

# Integration with Apigee



## Overview

OpenLegacy's API integration platform is the fastest and most standard way for legacy applications to be part of the Apigee API management and analytics software platform. OpenLegacy quickly and efficiently generates APIs for any legacy asset by connecting directly to the legacy system, automating code generation, and using microservices for deployment flexibility. With a couple of clicks, users can generate a consumable API for management by Apigee. There is no hand coding or additional configuration needed to generate an API that is compatible with Apigee's API proxies.

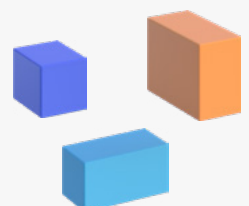
---

### Key benefits of OpenLegacy

- Enable Apigee customers to create additional APIs to be managed
- Direct Connection to almost any legacy and on-premise system
- Automatic code generation of APIs inside microservices.
- Parses metadata and generates SDK that includes run-time connection to legacy system
- Easily deployed into any infrastructure (Docker, PCF, Tomcat, etc).

### Key benefits of Apigee

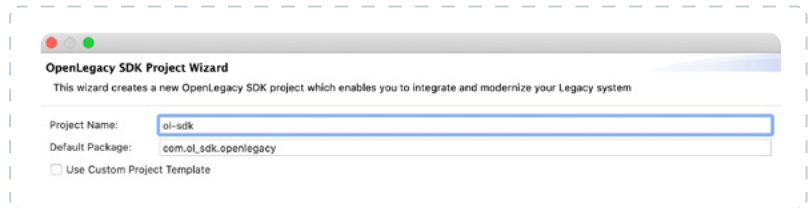
- Manage interaction with API consumers to optimize performance
- Transformation capabilities for easy consumption
- Security at the API layer to add additional protection to the legacy assets
- Development Portal dashboard for easy documentation of API contracts
- Analytics at the API level



# How OpenLegacy works: Legacy API creation in 10 easy steps

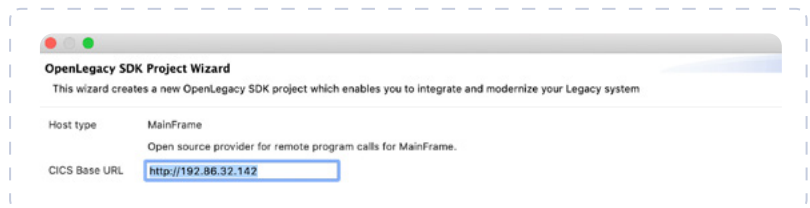
## 01

### Create an SDK project in the OpenLegacy IDE



## 02

### Populate the connection configuration (host, port, username, pass, etc.)

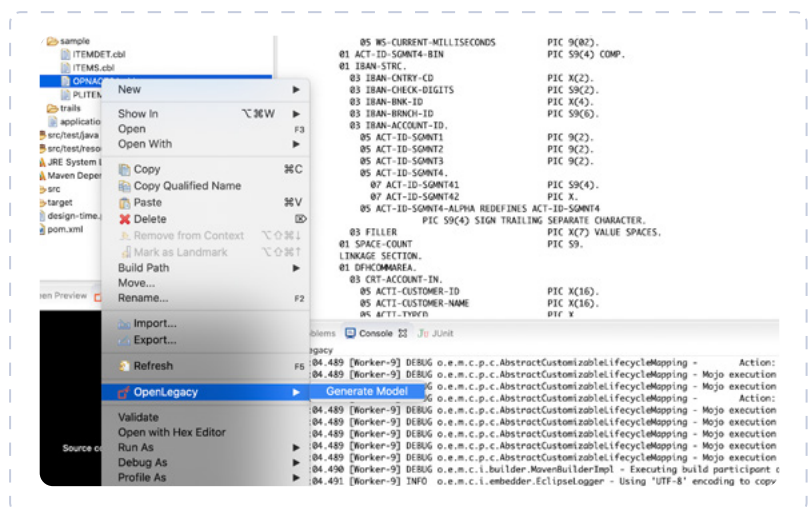


This enables OpenLegacy to build the connection information about the backend into the SDK project. It also can retrieve any metadata from the legacy system for parsing.

## 03

### Generate Java code based on metadata of back-end program

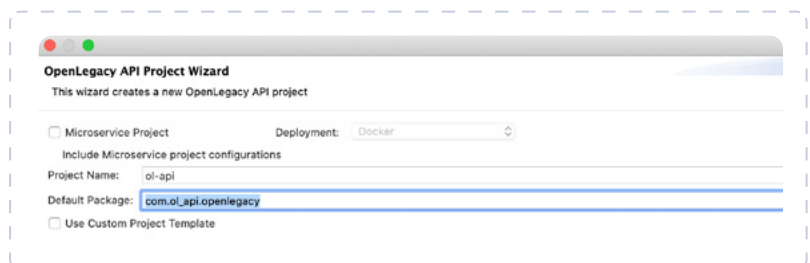
The code goes into the SDK project for use by the APIs.



## 04

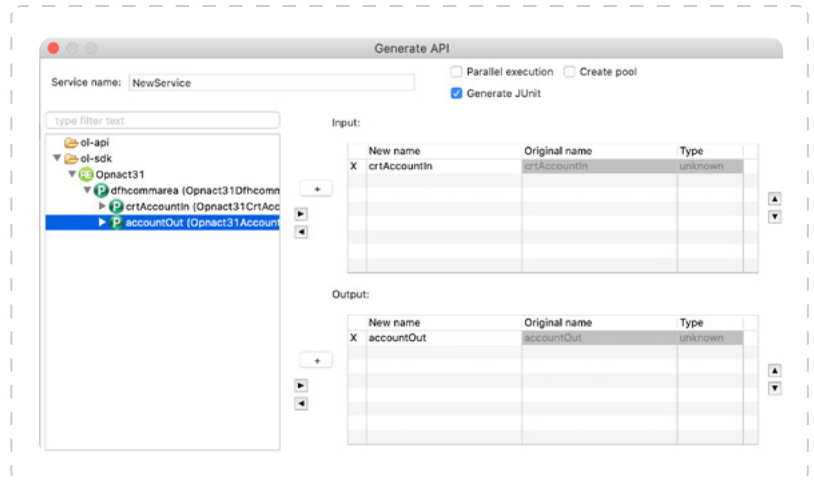
### Create an API project in the OpenLegacy IDE

API data gets populated from the SDK



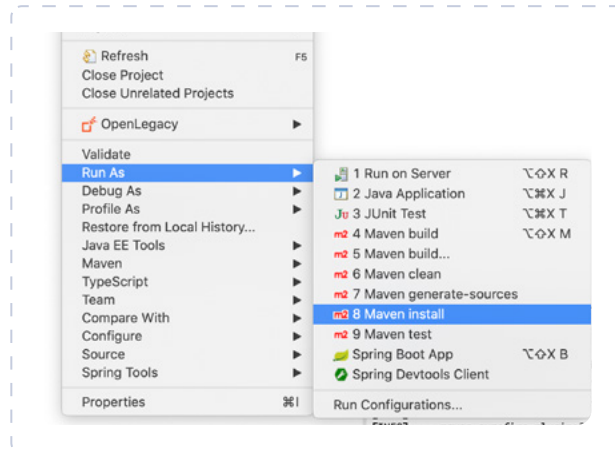
## 05

Create API inputs and outputs based on the back-end asset generated into the SDK



## 06

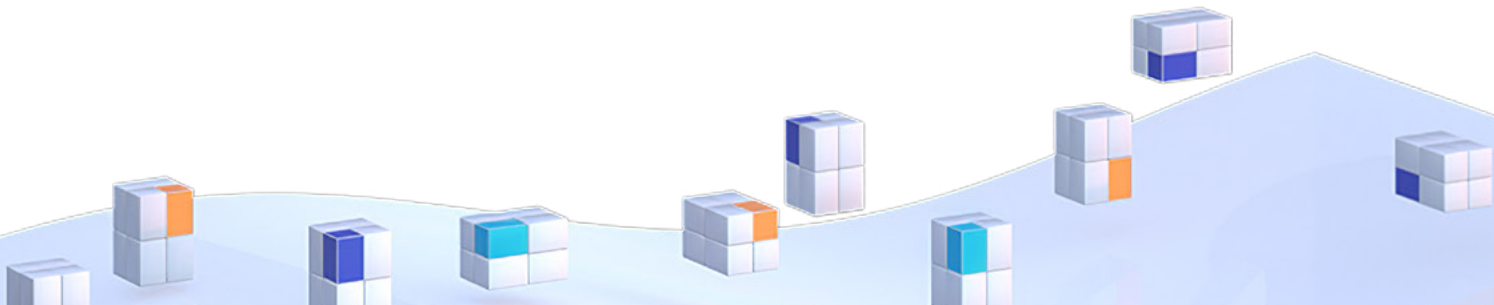
Put the microservice-based API into a JAR file by choosing "Maven Install"



## 07

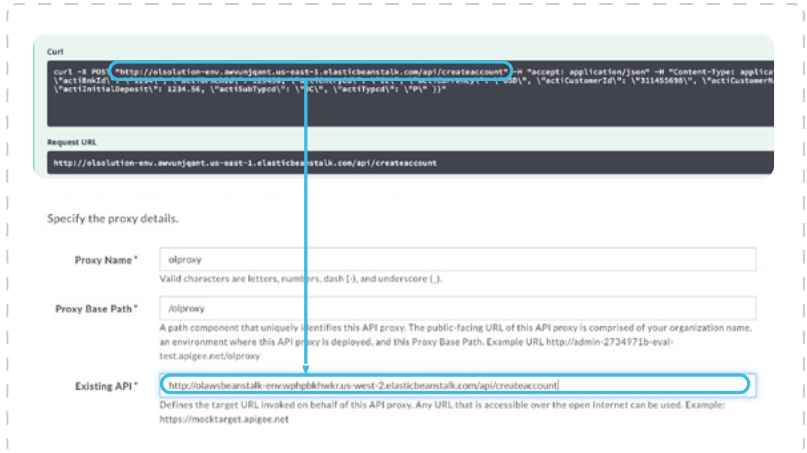
Configure an API Proxy in the Apigee dashboard

Preparation for deployment and management



## 08

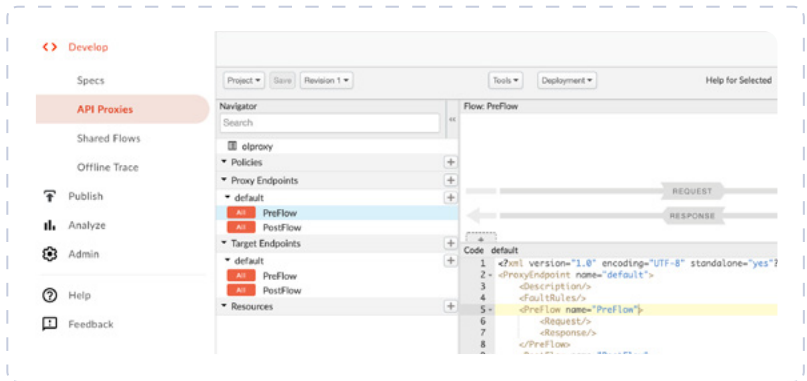
Use the OpenLegacy generated Swagger page to define the existing API for retrieval and deployment



The screenshot shows the Apigee console interface for configuring an API proxy. At the top, a cURL command is displayed, with a blue box highlighting the URL `http://olsolution-env.amazonaws.com-east-1.elasticbeanstalk.com/api/createaccount`. Below this, the 'Request URL' field is populated with the same URL. The 'Specify the proxy details' section contains three input fields: 'Proxy Name' (value: `olproxy`), 'Proxy Base Path' (value: `/olproxy`), and 'Existing API' (value: `http://olawsbeanstalk-env-wpfbkhwkr-us-west-2.elasticbeanstalk.com/api/createaccount`). A blue arrow points from the highlighted URL in the cURL command to the 'Existing API' field.

## 09

Apigee generates API metadata so platform can manage the active instance of the API



The screenshot shows the Apigee console interface for viewing API proxy metadata. The 'API Proxies' section is selected in the left sidebar. The main area displays the 'Flow: PreFlow' configuration. The 'Code' section shows the following XML snippet:

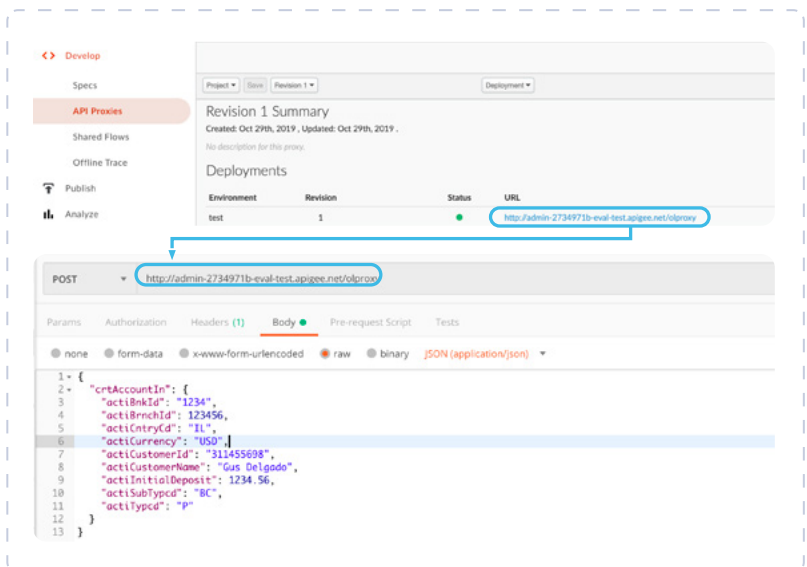
```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <ProxyEndpoint name="default">
3   <description/>
4   <faultRules/>
5   <preFlow name="PreFlow">
6     <request/>
7     <response/>
8   </preFlow>

```

## 10

Test the newly Apigee managed API from a service client, i.e. Postman

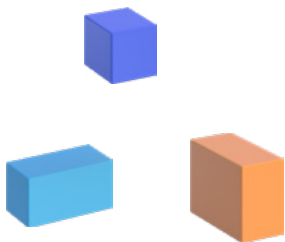


The screenshot shows the Postman interface for testing the API. The 'Revision 1 Summary' section shows the API was created on Oct 29th, 2019, and updated on Oct 29th, 2019. The 'Deployments' table shows a deployment in the 'test' environment with a status of 'Success' and a URL of `http://admin-2734971b-eval-test.apigee.net/olproxy`. Below this, the 'POST' request is shown with the URL `http://admin-2734971b-eval-test.apigee.net/olproxy`. The 'Body' tab is selected, showing the following JSON payload:

```

1 - {
2   "createAccountIn": {
3     "actiBankId": "1234",
4     "actiBranchId": "123456",
5     "actiEntryCd": "IL",
6     "actiCurrency": "USD",
7     "actiCustomerId": "311455698",
8     "actiCustomerName": "Gus Delgado",
9     "actiInitialDeposit": "1234.56",
10    "actiSubTypcd": "BC",
11    "actiTypcd": "P"
12  }
13 }

```



## About OpenLegacy

OpenLegacy's Digital-Driven Integration enables organizations with legacy systems to release new digital services faster and more efficiently than ever before. It connects directly to even the most complex legacy systems, bypassing the need for extra layers of technology. It then automatically generates APIs in minutes, rapidly integrating those assets into exciting new innovations. Finally, it deploys them as standard microservices or serverless functions, giving organizations speed and flexibility while drastically cutting costs and resources. With OpenLegacy, industry-leading companies release new apps, features, and updates in days instead of months, enabling them to truly become digital to the core.

