API Security : Securing APIs with API Key

Duration : 30 mins
Persona : API Team

Use case

You have an API Proxy that you want to secure, so that you can manage access and perform business logic base on the client making the call. In addition authorizing users, you want to know which Developer App is making calls so you can leverage that data to customize your API behavior based on the entitlement level of the caller, or even the specific caller. You also would like to be able to see who is calling your API Proxies in your Analytics dashboards.

How can Apigee Edge help?

The Verify API Key Policy in Edge authenticates that the call is coming from an approved application in a valid state. App developers who wish to access secure operations must request API keys for their apps via the developer portal.

In addition to authenticating requests, the Verify API Key Policy provides context about who is making the call. This context can be used to apply policies such as Quota management or routing based on the client app. Upon successful verification of the API Key, the API Context is populated with details about the App, Developer, and API Product associated with the call. This data can be used for applying business logic as well as gaining business insights through analytics.

In this lab, you will protect an existing API Proxy with the Verify API Key Policy and use the trace tool to see the policy in action. To accomplish this you will modify an existing API Proxy to add a security policy to handle the authorization. You will also create several artifacts in your Organization.

- App Developer
- Developer App
- API Product

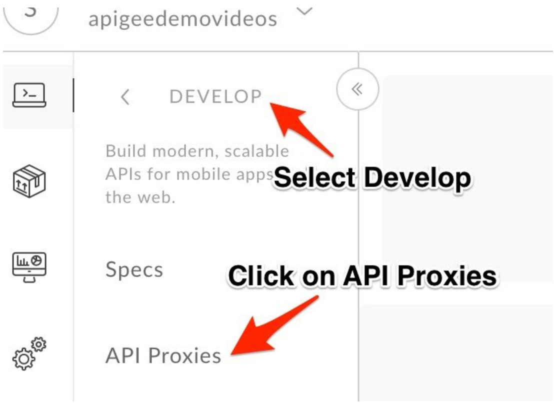
Pre-requisites

For this lab, you will need an API Proxy that is not currently secured. If you do not have an API Proxy available for this lab, revisit the lab “API Development - Create a Reverse Proxy” and then return here to complete these steps.

Instructions

Choose (and invoke) an API Proxy to secure

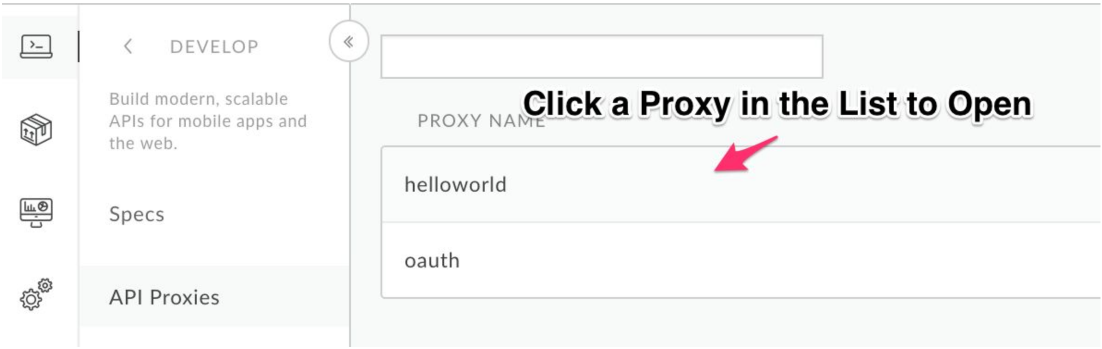
- Go to https://apigee.com/edge and log in. This is the Edge management UI
- Select **Develop → API Proxies**



Select Develop

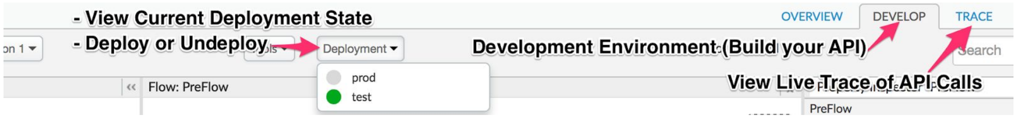
Click on API Proxies

- Find and open the API Proxy you want to secure by clicking on it.



Click a Proxy in the List to Open

- Verify that the API Proxy is deployed to an environment from the Overview page. Environment(s) to which the selected revision of the API Proxy is deployed will be indicated by a green circle. If it is not deployed, click an environment from the “Deployment” pull-down to deploy the API Proxy to that environment.

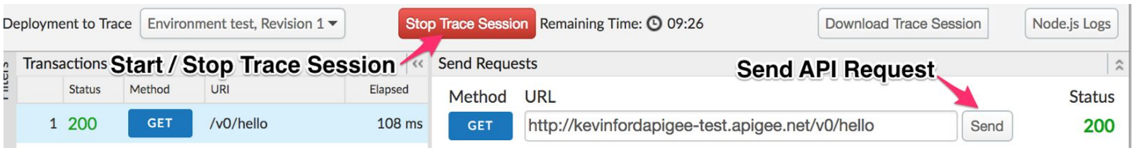


- View Current Deployment State
- Deploy or Undeploy

Development Environment (Build your API)

View Live Trace of API Calls

- Verify that you can successfully make calls using the built-in trace tool
- Click the “Trace” tab near the top of the window.
- The Trace view allows you to initiate tracing for up to 10 minutes, during which time all requests (regardless of whether they are sent from the trace tool or any other client) will be captured and their traces made visible to you.



Start / Stop Trace Session

Send API Request

	Status	Method	URI	Elapsed
1	200	GET	/v0/hello	108 ms

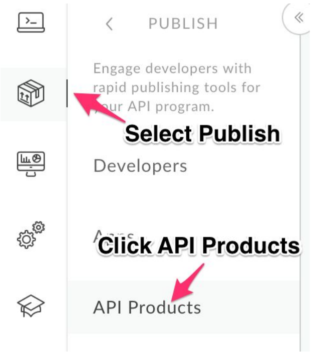
Method	URL	Status
GET	http://kevinfordapigee-test.apigee.net/v0/hello	200

- Click “Start Trace Session” to begin a trace session.
- Click “Send” to send a request. If your API Proxy requires query parameters add them here prior to sending.
- You should see a successful 2xx response for your API Call (it may take a few seconds for the trace results to appear)
- If you are not able to successfully test an API Proxy in the Trace Tool, revisit the lab “API Development - Create a Reverse Proxy”

Publish API as part of API Product

Once secured, consuming apps will need an API Key to successfully invoke your API. The way that Developer (consumer) Apps request API Keys is via an API Product. In short, API Products are the unit of deployment to the Developer Portal, where App Developers can learn about, register for, and consume your APIs. Read more about API Products here.

- Select **Publish** → **API Products** from the side navigation menu


PUBLISH
Engage developers with rapid publishing tools for your API program.
Developers
Apps
API Products

Select Publish

Click API Products

- Click **+API Product** and populate the following fields
  - Section: Product Details
    - Name: {your_initials}_{api_name}_product
    - Environment: test
    - Access: Public
  - Section: Resources
    - Section: API Proxies

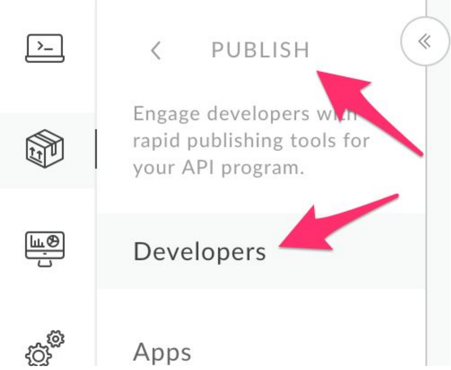
- Click the +API Proxy button
- Select your API Proxy
- Save the API Product

Note: We are adding the entire API Proxy to the API Product. We can just as easily select one or more operations from one or more API Proxies and bundle them together in an API Product.

Create An App Developer

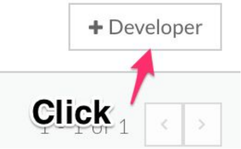
Next we will create an App Developer who can consume the new API Product.

- Select **Publish** → **Developers** from the side navigation menu

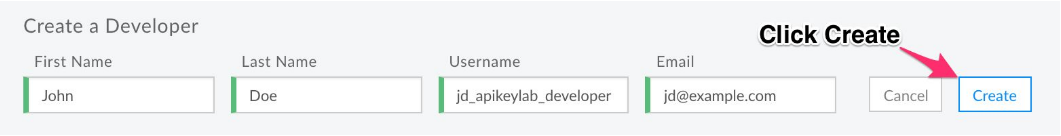


Engage developers with rapid publishing tools for your API program.

- Click **+Developer**



- Populate the following fields
  - First Name: {your_first_name}
  - Last Name: {your_last_name}
  - Email: {your_email}
  - Username: {your_initials}_apikeylab_developer
- Click **Create** to save the new App Developer.



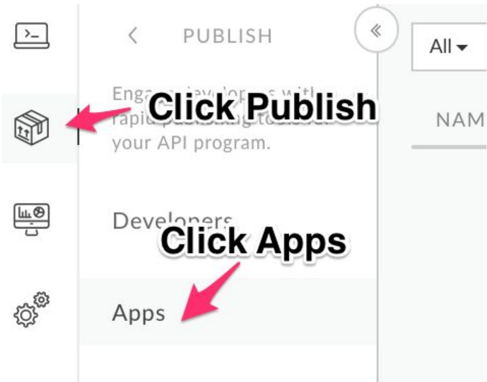
Create a Developer

First Name	Last Name	Username	Email
John	Doe	jd_apikeylab_developer	jd@example.com

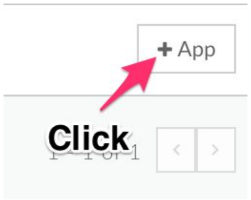
Create An App And Retrieve Its API Key

An App Developer can create any number of Apps. Each App can register for any number of products. We will create an App for our new App Developer, and register it with the API Product we created earlier in the lab. Read more about Developer Apps here.

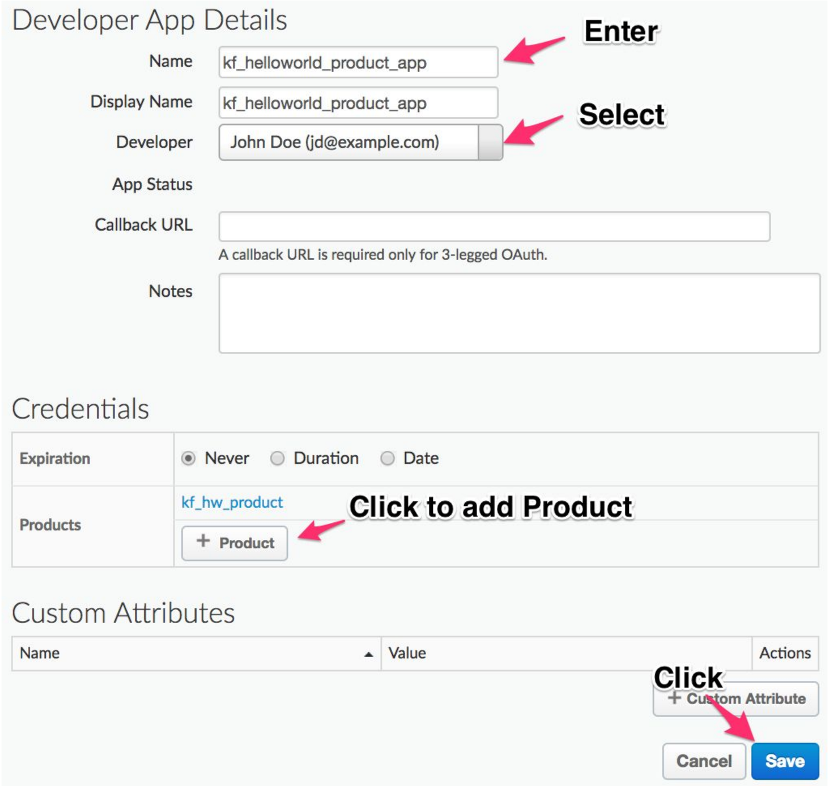
- Click **Publish** → **Apps** in the side navigation



- Click **+App**



- Populate the following fields
    - Name: {your_initials}_{product_name}_app
    - Developer: Select the developer you created from the pulldown
    - Product: Click +Product to add your API Product to this App



Developer App Details

Name	kf_helloworld_product_app
Display Name	kf_helloworld_product_app
Developer	John Doe (jd@example.com)
App Status	
Callback URL	A callback URL is required only for 3-legged OAuth.
Notes	

Credentials

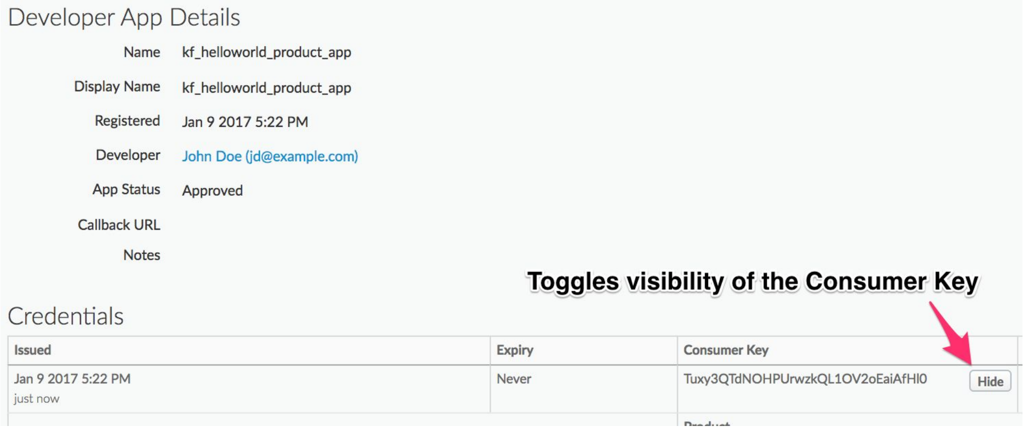
Expiration	● Never ○ Duration ○ Date
Products	kf_hw_product + Product

Custom Attributes

Name	Value	Actions

- Click Save

- Open the newly created App and click “Show” under “Consumer Key”. This will reveal the API Key that must be used to invoke the API when API Key verification is in use. Copy this key into a text document for later use



Developer App Details

Name	kf_helloworld_product_app
Display Name	kf_helloworld_product_app
Registered	Jan 9 2017 5:22 PM
Developer	John Doe (jd@example.com)
App Status	Approved
Callback URL	
Notes	

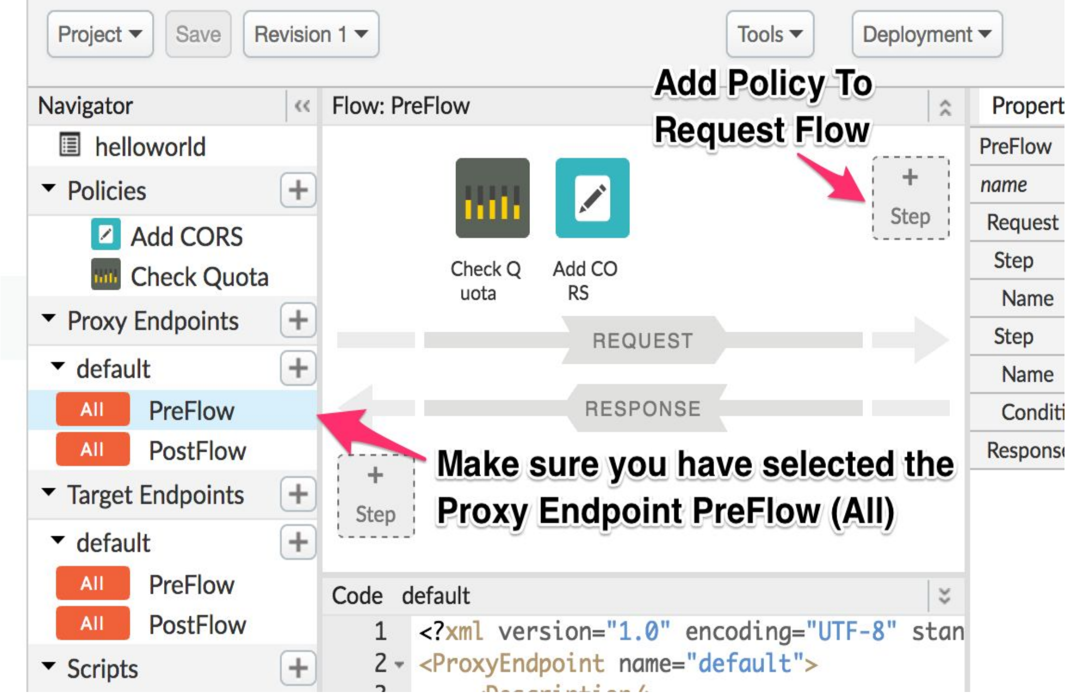
Toggles visibility of the Consumer Key

Credentials

Issued	Expiry	Consumer Key
Jan 9 2017 5:22 PM just now	Never	Tuxy3QTdNOHPUrwzkQL1OV2oEaiAfHl0 Hide

Add a “Verify API Key” Policy

- Menu: **Develop > API Proxies**
- Open your API Proxy and click the “Develop” tab to see the flow editor (you may have to move the panes to see the full request and response flow lines)
- Click **+Step** on the request flow and select “Verify API Key” policy from the “Security” section of the list. The name can be changed or left at the default.



Add Policy To Request Flow

Make sure you have selected the Proxy Endpoint PreFlow (All)

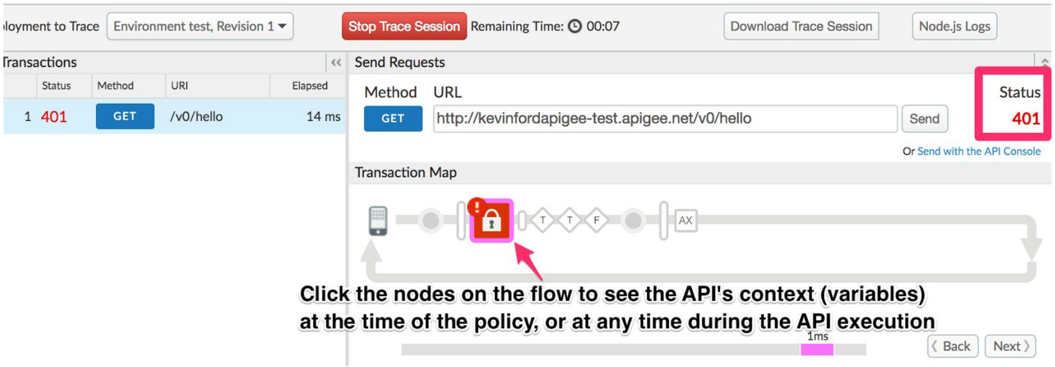
```
1  <?xml version="1.0" encoding="UTF-8" stan
2  <ProxyEndpoint name="default">
```

- Click **Add.**

- The policy will be added after any policies you previously had in the Request flow. Since we likely want this to occur first, drag the new policy to be the leftmost.
- With the “Verify API Key” policy selected, you can see its configuration (the default policy configuration is below). Note that the API Key is being retrieved from the context as the variable “request.queryparam.apikey”. This is the default but the policy can be configured to retrieve the key from any parameter key you prefer.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  <VerifyAPIKey async="false" continueOnError="false"
enabled="true" name="Verify-API-Key-1">
    <DisplayName>Verify API Key-1</DisplayName>
     <Properties/>
    <APIKey ref="request.queryparam.apikey"/>
</VerifyAPIKey>
```

- Save the API Proxy.
- Click the “Trace” tab near the top of the window.
- Click “Start Trace Session” to begin a trace session.
- Click “Send” to send a request. If your API Proxy requires query parameters add them dire prior to sending (Do not add the API Key yet)
- You should see a 401 (unauthorized) response for your API Call because the API Proxy was expecting an API Key as a query parameter. See the trace session below
- Now add the query parameter ?apikey={your_api_key} to the URL in the trace tool and try again. (Use the API Key you created here) and resend the request.



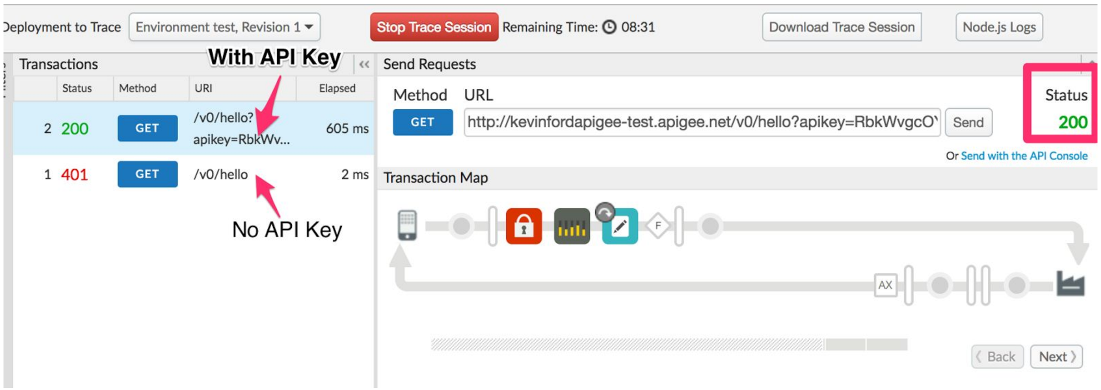
Transactions

	Status	Method	URI	Elapsed
1	401	GET	/v0/hello	14 ms

Method: GET
URL: http://kevinfordapigee-test.apigee.net/v0/hello
Status: 401

Click the nodes on the flow to see the API's context (variables) at the time of the policy, or at any time during the API execution

- You should see a 2xx response code and the Trace for that request should show that the Verify API Key policy is now passing.



	Status	Method	URI	Elapsed
2	200	GET	/v0/hello? apikey=RbkWv...	605 ms
1	401	GET	/v0/hello	2 ms

Method: GET
URL: http://kevinfordapigee-test.apigee.net/v0/hello?apikey=RbkWvgcO
Status: 200

Lab Video

If you would rather watch a video that covers this topic, point your browser here.

Earn Extra-points

Now that you have secured an API Proxy with API Key, you have access to details about the calling App, Developer, and associated API Product in the API flow. See if you can locate these details for a protected API call.

A few examples of where this might be useful.

- Route to a sandbox backend when a Product has the custom attribute of sandbox=true.
- Implement different quota policies for Apps that have been approved but not yet verified.
- Analyze traffic by calling App, Developer, or Product

Quiz

1. What would happen if a Quota Policy were placed before the Verify API Key policy?
2. Why is the Verify API Key policy typically found as the first policy in the Request PreFlow? When might it be in a conditional PreFlow instead of the "All" PreFlow?
3. How would you configure the policy to get the API Key from a header called "Api-Key" instead of the default query parameter location?

Summary

In this lab you learned how to protect your API Proxy using the Verify API Key policy. You implemented the policy and tested it using the built-in Trace Tool.

References

- Link to Apigee docs page
  - Verify Api Key Policy
    http://docs.apigee.com/api-services/reference/verify-api-key-policy

Rate this lab

How did you link this lab? Rate here.