportation***, the API in Exchange will be named ***MyExample\_Transportation***.

**v1** is the version to use in Exchange. If the version of the API already exists in Exchange, then the existing version will be used.

**myEnvironmentName** is the environment within the business group that the API instance will be registered into.

**keyId** is the encryption keyId (keystore alias) that should be used for encrypting the configuration properties. This is used in conjunction with the configuration property service to store the API registration properties. The sensitive values (such as client secrets) would be stored as encrypted values.

**my-policies.json** is a file that contains the list of policies that will be applied to the API Instance being registered. See the following section on “Defining Policies” for more information on this file.

The file name specified here must either be in the current running directory or on the Java classpath as a resource file. The default file is distributed in the project as client‑credentials‑policy which applies client credential enforcement using HTTP headers client\_id and client\_secret.

**my-clients.json** is a file that contains the list of client applications that will be registered as consumers of the API. As noted earlier, these client applications will be created if they do not already exist. Do to the current structure of Anypoint Access Management, the client application will fail if it already exists in another part of the Master Org that the specified Anypoint user performing the registration does not have access to. See the following section on “Defining a Client List” for more information on this file contents.

The file name specified here must either be in the current running directory or on the Java classpath as a resource file. The default file is distributed in the project as empty‑client‑access‑list which is an empty list resulting in no client applications being registered to use the API Instance.

### Defining Policies

This is a file that can be current directory where ApiDeployTool is running, or in the Java classpath. The current directory is searched first.

The file is in json format and lists the policies that should be applied to the API Instance. There are several examples of policy definitions in the resources directory of the ApiDeployTool:

* client-credentials-policy
* ip-whitelist-policy
* simple-basic-auth-policy

To determine more policies, use the “Developer view” of a Chrome browser when adding policies through API Management to determine what properties and names to use. These are not really documented anywhere.

Here is an example of client-credentials-policy:

[{

"policyTemplateId": "client-id-enforcement",

"configurationData": {

"credentialsOrigin": "customExpression",

"clientIdExpression": "#[message.inboundProperties['client\_id']]",

"clientSecretExpression": "#[message.inboundProperties['client\_secret']]"

}

}]

Here is an example of basic authentication with a simple authentication manager:

[{

"policyTemplateId": "simple-security-manager",

"configurationData": {

"username": "username",

"password": "password"

}

}, {

"policyTemplateId": "http-basic-authentication",

"configurationData": {}

}]

### Defining a Client List

This is a file that can be current directory where ApiDeployTool is running, or in the Java classpath. The current directory is searched first.

The file is in json format and lists the client applications that should be created (if they don’t already exist) and then registered to use the API Instance.

The registration assumes no SLA’s are configured for the API. Here is an example of registering two applications:

[{

"applicationName": "auto-api-registration"

},

{

"applicationName": "my-web-app"

}

]