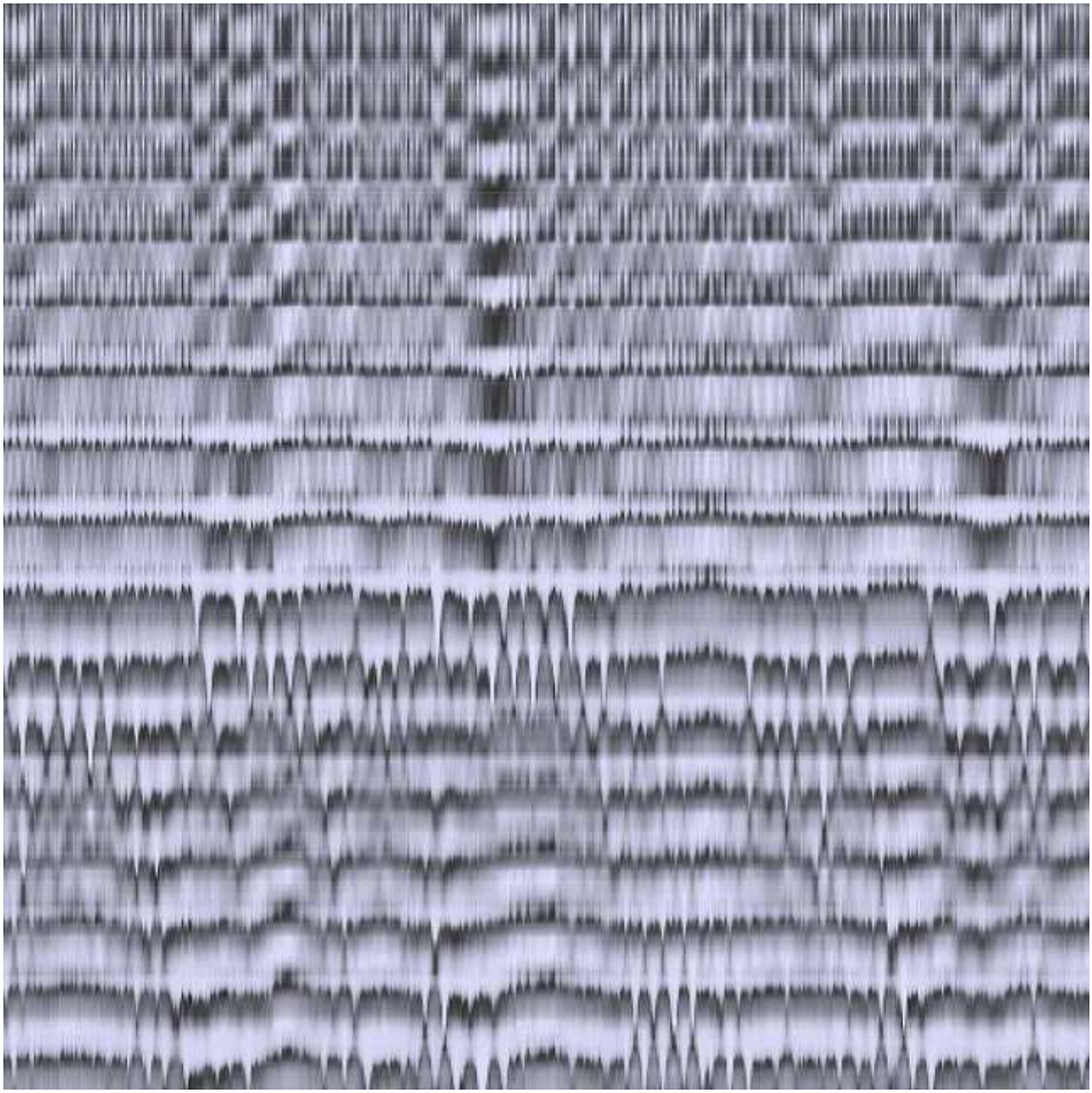
Apigee Edge OAuth2 and Third-Party Identity Providers

Robert Broeckelmann · Follow

8 min read · Feb 5, 2018

19 1



Patterns / fdecomite

This blog post summarizes the details of how to deploy and configure the the Apigee Edge OAuth2 example that I put together. This example will use the OAuth2 Authorization Code Grant and Refresh Token Grant to demonstrate how OAuth2 can be used with Apigee Edge in a real-world application.

The sample API Proxy also supports the OAuth2 Client Credentials Grant.

This post is a summary of the instructions provided across several different GitHub repos.

These instructions use the OAuth2 + OIDC Debugger.

Configure Red Hat SSO (as Third-Party Identity Provider)

- Configure Red Hat SSO v7.1 as described here for the OIDC Authorization Code Flow (which is also compatible with the OAuth2 Authorization Code Grant).

> *Make note of the following information:*
>
> *Authorization Endpoint*
>
> *Token Endpoint*
>
> *UserInfo Endpoint*
>
> *Test user name*
>
> *Test user password*

Setup the OAuth2 + OIDC Debugger

- Follow the instructions here to setup the OAuth2 + OIDC Debugger app on your local machine. This is a simple test application that simulates the interaction between a real app and an IdP using the OAuth2 or OIDC protocols.

Configure Apigee OAuth2 Provider API Proxy

- Clone the OAuth2 Wrapper API Proxy GitHub repository to a local file system by running:

```
git clone https://github.com/rcbjLevvel/apigee-api-proxy-oauth2-rh-sso-wrapper.git
```

- Install the npm (Node Package Manager) tool on your system using the appropriate packaging system.
- Install the apigeetool by running “npm -g install apigeetool”.
- Deploy the API Proxy by running:

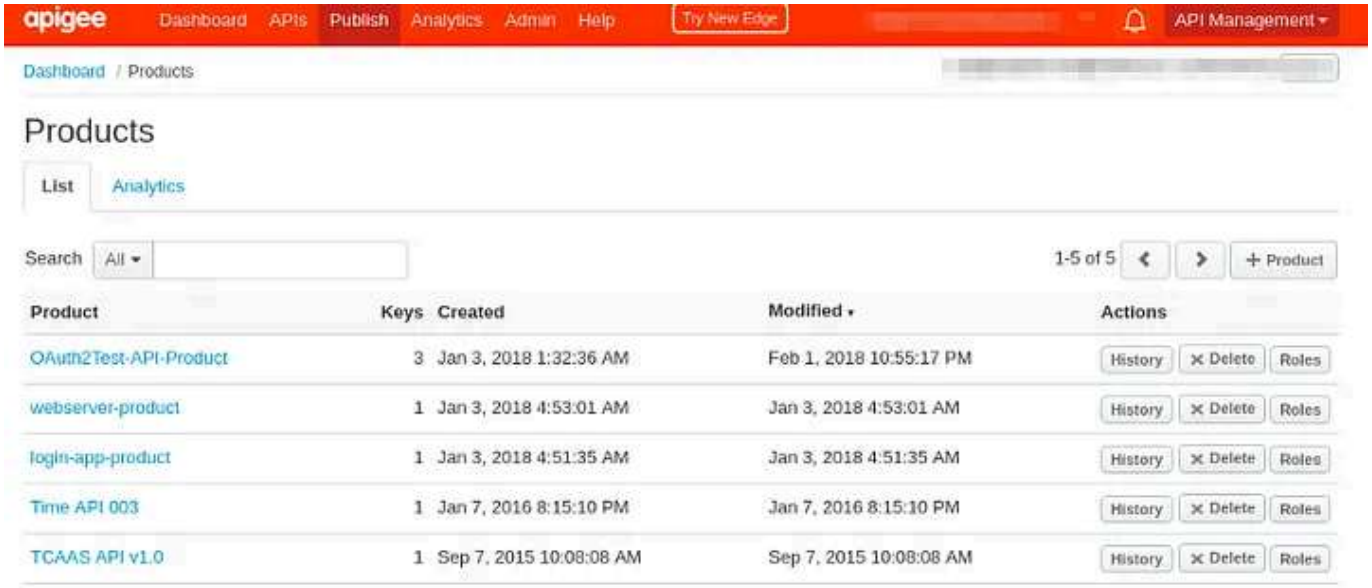
```
apigeetool deployproxy  -u admin_user_for_org -p admin_password -o apigee_org  -e env_name -n blog-rh-sso-integration -d ${REPOSITORY_HOME}/proxy
```

where

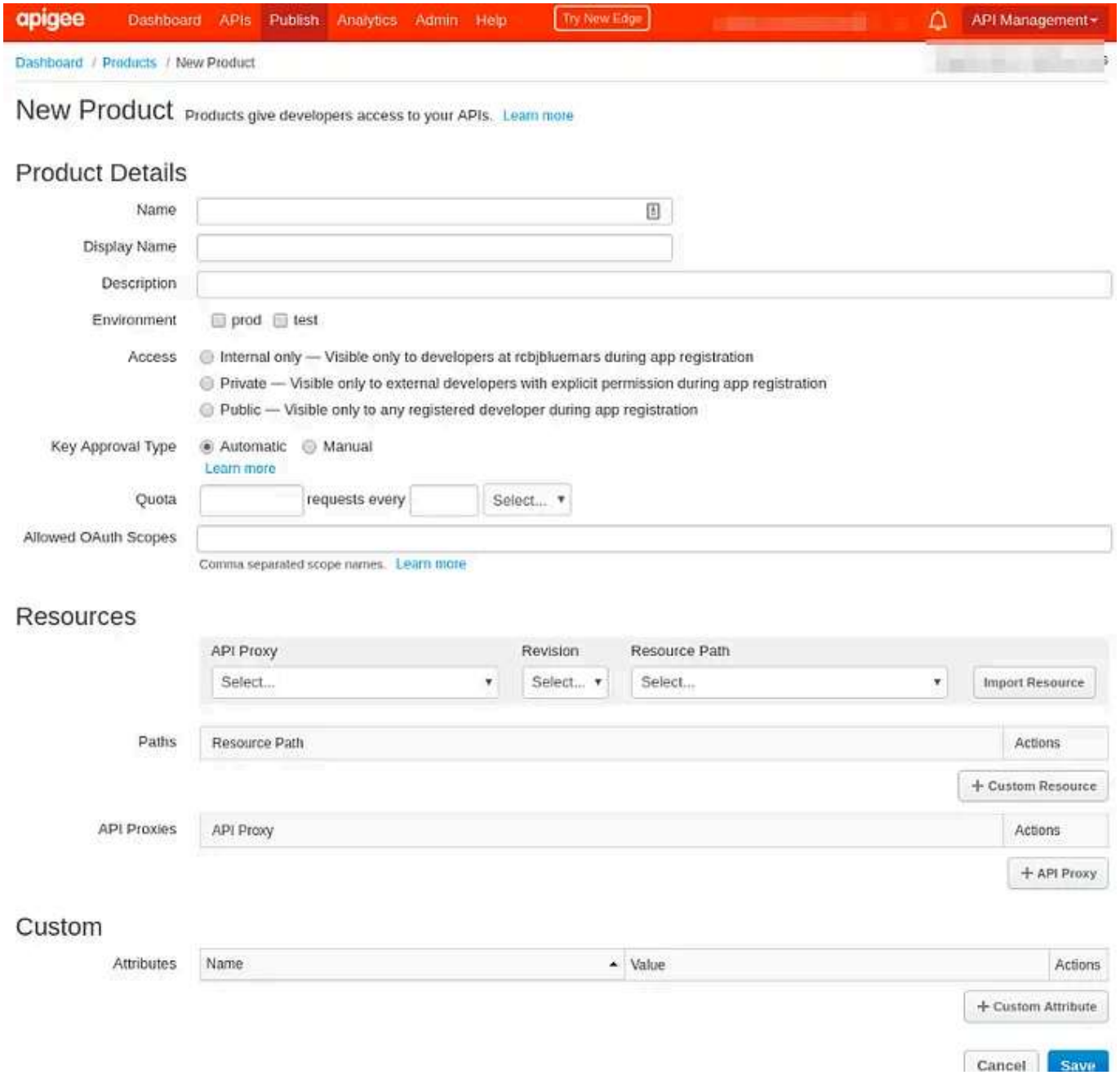
> *admin_user = an admin user for your apigee organization*
>
> *admin_password = the admin user’s password*
>
> *apigee_org = your apigee org*
>
> *env_name = the name of the environment where the API Proxy is to be deployed*
>
> *REPOSITORY_HOME = path to the root directory of the git repo.*

- Alternatively, if you are having trouble with apigeetool, you can also deploy the built API Proxy bundle (available here) through the web console.
- Open your favorite web browser (Chrome & Safari are known to work).
- Log into the Apigee Edge Public Cloud console here (https://enterprise.apigee.com).

- Go to Publish->Products.



- Click the “+Product” button in the upper left-hand corner.

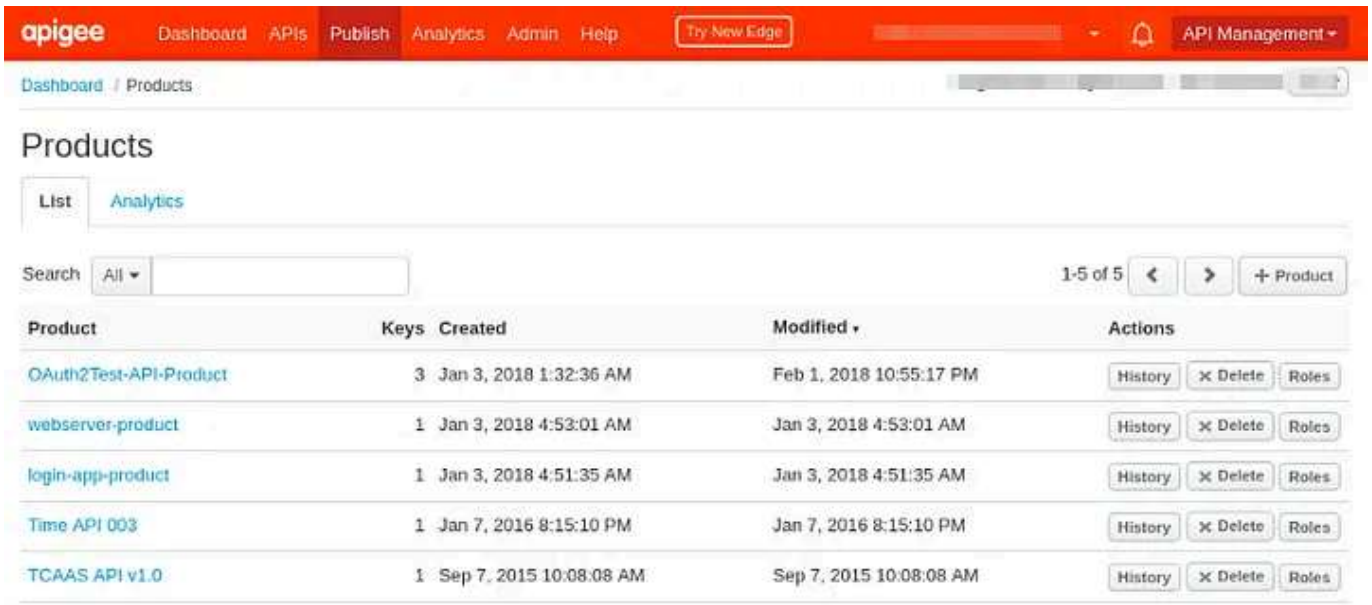


- Give the new Product a name of “OAuth2Test-API-Product”.
- Fill in the additional fields:

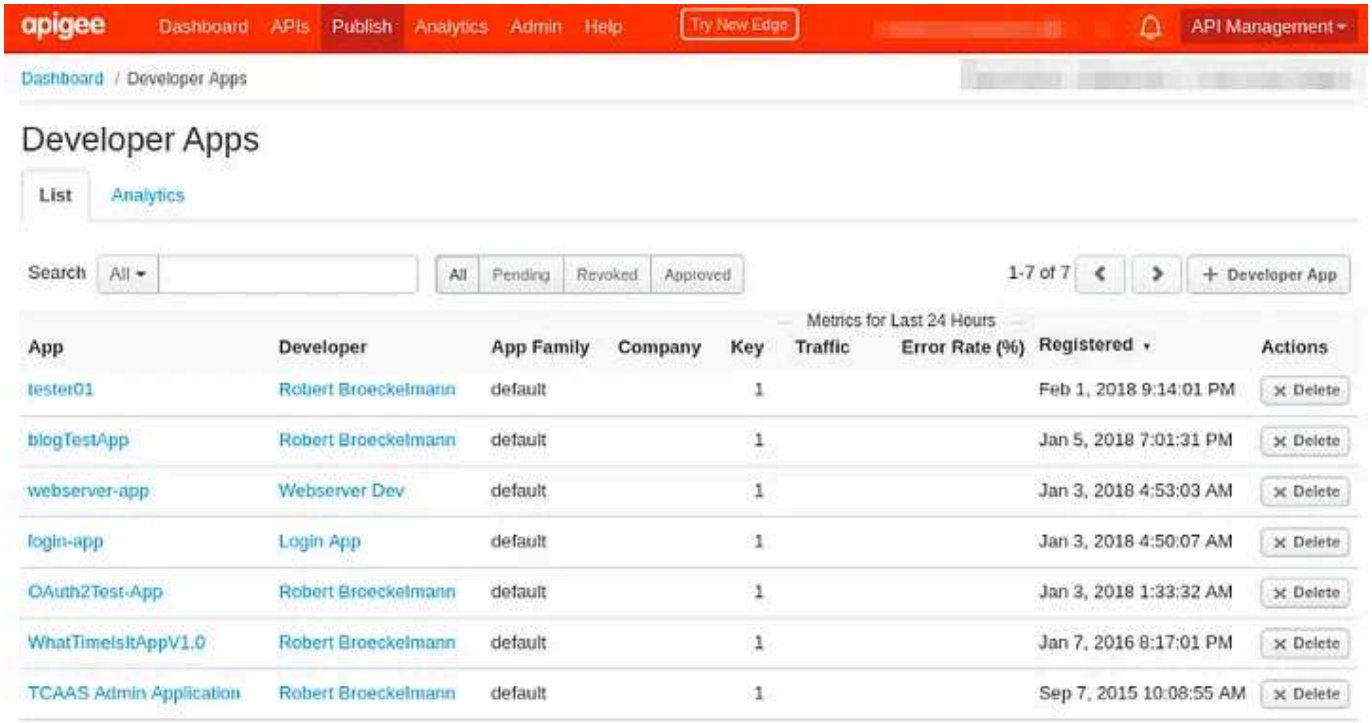
> *Display Name: Provide a meaningful display name.*
>
> *Description: Provide a meaningful description.*
>
> *Environment: Test*

> *Key Approval Type: Automatic*
>
> *Access: Public*
>
> *Quota: Can be left blank*
>
> *Allowed OAuth scopes: User*
>
> *Paths: /*
>
> *API Proxies: blog-rh-sso-integration*

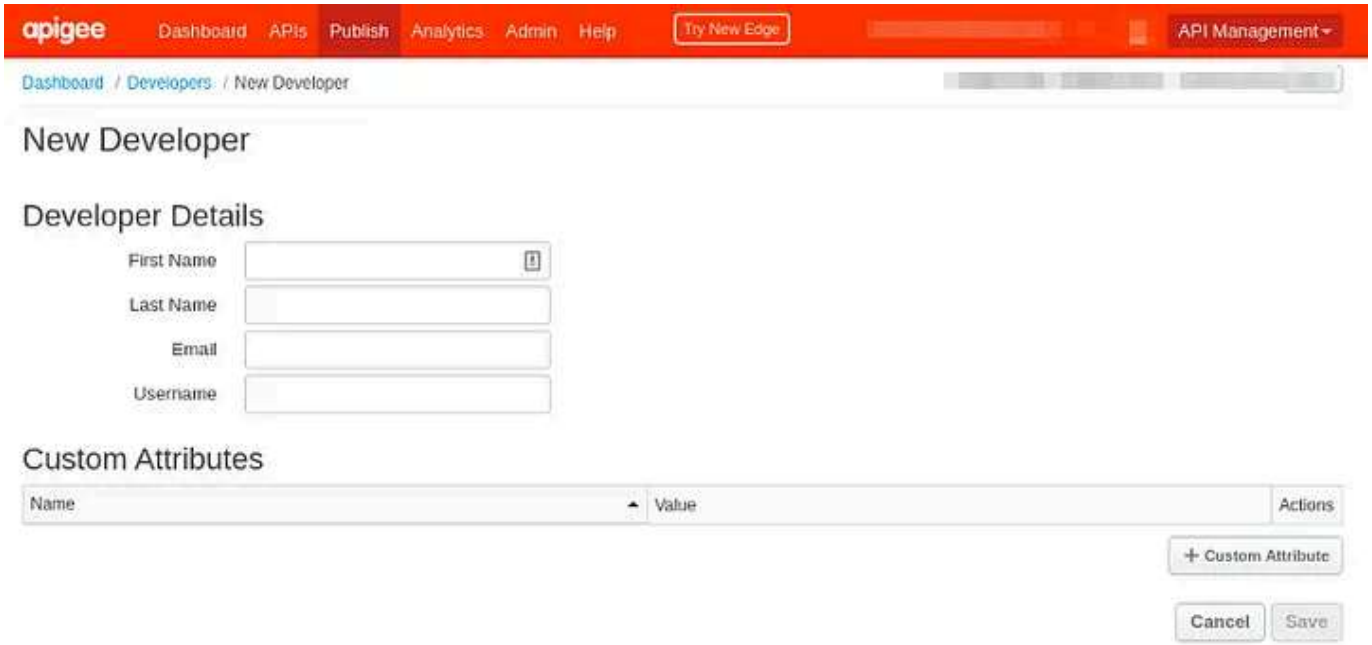
- Click Save.



- Go to Publisher->Developer Apps.



- Click the “+Developer App” button.

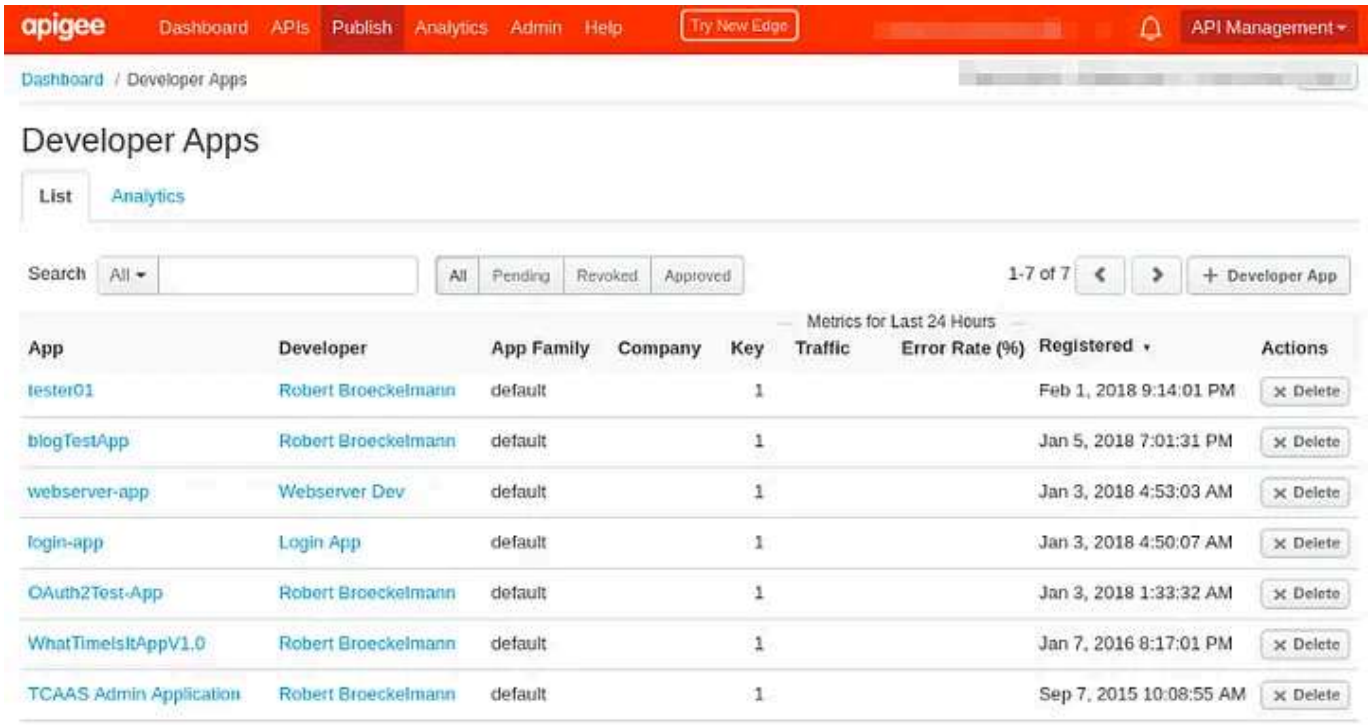


- Fill in the following parameters:

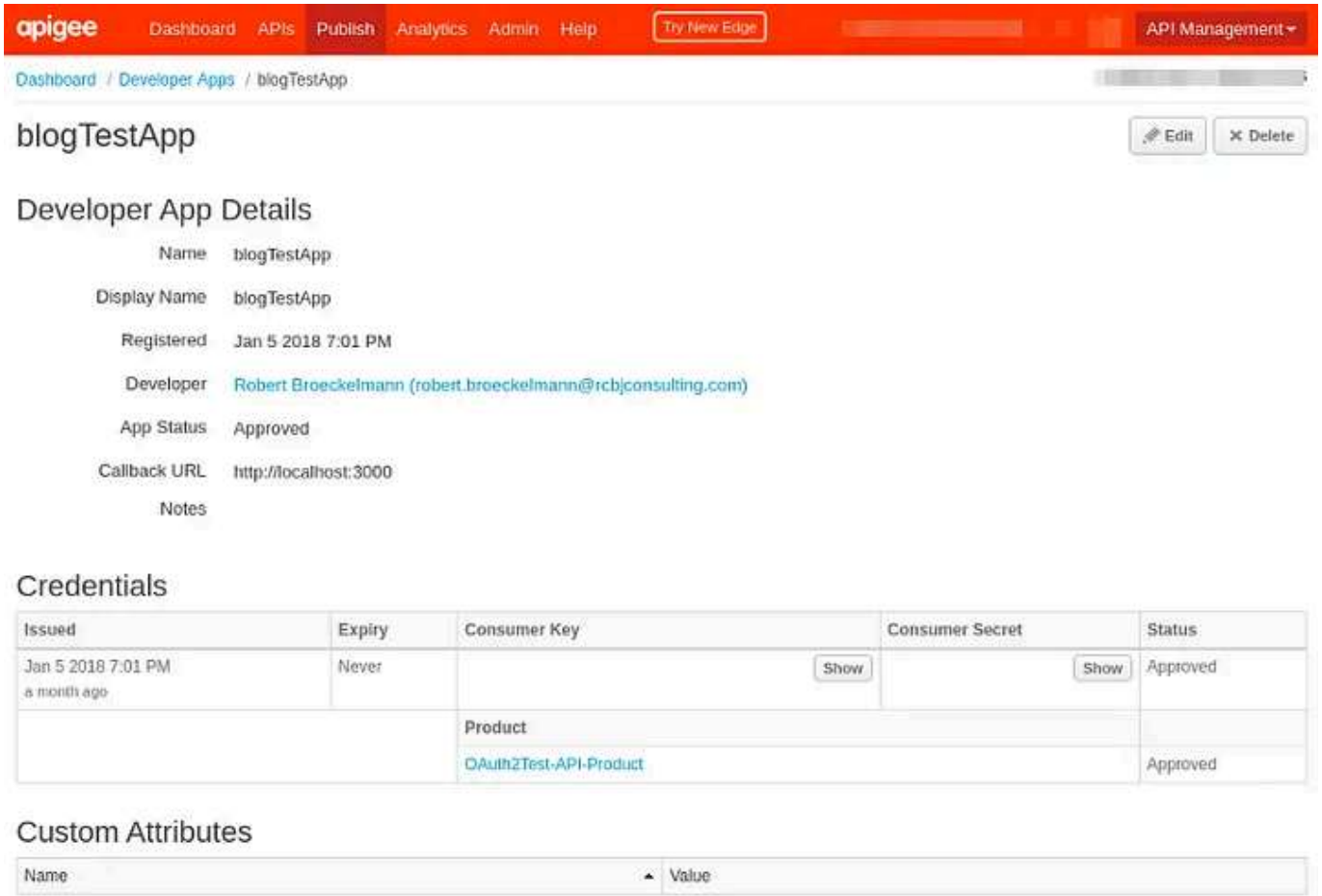
> *Name: blogTestApp*

> *Display Name: A meaningful display name.*
>
> *Developer Name: Yourself*
>
> *Callback URL: http://localhost:3000/callback (so this can be used with the OAuth2 + OIDC Debugger*
>
> *Expiration: Never*
>
> *Products: The product created above (OAuth2Test-API-Product).*

- Click the Save button.



- Click on “blogTestApp” in the list of Developer Apps.



- Under Credentials, click on the Consumer Key button.

- Save this value for later reference (this is the OAuth2 client identifier).
- Under Credentials, click on the Consumer Secret button.

Credentials

Issued	Expiry	Consumer Key	Consumer Secret	Status
Jan 5 2018 7:01 PM a month ago	Never	Show	Hide	Approved
		Product		
		OAuth2Test-API-Product		Approved

- Save this value for later reference (this is the OAuth2 client secret).
- If you are using Red Hat SSO v7.1 as the third-party Identity Provider, copy the following script to your local file system to create a mirror OAuth2 client definition in Red Hat SSO:

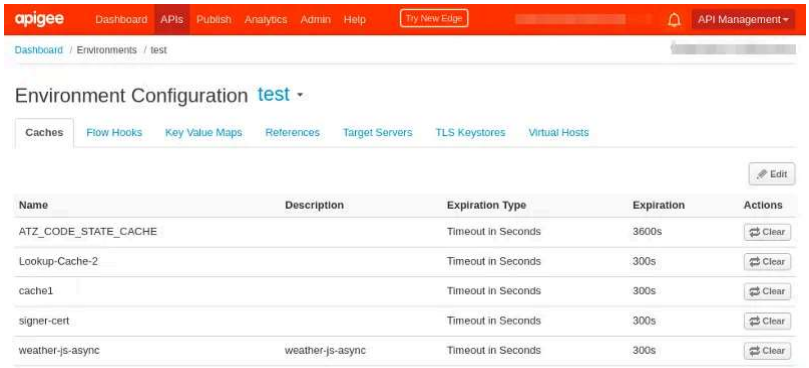
```
#!/bin/bash
#Update these variables with the values obtained earlier.
CLIENT_ID=
CLIENT_SECRET=
REDIRECT_URI=
KEY=
REALM=
RH_SSO_HOST=
curl -v -X POST \
-d ‘{ “clientId”: “${CLIENT_ID}”, “secret”:
“${CLIENT_SECRET}”,”redirectUris”:[“${REDIRECT_URI}"] }’ \
-H “Content-Type:application/json” \
-H “Accept: application/json” \
-H “Authorization: Bearer KEY” \
https://${RH_SSO_HOST}:8443/auth/realms/${REALM}/clients-
registrations/default --insecure -D headers.out
```

- Update the following values in the shell script:

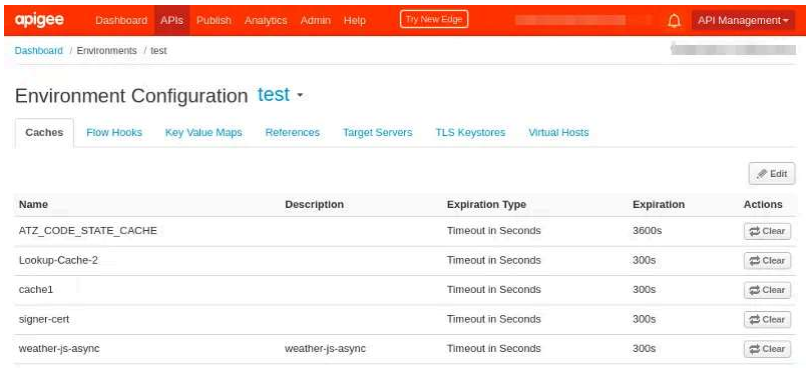
> *CLIENT_ID=the Apigee test application client_id that was just created.*
>
> *CLIENT_SECRET=the Apigee test application client_secret that was just created.*
>
> *REDIRECT_URI=the 3Scale test application redirect_uri that was just created.*

> *KEY=INITIAL_ACCESS_TOKEN just created in Red Hat SSO*
>
> *REALM=RH_SSO_REALM_NAME*
>
> *RH_SSO_HOST=resolvable Red Hat SSO URL hostname*

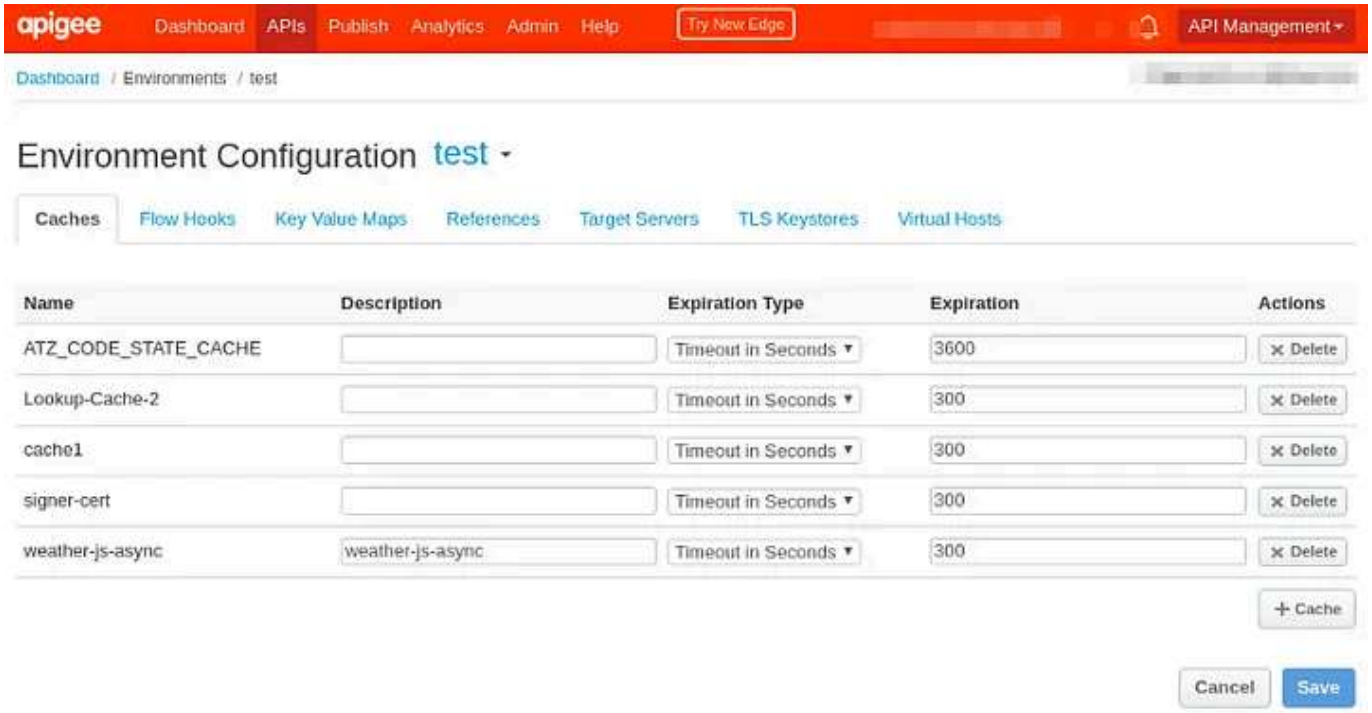
- Run the shell script to create the client definition in Red Hat SSO.
- Back in the web browser with the Apigee Web Console open, go to APIs->Environment Configuration.



- Go to the Caches tab (should be the default).



- Click the Edit button.



- Click the “+Cache” button.

- Enter a name for the new Cache: ATZ_CODE_STATE_CACHE
- Enter a description, if needed.
- Set the timeout to 3600 seconds. For real world purposes, the cache expiration should be a few seconds less than the access token timeout configured in Red Hat SSO for the test client.
- Click the Save button.

Dashboard / Environments / test

Environment Configuration test

Caches | Flow Hooks | Key Value Maps | References | Target Servers | TLS Keystores | Virtual Hosts

Edit

Name	Description	Expiration Type	Expiration	Actions
ATZ_CODE_STATE_CACHE		Timeout in Seconds	3600s	Clear
Lookup-Cache-2		Timeout in Seconds	300s	Clear
cache1		Timeout in Seconds	300s	Clear
signer-cert		Timeout in Seconds	300s	Clear
weather-js-async	weather-js-async	Timeout in Seconds	300s	Clear
tester1	tester1	Timeout in Seconds	3600s	Clear

- Go to the Key Value Maps tab.

Dashboard / Environments / test

Environment Configuration test

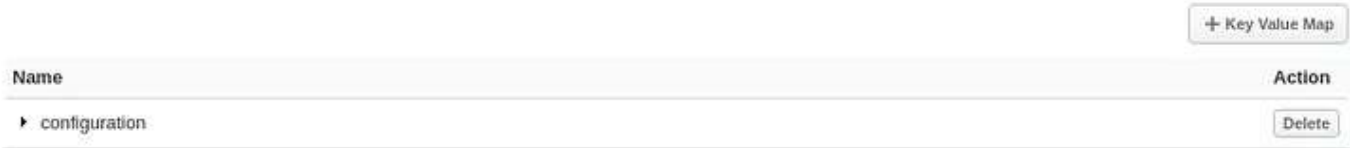
Caches | Flow Hooks | Key Value Maps | References | Target Servers | TLS Keystores | Virtual Hosts

+ Key Value Map

Name	Action
configuration	Delete
passwords	Delete

- Click the “+ Key Value Map” button.

- Give the new KVM a name: configuration
- Click Add.
- Wait for the screen to refresh.



- Add the following values to the configuration KVM by expanding the KVM, clicking the “+ Entry” button, adding the key:value pairs, and clicking save.

> *idpUserInfoEndpoint: The OIDC UserInfo Endpoint (example: /auth/realms/demo_project_sf/protocol/openid-connect/userinfo)*
>
> *idpTokenEndpoint: The OIDC Token Endpoint (example: /auth/realms/demo_project_sf/protocol/openid-connect/token)*
>
> *idpAuthorizationEndpoint: The OIDC Authorization Endpoint (example: /auth/realms/demo_project_sf/protocol/openid-connect/auth)*

> *idpHost: (example: ec2-blah.compute-1.amazonaws.com:8443)*

- The final result should look similar to the following (note, the Apigee Service Callout Policy requires that the host:port and path be defined in separate variables):

Deploy OAuth2 Validating API Proxy

- Clone the OAuth2-Validating API Proxy GitHub repository to a local file system.
- Install npm (Node Package Manager) on your system using whatever package manager is appropriate.
- Install the apigeetool by running “npm -g install apigeetool”.
- Deploy the API Proxy by running (you can also deploy the API Proxy bundle through the console):

```
apigeetool deployproxy  -u admin_user_for_org -p admin_password -o
apigee_org  -e env_name -n blog-rh-sso-integration -d
${REPOSITORY_HOME}/proxy
```

Setup the OAuth2 + OIDC Debugger

- Clone the OAuth2 + OIDC Debugger GitHub repository by running:

```
git clone https://github.com/GetLevvel/oauth2-oidc-debugger.git
```

- Follow the instructions in this repo's README.md to build and start the docker image.
- The short version is:

> *Run "cd ${REPO_HOME}/client".*
>
> *Run "docker build -t oatuh2-oidc-debugger .".*
>
> *Run "docker run -p 3000:3000 oauth2-oidc-debugger".*

Obtain An Access Token

- Open a browser.
- Go to http://localhost:3000.
- Using the following values, use the OAuth2 + OIDC Debugger to obtain an access token:

> *Authorization Endpoint: https://org-env.apigee.net/oauth2/authorization (org = your org, env = env name)*
>
> *Token Endpoint: https:/org-env.apigee.net/oauth2/token (org = your org, env = env name)*

> *Client Identifier: Obtained above from the Consumer Key*
>
> *Client Secret: Obtained above from the Consumer Secret*
>
> *Callback: http://localhost:3000/callback*
>
> *Scope: User*
>
> *Username: configured in the third-party IdP (User1, if following the Red Hat SSO v7.1 post referenced above).*
>
> *Password: configured in the third-party IdP (secret, if following the Red Hat SSO v7.1 post referenced above).*
>
> *Validate Token Endpoint SSL: Yes*
>
> *Display OIDC Artifacts: No*
>
> *Use Refresh Tokens?: Yes (if needed)*
>
> *Validate SSL Certificate?: Yes (the Apigee endpoint uses a certificate issued by a public CA, by default)*

Make an API Call With The Access Token

- Run the following the command:

```
curl -X GET https://${ORG}-${ENV}.apigee.net/oauth2test/posts/1 --
insecure -H "Authorization: Bearer ${TOKEN}" -D headers.out
```

where

> *ORG=your apigee org name*
>
> *ENV=environment where the proxy is deployed (test or prod by default)*
>
> *TOKEN = OAuth2 access token obtained in the last step*

- The output should look something like:

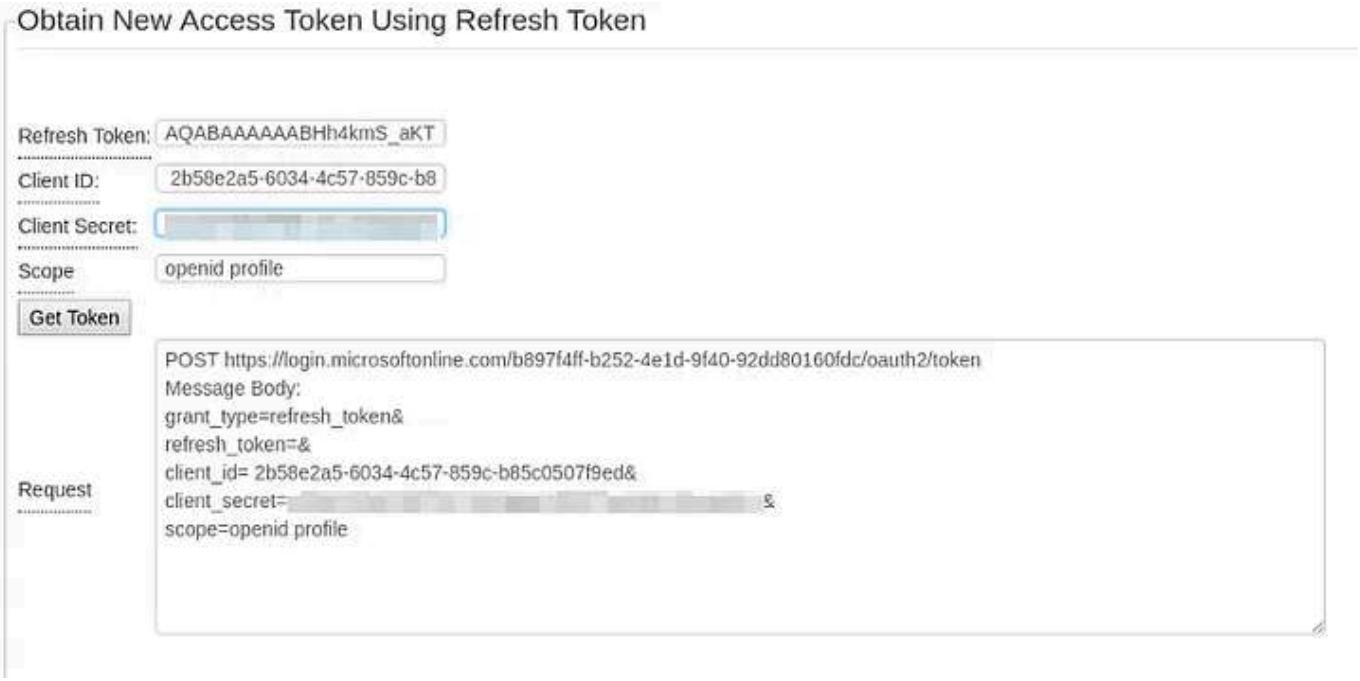
```
{
  "userId": 1,
  "id": 1,
  "title": "sunt aut facere repellat provident occaecati excepturi
optio reprehenderit",
  "body": "quia et suscipit\nsuscipit recusandae consequuntur
expedita et cum\nreprehenderit molestiae ut ut quas totam\nnostrum
rerum est autem sunt rem eveniet architecto"
}
```

Use a Refresh Token To Obtain a New Access Token

- Use the steps described in this post to obtain a new access token when the original expires.

The important steps are:

- Make sure that you selected Yes next to “Use Refresh Tokens” in the Configuration section at the top.
- You will see something similar to the following.



- Enter the following values:

> *Enter your client's client identifier in the Client ID field.*
>
> *Enter your client's client secret in the client Secret field (if applicable).*
>
> *Enter "openid profile" in the Scope field.*

- Click the "Get Token" button.
- You will see something similar to the following:

Token Endpoint Results for Refresh Token Call:

access_token

AQABAAAAAABHh4kmS_aKT5XrjzxRAtHzE72o2P5WWgaYRSlsQjxle0dHY9bhTA8j9zgiBkjXGo0XJrq_cfQ93erLmbshU3xH7hQ
9jKNGpE_TbWFoQCt3nMgLoJlU1-
ty5xQ8IjdspHeyCOr_Y5q6CTlvGPtTogvBKe1c5yzvYD807BCsW5ADqiVcLlpAD0dsxetuFZm4YZgQ41cT7-
KMNmnoJZtfrXUJY1d0jnqY8w8wWkIKx0Uu8JB8fr0au
KxtNi53oX2bnvYZlVWy3hSo6q2DR-61EZ6KFMdhwbx
dJX8mP2-fgA6cYDGwascxMIjxkmoElTR0IwsZ4XiNDUWkcj6MunTseJby_1XGkkyVo-evFds7dwm69q-
pb9ry7SDNQ_Jf_sJOoo9BUdpQxDmM_eAi4J2i2XYepEmYNsaWAAB8Nk1x1lL1T4dVnMGiuNmrEDQLUQrGjwo0Z8hkwwNinFof
NEDov6Af_44bF92LkQgFvPDNSblllW5L4vNF-
rLQNF82HJGHWnwv_iNyWO8ahebqqEANmd62eMVP1KASZIo5Xri6180DPSgisJuk330qKzR8MjtjjX86sNYhmeVIDBxSnGwzm2
D_CJkiRVMB1gMBTUJ94_RKUOZxN29mkU__6FgLq7T0OHeens2pyeW--

refresh_token

AQABAAAAAABHh4kmS_aKT5XrjzxRAtHzy90cMmGyJ4VUv9VAn115guIO3nqk-iJdqycy-5xjbTp-
scO8ucaP8AECaKziNnuPW3qlbGrlYNqA0NyWWDJY44RoqwLbW34OecPCyHShbzOn9CRo18pFa3VQQfpN0eKJI53l8KLOdw-
RaVI_tte8BGcACewFpNuceLN0p21Wy9fNGXoq8K
r_gG6Arg_sh87q5oWS_cfKGz6rAuTE_KeWNbCV
1B6qAJgt_1_vkcEOUoADZk7EPZTNxg9lKPT9agf
HyqQgtONh_wHPyT1NYU9dIm9n6mFJhnIs1dtlZn
EuTdf0VwMb4vHw8xUQ3uFLs7oIMGQ4WzupkuJwzYD4SSHO9wwi4EgBoZyLiro27L8hYym3urTp03TqLzRQ-
4R81nfFvIApOzii_b_obs2LSCYOC46K8VNEOaITzvZ13uA-ZAKqV_MAFx2sKwFmzePBlEee-xIuCtYu2O9Ynz-
e9MNtVCQBfN4mmIW6BbmOqgY83BOztxI5A6r2W0XLCs05lnxbQyPjvRvCIugH9S5oYX083toCrsv2kwn-fEYbtbaaj-
xYaRUAfsoJg5LYuTo6gEQ4L04ocApPYiVDWwGDBZhZ6MvII1rD-AcLYpgJSGAj91pX-

id_token

eyJ0eXAiOiJKV1QiLCJhbGciOiJub25lIn0.eyJhdWQiOiIyYjU4ZTJhNS02MDM0LTRjNTctODU5Yy1iODVjMDUwN2Y5ZWQiLCJpc3
MiOiJodHRwczovL3N0cy53aW5kb3dzLm5ldC9iODk3ZjRmZi1iMjUyLTRlMWQtOWY0MC05MmRkODAxNjBmZGMvIiwiaWF0IjoxN
TE3NzI5MjA5LCJuYmYiOjE1MTc3MjkyMDksImV4cCI6MTUx
JjYmoubmV0IiwiZmFtaWx5X25hbWUiOiI1MzViZmNlYi0xMTg
YnJvZWNrZWxAcmNiai5uZXQiLCJpZHAiOiJsaXZlLmNvbSIsImlwYWRkcil6IjI0LjIwLjkyLjEzNCIsIm5hbWUiOiJicm9lY2tlbEByY2JqL
m5ldCA1MzViZmNlYi0xMTg1LTQxNjMtODRmMC02NWMxYTZhMmZlYTgiLCJub25jZSI6Ijg4NzViMTA5LTc1ZjktNDkzYS1hOWI2L
WI4MjcyODkxMDFkZCIsIm9pZCI6ImZiMmIwMDg0LWU0ZmItNDcxNi05ZDFhLTAwMmJhZjJhMjIyMCIsInN1YiI6IkYySFVucnBxeHB
Tbk1wOGNISmc0dDZiVnJUbnZPVjJwS2REQ2xmRUVhRU0iLCJ0aWQiOiJiODk3ZjRmZi1iMjUyLTRlMWQtOWY0MC05MmRkODA
xNjBmZGMiLCJ1bmlxdWVfbmFtZSI6ImxpdmUuY29tI2Jyb2Vja2VsQHJjYmoubmV0IiwidmVyIjoiMS4wIn0.

iteration 1

- You can continue to click the “Get Token” button to obtain new access tokens for as long as the Identity Provider will allow it.

Token Endpoint Results for Refresh Token Call:

access_token
AQABAAAAAABHh4kmS_aKT5XrjzxRAtHzmxUi_cQDU91Q4KsZkT5iTUmJqTuNbpRnl04la9-
72FM9qXf9Bicd8UwI3q9F9Z7ivGNJ69xnCxWQSGL7jIJ4bUXpNHjyynibn8TBjF1Mc4wD72ETwkOsdHaIJSMBUxPV_X6oBJtLPF
UmQf_2I2Y2EkUSFAe9GsslPVlDzEbQqe53cWjqssG_mYoJ_5CI8orEOWXxu5dl5h2zlvnVV3LnEzRRLnEOCCUSELtmHhgSqup
K8wZV2nDFhHYJuUeTM_7rRa0U4FoUWxikq1HVCXdtRTOR
nyjhjqJV7AX-Bxwmtas1n6YSDYh2-
2bhp1bn7smxJArXGytasCSc4bCXDwTAlAkkwFl1sNyYFRBxzgZ_oAt4LU7GwKzfKvFBoByhQrkDBirk0oIqxdNeNG9cMgd8_ZNw
V9n4SRDskDxvRUMS35TmOs9JAtaX5lzLX1szOl9UCzzg7XjwQheJQ_ZwTwJwtniWGiiw-MvFKk_A9ocAlKUU-lVeJkozjt-
nxpYdydxTjXuRX6hEWvE1EoFhhqwy2ez6durgeKZOZG6Z01khBUcQ6LxvltBV2Z3F_Tvalr_-
QpFo7mSLKm8iXV4jtEwFSn6jYTjmdt3ogwc0XC1fHItC_eoD_1OIni9UzZfNJ3uHCGxwUt832sNf5ykpr_ev5594p2V4zBavvc1PCQ
2ShyLwo4ZZ8xo7phZDZO6BBwkWeDDJLh6ZBrMuOdkIoGFuFMpmx5h1BlG-

refresh_token
AQABAAAAAABHh4kmS_aKT5XrjzxRAtHzNrRTFLhjRDGAUyGjtaTgUxwZittO7TVRMI5NVgzjlXait-
YwnS_suSAUyVhhDhsinRujbgrbMKru8srOAGymx5CLOBtMNDzsbmRb5nYRPMhf51YPNvmGsLCxJMrE30CA1Z1cNFVnm4BT
Ztn9KE1WKMdXRodyl6plKVF7KYgc18dQvWQlTi7U2wfuP452ayh_4_tAviCTitCFVVsYK-
7pcs_Yvqeh6gqKxb6QD9lPlBgvFEZjIyqlFbtd_l6deD
AImOcQbmawdvj4xtJ58m47aBT2Vi-g_XF75ixgQ_3r
Iyi2xp8Od7abLzqXkGuD8hUCzlkNNO_OVF-
gD7rfhCgfMjU6EJCUqkuawC79x2hzCh60PJ2ZsC6CWeerdsePQ33SAEi8BFa3rW9bzJ1yO_ewwtE7fj4M4S1FtFWZegZ5Lt5tlW
1wQSXghoP0P0X4wm9lOoIoSjjqHLIXWYZxenWSXdOAtC6c6tZ_p6psRyz7Ih-4i-
_psiozLt6qcgsgsV0It8MmH5FGTxdq2x91ryxhNMckoOXFv6Lm8sNnOg7ST_TQC8Rk74Sr4_jDr1Avd_LcOcg7NtqtqQviv_o7ZdD
dJckIvILd1md9LZpg0ZIyyjGEQraRgJkbHJaNXPSlTX6JGwSjTUggqaCMLNJMR5A56p8V0KtZFcAU1R4AHqSW8Bq_zcqPb_pE

id_token
eyJ0eXAiOiJKV1QiLCJhbGciOiJub25lIn0.eyJhdWQiOiIyYjU4ZTJhNS02MDM0LTRjNTctODU5Yy1iODVjMDUwN2Y5ZWQiLCJpc3
MiOiJodHRwczovL3N0cy53aW5kb3dzLm5ldC9iODk3ZjRmZi1iMjUvLTRlMWOtOWY0MC05MmRkODAxNiBmZGMvIiwiaWF0IioxN
TE3NzI5NDAyLCJuYmYiOjE1MTc3Mjk0MDIsImV4cCI6MTUxNzczI
JjYmoubmV0IiwiZmFtaWx5X25hbWUiOiI1MzViZmNlYi0xMTg1LTQ
YnJvZWNrZWxAcmNiai5uZXQiLCJpZHAiOiJsaXZlLmNvbSIsImlwY
m5ldCA1MzViZmNlYi0xMTg1LTQxNjMtODRmMC02NWMxYTZhMmZlYTgiLCJub25jZSI6IjQ5MGNmMjJhLTU3NTQtNGY0Zi05OG
NiLTU2YWI2NjI2MTQyYSIsIm9pZCI6ImZiMmIwMDg0LWU0ZmItNDcxNi05ZDFhLTAwMmJhZjJhMjIyMCIsInN1YiI6IkYySFVucnBxe
HBTbk1wOGNISmc0dDZiVnJUbnZPVjJwS2REQ2xmRUVhRU0iLCJ0aWQiOiJiODk3ZjRmZi1iMjUyLTRlMWQtOWY0MC05MmRkO
DAxNjBmZGMiLCJ1bmlxdWVfbmFtZSI6ImxpdmUuY29tI2Jyb2Vja2VsQHJjYmoubmV0IiwidmVyIjoiMS4wIn0.

iteration
2

- You can make additional calls to the API endpoint now with the new access token in the HTTP Request Authorization Header as described above in the “Make an API Call with The Access Token” Section.

Just as I’ve described in numerous other other posts on these topics, usually, the best approach is to use an OAuth2 (or OIDC) library that handles all of these details for you. However, when troubleshooting problems or for initial testing, working directly with the protocol through a debugger can be very helpful. Likewise, if troubleshooting the server-side Identity Provider components (or API Gateway in this case), interacting with the system in this way is helpful.

Image: Patterns / fdecomite

Oauth

Written by Robert Broeckelmann

1.99K Followers · 1 Following

Follow

My focus within Information Technology is API Management, Integration, and Identity–especially where these three intersect.

Responses (1)

What are your thoughts?

Respond

bhanu prakash
almost 7 years ago

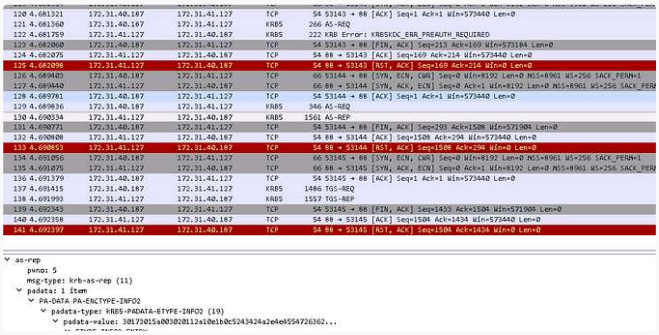
How can we implement Azure AD as the Identity Provider and Apigee as the OAuth Provider for aceessing and protecting APIs covered by API Proxies in Apigee Edge ?

I am new to this .....Can you please help ?

1 reply

Reply

More from Robert Broeckelmann



Robert Broeckelmann

Kerberos Wireshark Captures: A Windows Login Example

This blog post is the next in my Kerberos and Windows Security series. It describes the...

May 17, 2018 254 4

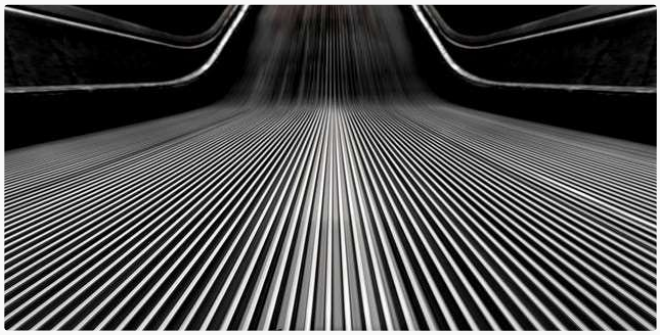
Robert Broeckelmann

OpenID Connect Logout

The OpenID Connect (OIDC) family of specs supports logout (from a single application)...

Jul 12, 2017 490 5

Robert Broeckelmann



Robert Broeckelmann

HTTP POST vs GET: Is One More Secure For Use In REST APIs?

The use of HTTP POST vs HTTP GET for read-only (or query) operations in REST APIs...

Feb 6, 2021 78

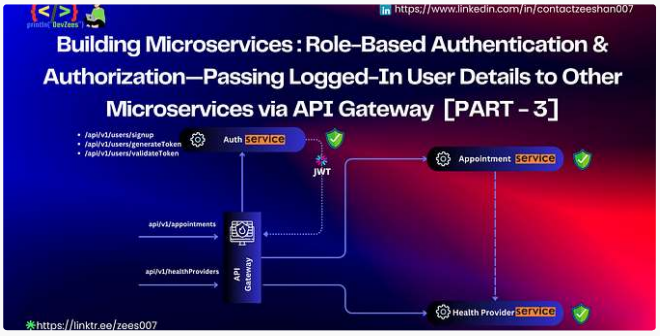
Authentication vs. Federation vs. SSO

Authentication. Federation. Single Sign On (SSO). I've mentioned these concepts many...

Sep 24, 2017 859 6

See all from Robert Broeckelmann

Recommended from Medium

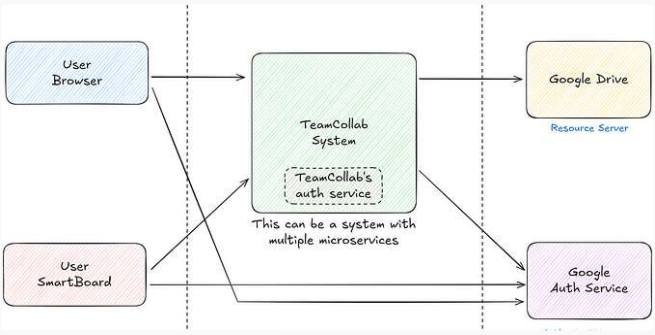


In Level Up Coding by Zeeshan Adil

Building Microservices [PART-3]: Role-Based Authentication &...

Welcome Back to DevZees ♥

Nov 12 106



In CodeX by Isuru Cumaranathunga

Choosing the Right OAuth 2.0 Grant Type. A Scenario-Based...

Apply the Right Grant Type to Your Use Case

Oct 13 2

Lists

Staff picks

776 stories · 1484 saves

Stories to Help You Level-Up at Work

19 stories · 887 saves

Self-Improvement 101

20 stories · 3110 saves

Productivity 101

20 stories · 2615 saves

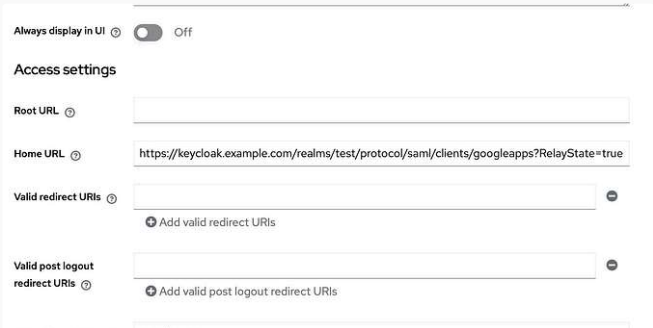
uthorization ype	Pros	Cons	Use Cases
BAC	Simplicity, Manageability, Performance	Limited Flexibility, Scalability Issues	Corporate intranet systems, content management systems
BAC	Flexibility, Granularity	Complexity, Performance Overhead	Cloud services, large-scale enterprise systems
CL	Granularity, Flexibility	Manageability, Scalability	File systems, databases, specific resource permissions
Auth 2.0	Delegated Access, Standardization, Security	Complexity, Implementation Effort	Third-party API access, social media integrations, SSO systems
WT	Stateless, Scalability, Interoperability	Security Risks, Token Management	RESTful APIs, microservices, single-page applications

Adnan taşdemir

Comparing Authorization Types in PHP: RBAC, ABAC, ACL, OAuth 2....

Authorization is a crucial aspect of web application security. It ensures that users ca...

Jun 30 2



MszPro

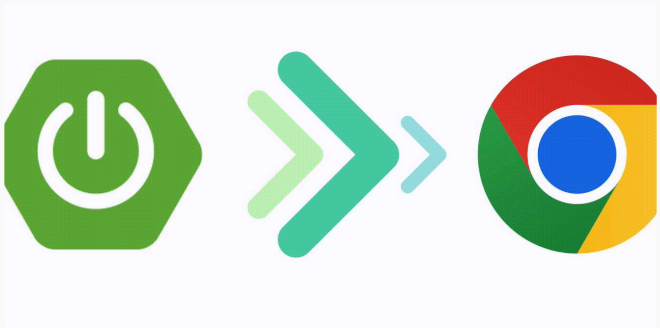
Use Keycloak to log into Google Workspace (KeyCloak 25 SAML...

This guide talks about setting your enterprise Keycloak as a way your employee can log int...

Aug 1

Yogendra Pratap Singh

Authentication



In Code Like A Girl by Mónika Lombos

In this article, I explore several key concepts related to authentication:

Oct 20

Social login with Spring Boot 3.1 — part 1

OAuth 2.0 visualization with step-by-step explanations and code examples

Oct 6, 2023 281 4

See more recommendations