Draft Document for Review August 1, 2016 5:56 pm REDP-5350-00

Redbooks
ibm.com/redbooks

Getting Started with IBM API Connect: Scenarios Guide

Alex Seriy
Bhargav Perepa
Christian E Loza
Christopher P Tchoukaleff
Gang Chen
Ilene Seelemann
Kurtulus Yildirim
Rahul Gupta
Soad Hamdy
Vasfi Gucer

Cloud

Mobile

IBM

Redpaper

International Technical Support Organization

Getting Started with IBM API Connect: Scenarios Guide

July 2016

Note: Before using this information and the product it supports, read the information in “Notices” on page vii.

First Edition (July 2016)

This edition applies to IBM API Connect Version 5.0.

This document was created or updated on August 1, 2016.

Contents

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, MD-NC119, Armonk, NY 10504-1785, US

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided “AS IS”, without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks or registered trademarks of International Business Machines Corporation, and might also be trademarks or registered trademarks in other countries.

DataPower®	Redbooks®	WebSphere®
IBM®	Redpaper™	
Rational®	Redbooks (logo)  ®	

The following terms are trademarks of other companies:

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Other company, product, or service names may be trademarks or service marks of others.

Find and read thousands of IBM Redbooks publications

- Search, bookmark, save and organize favorites
- Get personalized notifications of new content
- Link to the latest Redbooks blogs and videos

Get the latest version of the Redbooks Mobile App

iOS **Download Now** Android

Promote your business in an IBM Redbooks publication

Place a Sponsorship Promotion in an IBM® Redbooks® publication, featuring your business or solution with a link to your web site.

Qualified IBM Business Partners may place a full page promotion in the most popular Redbooks publications. Imagine the power of being seen by users who download millions of Redbooks publications each year!

ibm.com/Redbooks
About Redbooks → Business Partner Programs

THIS PAGE INTENTIONALLY LEFT BLANK

Preface

IBM® API Connect is an API management solution from IBM that offers capabilities to create, run, manage and secure APIs and microservices, thus managing the full lifecycle of APIs for both on-premises and cloud environments.

This IBM Redpaper™ publication covers practical scenarios showing the API Connect capabilities for managing the full API lifecyle, for creating, running, security and managing the APIs. This book is targeted for users of an API Connect based API strategy, developers, IT architects and technical evangelists.

If you are not familar with APIs or AI Connect, we suggest that you read the following companion IBM Redpaper first: *Getting Started with IBM API Connect: Concepts, Architecture and Strategy Guide*, REDP-5349.

Authors

This paper was produced by a team of specialists from around the world working at the International Technical Support Organization, Austin Center.

Alex Seriy is an accomplished technologist and DevOps evangelist with over 25 years of experience under the belt. He is an automation enthusiast and usability guru, passionate about uncovering new use cases and innovating applications for platforms, frameworks and tools. At IBM, Alex is a subject matter expert in Continuous Integration, Delivery and Testing. He currently focuses on integration testing in the API Economy through service virtualization. Prior to joining IBM, Alex served as the Senior DevOps Architect at TIAA-CREF for 4 years. Before that, he was a consultant specializing in SCM, Release Engineering, QA and DevOps for over 10 years.

Bhargav Perepa has been with IBM for 22 years. He is an IBM IT Specialist/Architect, working with various US Federal Civilian and Department of Defense government agencies in the IBM Federal Software Group in the Washington, DC, area. He received his M.S. in Computer Science from Illinois Institute of Technology, Chicago IL, and an MBA from University of Texas at Austin, TX. His current interests are in cloud, analytics, mobile, social, security, and Watson technologies.

Christian E Loza iChris is a data scientist with a strong background in Natural Language Processing and Cognitive computing, with 15 years of experience in the industry. His area of expertise includes the development of analytic solutions in Big Data, cognitive solutions and cloud technologies. He currently works for the Public Sector. He worked previously supporting the Communications sector, developing solutions for the Telecommunications and Media and Entertainment industries. During this time, he worked developing multiple solutions including Audience Insights, Smarter Cities solution in collaboration with AT&T, Bluemix showcase solutions for the industry, and Watson for Network Operations.

Christopher P Tchoukaleff is a Consultant in the Hybrid Cloud Services team. He has 10+ years of Enterprise development and consulting experience in several different industries, and focuses on the MobileFirst platform as well as API Connect. You can reach Chris at cptchouk@us.ibm.com.

Gang Chen is an IBM Executive IT Specialist from IBM Cloud Architecture and Solution Engineering tem. He takes the leadership role in Developer Experience for Cloud adoption in areas of Microservices, API, Integration, Docker, Cloud Foundry, Bluemix and Programming Model.

Ilene Seelemann is a senior consultant for the IBM Bluemix Garage in Toronto, Canada where she works with clients to develop cloud solutions on the Bluemix platform. She specializes in API strategy and application security. Prior to this role, Ilene worked for over 20 years in software development at IBM, most recently on the API Management product. She holds a Masters degree in Mathematics from the University of Waterloo.

Kurtulus Yildirim is a Certified and Senior IT Specialist in IBM Cloud Unit, Turkey. He has 12 years of experience in application design, development and consulting. He holds a master degree on Software Management from Middle East Technical University. His areas of expertise include Change and Configuration, Requirement, Quality and API Management. At his current assignment, he works as an IBM Rational® consultant for various clients in Middle East and Africa Region.

Rahul Gupta is a Senior Software Architect in the IBM Watson Internet of Things group in Austin,TX. He is a Certified SOA Architect with 11 years of professional experience in IBM messaging technologies. Rahul has been a technical speaker for messaging related topics at various IBM conferences. He has also worked as a Services Architect with various IBM customers in North America. He has authored several IBM Redbooks® publications on messaging, mobile and cloud computing and is a also a recognized inventor by IBM innovation community.

Soad Hamdy is a Software Engineer in Cairo Technology Development Center, IBM Egypt. She has more than six years of experience in developing web and mobile applications. She obtained twelve product certificates like IBM Worklight, Oracle Java Business Component, IBM SOA Associate and Oracle Java Programmer. She also got many awards for her achievements like Duke's Choice Award and Manager Choice Award. She spoke at many events in different countries like Africa Technical Academy in Morocco, Cairo and South Africa and Kenya. She published many technical articles like *Getting started with MongoDB in Node.js*, *MongoDB for Beginners*, *Overview of IBM StrongLoop, Worklight Common errors, RPT in a nut shell, Getting started with android development and Android Google Maps*. She has BSc. degree in Computer Science.

Vasfi Gucer is an IBM Redbooks Project Leader with the IBM International Technical Support Organization. He has more than 18 years of experience in the areas of systems management, networking hardware, and software. He writes extensively and teaches IBM classes worldwide about IBM products. His focus has been on cloud computing for the last three years. Vasfi is also an IBM Certified Senior IT Specialist, Project Management Professional (PMP), IT Infrastructure Library (ITIL) V2 Manager, and ITIL V3 Expert.

A complete and detailed IBM Redpaper publication on a topic such as this would not be possible without generous support and guidance from key members in the IBM API Connect product and development organizations, as well as many other IBMers.

The authors team would like to acknowledge the following people for their contributions to this project:

Mohammed Abdula, Dennis Ashby, Roland Barcia, Rick Blalock, Vince Brunssen, Shane Clausen, Jim Colson, Jason Gartner, Marc E Haberkorn, Rob High, David K Hodges, Prasad Imandi, Stephen Kenna, David Kerr, Heather Kreger, Christian E Loza, Christopher F Markes, Tien Nguyen, Robert Sawyer, Benjamin C Wisnewski
IBM USA

Johan H Thole
IBM Netherlands

Dinesh G Shetty
IBM India

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our papers to be as helpful as possible. Send us your comments about this paper or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks
- Send your comments in an email to:

  redbooks@us.ibm.com
- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks
- Follow us on Twitter:

  http://twitter.com/ibmredbooks
- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806
- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm
- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

1

Introduction to API Connect and environment setup

This chapter provides an overview of the sample scenario that we will use to demonstrate the activities of both an API Provider and an API Consumer. The scenario illustrates how API Connect provides the foundation for building, securing, monitoring, publishing and consuming APIs to support a digital ecosystem.

This chapter also gives a brief high level overview of API Connect and provides instructions for setting up your local environment and the API Connect service in Bluemix so that you can follow along and try out the scenario described in this book.

This chapter has the following sections:

1.1 Introduction to IBM API Connect

IBM API Connect is an integrated creation, runtime, management and security platform for enterprise APIs and microservices. It enables you to build and test your microservices and APIs locally or in the cloud. API Connect is available as an on-premises offering, a SaaS offering through IBM Cloud Marketplace and as a Bluemix service. When provisioned from the Bluemix catalog, it provides integration points with Bluemix runtimes and security. The scenario in this IBM Redpaper demonstrates API Connect as a Bluemix service.

Note: For more information on API Connect, you can refer to IBM Redpaper *Getting Started with IBM API Connect: Concepts, Architecture and Strategy Guide.*

The API Connect Bluemix service has three interfaces:

- Developer Toolkit (API Designer console and command-line interface)
- API Manager console
- Developer Portal

1.1.1 Developer Toolkit

The Developer Toolkit is installed locally and provides an offline development experience. It is used for configuring, securing and testing APIs locally using the Micro Gateway.The Micro Gateway is a node.js based gateway built on StrongLoop technology. It provides a subset of the policies available in the DataPower Gateway. Once APIs are designed and tested, they can be published to an API Manager catalog using either the Designer console or the command line interface.

The Developer Toolkit is integrated with the LoopBack Framework. Developers can use LoopBack to implement REST APIs, test them locally and publish them to Bluemix. For a detailed overview of LoopBack, see Chapter 2. This paper describes a scenario that uses LoopBack to implement a REST API. If you already have existing REST or WSDL assets, you can use the API Designer to configure your APIs to directly connect to those existing assets.

1.1.2 API Manager Console

When provisioned through Bluemix, the API Manager console is launched from a Bluemix space. The API Manager console provides all of the API configuration and publishing capability in the API Designer, as well as:

- User management
- TLS security configuration
- REST over SOAP assembly
- API usage analytics
- Detailed catalog and product views
- Application subscription approval

1.1.3 Developer Portal

The Developer Portal is an entry point for consumers to browse and subscribe to APIs. The Developer Portal provides access to documentation, sample code for API invocations and

live "Try It" capability. Consumers get API credentials by registering applications in the Developer Portal. They can also monitor API usage.

The Developer Portal has an administrator interface that can be used by the API Provider Portal to customize the consumer's user experience. The Portal administrator can customize the theme for their business, create and control forums and customize content. For more information, see:

http://www.ibm.com/support/knowledgecenter/SSMNED_5.0.0/com.ibm.apic.devportal.doc/capim_devportal_admin.html

1.2 IBM API Connect sample scenario

This paper will use a sample application to walkthrough an end-to-end API lifecycle using API Connect. The scenario will include the following activities:

1. Implement an API as a node.js application using LoopBack
2. Secure API
3. Test API
4. Deploy API implementation (app) to Bluemix
5. Publish API to a Developer Portal for discovery and use by an API consumer
6. As an API consumer, sign up to use the API and develop an iOS and web application that invoke the API

Bob (the persona) downloads a mobile application called IBM Redbooks Mobile App A. This application built by Digital Agency Company A. The application lists amazing IBM's historical computers and computing devices. The list is provided by a third party API producer (e-Commerce Company B).

Bob browses through the list of inventory and interested in "Punched-card tabulating machines". He taps it and the Mobile application shows the detail about this item. The IBM Redbooks Mobile Application A pulls the Item/inventory detail such as images and description. It also pulls down user reviews and comments from another Third Party API provider - Social Sharing Provider Company C. (Bob believes other people using this website share the same interests as he does).

Bob really liked what he saw and decided to share some thoughts on Social Sharing Provider Company C, he tabs the button to write/share a comment. The Mobile application starts the OAuth flow provided by Social Sharing Provider Company C, Bob grants the (IBM Redbooks Mobile Application A access to his Social Sharing Provider Company C account, he then posted some amazing comments associated with this item.

Figure 1-1 on page 4 shows the high level architecture of this scenario.

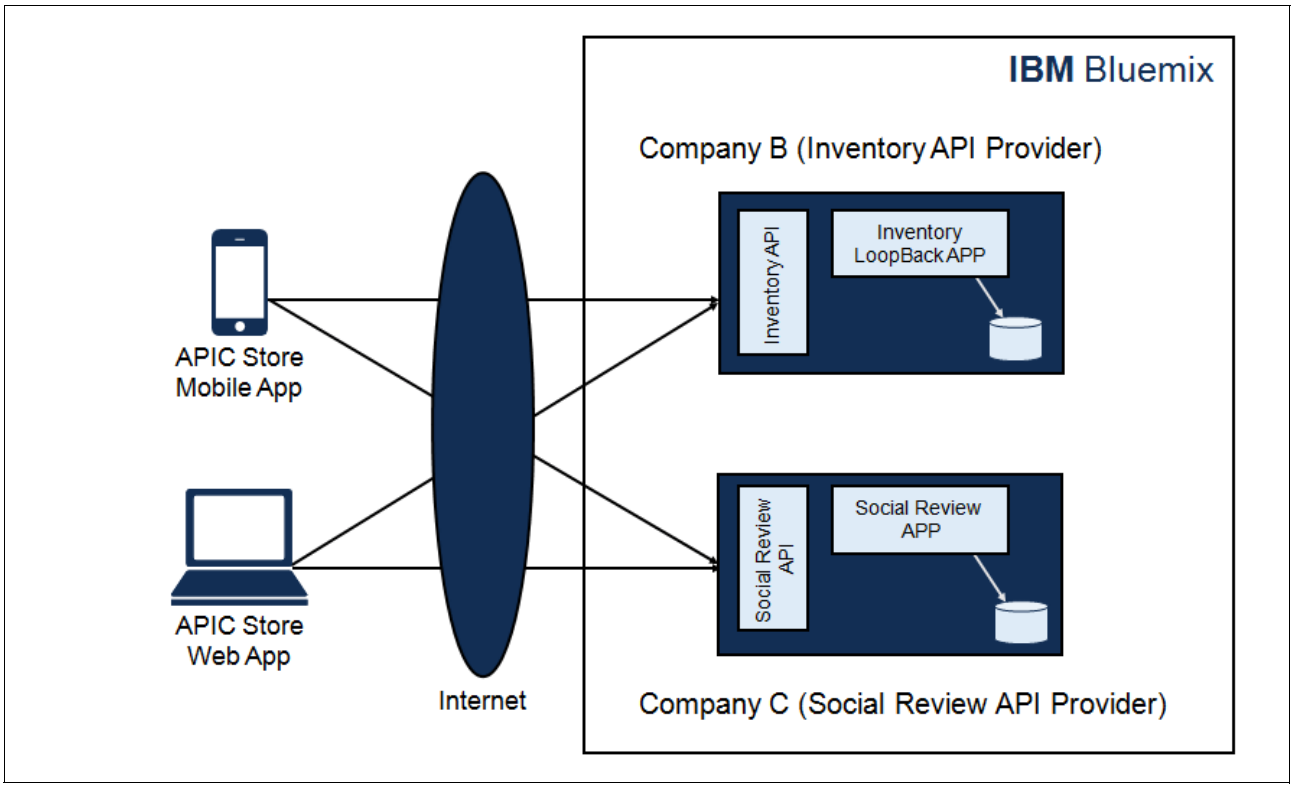

Figure 1-1 High level architecture of the sample scenario

1.3 Environment setup

Before we start creating a LoopBack application and API, we must first set up your environment by provisioning API Connect on Bluemix and installing the Developer Toolkit locally.

1.3.1 Provisioning the API Connect service in Bluemix

To provision API Connect, you must have a Bluemix account. Login to your Bluemix account or register for a new Bluemix account using the URL below:

`https://bluemix.net/registration`

Once logged in create a new space for hosting the sample application that we will create with LoopBack and for provisioning the API Connect service to manage the API.

Create a new space in Bluemix organization

Perform the following steps to create a new space in Bluemix organization:

1. Click on the **Bluemix account** in top right corner.
2. Create a new space.
3. Create a space with name IBM Redbooks.

Create a new API Connect service

Perform the following steps to create a new API Connect service:

1. Click on console and select **APIs** as shown in Figure 1-2.

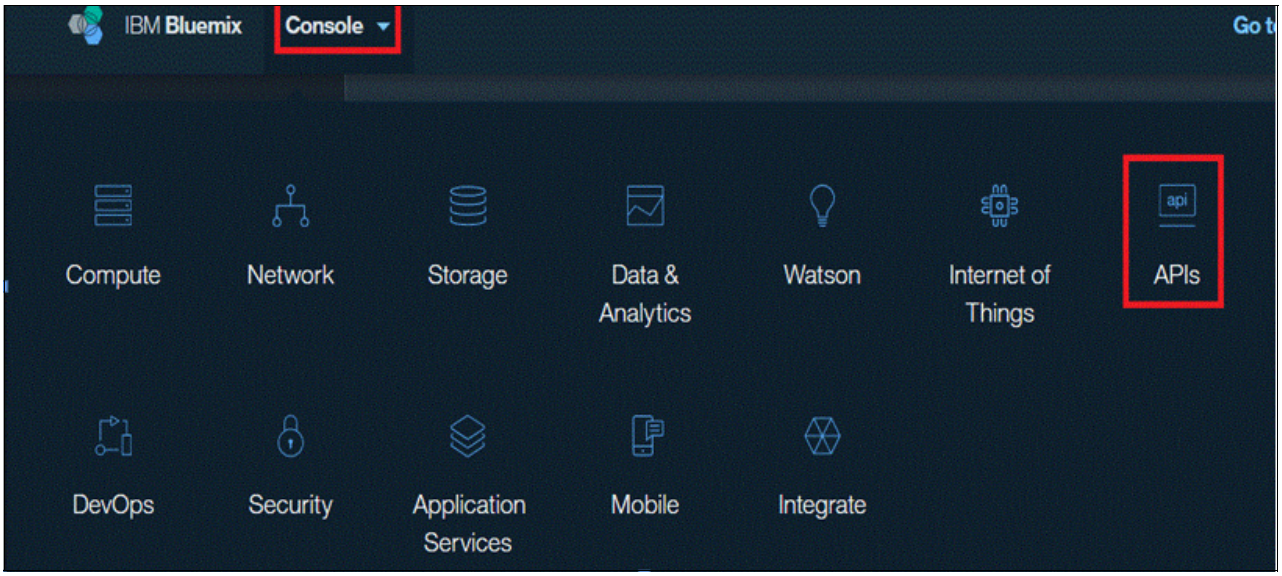

Figure 1-2 Select API

2. Select the **API Connect** service as shown in Figure 1-3 on page 6.

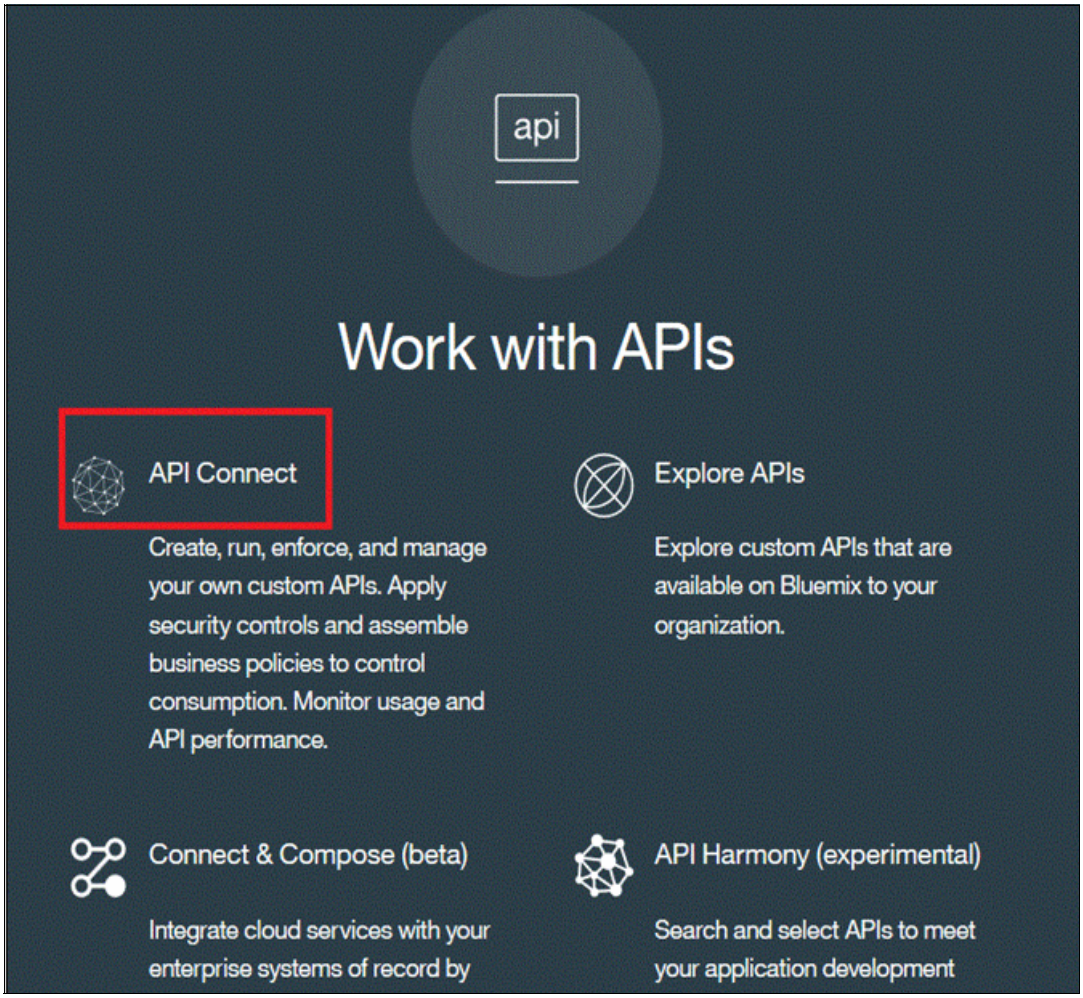

Figure 1-3 Select API Connect

3. On the next page click **Create**
4. Select the free **Essentials Plan and** Click **Create** again
5. Click on **Launch API Manager**. See Figure 1-4 on page 6.

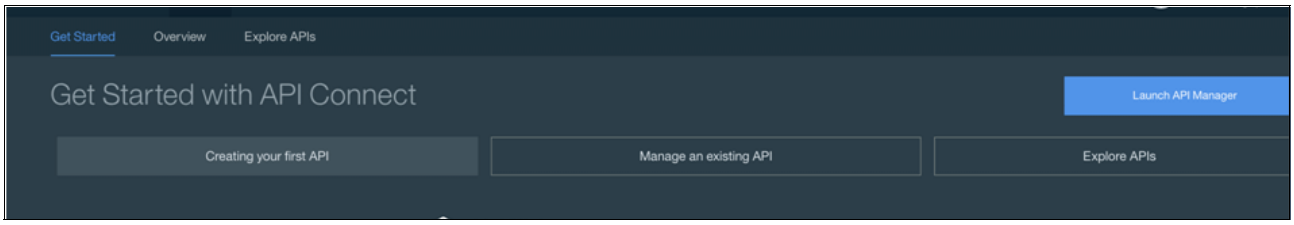

Figure 1-4 Get Started with API Connect panel

6. If prompted for your credentials, enter your Bluemix account userid and password.

   We will now create a catalog. A catalog contains a collection of API Products and is associated with a Developer Portal. An API Product contains one or more APIs and Plans. Plans can be used to group APIs and resources and to set rate limits. You might have a free plan with low rate limits and a chargeable plan with much higher rate limits and QoS characteristics. Plans are unique to an API Product, while APIs can be packaged by multiple Products. See Figure 1-5 on page 7.

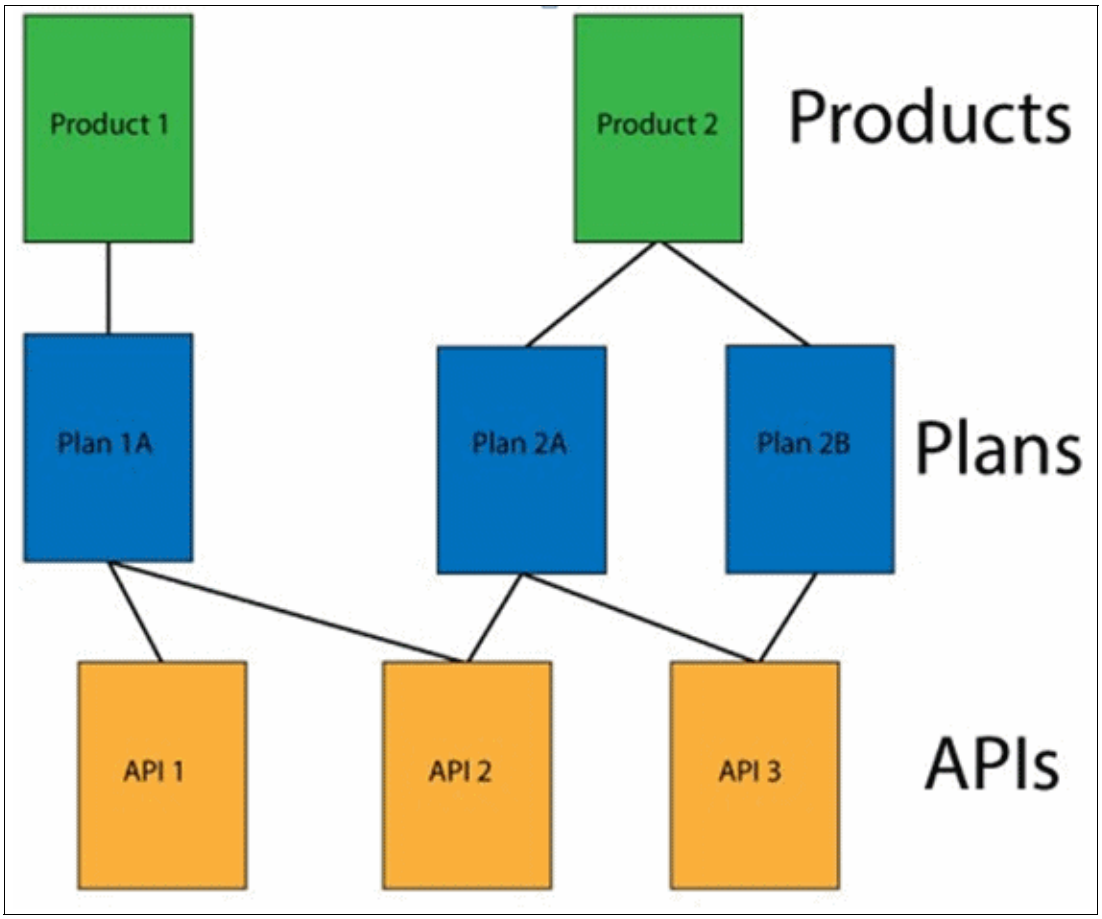

Figure 1-5 Products, Plans and APIs

When you publish an API Product to a catalog, that API Product becomes available on the Developer Portal associated with that catalog. An API Product can be published to multiple catalogs. You might have different catalogs for different consumers, such as one catalog for business partners and another for internal usage. You might also use different catalogs for continuous integration; for example, one catalog for preproduction activities such as quality assurance and test and another catalog for production use.

When someone signs up or is invited by the API Provider to a Developer Portal, a new Developer Organization is created and the new member becomes the owner of that Developer Organization. When publishing your API Products to a catalog or portal, you can configure which Developer Organizations can see each of your API Products and which Developer Organizations can subscribe to them. You can also make API Products visible and subscribable to the public to users who have not even authenticated or signed into the Developer Portal. An example of how you might use Developer Organizations would be to have a different Developer Organization for each Business Partner. This would allow you to control which API Products each business partner could see and subscribe to, and it would also isolate Business Partners from each other in terms of registered applications and usage data.

For our sample scenario, we will create a single catalog and call it ApicStore Catalog.

7. After launching the API Manager, navigate to the API Connect Dashboard and select **Add Catalog** at the top left. You will notice that a *Sandbox Catalog* has already been

automatically generated for you. That is OK. We're going to create a new one for our scenario.

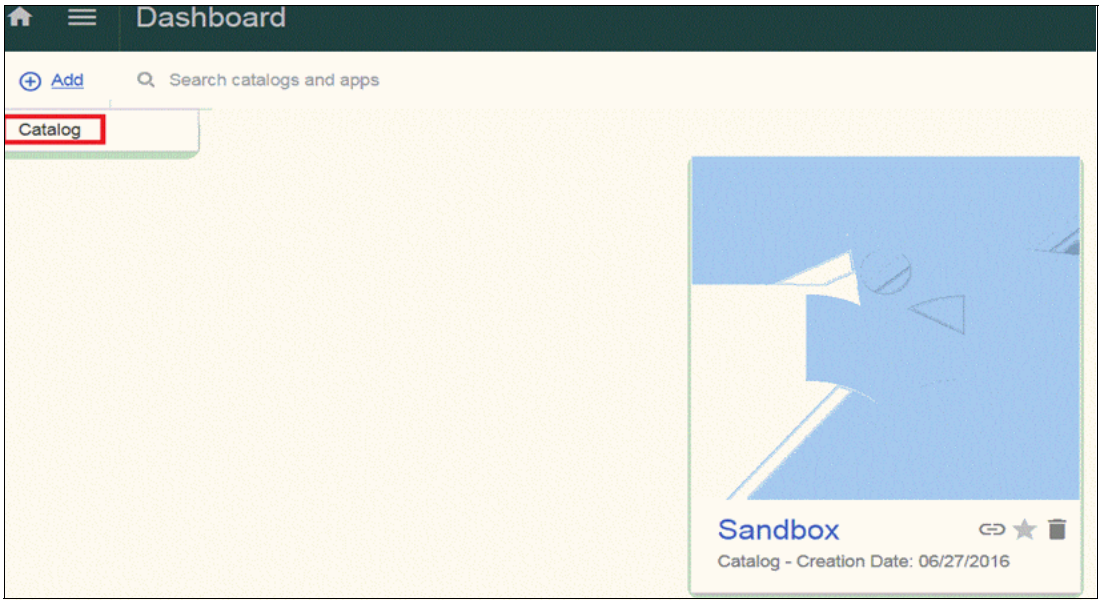

Figure 1-6 Create a catalog

8. Name the catalog ApicStore Catalog and press the **Add** button. See Figure 1-7.

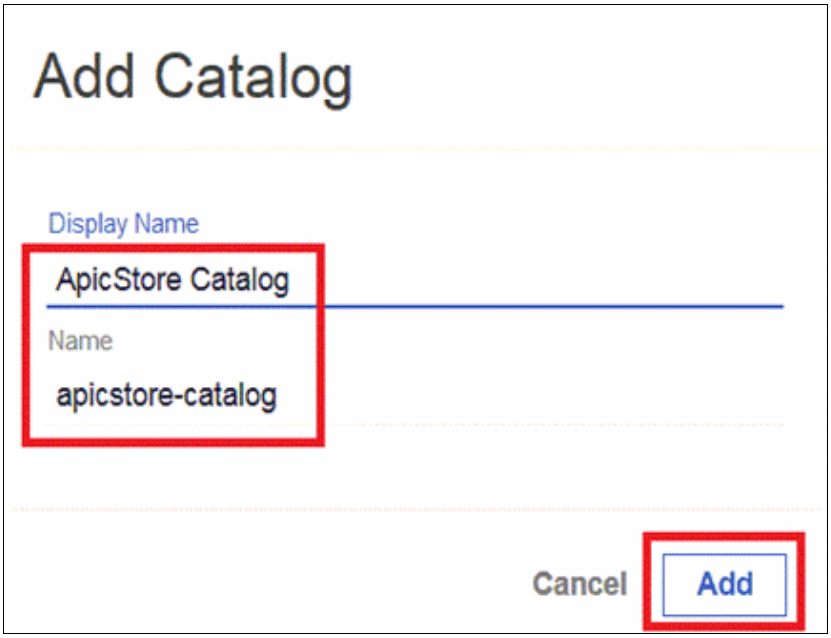

Figure 1-7 Add Catalog

9. Select the catalog and then the **Settings** tab and the **Portal** sub-tab, as shown in Figure 1-8 on page 9.

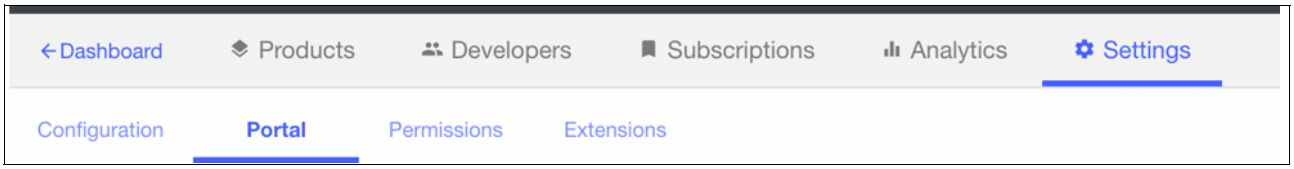

Figure 1-8 Settings and Portal tabs

10. To set up a Portal that your consumers can use to explore your APIs, select the **IBM Developer Portal** radio button. This will provision a Portal for you. You will get a message like the following. See Figure 1-9.

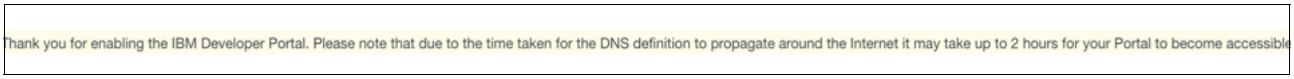

Figure 1-9 Provisioning a Portal

Once the new Developer Portal is created and active you will receive an email.

Tip: Make sure that you take note of your administrator password when you reset it the first time.

11. Keep the User Registry configured to the *default IBM ID*. Users that sign up to your Developer Portal will authenticate with their IBM ID (Bluemix account) credentials. Leave the remaining defaults as well. See Figure 1-10. Click **Save** after making these changes.

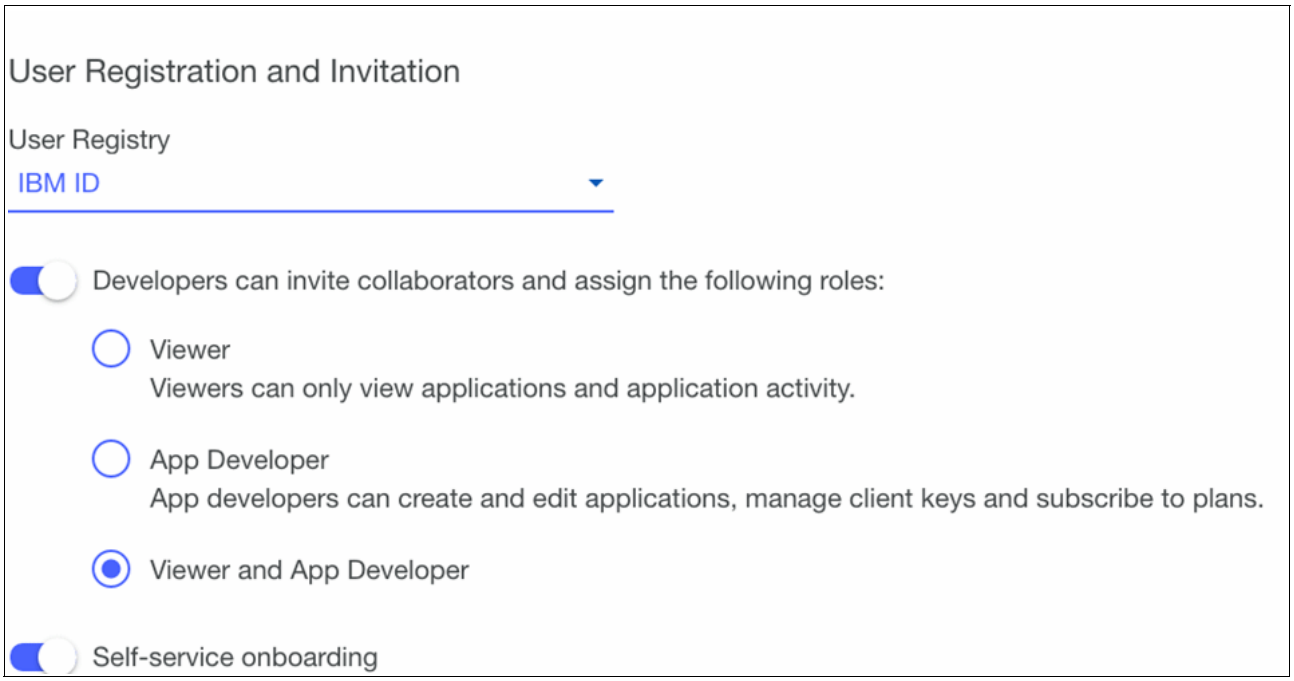

Figure 1-10 User Registration

1.3.2 Installing the IBM API Connect Developer Toolkit

Now that you have set up the API Connect service, you need to install the Developer Toolkit. Developers develop APIs and LoopBack applications by using the IBM API Connect Developer Toolkit. The IBM API Connect Developer Toolkit provides both the API Designer

user interface and a command line interface that developers could use to develop APIs and LoopBack applications and publish them to the IBM API Connect runtime.

Developers publish APIs by including them in a product and then publishing the product. Developers define their APIs and products by creating and validating YAML definition files in their local development environment. Then, they interact with IBM API Connect by using either the API Designer console or the toolkit command line interface.

The toolkit includes the following features:

- **API Designer**: A visual tool for creating, testing, and publishing APIs and applications.
- **LoopBack**: A highly-extensible, open-source Node.js framework for quickly creating dynamic end-to-end REST APIs.
- **Micro Gateway**: A gateway to support unit testing of policies to secure and enforce APIs as part of the local development experience.
- **The APIC CLI**: A set of commands to augment the local API create experience and for publishing APIs and applications to APIConnect clouds in Bluemix and on premises.

  IBM API Connect Developer Toolkit can be installed either from **npm** or from the management server in IBM API Connect cloud.The Knowledge Centre link below provides the steps to install the toolkit.

  http://www.ibm.com/support/knowledgecenter/SSFS6T/com.ibm.apic.toolkit.doc/tapim_cli_install.html

Note: Please install the prerequisites before installing the IBM API Connect Developer Toolkit.

1.3.3 Download the applications from GitHub

The LoopBack, security, web application and mobile applications developed through this redpaper are provided in the public GitHub repository for as-is use. Figure 1-11 below shows the different projects in this repository.

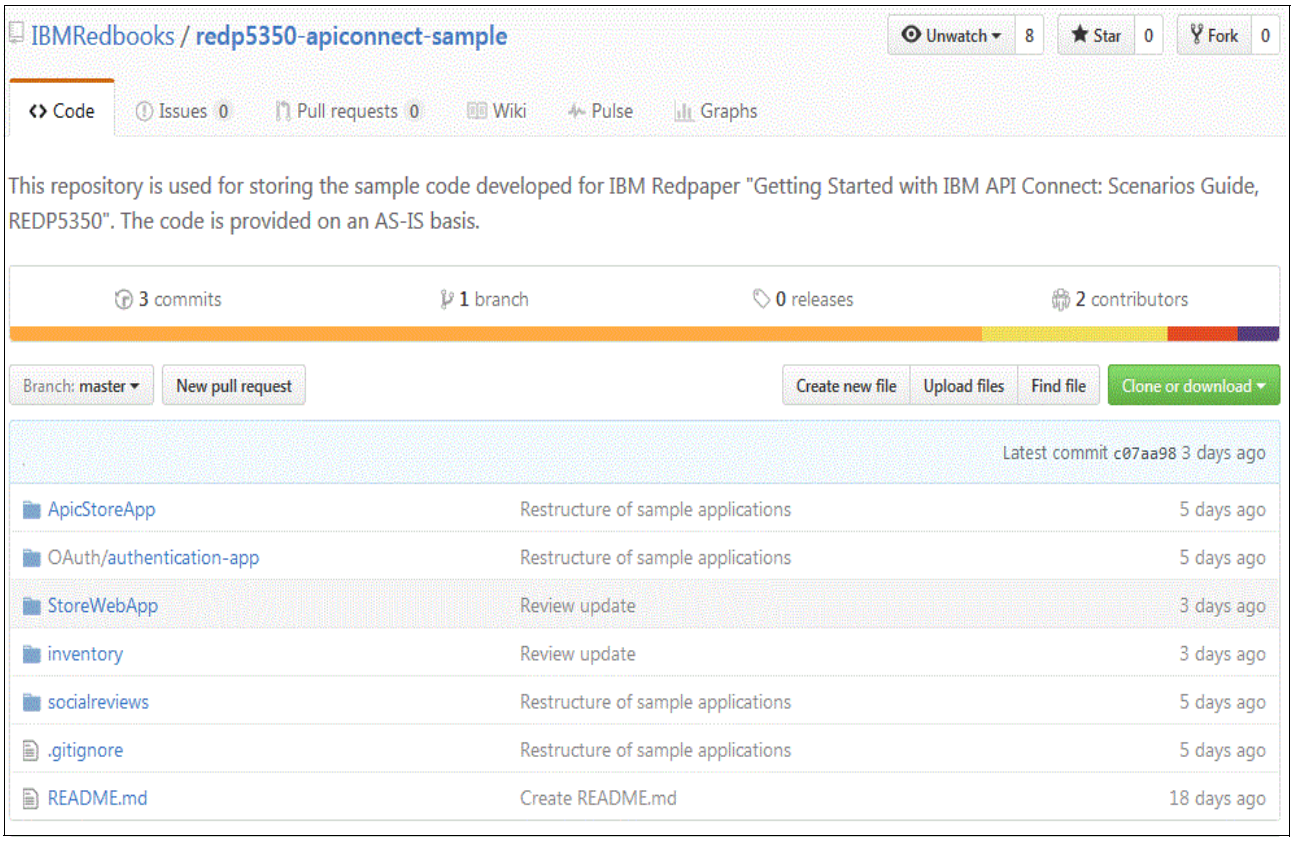

Figure 1-11 GitHub repository for IBM Redpaper REDP5350

This repository could be accessed using the following GitHub URL:

`https://github.com/IBMRedbooks/redp5350-apiconnect-sample`

Table 1-1 below lists the different projects in the GitHub repository.

Table 1-1 Projects in GitHub repository

Project	Details
inventory	Inventory LoopBack application project
socialreviews	Socialreviews LoopBack application project
Oauth	Oauth security project
StoreWebApp	Web app for accessing the items and reviews through APIs
ApicStoreApp	Mobile app for accessing the items are reviews through APIs

Clone the GitHub repository

To clone the GitHub repository use the **`git clone`** command as shows in the example below:

Example 1-1 Clone the redpaper repository to local

```
C:\Users\IBM_ADMIN> git clone
https://github.com/IBMRedbooks/redp5350-apiconnect-sample.git
Cloning into 'redp5350-apiconnect-sample'...
remote: Counting objects: 235, done.
```

```
          bjects:  89% (210/235), 748.01 KiB | 724.00 KiB/s
Receiving objects: 100% (235/235), 839.50 KiB | 724.00 KiB/s, done.
Resolving deltas: 100% (53/53), done.
Checking connectivity... done.
C:\Users\IBM_ADMIN> cd .\redp5350-apiconnect-sample
C:\Users\IBM_ADMIN\redp5350-apiconnect-sample [master]> dir

    Directory: C:\Users\IBM_ADMIN\redp5350-apiconnect-sample


Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
d----         7/11/2016   4:56 PM            ApicStoreApp
d----         7/11/2016   4:56 PM            inventory
d----         7/11/2016   4:56 PM            OAuth
d----         7/11/2016   4:56 PM            socialreviews
d----         7/11/2016   4:56 PM            StoreWebApp
-a---         7/11/2016   4:56 PM        529 .gitignore
-a---         7/11/2016   4:56 PM       1694 README.md
```

Note: Install Git client (https://git-scm.com/downloads) if it's not installed before cloning the code repository.

2

Developing and deploying LoopBack applications and APIs

This chapter discuses the LoopBack framework and how IBM API Connect uses LoopBack framework. It also provides a step by step procedure to develop and deploy the LoopBack applications for IBM API Connect. This chapter includes following sections:

2.1 What is LoopBack?

LoopBack is a highly extensible, open source Node.js framework that enables developers to:

- Create dynamic end-to-end REST APIs with model driven approach.
- Access data from major relational databases, MongoDB, SOAP and REST APIs.
- Incorporate model relationships and access controls for complex APIs.
- Separable components for file storage, third-party login, and OAuth 2.0.
- Easily create client apps using Android, iOS, and JavaScript SDKs.
- Run the application on-premises or in the cloud.

2.1.1 IBM API Connect and LoopBack

IBM API Connect is an end-to-end API management solution. It provides Node.js and Java microservice runtime for API implementation. API Connect has the Loopback framework built-in.

For more information refer the following website:

https://docs.strongloop.com/display/APIC/Using+LoopBack+with+IBM+API+Connect

For the sample application walkthrough, this chapter uses the LoopBack framework to build the microservices and expose them as consumable APIs.

2.1.2 LoopBack core concepts

In this section we introduce LoopBack core concepts.

LoopBack models

LoopBack models are at the heart of LoopBack application, and represent backend datasources such as databases or other back end services (REST, SOAP, and so on). LoopBack models are JavaScript objects with both Node and REST APIs. A key powerful feature of LoopBack is that when a developer defines a model it automatically comes with a predefined REST API with a full set of create, read, update, and delete operations. Every LoopBack application has a set of predefined built-in models such as User, Role, and Application, so a developer doesn't have to create these common models from scratch. Developers can define their own custom models specific to their application.

Figure 2-1 on page 15 shows LoopBack model inheritance. When a developer attaches a model to a persistent datasource it becomes a connected model with create, retrieve, update, and delete operations; LoopBack's built-in models inherit from it.

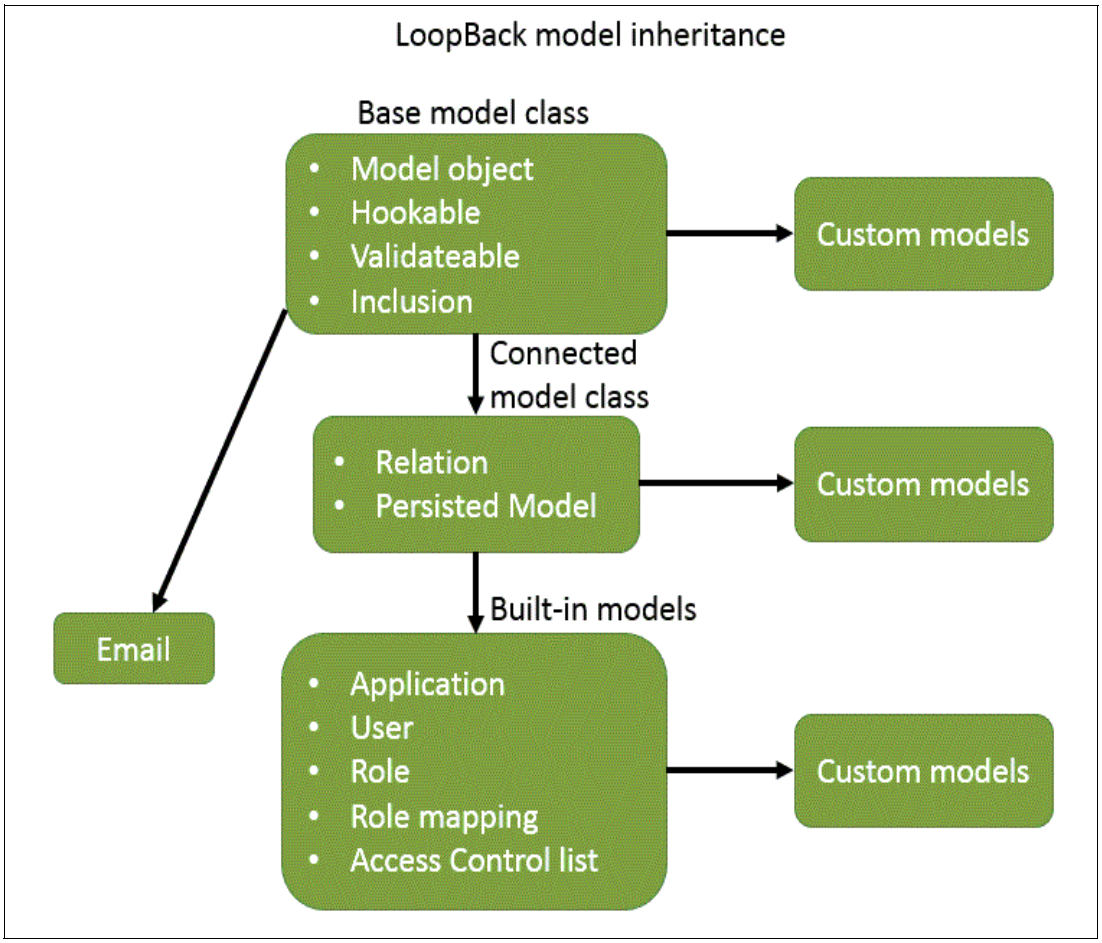

Figure 2-1 LoopBack model inheritance

Application logic

Developers can add custom application logic by using several techniques:

- Add application logic to models through remote methods (custom REST endpoints), remote hooks that are triggered by remote methods, and operation hooks that are triggered by model create, retrieve, update, and delete methods.
- Add boot scripts that run when the application starts.
- Define custom middleware, similar to Express middleware framework.

Read more about the Express framework through the website below:

`http://expressjs.com/`

Middleware phases

Middleware refers to functions executed when HTTP requests are made to REST endpoints. Since LoopBack is based on Express, LoopBack middleware is the same as Express middleware. However, LoopBack adds the concept of phases, to clearly define the order in which middleware is called. Using phases helps to avoid ordering issues that can occur with standard Express middleware.

Datasources and connectors

LoopBack generalizes backend services such as databases, REST APIs, SOAP web services, and storage services as datasources.

Datasources are backed by connectors that then communicate directly with the database or other back-end service. Applications don't use connectors directly, rather they go through datasources using the datasource and PersistedModel APIs.

Figure 2-2 shows the datasources and connectors in a LoopBack application.

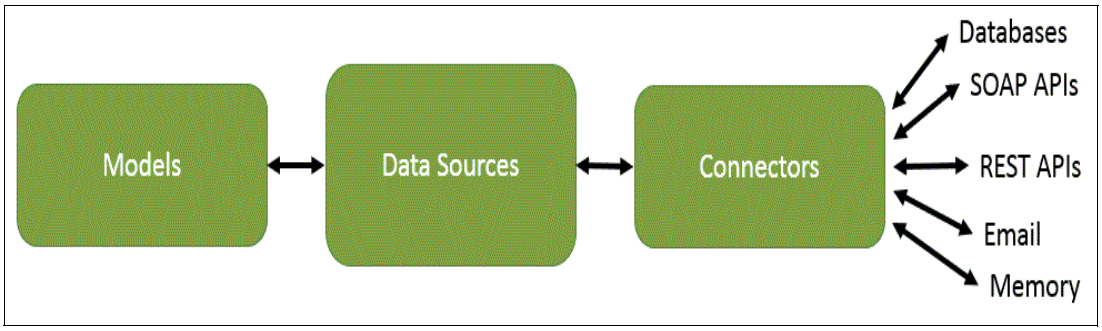

Figure 2-2 LoopBack datasources and connectors

2.2 IBM API Connect Developer Toolkit

In this section we will introduce IBM API Connect Developer Toolkit.

2.2.1 Installation of IBM API Connect Developer Toolkit

To get started with IBM API Connect the developer toolkit should be installed. Steps for installing API Connect Developer Toolkit are provided in 1.3.2, “Installing the IBM API Connect Developer Toolkit” on page 9.

Note: Please install the prerequisites before installing the IBM API Connect Developer Toolkit.

2.2.2 IBM API Connect Developer Toolkit command line interface

The API Connect Developer Toolkit delivers a command line interface named **`apic`** to augment the toolset developers use to create and test APIs to be run, managed, and secured by API Connect. The **`apic`** command set also provides capability to support devops oriented engineering tasks (e.g. continuous integration and delivery).

Example 2-1 on page 16 shows the various command line options for the **`apic`** command.

Example 2-1 apic command options

```
>apic -h

Usage: apic COMMAND OPTIONS

  Options
  -h, --help        command usage
```

```
  -v, --version     toolkit version

Commands (type apic COMMMAND -h for additional help):

  Creating and validating artifacts
    config          manage configuration variables
    create          create development artifacts
    edit            run the API Designer
    validate        validate development artifacts

  Creating and testing applications
    loopback        create and manage LoopBack applications
    microgateway    create Micro Gateway applications
    start           start services
    stop            stop services
    logs            display service logs
    props           service properties
    services        service management

  Publishing to the cloud
    login           log in to an IBM API Connect cloud
    logout          log out of an IBM API Connect cloud
    organizations   manage organizations
    catalogs        manage catalogs in an organization
    publish         publish products and APIs to a catalog
    products        manage products in a catalog
    apps            manage provider applications
    drafts          manage APIs and products in drafts
```

All the apic commands use a `command:action` syntax (for example **`apic apps:publish`**). For the most popular commands, either the command or action portion is optional to simplify usage. For example:

- **`apic auth:login`** → **`apic login`**
- **`apic local:create`** → **`apic create`**
- **`apic products:publish`** → **`apic publish`**
- **`apic products:list`** → **`apic products`**

All of the commands take a -h/--help flag which provides the command usage and a handful of useful examples (for example **`apic publish -h`**).

2.2.3 IBM API Connect Developer Toolkit API Designer

API Designer is the graphical design tool provided by the toolkit to support most of the capability available via the other command lines. Use the **`apic edit`** command as shown in Example 2-2 to run the API Designer tool:

Example 2-2 Command to start designer

```
apic edit
```

After the command in Example 2-2 is executed from the command line, it will open the designer tool in the web browser. Figure 2-3 shows the API Designer tool.

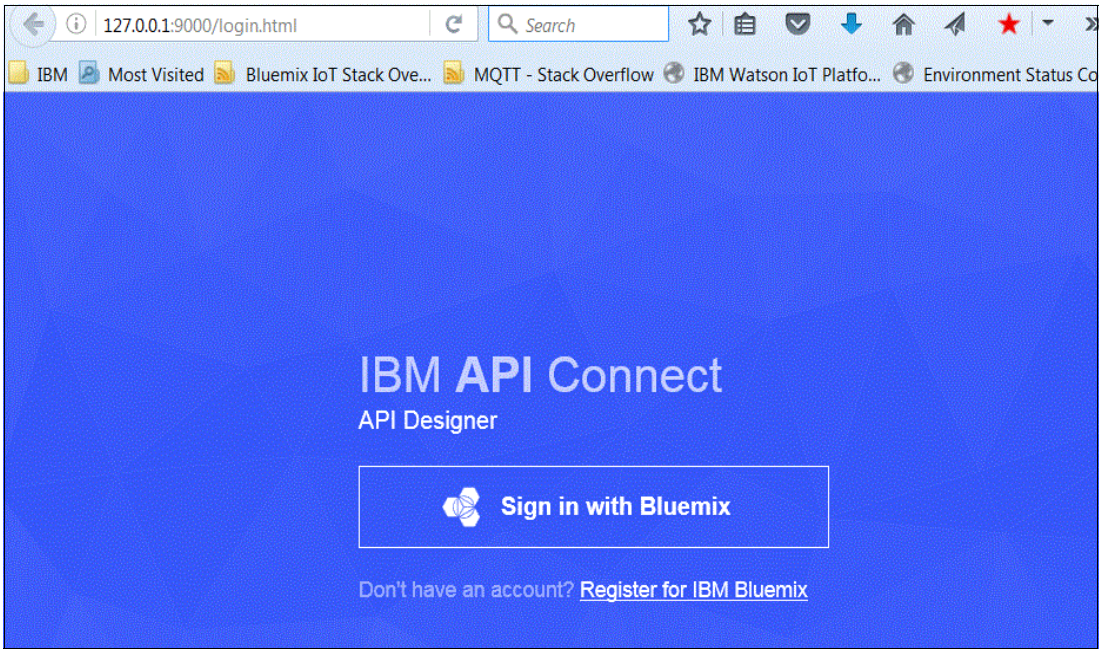

Figure 2-3 API Designer tool

Use the IBM Bluemix login credentials to login into the designer tool. Once logged in the process of developing products and API could be started. Figure 2-4 on page 18 shows the designer tool to add a new API product for a company.

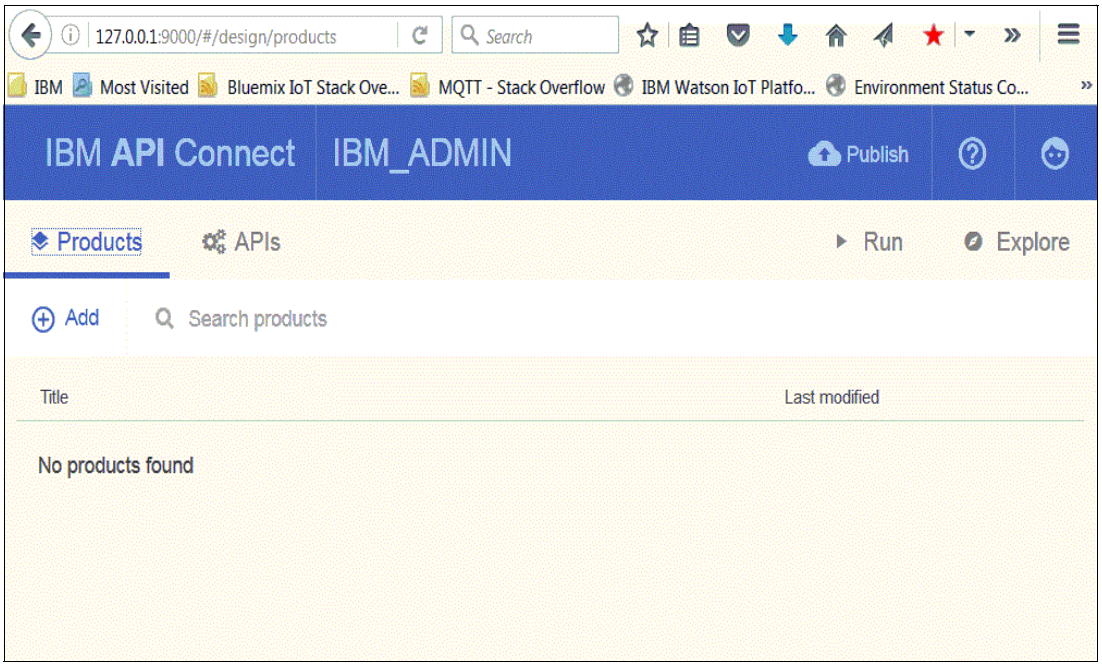

Figure 2-4 Add a new API product for a company

2.2.4 Products in the API Designer

In IBM API Connect, Plans and APIs are grouped together in Products, with which a company can manage the availability and visibility of APIs and Plans. Developers use the API Designer to create, edit, and stage a product, and the API Manager to manage the lifecycle of the

Product. Plans belong to only one Product, but they can possess different APIs to other Plans within the same Product, and they can share APIs with Plans from any Product.

Figure 2-5 on page 19 how Products, Plans, and APIs relate to one another. Products provide a method by which a company can group APIs into a package that is intended for a particular use. Additionally, they contain Plans, which can be used to differentiate between different offerings. Plans can share APIs, but whether subscription approval is required depends upon the Plan itself. Additionally, the offering company can enforce rate limits through these Plans, or through operations within the APIs of a Plan that override the rate limit of the Plan.

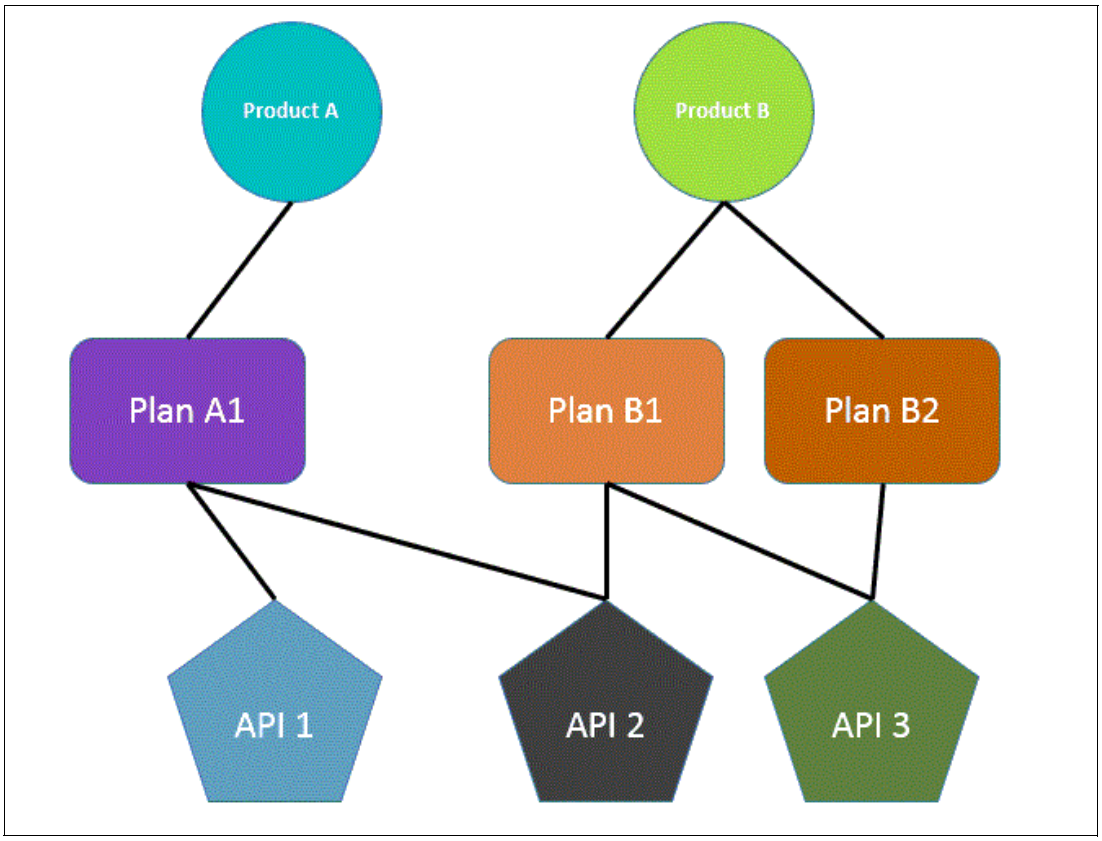

Figure 2-5 Products plans and APIs in API Designer

2.3 Sample application scenario

Bob (the end user) downloads a mobile application called IBM Redbooks ApicStore Mobile App A. This application is built by Digital Agency Company A. The application lists amazing historical computers and computing devices. The list is provided by a third party API producer (E-Commerce Company B).

Bob browses through the list/inventory and interested in "Punched-card tabulating machines". He taps it and the Mobile application shows the detail about this item. The IBM Redbooks Mobile App A pulls the Item/inventory detail such as images and description. It also pulls down user reviews and comments from another Third Party API provider - Social Sharing Provider Company C (Bob believes other people using this website share the same interests as he does).

We'll walk through building the LoopBack *Inventory* application developed by E-Commerce Company B and the *SocialReviews* application developed by social sharing provider Company C.

Figure 2-6 on page 20 shows the Inventory and the Social Reviews applications and APIs.

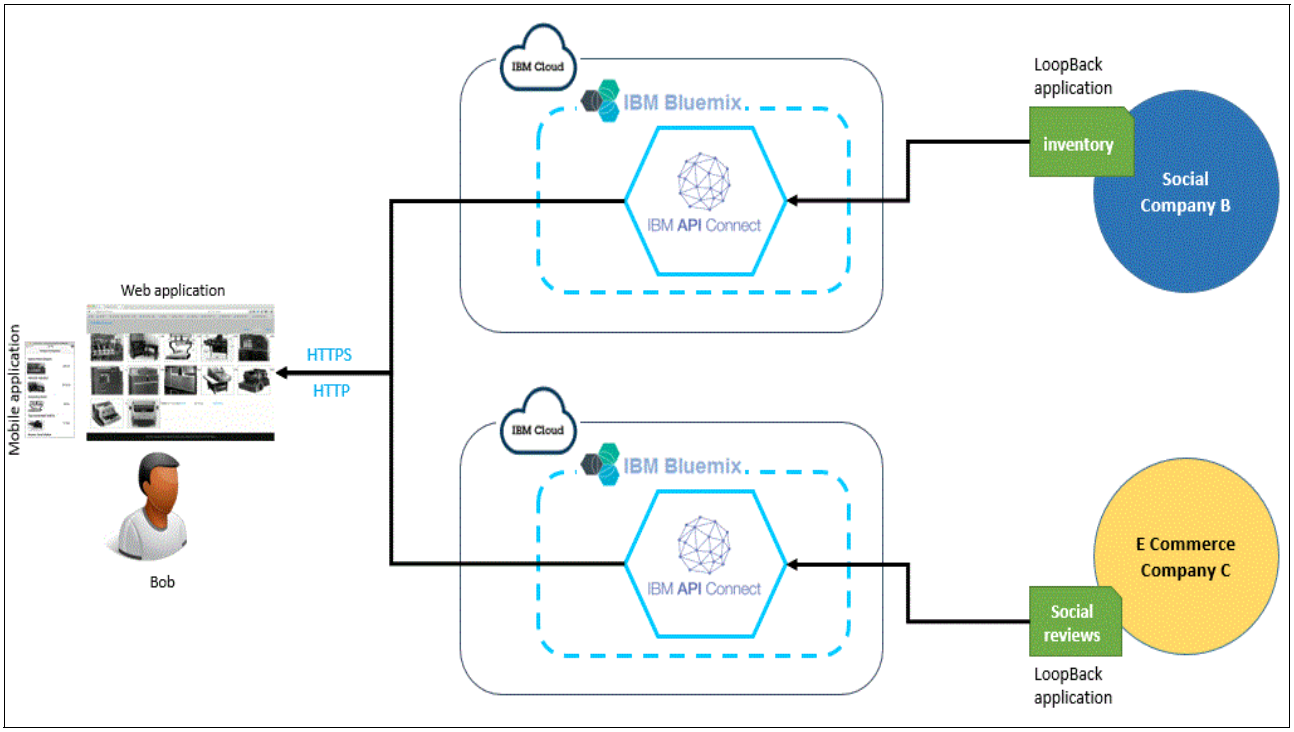

Figure 2-6 LoopBack application developed by Company B and Company C

2.4 Develop LoopBack applications and APIs

This section describes the step-by-step process to develop the API applications from company B and company C.

2.4.1 Inventory LoopBack application from Company B

Company B provides an *inventory* API product and has *inventory* APIs. The inventory API has two LoopBack models listed below:

- item - item model provides functionality to work with the individual items
- container - container model provides functionality to work with images of individual items

Note: You can skip the walk through of creating the LoopBack applications and could clone the inventory and socialreviews application from GitHub using the following git repository and can continue with 2.5, “Start and quickly test the LoopBack applications locally” on page 39.

https://github.com/IBMRedbooks/redp5350-apiconnect-sample.git

Steps to clone Github project repository are provided in 1.3.3, “Download the applications from GitHub” on page 10.

Use **npm install** command from command line to install the dependencies of the LoopBack project. This command should be executed from command line where the LoopBack project package.json file exist.

Create a LoopBack project

Create a LoopBack project following the steps below:

1. Create a directory with name APIConnect from command line.
2. Change the directory to APIConnect.
3. Create the LoopBack application by executing the **api** command.

   ```
   apic loopback -n <applicationnname>
   ```

   Example 2-3 shows steps to create the inventory application from companyB.
4. Input the application name as `inventory`, and directory name also as `inventory`.
5. Choose an empty server application.
6. Choose the LoopBack version 3.x.

Example 2-3 Inventory LoopBack application from CompanyB

```
C:\APIConnect>apic loopback -n inventory

     _-----_
    |       |    .--------------------------.
    |--(o)--|    |  Let's create a LoopBack |
   `---------'   |       application!       |
    ( _'U`_ )    '--------------------------'
    /___A___\
     |  ~  |
   __'.___.'__
 ' `  `  |° ' Y `

? What's the name of your application? inventory
? Enter name of the directory to contain the project: inventory
   create inventory/
     info change the working directory to inventory

? What kind of application do you have in mind? empty-server (An empty LoopBack
API, without any configured models or datasources)
? Which version of LoopBack would you like to use? 3.x
Generating .yo-rc.json

I'm all done. Running npm install for you to install the required dependencies.
If this fails, try running the command yourself.
```

```
    create .editorconfig
    create .eslintignore
    create .eslintrc
    create server\boot\root.js
    create server\middleware.json
    create server\middleware.production.json
    create server\server.js
    create .gitignore
    create client\README.md

Done running loopback generator

Updating swagger and product definitions
Created C:\APIConnect\inventory\definitions\inventory.yaml swagger description
Created inventory-product.yaml product definition [inventory:1.0.0]
Next steps:
  Change directory to your app
    $ cd inventory

  Create a model in your app
    $ apic create --type model

  Compose your API, run, manage, enforce and deploy it with API Connect
    $ apic edit

  Run the app
    $ apic start
```

7. From command line change the work directory to inventory.

Create datasource and models

A LoopBack model represents data in backend systems such as databases, and by default has both Node and REST APIs. Additionally functionality for validation rules and business logic could be added to the LoopBack models. Inventory application has two models item and container. LoopBack models can manipulate data via the datasource object.

A datasource enables a model to access and modify data in backend system such as a relational database. Datasources encapsulate business logic to exchange data between models and various back-end systems such as relational databases, REST APIs, SOAP web services, storage services, and so on. Datasources generally provide *create, retrieve, update, and delete (CRUD)* functions.

Item model

Item model represents data in backend for storing the inventory of computer items from companyB. The item model will also create the REST API for working with item data. Table 2-1 shows the properties of item model.

Table 2-1 Properties in item model

Property name	Property type	Description	Required	ID
id	number	Item Id	Yes	Yes
name	string	Item name	Yes	NA
description	string	Item description	Yes	NA

Property name	Property type	Description	Required	ID
price	number	Item price	Yes	NA
rating	number	Item rating	No	NA
img_alt	string	Item image title	Yes	NA
img	string	Item image location	Yes	NA

This model uses in-memory database. Data for in-memory database is persisted in flat file on the local storage of application. Follow steps below to configure the in-memory datasource for item model.

1. On command line change the directory to APIConnect/inventory.
2. Create a directory with name `db`.
3. Create a file inside db directory with name `memdb.json`.
4. Get back to directory inventory.
5. Execute command shown in Example 2-4 on page 23 to create a datasource called memdb.
6. Provide the datasource name as `memdb` and select the **in-memory** datasource.
7. Leave the window.locastorage blank and press Enter.
8. Provide full path to file as `db/memdb.json`.

Example 2-4 Create memdb datasource for item model

```
C:\APIConnect\inventory>apic create --type datasource
? Enter the data-source name: memdb
? Select the connector for memdb: In-memory db (supported by StrongLoop)
Connector-specific configuration:
? window.localStorage key to use for persistence (browser only):
? Full path to file for persistence (server only): db/memdb.json
Done running loopback generator

Updating swagger and product definitions
Created C:\APIConnect\inventory\definitions\inventory.yaml swagger description
```

Following the steps above creates the in-memory datasource for item model. Steps below now would help create the item model using the user interface of API Connect Toolkit.

9. To start the Designer toolkit, use the command shown in Example 2-5 below.

Example 2-5 Start API Connect Toolkit

```
C:\APIConnect\inventory>apic edit
Express server listening on http://127.0.0.1:9000
```

This will open the Designer toolkit application in browser. Login using your IBM Bluemix credentials as shown in 2.2.3, “IBM API Connect Developer Toolkit API Designer” on page 17.

Note: API Connect Toolkit Designer tool could now also be accessed using the url below:

`http://127.0.0.1:9000/#/design/apis`

10. Click on the **Models** tab in the user interface.
11. Click on the **Add** button to create a new model.
12. Create a new LoopBack model with name `item` and then click the **New** button as shown in Figure 13.

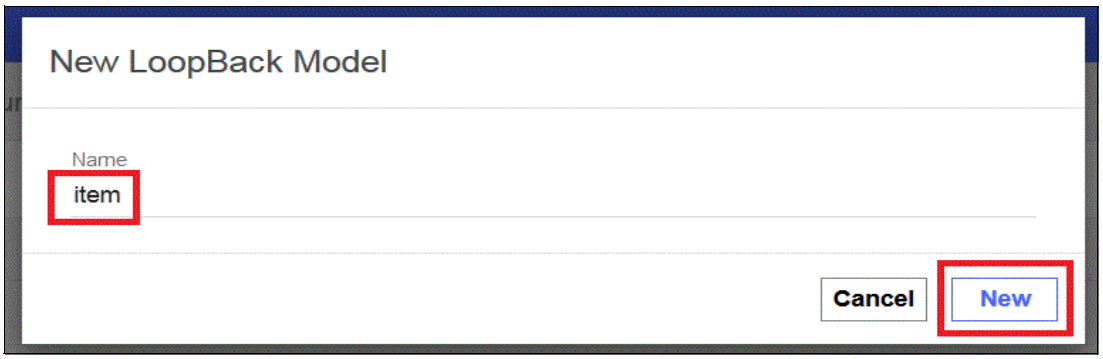

Figure 2-7 Create a new model name item

13. Now add new properties in this model referencing the properties listed in Table 2-1 on page 22.

    Figure 2-8 on page 24 shows the properties created in item model

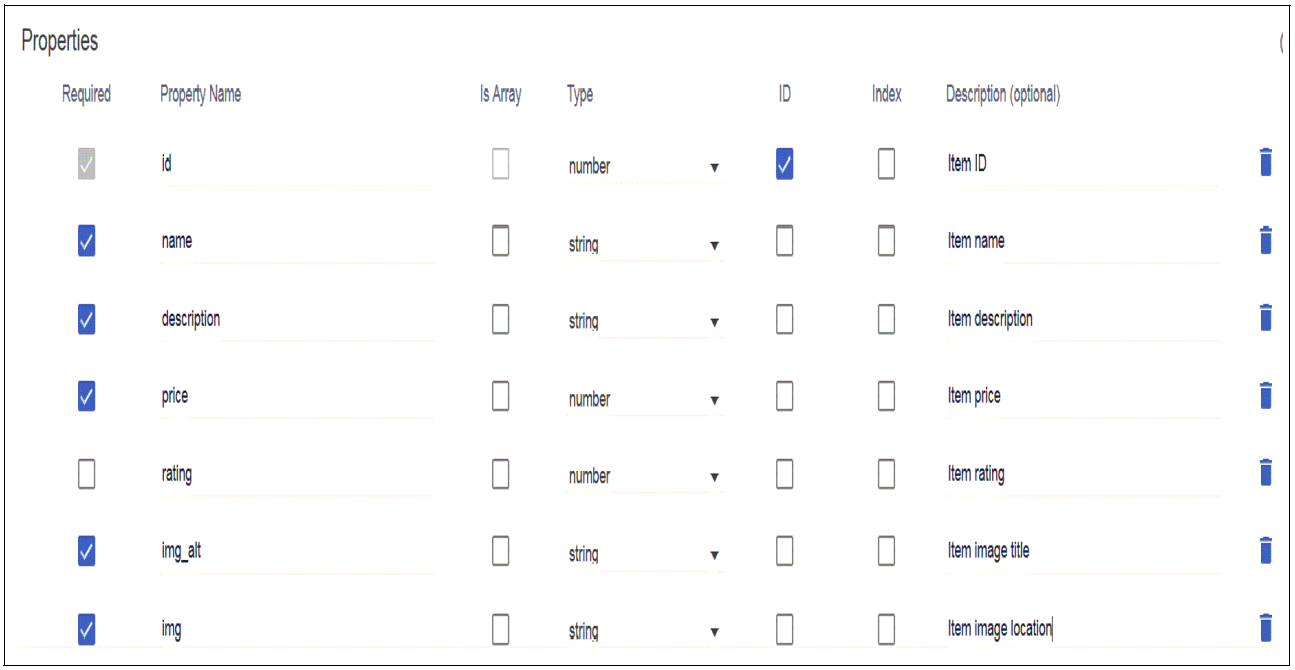

Figure 2-8 Properties of item model

14. In item model set the datasource to use the in-memory datasource memdb as shown in Figure 2-9.

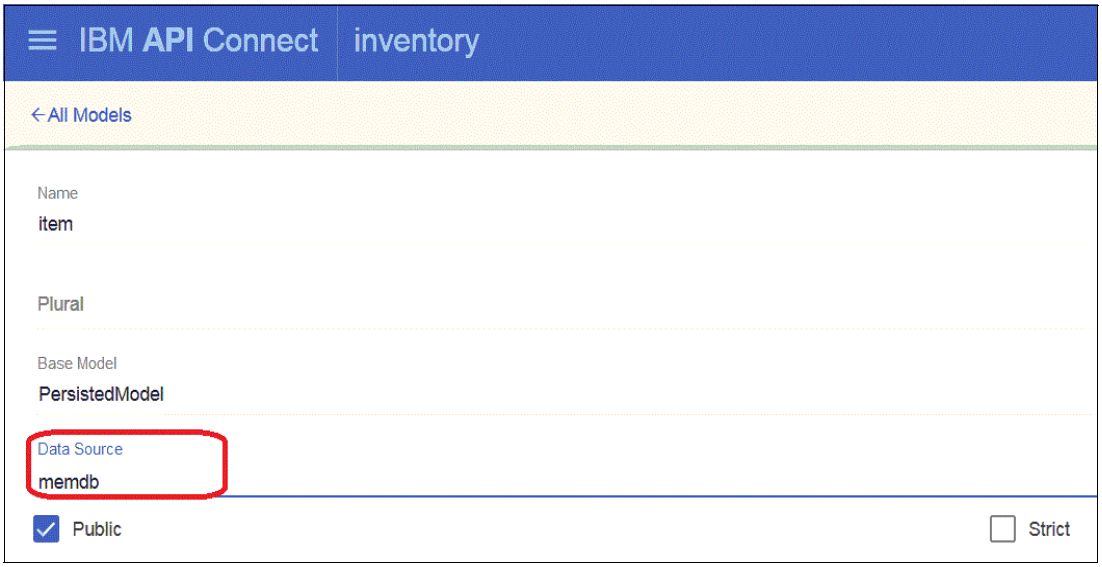

Figure 2-9 Set the item model to use the in-memory datasource

15. Click the **Save** button on top right corner to save the changes made to the item model.

Container model

Container model is used for storing item image files locally and then exposing through REST API. This model uses the storage connector and uses the l*oopback-component-storage* module. Website below provide more details on using the storage connector.

`https://docs.strongloop.com/display/APIC/Storage+connector`

Follow steps below to implement the storage connector datasource and the configure the container model.

1. Change directory to APIConnect/inventory/server.
2. Create a directory called storage.
3. Change directory storage.
4. Create directory items under storage.
5. Copy all the item image files under this directory as shown in Figure 2-10.

Note: Item images are available for download from the IBM Redbooks FTP server. Check the instructions in Appendix B, “Additional material” on page 129 to download the item images.

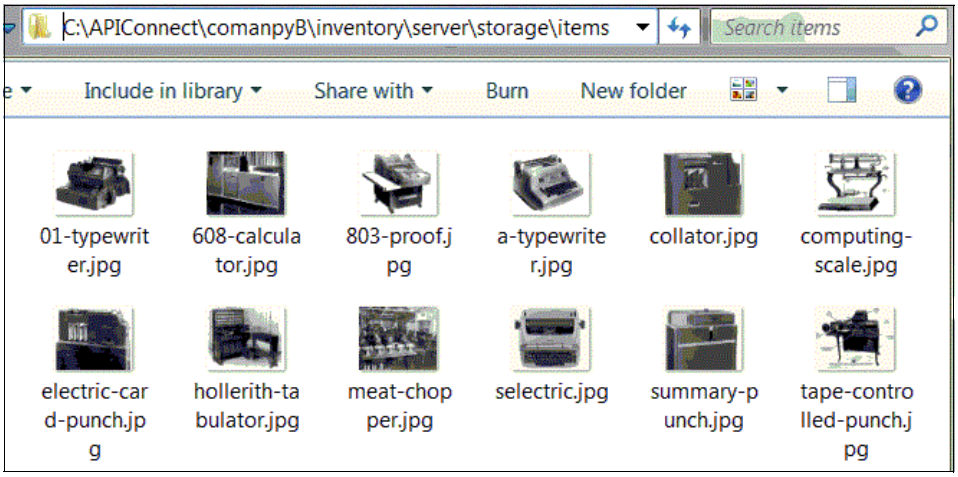

Figure 2-10 Item images in storage for container model

6. Change directory to APIConnect/inventory.
7. Create datasource storage using the command shown in Example 2-6.

Example 2-6 Create storage datasource for container model

```
c:\APIConnect\inventory>apic create --type datasource
? Enter the data-source name: storage
? Select the connector for storage: other
? Enter the connector name without the loopback-connector- prefix:
loopback-component-storage
? Install loopback-component-storage (Y/n) Y

Done running loopback generator

Updating swagger and product documentation
```

8) Update the storage datasource JSON file to update the provider and root properties of the datasource. Edit the *datasources.json* file in the server folder like shown in Example 2-7 below.

Example 2-7 Update datasources.json

Before editing

```
{
  "memdb": {
    "name": "memdb",
    "localStorage": "",
    "file": "db/memdb.json",
    "connector": "memory"
  },
  "storage": {
    "name": "storage",
    "connector": "loopback-component-storage"
  }
}
```

After editing

```
{
  "memdb": {
    "name": "memdb",
    "localStorage": "",
    "file": "db/memdb.json",
    "connector": "memory"
  },
  "storage": {
    "name": "storage",
    "connector": "loopback-component-storage",
    "provider": "filesystem",
    "root": "./server/storage"
  }
}
```

8. Create the container model using the command shown in Example 2-8 below. Use the connector as *loopback-component-storage*. npm will install this module in node_modules.

Example 2-8 Create container model

```
c:\APIConnect\inventory>apic create --type model
? Enter the model name: container
? Select the data-source to attach container to: storage
(loopback-component-storage)
? Select model's base class Model
? Expose container via the REST API? Yes
? Custom plural form (used to build REST URL):
? Common model or server only? common
Let's add some container properties now.

Enter an empty property name when done.
? Property name:
Done running loopback generator
Updating swagger and product definitions
Created c:\APIConnect\inventory\definitions\inventory.yaml swagger description
```

Configure the inventory product

To configure the inventory product through API Designer toolkit perform the following steps:

1. Click on **Products** tab in the designer. Figure 2-11 shows the inventory product. Configure the plans and provider details of inventory product by clicking on the product.

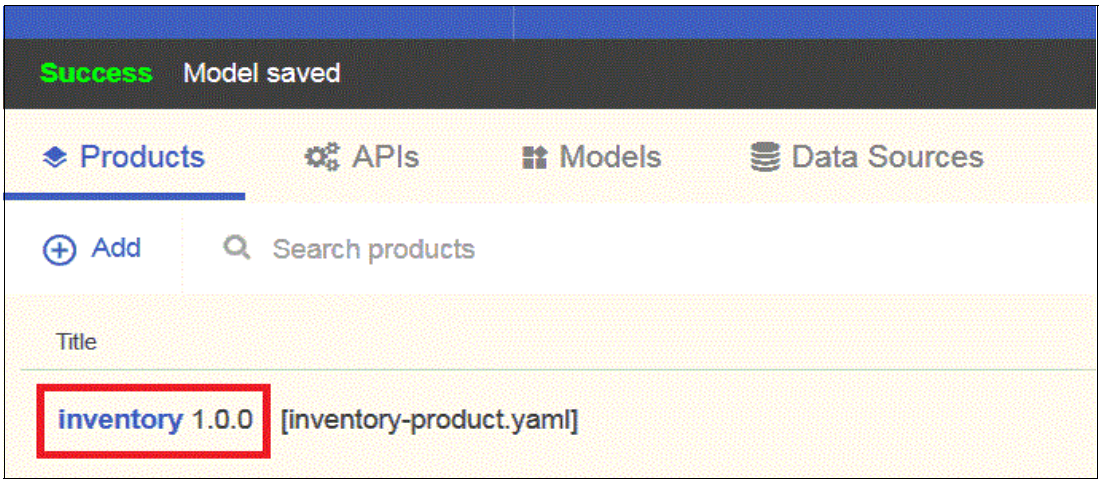

Figure 2-11 Inventory product

2. Configure the inventory product using the property values listed in Table 2-2.

Table 2-2 Inventory product design details

Inventory product design details		
Info	Title	inventory
	Name	inventory
	Version	1.0.0
	Description	inventory application from CompanyB
Contact	Name	Admin
	Email	admin@companyb.com
	URL	http://www.companyb.com
License	Name	companyB license
	URL	http://www.companyb.com
Terms of Service	Term of Service	companyB terms of service
Visibility	Visible to	Public
	Subscribable by	Authenticated users
API		inventory 1.0.0
Plans	Silver-Plan	1000 invokes/hr (Hard Limit Enforced)
	Gold-Plan	unlimited access

3. Save the product configuration after making all the changes.

Configure inventory API

Inventory application has two models; item and container. Both models expose the related API under the inventory API. We will be using following API (see Table 2-3 on page 29) in the application and other default API for these models could be deleted.

Table 2-3 Item and container model APIs

Model	Path	Operation	Description
Item	/items	POST	Create a new item
Item	/items	GET	Get all the items
Item	/items/{id}	PUT	Update one item
Item	/items/{id}	DELETE	Delete one item
Item	/items/{id}	GET	Get one item
Container	/containers/{container} /download/{file}	GET	Gets the image file for item from storage

1. Navigate to inventory API by clicking on **inventory 1.0.0**
2. Disable **clientSecretHeader** in the security section as shown in Figure 2-12 below.

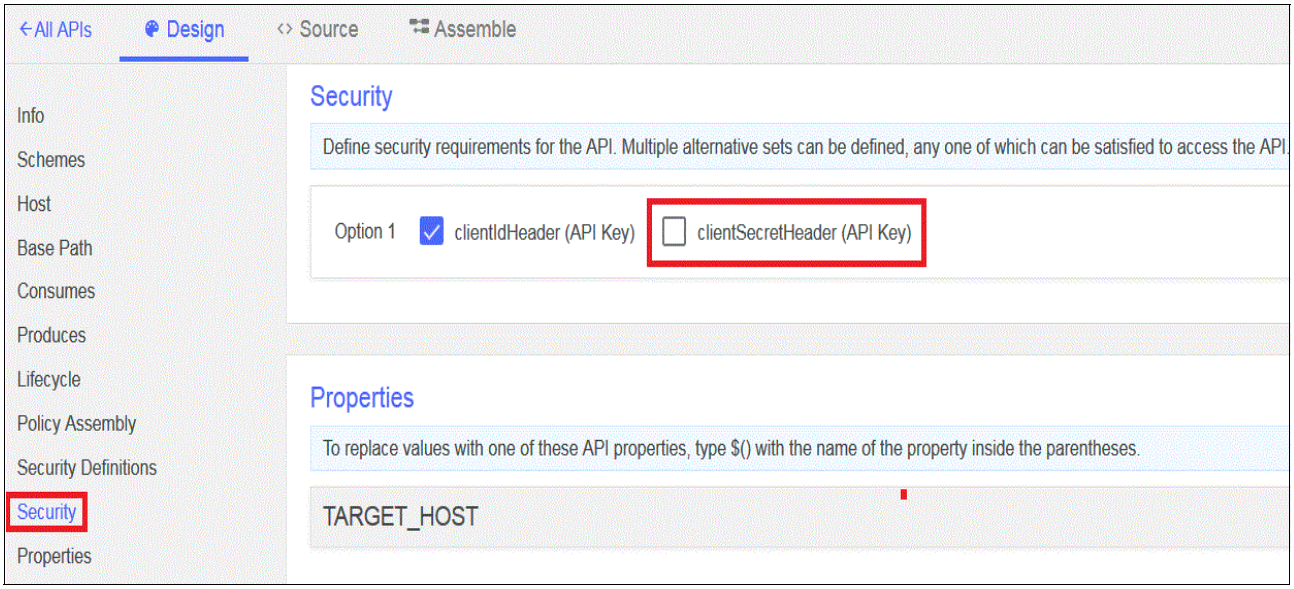

Figure 2-12 Disable clientSecretHeader in inventory API

3. Create a new property called TARGET_HOST. This will be the property to tell the assembly where the application will be actually deployed. This property must be changed before the actual deployment to Bluemix.

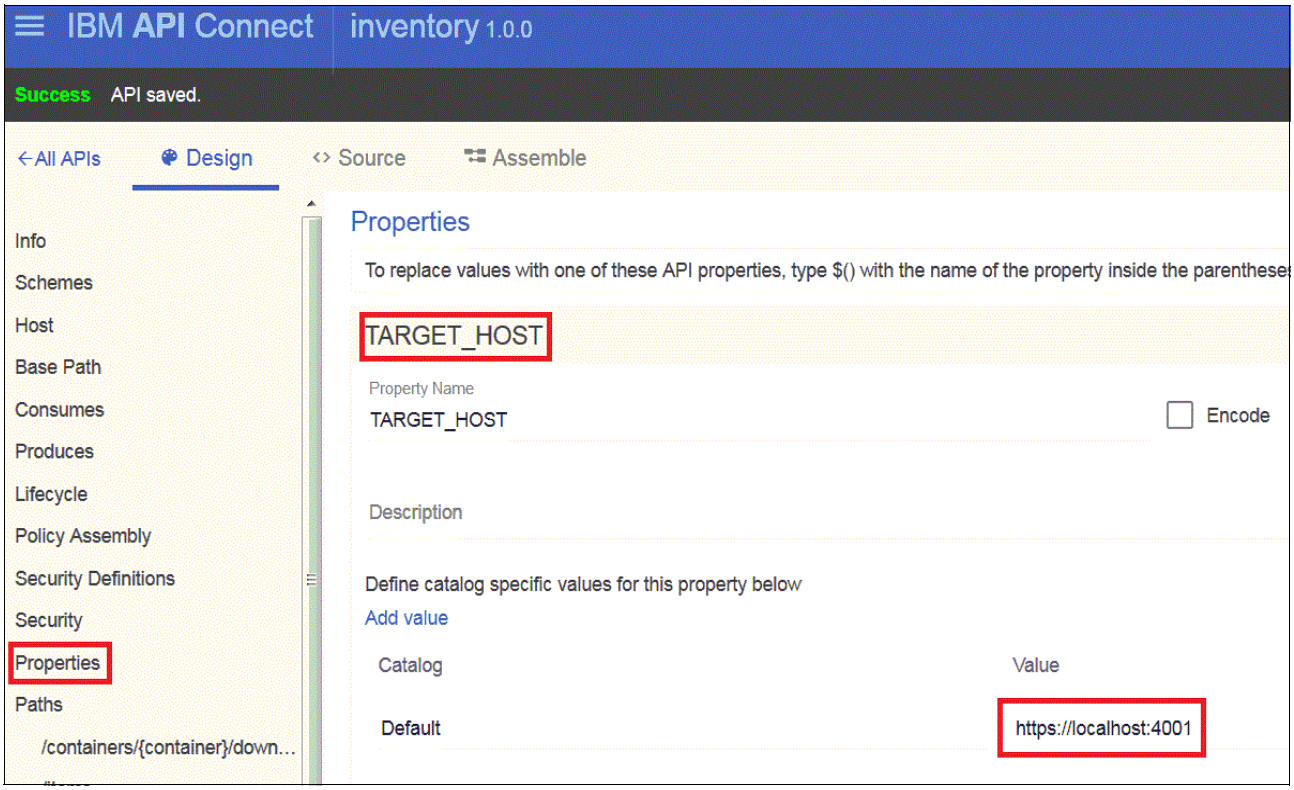

Figure 2-13 TARGET_HOST property for API assembly

4. This completed the creation of inventory LoopBack application from CompanyB.
5. Stop the APIConnect development toolkit from command line using ***Ctrl+C*** keys.

2.4.2 Social Reviews LoopBack application from Company C

Company C provides *socialreviews* product and has *socialreviews* APIs. The socialreviews API have one LoopBack model named review. Review model provides functionality to work with individual review for an item.

Create a LoopBack project

Create a LoopBack project following the steps below:

1. Change the directory to APIConnect.
2. Create LoopBack application by executing the api command:

   ```
   apic loopback -n <applicationn name>
   ```

3. Example 2-9 shows steps to create the socialreviews application from companyC
4. Input the application name as socialreviews, directory name as socialreviews.
5. Choose an empty server application.
6. Choose the LoopBack version 3.x.
7. Change the directory to socialreviews once the LoopBack application is created.

Example 2-9 SocialReviews LoopBack application from CompanyC

```
C:\APIConnect>apic loopback -n socialreviews

     _-----_
    |       |    .--------------------------.
    |--(o)--|    |  Let's create a LoopBack |
   `---------'   |       application!       |
    ( _'U`_ )    '--------------------------'
    /___A___\
     |  ~  |
   __'.___.'__
 ' `  `  |° ' Y `

? What's the name of your application? socialreviews
? Enter name of the directory to contain the project: socialreviews
   create socialreviews/
     info change the working directory to socialreviews

? What kind of application do you have in mind? empty-server (An empty LoopBack
API, without any configured models or datasources)
? Which version of LoopBack would you like to use? 3.x
Generating .yo-rc.json


I'm all done. Running npm install for you to install the required dependencies. If
this fails, try running the command yourself.


   create .editorconfig
   create .eslintignore
   create .eslintrc
   create server\boot\root.js
   create server\middleware.json
   create server\middleware.production.json
   create server\server.js
   create .gitignore
   create client\README.md

Done running loopback generator

Updating swagger and product definitions
Created C:\APIConnect\socialreviews\definitions\socialreviews.yaml swagger
description
Created socialreviews-product.yaml product definition [socialreviews:1.0.0]

Next steps:

  Change directory to your app
    $ cd socialreviews

  Create a model in your app
    $ apic create --type model

  Compose your API, run, manage, enforce and deploy it with API Connect
    $ apic edit
```

```
  Run the app
    $ apic
start
```

Create datasource and models

A LoopBack model represents data in backend systems such as databases, and by default has both Node and REST APIs. Additionally functionality for validation rules and business logic could be added to the LoopBack models. Socialreviews application has one model *review*. LoopBack models can manipulate data via the datasource object. Attaching a datasource to a model adds instance methods and static methods to the model.

Review model

Review model represents data in backend for storing the reviews from companyC. The review model will also create the REST API for working with reviews data. Table 2-4 shows the properties of review model.

Table 2-4 Properties in review mode

Property name	Property type	Description	Required	ID
itemId	number	Item Id	Yes	Yes
date	date	review date	No	NA
comment	string	reviewer comment	No	NA
rating	number	reviewer item rating	No	NA
reviewer_name	string	reviewer name	No	NA
reviewer_email	string	reviewer email	No	NA

This model uses in-memory database. Data for in-memory database is persisted in flat file on the local storage of application. Follow steps below to configure the in-memory datasource for review model.

1. On command line change the directory to APIConnect/socialreviews.
2. Create a directory with name `db`.
3. Create a file inside db directory with name `memdb.json`.
4. Get back to directory APIConnect/socialreviews.
5. Execute command shown in Example 2-10 to create a datasource called `memdb`.
6. Provide the datasource name as memdb and select the **in-memrory** datasource.
7. Leave the window.locastorage blank and press Enter.
8. Provide full path to file as `db/memdb.json`.

Example 2-10 Create memdb datasource for review model

```
C:\APIConnect\socialreviews>apic create --type datasource
? Enter the data-source name: memdb
? Select the connector for memdb: In-memory db (supported by StrongLoop)
Connector-specific configuration:
? window.localStorage key to use for persistence (browser only):
```

```
? Full path to file for persistence (server only): db/memdb.json
Done running loopback generator

Updating swagger and product definitions
Created C:\APIConnect\socialreviews\definitions\socialreviews.yaml swagger
description
```

Following the steps above creates the in-memory datasource for review model. Steps below now would help create the review model using the user interface of API Connect Toolkit.

9. To start the designer toolkit use the command shown in Example 2-11.

Example 2-11 Start API Connect Toolkit

```
C:\APIConnect\comanpyC\socialreviews>apic edit
Express server listening on http://127.0.0.1:9000
```

This will open the designer toolkit application in browser. login using the IBM Bluemix credentials as shown in 2.2.3, “IBM API Connect Developer Toolkit API Designer” on page 17

Note: API Connect Toolkit designer tool could now also be accessed using the url below:

`http://127.0.0.1:9000/#/design/apis`

10. Click on the **Models** tab in the user interface.
11. Click on the **Add** button to create a new model.
12. Create a new LoopBack model with name `review` and then click the **New** button.
13. Now add new properties in this model referencing the review model properties listed in Table on page 32.

    Figure 2-14 below shows the properties created in review model.

Figure 2-14 Properties of review model

14. In review model set the datasource to use the in-memory datasource memdb as shown in Figure 2-15 on page 34 below.

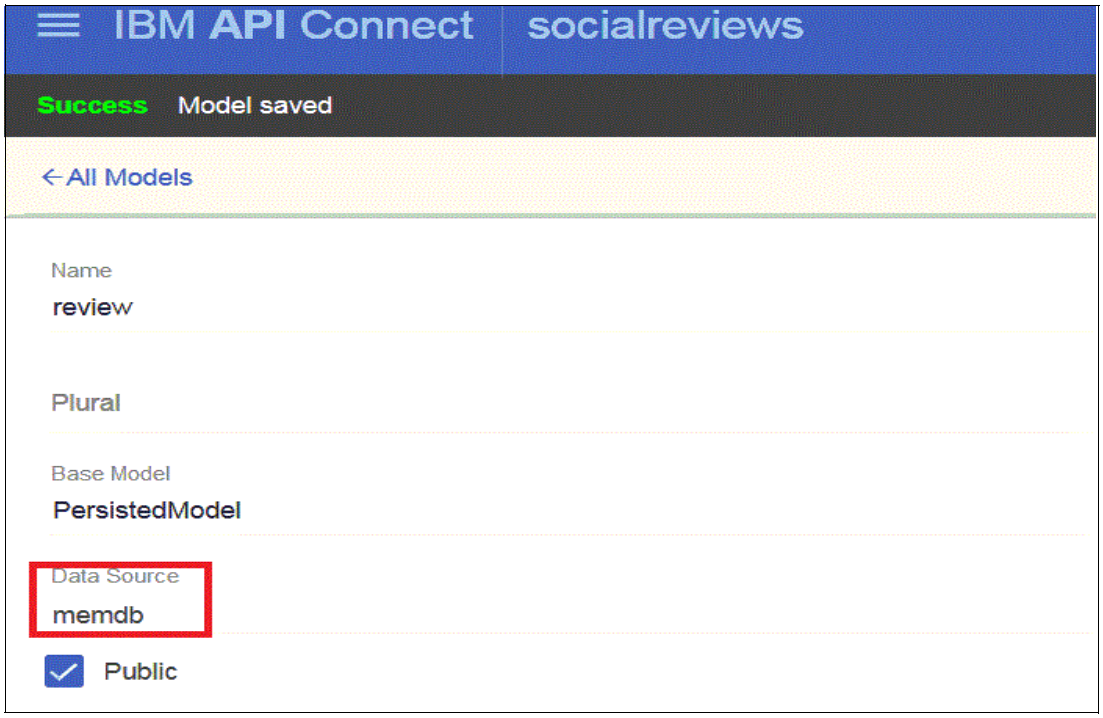

Figure 2-15 Set the review model to use the in-memory datasource

15. Click the **Save** button on top right corner to save the changes made to the review model.

Configure the socialreviews product

To configure the socialreviews product through API Designer toolkit perform the following steps:

1. Click on the **Products** tab in the API Designer. Figure 2-16 below shows the socialreviews product. Configure the plans and provider details of socialreviews product by clicking on the product.

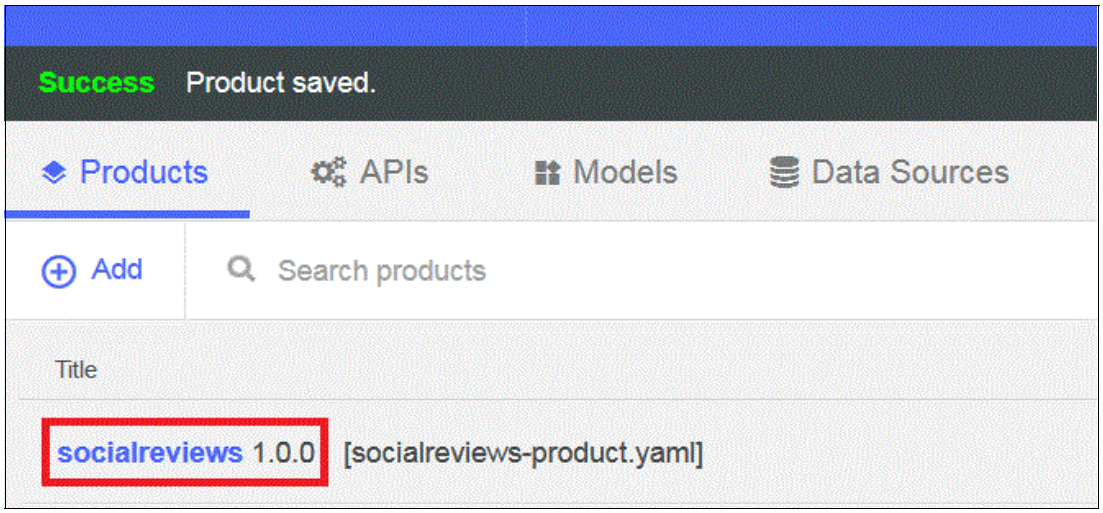

Figure 2-16 Socialreviews product

2. Configure the socialreviews product using the property values listed in Table 2-5 on page 35 below:

Table 2-5 Socialreviews product design details

Socialreviews product design details		
Info	Title	socialreviews
	Name	socialreviews
	Version	1.0.0
	Description	socialreviews application from CompanyC
Contact	Name	Admin
	Email	admin@companyc.com
	URL	http://www.companyc.com
License	Name	companyC license
	URL	http://www.companyc.com
Terms of Service	Terms of Service	CompanyC terms of service
Visibility	Visible to	Public
	Subscribable by	Authenticated users
API		socialreviews 1.0.0
Plans	Silver-Plan	1000 invokes/hr (Hard Limit Enforced)
	Gold-Plan	unlimited access

3. Save the product configuration after making all the changes.

Configure socialreview API

Socialreview application has one LoopBack model with name review. This application will be using following APIs as shown in Table 2-6. You can delete the other default API.

Table 2-6 Review model APIs

Model	Path	Operation	Description
review	/reviews	GET	Get all the reviews for a item by filter
review	/reviews	POST	Create a new review for an item
review	/reviews/avgrating	GET	Get average rating for an item
review	/reviews/counter	GET	Get review count for an item
review	/reviews/oauth	GET	Get empty message to trigger Oauth flow

The GET and POST methods for path /reviews are created by the designer tool. Follow the steps in sections below to create the remaining three API.

Navigate to socialreviews API by clicking on **socialreviews 1.0.0** and then Disable **clientSecretHeader** in the security section, as shown in Figure 2-17 below.

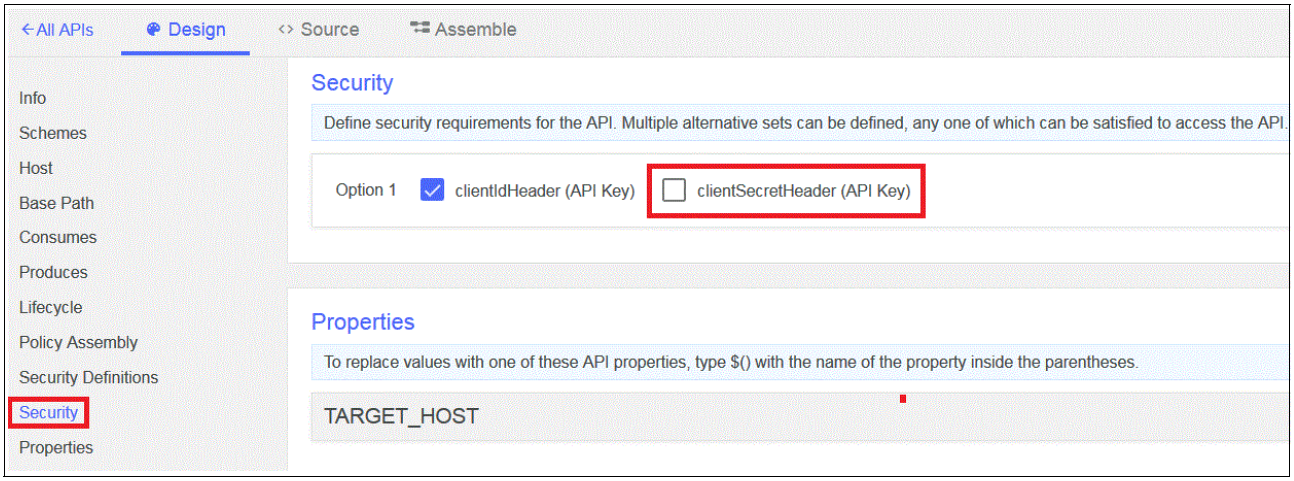

Figure 2-17 Disable clientSecretHeader in socialreviews API

Create the reviews average rating path

You can create the reviews average rating path using the following steps:

1. In socialreviews 1.0.0 API click on **Path** → **Add new Path**.
2. Provide the path name as /reviews/avgrating.
3. A default GET method is created for this path, expand the GET method.
4. Update summary of GET method with “Find average rating of an item”.
5. Update operation ID of GET method “review.avgrating”.
6. Click **Save**.
7. A remote hook should be defined to implement the logic to calculate the average rating.

Note: A *remote method* is a static method of a model, exposed over a custom REST endpoint. Use a remote method to perform operations not provided by LoopBack's standard model REST API. See:

https://docs.strongloop.com/display/public/LB/Remote+methods

A *remote hook* enables to execute a function before or after a remote method is called by a client. See:

https://docs.strongloop.com/display/public/LB/Remote+hooks

To add a remote hook for average rating, add methods shown in Example 2-12 on page 36 inside the review.json file inside the LoopBack application. The file will be located in following location:

APIConnect/socialreviews/common/models/review.js

Example 2-12 Remote method for average rating for an item

```
module.exports = function(Review) {

Review.avergateratingbyItem = function(itemId, cb) {
```

```
   Review
       .find( {"where": {"itemId" : itemId} })
       .then(function(rev) {
         var count = 0;
            var avgr  = 0;
            var sum   = 0;

        var rCount = rev.length;
        for (var i = 0; i < rCount; i++) {
             var rating = rev[i].rating;
                sum = sum + rating;
                if(rating > 0 )
                    count=count+1;
         }

            if (count > 0)
            {
                avgr = sum/count;
            }
            else
                avgr = 0;

            cb(null, avgr);
       })
};

Review.remoteMethod('avergateratingbyItem',
    {
        http: {path:'/avgrating', verb:'get'},
        accepts:{arg:'itemId', type:'number'},
        returns:{arg:'avgrating', type:'integer'}
    }
);

};
```

8. Save the review.js file

Create the reviews counter path

Follow the steps below to create the reviews counter path:

1. In socialreviews 1.0.0 API add new path by clicking **Path** → **Add New Path**.
2. Provide the path name as /reviews/counter.
3. A default GET method is created for this path, expand the GET method.
4. Update summary of GET method with “Find the count of reviews for an item”
5. Update operation ID of GET method “review.counter”
6. Click **Save**.
7. A remote hook should be defined to implement a logic to calculate the count of review for an item. To add a remote hook for review counter, add methods shown in Example 2-13 on page 37 inside the review.js file of LoopBack application.

Example 2-13 Remote method for counting the reviews for an item

```
Review.reviewCounter = function(itemId, cb) {
```

```
   Review
       .find( {"where": {"itemId" : itemId} })
       .then(function(rev) {

      var rCount = rev.length;
        cb(null, rCount);
       })
};

Review.remoteMethod('reviewCounter',
    {
        http: {path:'/counter', verb:'get'},
        accepts:{arg:'itemId', type:'number'},
        returns:{arg:'reviewCount', type:'integer'}
    }
);
```

8. Save the review.js file.

Create the reviews oauth path

This is a utility API that will be used to trigger the OAuth flow, Chapter 5, “Consuming the APIs” on page 79 will provide more details on this API.

1. In socialreviews 1.0.0 API click on **Path** → **Add new Path**.
2. Provide the path name as /reviews/oauth.
3. A default GET method is created for this path, expand the GET method.
4. Update summary of GET method with “Oauth trigger API”.
5. Update operation ID of GET method “review.oauth”.
6. Click **Save**.
7. A remote hook should be defined to implement a logic to send a empty response for the oauth trigger. To add a remote hook for review oauth, add methods shown in Example 2-14 inside the review.js file of LoopBack application.

Example 2-14 Remote method for Oauth trigger

```
Review.ouathtrigger = function(cb) {
           cb(null, '');
};

Review.remoteMethod('ouathtrigger',
    {
        http: {path:'/oauth', verb:'get'},
    }
);
```

8. Save the review.js file.
9. Create a new property called TARGET_HOST. This will be the property to tell the assembly where the application will be actually deployed. This property must be changed before the actual deployment to Bluemix.

   This now completes the creation of LoopBack application from companyB and companyC. Stop the API Connect Developer Toolkit using Ctrl+C keys.

2.5 Start and quickly test the LoopBack applications locally

In this section we will show you how to start and quickly test the LoopBack applications locally.

2.5.1 Start the inventory application

To start the inventory application from command line, navigate to the inventory directory in a terminal or command line window. Use the **apic start** command in Example 2-15 to start the inventory application for local testing. Once the application is started it will create two node processes as shown in Example 2-15 below.

Example 2-15 Start the inventory application

```
C:\APIConnect\inventory>apic start
Service inventory started on port 4001
Service inventory-gw started on port 4002
```

2.5.2 Test the inventory application

Tests below will be a quick test to confirm if the API are reachable are not. More detailed testing of the API will be explained in Chapter 5, “Consuming the APIs” on page 79.

Test the API to get the list of all items in inventory

In a browser window, access the URL below (Figure 2-18).

```
GET http://localhost:4001/api/items
```

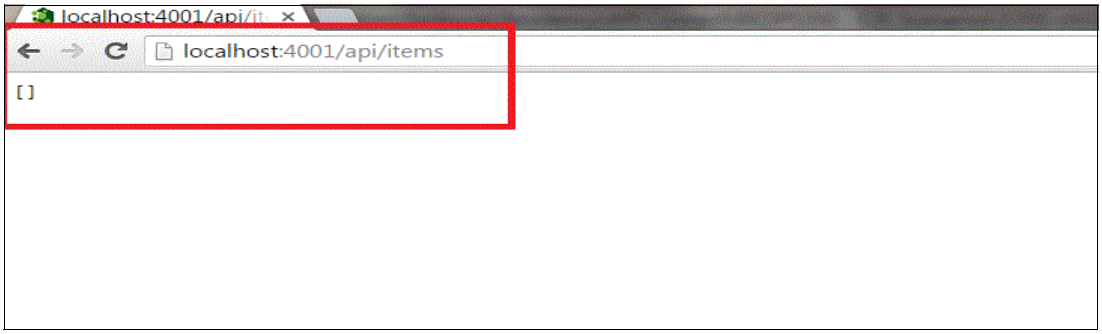

Figure 2-18 Test the GET Items inventory API

This test confirms that the inventory application is up and running and it also renders the empty list of items as no item exists in the in memory database as shown in Figure 2-18 above.

Test the API to get the item image file through the container API

In a browser, access the URL below:

```
GET http://localhost:4001/api/containers/items/download/hollerith-tabulator.jpg
```

This test confirms that the inventory application is up and running and it also renders the items image file from the local storage as shown in Figure 2-19 on page 40.

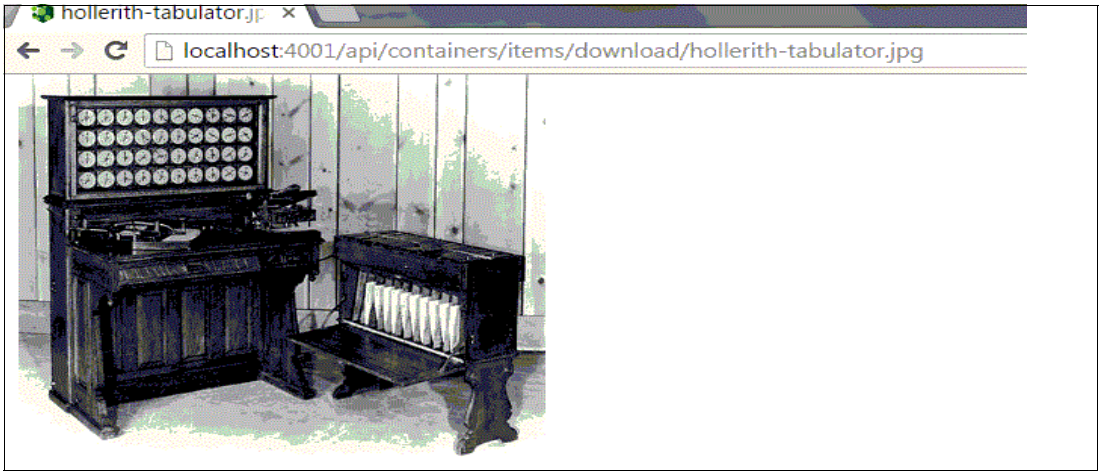

Figure 2-19 Test the download image container API

2.5.3 Stop the inventory application

To stop the inventory application use the **`apic stop`** command as shown in Example 2-16 below:

Example 2-16 Stop the inventory application

```
C:\APIConnect\inventory>apic stop
Stopped inventory
Stopped inventory-gw
```

This stops both the node processes for inventory application.

2.5.4 Start the socialreviews application

To start the socialreviews application from the API connect designer, start the socialreview designer using the **`apic edit`** command. From the API Designer, click the **Play** button to start the application, as shown in Figure 2-20 on page 40 below.

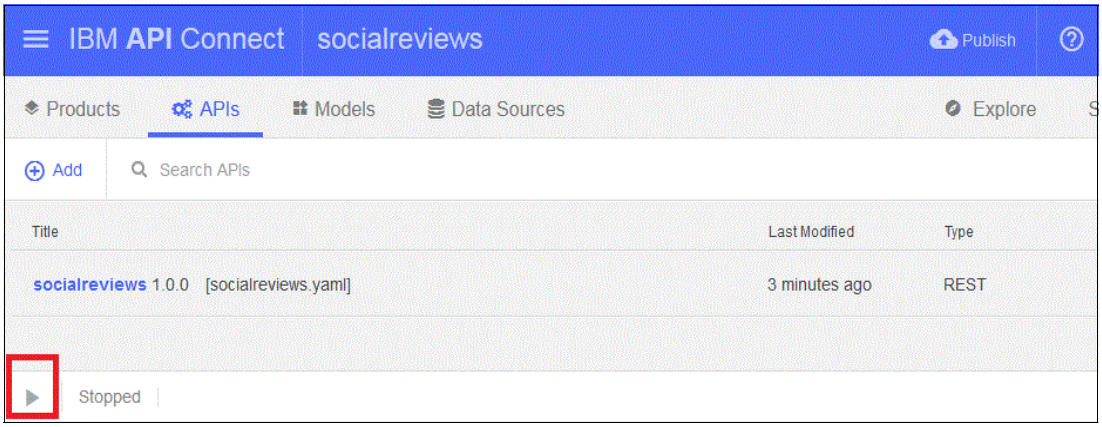

Figure 2-20 Start the socialreviews application

2.5.5 Test the socialreviews application

Tests below will be a quick test to confirm if the API are reachable are not. More detailed testing of the API will be explained in Chapter 5, “Consuming the APIs” on page 79.

Test the API to get the count of reviews for an item

In a browser window, access the URL below:

```
GET http://localhost:4001/api/reviews/counter?itemId=13412
```

This will return a JSON message with the count of reviews for ItemId 13412. As there are no reviews in the in-memory database the count is zero as shown in Example 2-17 below:

Example 2-17 Review count for an item

```
{"reviewCount":0}
```

Test the API to get the average rating for an item

In browser access the URL below:

```
GET http://localhost:4001/api/reviews/avgrating?itemId=13412
```

This will return a JSON message with the average rating for itemId 13412. As there are no reviews in the in-memory database the average rating for this item will be zero as shown in Example 2-18 below.

Example 2-18 Average rating for an item

```
{"avgrating":0}
```

2.5.6 Stop the socialreviews application

To start the socialreviews application from the API connect designer click on the stop button at the bottom left of the page and this will stop both the loopback application and micro gateway node processes as shown in Figure 2-21 on page 41 below.

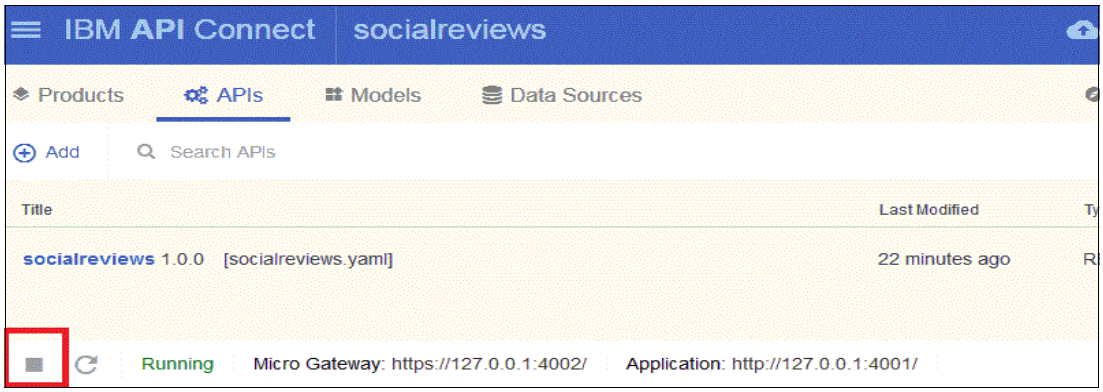

Figure 2-21 Stop the socialreviews application

2.6 Test the APIs

This section helps readers to understand how the use the API and populate the in-memory database for all the further testing.

2.6.1 Test the inventory application API

Inventory application has six REST API exposed through the LoopBack application. These API could be referenced back in Table 2-3 on page 29. Use **apic start** command from inventory directory to start the application locally.

Create items for the inventory application

Create twelve items using HTTP REST tool. To create items use the REST URL listed below:

```
POST http://localhost:4001/api/items
```

Figure 2-22 below shows how the HTTP REST tools could be used to invoke the POST call for /items API.

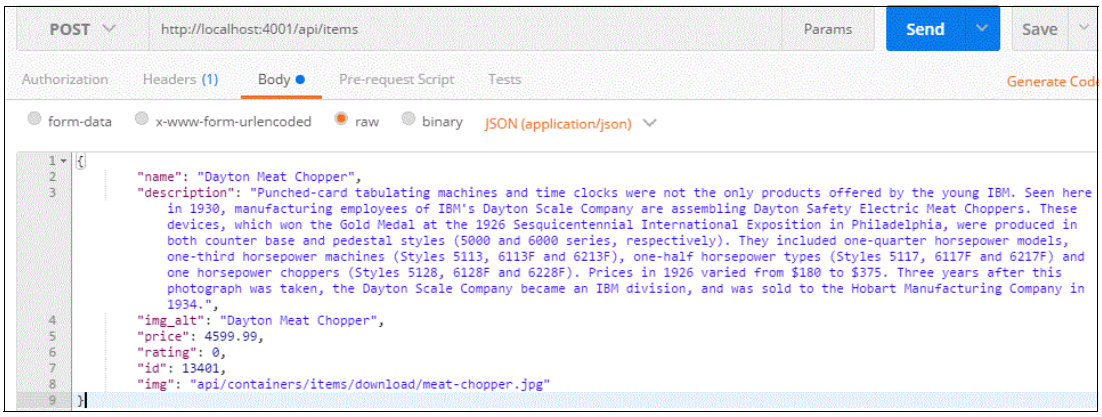

Figure 2-22 Create item 13401

Note: A pre-populated memdb.json file is available for download in IBM Redbooks FTP server. For more details please refer to Appendix B, “Additional material” on page 129.

You can download the memdb.json file from IBM Redbook ftp share and replace the existing memdb.json with the downloaded file. Restart the LoopBack application after copying the new memdb.json file. Once all the items are virtually created there will be a total of twelve items in the in-memory database.

Get all items for the inventory application

You could get all the twelve items created previously by using the GET /items API. URL below will return all the items in the inventory LoopBack application.

```
GET http://localhost:4001/api/items
```

Get an individual item for the inventory application

You could GET an individual item by using the GET /items/{id} API. URL below will return the details of item 13401

```
GET http://localhost:4001/api/items/13401
```

Figure 2-23 Get details for Item 13401 from inventory application

Update an individual item

You could update elements of an individual item by using the PUT /items/{id} API. The payload should consist of an JSON element to be updated. Example 2-19 shows how could we update the rating for item 13041

Example 2-19 Update the rating of item 13401

```
PUT http://localhost:4001/items/13401
{
“rating”:4
}
```

Note: Ensure that the HTTP request has the content type header set to application/json

Content-type: application/json

Deleting an individual item

An item could be deleted from the in-memory database by using DELTE /items/{id} API. URL below for a DELETE REST call will delete item 13401 from the in-memory database.

```
DELETE http://localhost:4001/items/13401
```

Accessing an image file using the container API

In browser access the URL below:

```
GET http://localhost:4001/api/containers/items/download/hollerith-tabulator.jpg
```

This test confirms that the inventory application is up and running and it also renders the items image file from the local storage as shown in Figure on page 40. Use **apic stop** command from inventory directory to stop the application.

2.6.2 Test the socialreviews application API

Socialreview application has five REST exposed through the LoopBack application. These API could be reference back in Table 2-6 on page 35. se **apic start** command from socialreviews directory to start the application locally.

Create a review for an item

Create reviews using the HTTP REST tools. To create the reviews use the REST URL listed below:

```
POST http://localhost:4001/api/reviews
```

Figure 2-24 shows how a review could be created for item with the socialreview application

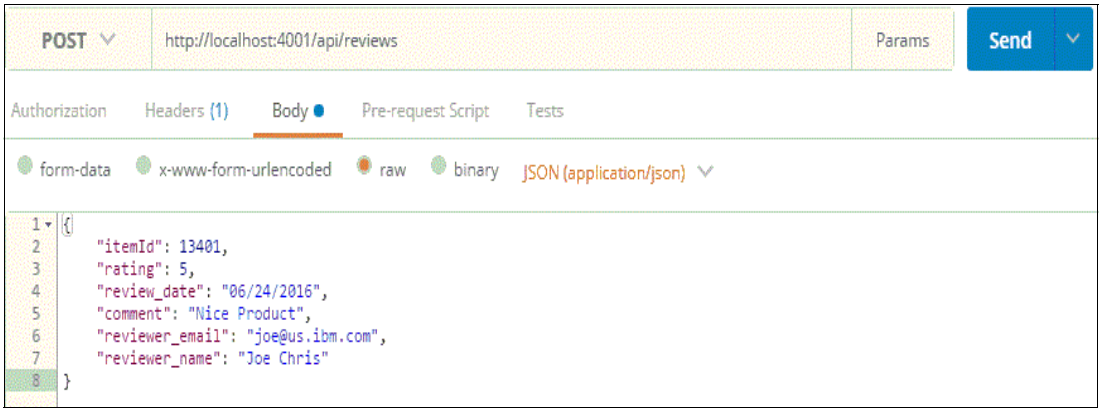

Figure 2-24 Create a review for item 13401

Example 2-20 contains payload for creating reviews.

Example 2-20 Payload for creating reviews

```
{
    "itemId": 13401,
    "rating": 5,
    "review_date": "06/24/2016",
    "comment": "Nice Product",
    "reviewer_email": "joe@us.ibm.com",
    "reviewer_name": "Joe Chris"
}
```

Note: Ensure that the HTTP request has the content type header set to application/json

Content-type: application/json

Get reviews for an item

You can get reviews for an item by applying filter on GET /reviews REST API. URL below will provide all the reviews for item 13401

```
GET http://localhost:4001/api/reviews?filter={"where":{"itemId":13401}}
```

To read more about LoopBack where filter please refer the URL below:

```
https://docs.strongloop.com/display/public/LB/Where+filter
```

Get the reviews count for an item

In browser access the URL below:

```
GET http://localhost:4001/api/reviews/counter?itemId=13401
```

This will return a JSON message with the count of reviews for ItemId 13401. As there is just one reviews in the in-memory database the count is one as shown in Example 2-17 on page 41 below:

Example 2-21 Review count for item 13401

```
{"reviewCount":1}
```

Get the average rating for an item

In browser access the URL below:

```
GET http://localhost:4001/api/reviews/avgrating?itemId=13401
```

This will return a JSON message with the average rating for itemId 13401. As there is only one review in the in-memory database so the average rating for this item will be five as shown in Example 2-18 on page 41 below.

Example 2-22 Average rating for an item 13401

```
{"avgrating":5}
```

Use **apic stop** command from socialreviews directory to stop the application.

2.7 Update the LoopBack application gateway

When a LoopBack application is created it is configured to use the Micro Gateway and that helps to test the API locally. When you publish during testing, you make your Product available through the Micro Gateway hosted on a API Connect Collective. You can then use a DataPower Gateway to make your APIs available when they are ready for production. The Micro Gateway runs on a API Connect Collective, as do LoopBack apps. The DataPower Gateway can run on either a physical or virtual DataPower appliance.

Micro Gateway - The Micro Gateway is a Gateway that is built on node.js, and provides enforcement for the authentication, authorization, and flow requirements of an API. The Micro Gateway is deployed on a API Connect Collective and has a limited number of policies available to it.

DataPower Gateway - The DataPower Gateway is a Gateway that is deployed on a virtual or physical DataPower appliance. DataPower Gateway has more policies available to it than the Micro Gateway and can handle enterprise level complex integration.

2.7.1 Update the inventory LoopBack application gateway

1. In the inventory API using the Designer Toolkit, go to the assemble tab as shown in Figure 2-25 below

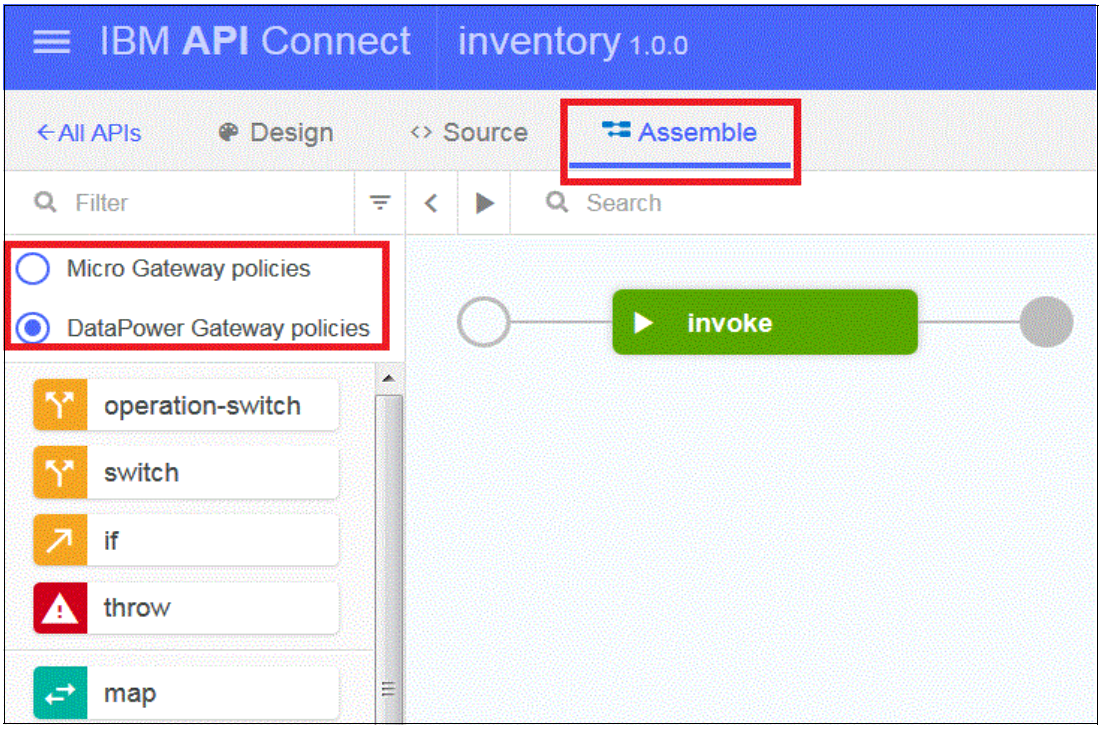

Figure 2-25 Update inventory API to use DataPower Gateway

2. Change to use DataPower gateway policy instead of Micro Gateway Policies.
3. Update the invoke node URL property as $(TARGET_HOST)$(request.path) to use the TARGET_HOST property.
4. Save the changes made to the LoopBack application.

2.7.2 Update the socialreviews LoopBack application gateway

Perform the following steps to update the socialreviews LoopBack application gateway:

1. In the socialreviews API using the Designer Toolkit, go to the **Assemble** tab as shown in Figure 2-26 on page 47
2. Drag and drop the **gatewayscript** node from the node menu in front of the assembly node
3. Update the gateway script node with script provided in Figure 2-26 on page 47. This script helps in using the filters.

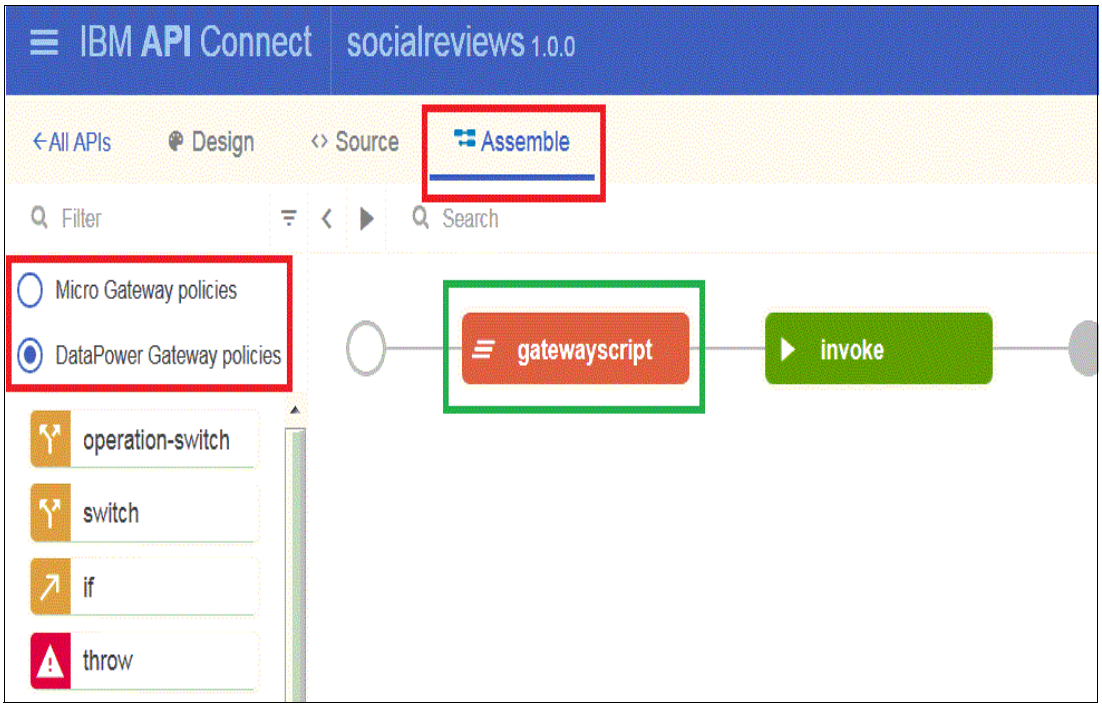

Figure 2-26 Update socialreviewsAPI to use DataPower Gateway and Gateway script node

Example 2-23 Gateway script to handle filters

```
var reqUrl = apim.getvariable('request.uri');
var filterQuery = reqUrl.split('?')[1];
if(filterQuery || filterQuery !== '')

{
      apim.setvariable('filterQuery', filterQuery);
}
else
 {
      apim.setvariable('filterQuery', "");
}
```

4. Update the invoke node URL property as $(TARGET_HOST)$(request.path)?$(filterQuery) to use the TARGET_HOST property.
5. Save the change made to the LoopBack application.

2.8 Deploy the LoopBack applications to IBM Bluemix

This section shows the step-by-step process of deploying the inventory and socialreviews LoopBack application in Bluemix environment,

2.8.1 Configure the Bluemix environment.

Section 1.3, “Environment setup” on page 4 provides step-by-step procedure to setup the Bluemix environment. Follow the steps in this section to configure your Bluemix environment.

2.8.2 Deploy inventory application in Bluemix

To deploy the inventory application using command line follow the steps below.

1. From the command line move to directory of inventory application.
2. Login using **apic login** command as shown in Example 2-24 on page 48

Example 2-24 Login into your apic account using command line

```
c:\APIConnect\inventory>apic login
Enter the API Connect server
? Server: us.apiconnect.ibmcloud.com
? Username: **************************
? Password (typing will be hidden) ********
Generate a passcode from https://mccp.ng.bluemix.net/login/showpasscode.jsp
? Enter Bluemix one-time passcode: IGJ7wy
Logged into us.apiconnect.ibmcloud.com successfully
```

3. Set the publish configuration variable using **apic config** command to publish the application as shown in Example 2-25. This command sets the application name to *inventory-loopback-app*.

Example 2-25 Publish configuration for inventory application

```
c:\APIConnect\inventory>apic config:set
app=apic-app://us.apiconnect.ibmcloud.com/orgs/rgupta1usibmcom-redbook/apps/inventory-loopback-app
app:
apic-app://us.apiconnect.ibmcloud.com/orgs/rgupta1usibmcom-redbook/apps/inventory-loopback-app
```

Note: The command has syntax

apic config:set
app=apic-app://us.apiconnect.ibmcloud.com/orgs/{bluemiOrg}-{bluemixSpace}/apps/inventory-loopback-app

You need to replace the {bluemixOrg} and {bluemixSpace} to the ones you have.

This information can also be captured from catalog identifier button on Catalog created in Bluemix,

4. Deploy the inventory LoopBack application using **apic apps:publish** command as shown in Example 2-26 below.

Example 2-26 Deploy the inventory application in Bluemix

```
c:\APIConnect\inventory>apic apps:publish
Deploying to Bluemix
...preparing project
...building package for deploy
...uploading package
Runtime published successfully.

Management URL:
https://new-console.ng.bluemix.net/apps/0203ebfb-9ff5-495c-81f6-267561ee14bc
```

```
API target urls:
apiconnect-0203ebfb-9ff5-495c-81f6-267561ee14bc.rgupta1usibmcom-redbook.apic.mybluemix.net
API invoke tls-profile: client:Loopback-client
```

5. Once the application is deployed it could be seen under the Cloud Foundry application in the Bluemix dashboard as shown in Figure 2-27 on page 49 below.

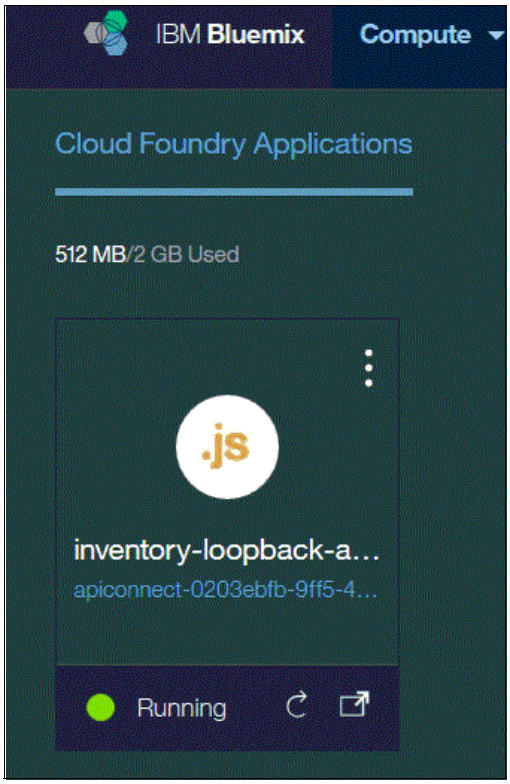

Figure 2-27 Cloud Foundry inventory application in Bluemix

6. Inventory application should now also be visible in the API connect service in Bluemix as shown in Figure 2-28.

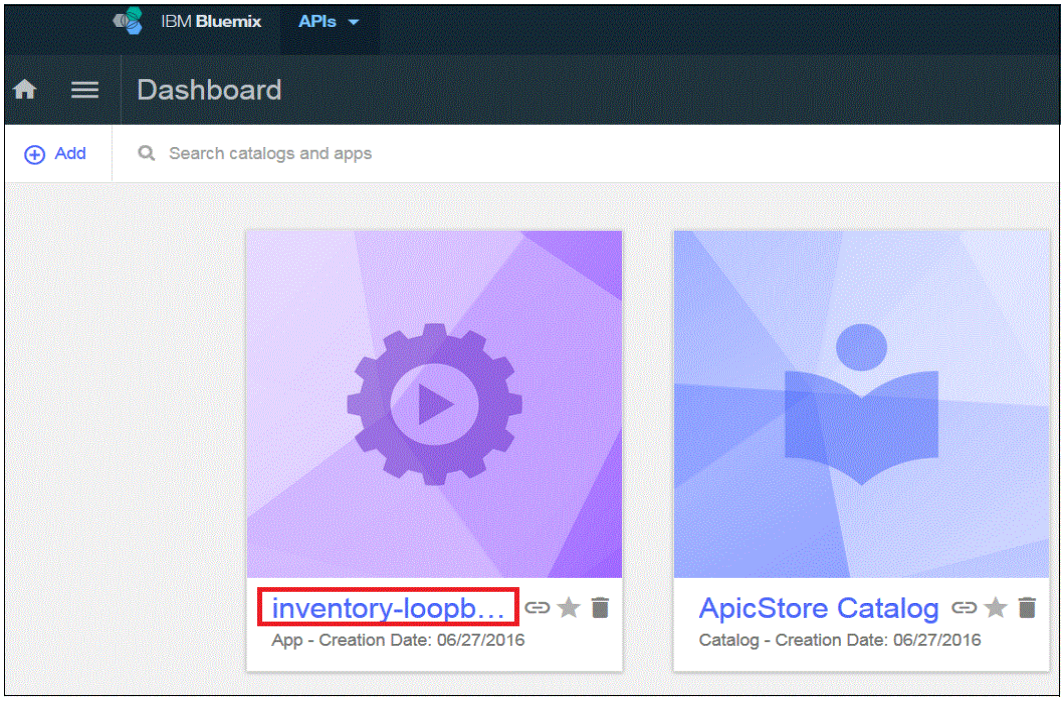

Figure 2-28 Inventory loopBack application available in API Connect service

Note: At this time also update the TARGET_HOST property created in Inventory LoopBack application with the Inventory application target url from the publish command:

```
apiconnect-0203ebfb-9ff5-495c-81f6-267561ee14bc.rgupta1usibmcom-redbook.apic.mybluemix.net
```

2.8.3 Deploy socialreviews application in Bluexmix

You can deploy the Loopback application using API Connect Designer UI as well. We will deploy the socialreviews application using Designer toolkit as described in the steps below.

1. Open the API Connect designer toolkit for socialreviews application using the **`apic edit`** command.
2. Click on the **Publish** button and then click on **Add and manage targets** to add the target for deployment as shown in Figure 2-29.

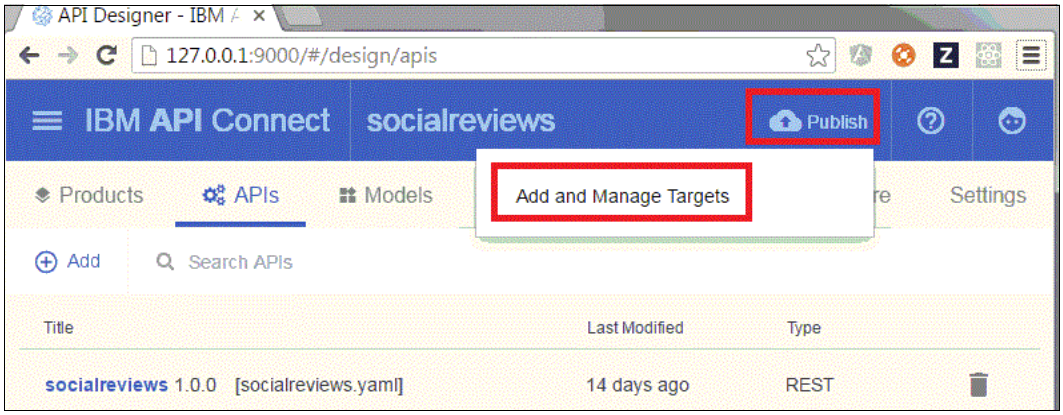

Figure 2-29 Add and manage targets for deployment

3. Add an IBM Bluemix target.
4. Select an organization and catalog and then click **Next** as shown in Figure 2-30 on page 51 below.

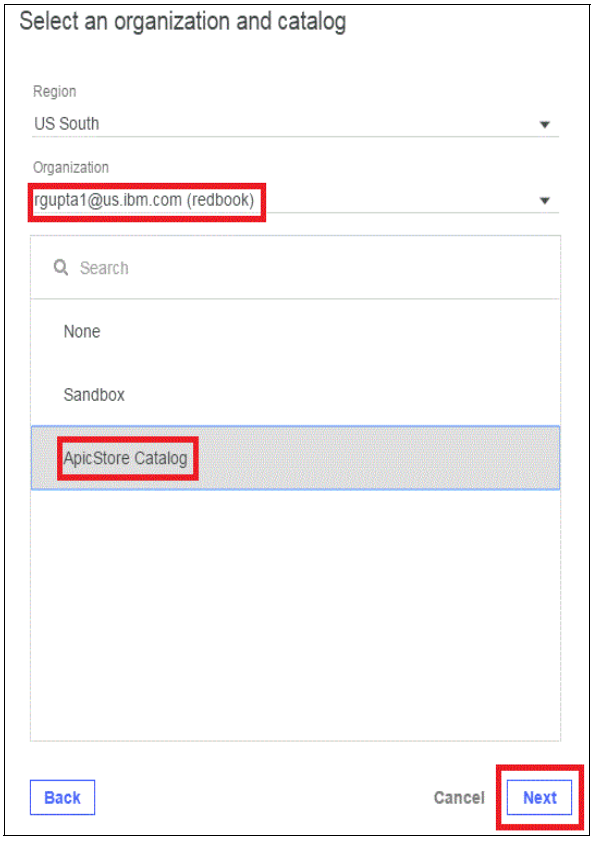

Figure 2-30 Select Bluemix organization and IBM API Connect catalog

5. Provide a Cloud Foundry application name as socialreviews-loopback-app, Click the **plus** icon and then save the configuration as shown in Figure 2-31 on page 52.

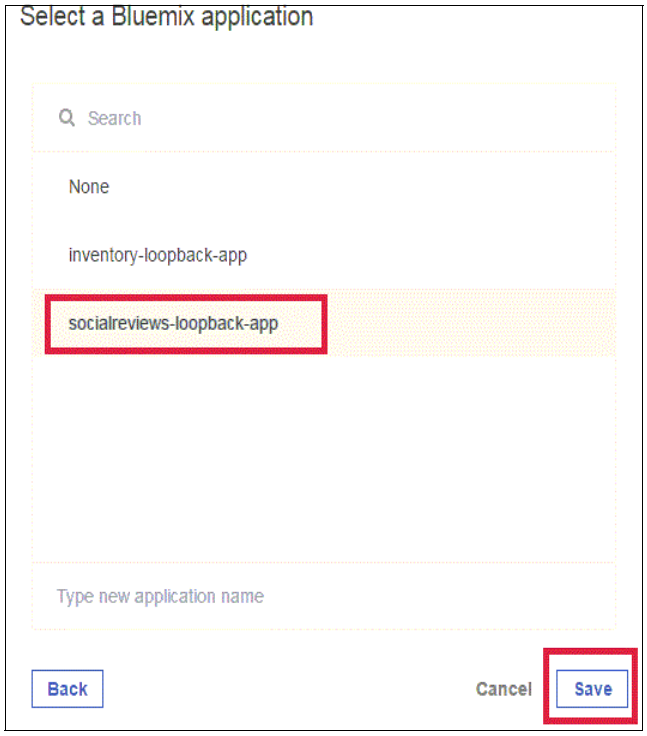

Figure 2-31 Save the publish configuration

6. To publish the socialreviews application click on the **Publish** button and select the **publish configuration** created in previous steps.

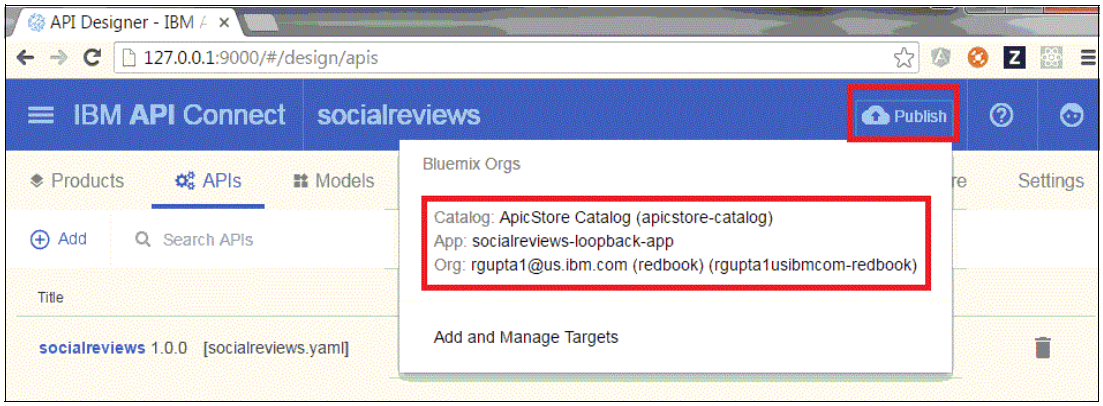

Figure 2-32 Select the publish target

7. Select **Publish** and then publish the socialreviews application to Bluemix.

Figure 2-33 Publish application to Bluemix

8. Socialreviews LoopBack application is deployed as a Cloud Foundry application and available in Bluemix as shown in Figure 2-34.

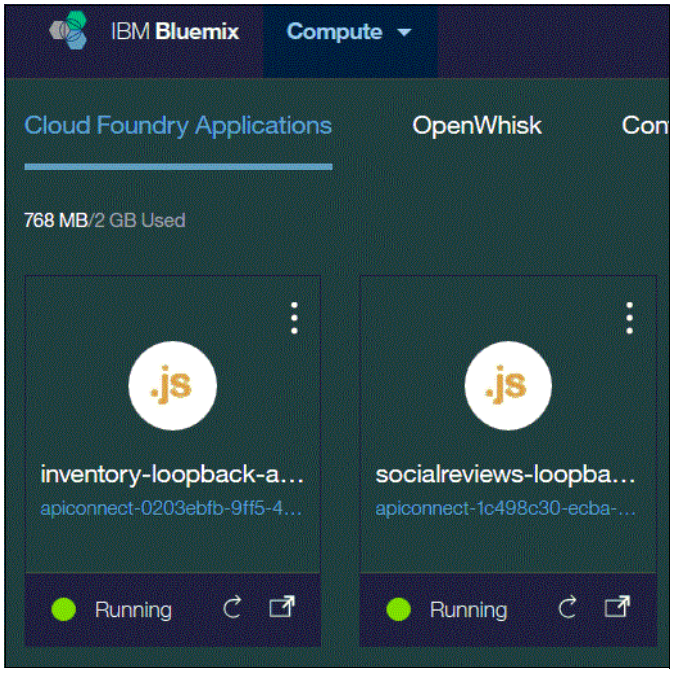

Figure 2-34 Socialreviews application deployed in Bluemix

Note: At this time also update the TARGET_HOST property created in socialreviews LoopBack application with the socialreviews application target url from the deployed Bluemix Nodejs application.

You can find the URL of this application from the deployed NodeJs application in Bluemix.

2.9 Summary

This chapter discussed the LoopBack applications and provided step-by-step process to create inventory and socialreview LoopBack application followed by local testing and application deployment in Bluemix.

3

Managing the APIs

In this chapter, we will explore how to manage the APIs developed in Chapter 2, “Developing and deploying LoopBack applications and APIs” on page 13. IBM API Connect provides several key API management capabilities, such as:

- Create and configure API Catalogs for publication
- Manage the Developer Organizations and community
- Manage the security of API environment
- Manage the API product lifecycle such as publishing API
- Analyze the API use

You have learned the catalog creation in Chapter 1, “Introduction to API Connect and environment setup” on page 1, and we'll explore the security in Chapter 4, “Securing the APIs” on page 65, Developer Organization in Chapter 5, “Consuming the APIs” on page 79, then analyze API use in Chapter 6, “Monitoring and analyzing the APIs” on page 103. In here, we are going to mainly look at publishing APIs, so developers can write applications to consume the Inventory and social review functionalities.

This chapter has the following sections:

3.1 Publish APIs with API Designer user interface

If you have followed through Chapter 1, “Introduction to API Connect and environment setup” on page 1, you should have the API Connect service added to your Bluemix space and the API catalog created. In Chapter 2, “Developing and deploying LoopBack applications and APIs” on page 13, you should have the Inventory and SocialReview LoopBack microservice applications deployed to Bluemix as well. Now, you can go ahead and publish your APIs to the catalog.

In order to make your API available for others to consume, two things need to happen:

- The Inventory application must be deployed and running so it can service the API calls.
- The API definition must be available for the consuming application developer so they can orchestrate the correct API invocation.

The second action is managed as part of the API Product lifecycle. It is via the API product that you manage the different stages of the APIs you developed. Figure 3-1 shows the possible states in IBM API Connect:

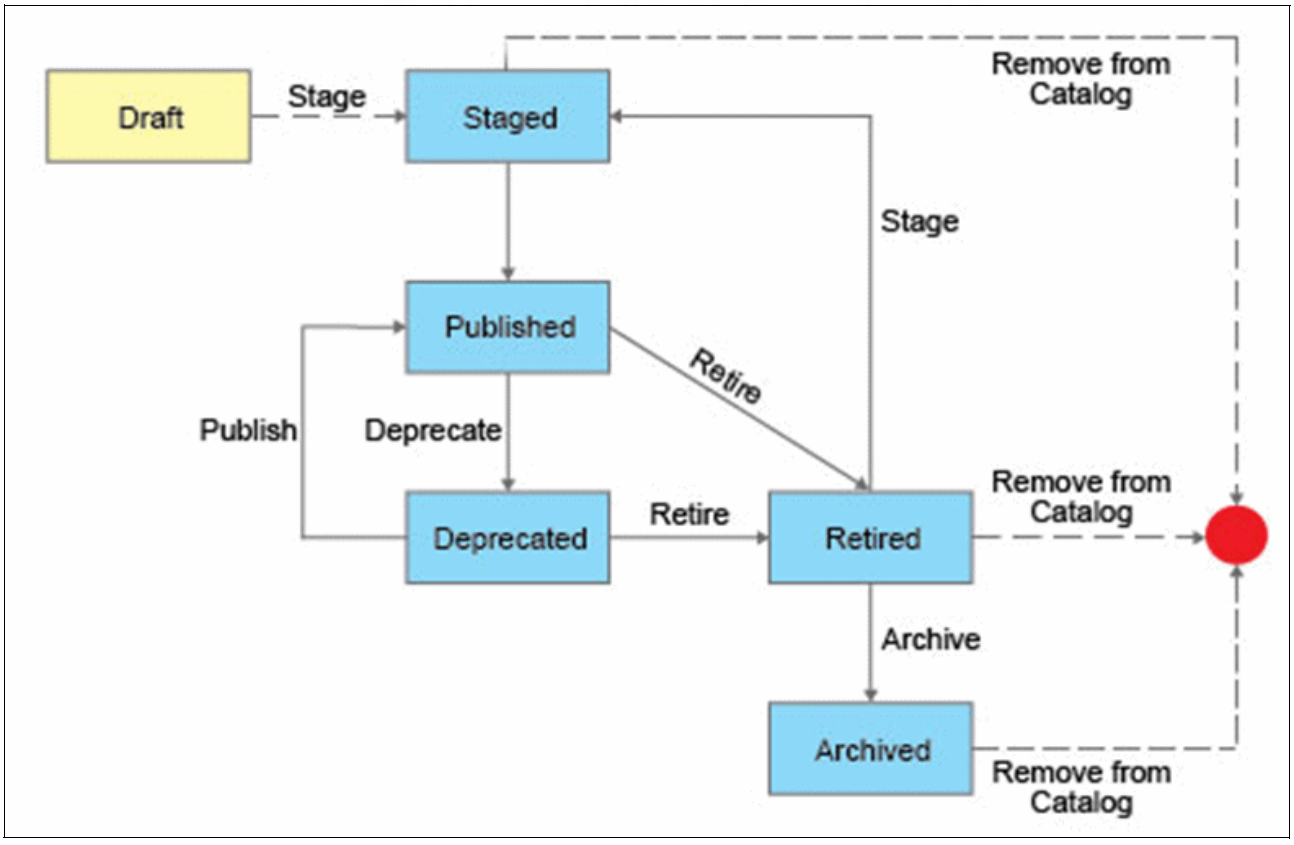

Figure 3-1 Lifecyle state diagram for Product versions

Understanding the API lifecycle: The API lifecycle is basic state machine - a set of states in which the API can be in along with the transitions between those states caused by an external signal such a request by the API manager to change the API state.

Let's now delve into the various states of the API lifecycle and discuss they *whys* and the *hows* of moving an API between the states.

- **STAGED:** The API isn't yet live but a click away from being published. It's not visible on the developer portal. Although it is possible to bypass this state and put an API directly into the PUBLISHED state, STAGED state may be advantageous if you deploy the underlying application that provides the API and publishing the API at once. Let's say the application doesn't deploy successfully then the API can't really be consumed from the developer portal. Putting the API into the STAGED state first provides the opportunity to ensure the service available before actually publishing the API. This is especially true if multiple services and the corresponding APIs are rolled out at once.
- **PUBLISHED:** The API is live and available from the developer portal for consumption. New applications can subscribe to use the API.
- **DEPRECATED:** The API is still live and available for the applications that are already using it but no new subscriptions allowed. This indicates that the API is on the to being retired and it's time for the applications consuming to upgrade to a new version of the API or a different API altogether. This is useful a large subscription based exists but the plan to eventually retire the API is on the roadmap.
- **RETIRED:** In this state, the API is no longer available for any application to consume. Any application currently using will lose access to the API. One retired, the API can't be re-published but I can be restaged at which point it gets a *second life*.
- **ARCHIVED:** Once archived, an API can't be ever used again. It can also be deleted at which point the API's life is terminated.

Note that when discussing the API lifecycle, API refers to a specific version of the API. From the API management standpoint, Inventory 1.0.0 and Inventory 1.0.1 are two different APIs with their own lifecycles. For example, while Inventory 1.0.0 is in production, Inventory 1.0.1 could be staged. Furthermore, both 1.0.0 and 1.0.1 can be in production and available to developers to consume simultaneously.

Before we dive into the details, we need to recap how an API needs to be package in order to be published to the API catalog.

1. In Chapter 2, “Developing and deploying LoopBack applications and APIs” on page 13, we used API Connect Designer to create the inventory application. Let's inspect the directory structure of our application (under `redp5350-apiconnect-sample/inventory` folder), as shown in Figure 3-2 on page 58.

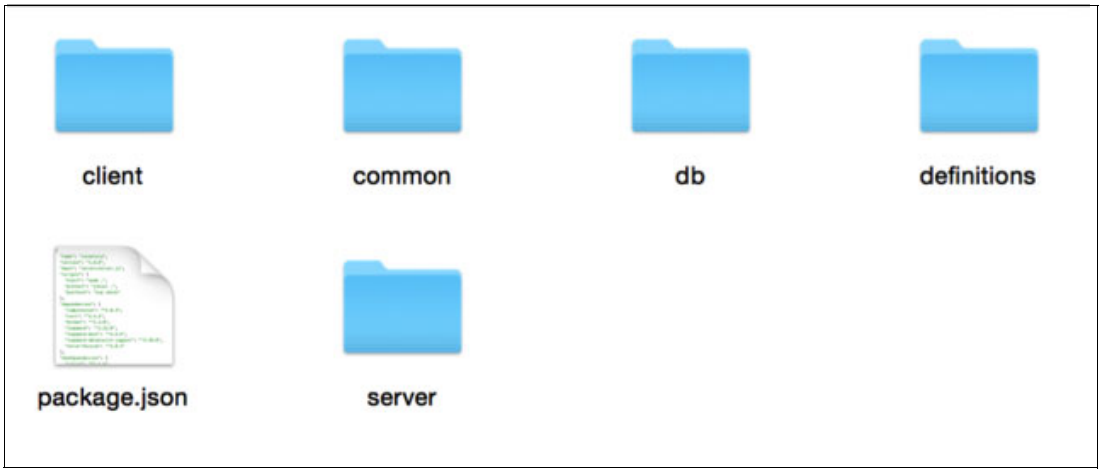

Figure 3-2 Directory structure of our application.

2. Navigate into the **definitions** directory. Inside this directory are two YAML files. See Figure 3-3.

Figure 3-3 yaml files

These files were generated by the API Connect Designer. The *inventory.yaml* is the Swagger definition of the API provided by the Inventory application. The second file, *inventory-product.yaml,* is the product file. See Figure 3-4 on page 59.

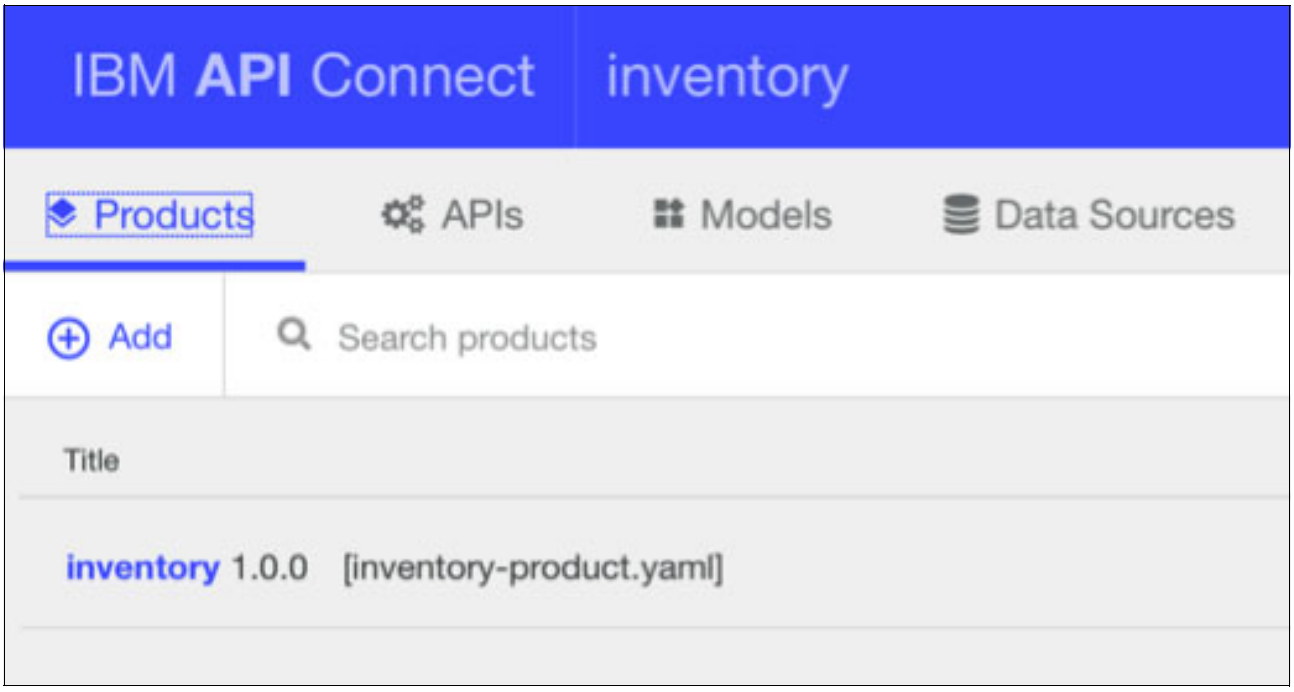

Figure 3-4 Products menu

3. You will start the API Designer by navigating into the Inventory directory and execute apic edit command: **`apic edit`**.
4. In API Designer console, sign in with your Bluemix ID.
5. In 2.8.3, “Deploy socialreviews application in Bluexmix” on page 50, you have configured the API Designer deployment target. Click on the **Publish** and select the **Bluemix target** you created earlier. Figure 3-5 shows that target has been created.

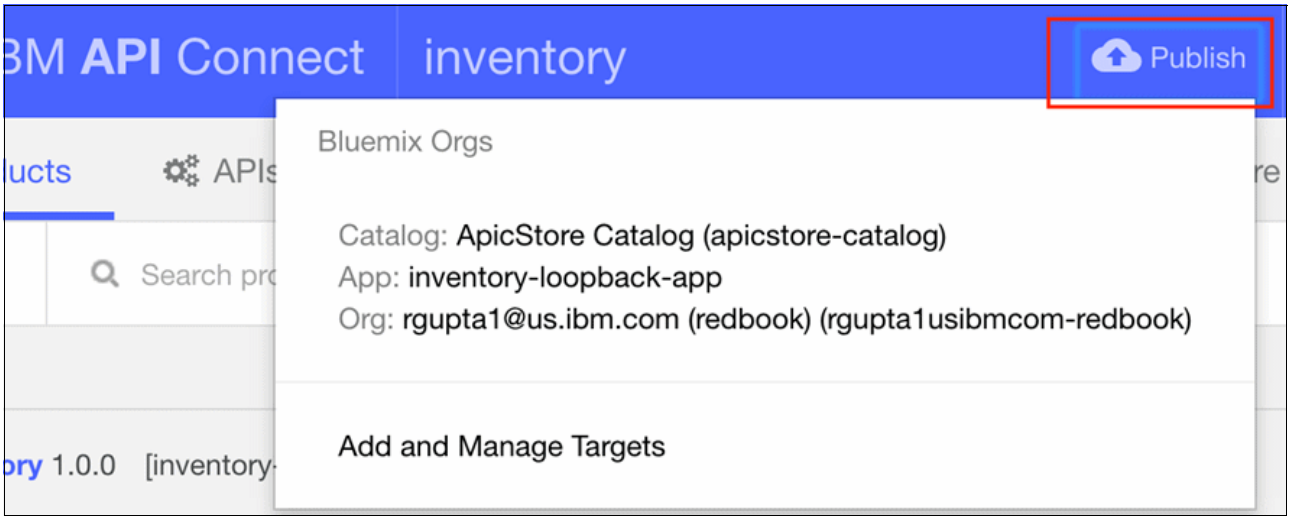

Figure 3-5 Publish target has been created

6. 6.In the Publish window, check **Stage and Publish products**. Then, check the **Stage Only** and **Select specific products**.
7. Check the I**nventory** product to publish as shown in Figure 3-6 on page 60. Click **Publish**.

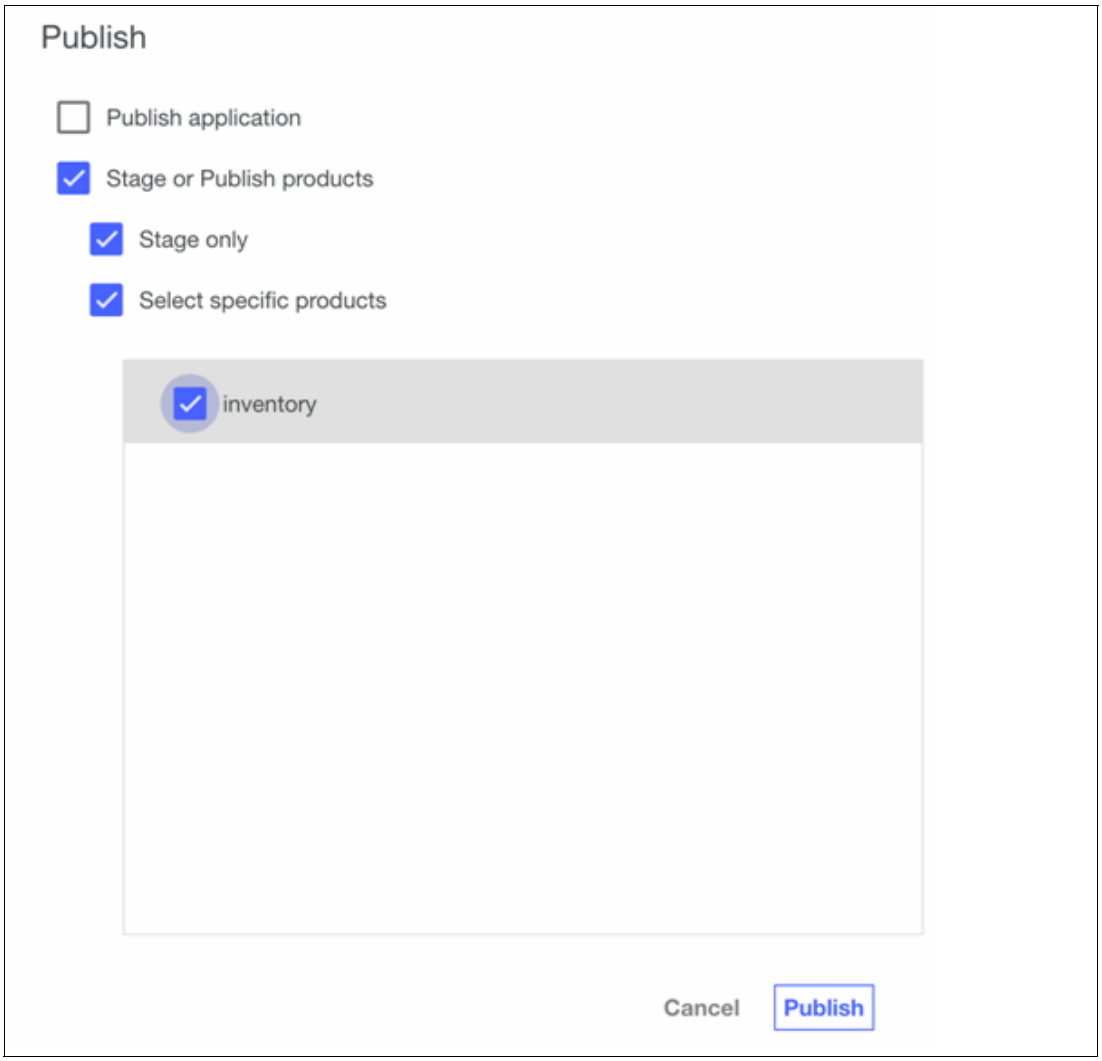

Figure 3-6 Click Publish

8. This will push the Inventory Product along with the Inventory API to Bluemix API Connect in Staged state. You can review this by logging into your Bluemix API Management console. Select the **Dashboard** section in the navigation pane, and click on the required catalog. The product is shown with a state of *Staged* as shown in Figure 3-7 on page 61.

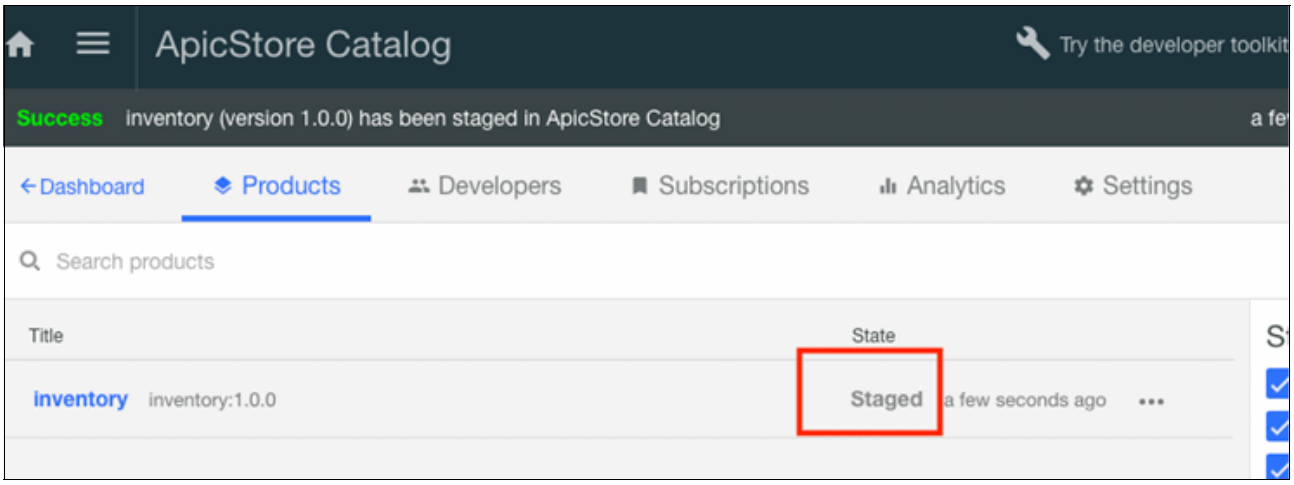

Figure 3-7 Staged state

9. Now, click the action "**...**" button next the state column to bring up Action context Menu. From there, click **Publish**. *This will change the API product state from Staged to Published*. See Figure 3-8.

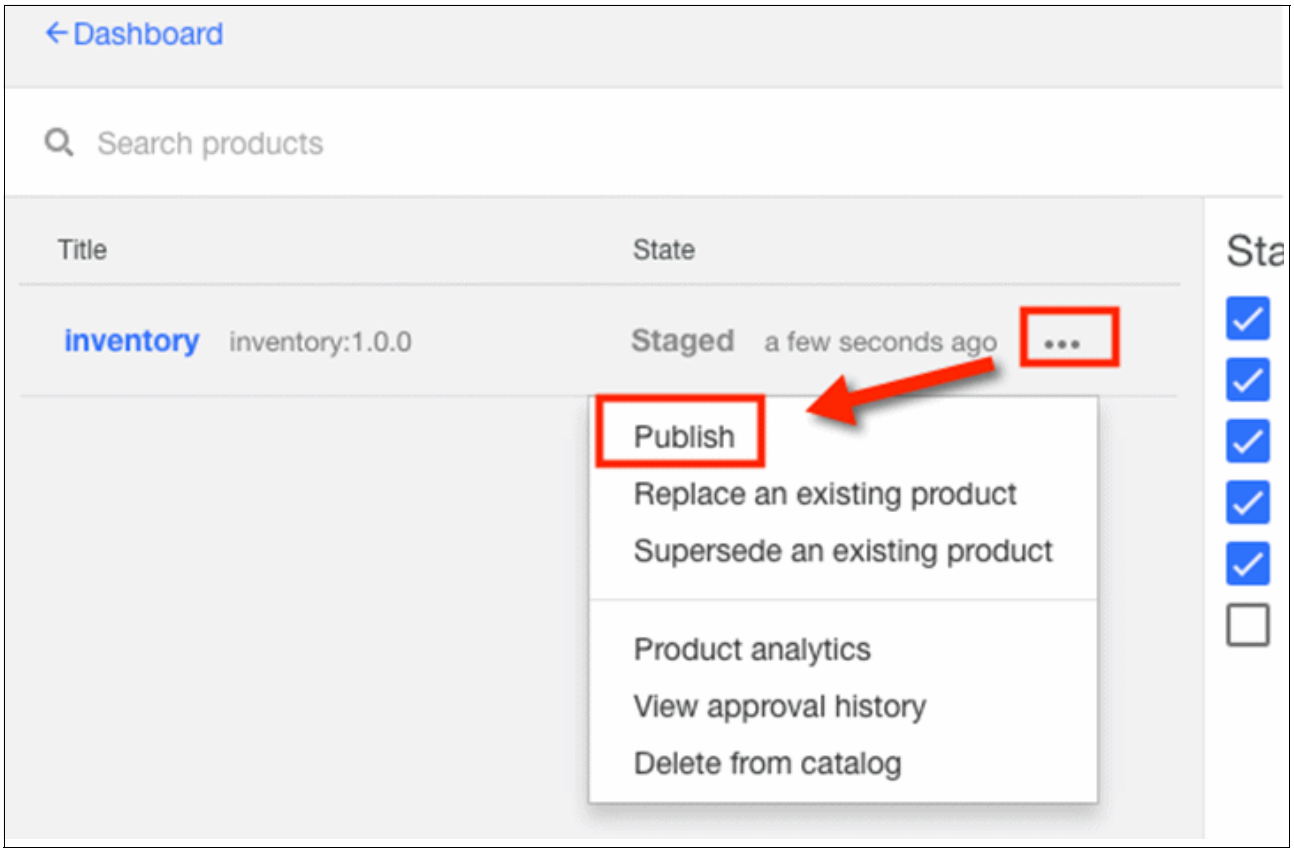

Figure 3-8 Select Publish

10. In the Edit Visibility and subscribers section, keep the default and click **Publish** button. See Figure 3-9 on page 62.

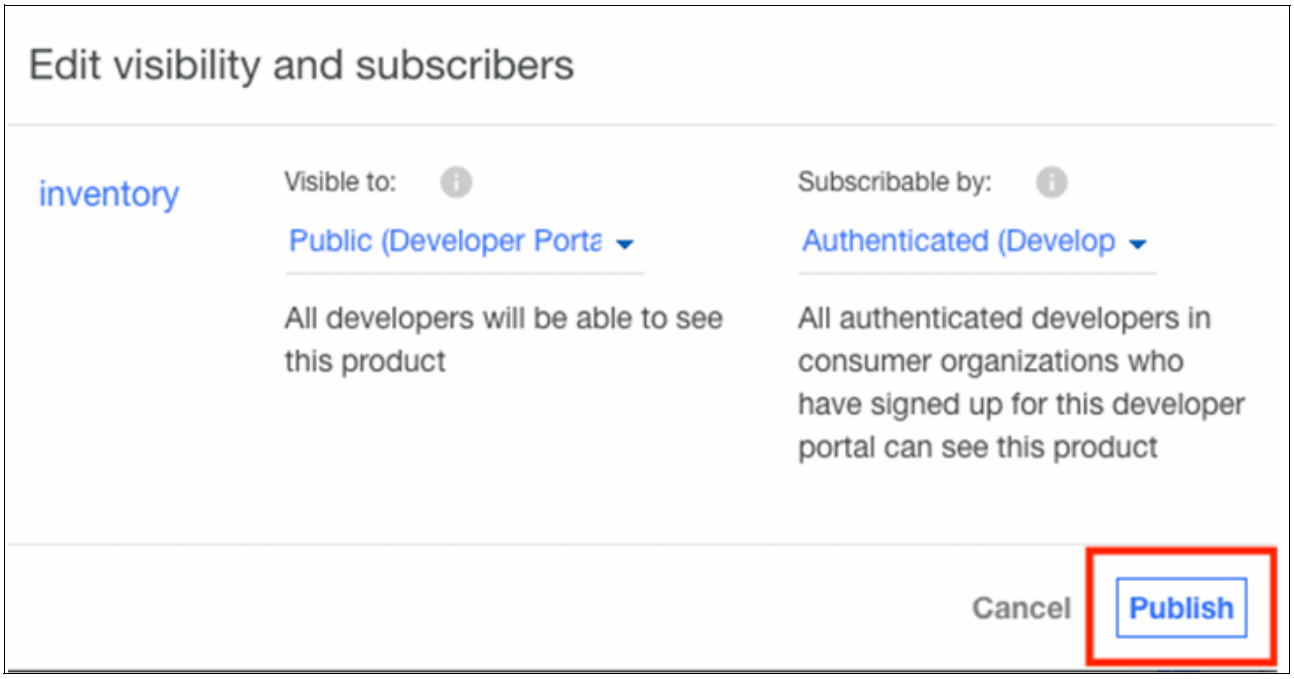

Figure 3-9 Publish the inventory product

11. The Inventory API version 1.0.0 is now ready to be consumed! From the dashboard, you should see the inventory product state has changed to *Published*.

3.2 Publish APIs with apic command line tool

Assuming you are working for CompanyC, it is time to publish the social review APIs. In this section, we want to take a different approach of publishing APIs. Instead of API Designer User Interface, we'll walk you through how to publish API and products using apic command line.

1. From the command line move to directory of Inventory application.
2. Login using **`apic login`** command as shown in Example 3-1.

Example 3-1 Login into your apic account using command line

```
c:\APIConnect\companyC\socialreviews>apic login
Enter the API Connect server
? Server: us.apiconnect.ibmcloud.com
? Username: rgupta1@us.ibm.com
? Password (typing will be hidden) ********
Generate a passcode from https://mccp.ng.bluemix.net/login/showpasscode.jsp
? Enter Bluemix one-time passcode: xxxxx
Logged into us.apiconnect.ibmcloud.com successfully
```

3. You will publish APIs and Product to ApicStore Catalog. So, you need to set the publish configuration variable using **`apic config`** command to the target catalog.

> **Tip:** You can get the information about this variable from the API Management Console Dashboard.

```
c:\APIConnect\redp5350-apiconnect-sample\socialreviews > apic config:set
catalog=apic-catalog://us.apiconnect.ibmcloud.com/orgs/rgupta1usibmcom-redbook/
catalogs/apicstore-catalog
```

Command syntax: This command has the following syntax:

```
apic config:set
app=apic-app://us.apiconnect.ibmcloud.com/orgs/{bluemixOrg}-{bluemixSpace}/a
pps/inventory-loopback-app
```

You need to replace the {bluemixOrg} and {bluemixSpace} with the ones you use in your environment.

4. Before you publish the API, it is always a good idea to validate your product and API definition with following command:

   ```
   c:\APIConnect\redp5350-apiconnect-sample\ socialreviews > apic validate
   definitions/socialreviews-product.yaml
   ```

   You should see the validation success message as shown in Example 3-2.

Example 3-2 Success message

```
validated socialreviews-product.yaml against API Connect product schema product definition
[socialreviews:1.0.0].
Successfully validated socialreviews.yaml against Swagger Version 2.0 schema API definition
[socialreviews:1.0.0].
Successfully validated socialreviews.yaml against API Connect swagger extensions schema API
definition [socialreviews:1.0.0].
Successfully validated socialreviews.yaml against IBM Swagger Version 2.0 schema API definition
[socialreviews:1.0.0].
```

5. Now, you need to push the socialreview Product to the Bluemix API Connect:

   ```
   c:\APIConnect\redp5350-apiconnect-sample\socialreviews> apic publish
   definitions/socialreviews-product.yaml
   ```

   You should get the following the message shown in Example 3-3 that shows operation was successful:

Example 3-3 Operation was successful

```
Staged definitions/socialreviews-product.yaml to gangchenusibmcom-apic:apicstore-catalog
[socialreviews:1.0.0]
Published definitions/socialreviews-product.yaml to gangchenusibmcom-apic:apicstore-catalog
[socialreviews:1.0.0]
```

6. Launch Bluemix API management console. Navigate to **Dashboard** from the top navigation pane. Click **ApicStore Catalog**. You should see both Inventory and socialreviews products in *Published state* in Bluemix.

   At this point, you should have both API products published to Bluemix API connect environment and ready to be consumed.

3.3 Summary

In this chapter, you learned how to manage the APIs from a provider perspective using the tooling provided by IBM API Connect on Bluemix.

4

Securing the APIs

This chapter discusses application security options available in API Connect and describes how each type of security is applied to the sample scenario used in the book. Note that this chapter provides more background material on security and the justification for the level of application security and end-user authentication that we have selected for the scenario. This information is interspersed with the configuration steps for the scenario.

This chapter has the following sections:

4.1 Overview

There are several areas where an API Provider must consider the type of security needed for their API.

- **Application identification and authentication**: Is the application required to identify itself by passing a *Client Id* during the API invocation? Is it required to pass a C*lient Secret*? Client id and secret can be thought of as userid and password credentials for an application. The client id also serves to identify an application which is important for monitoring, permitting and revoking access and rate limiting.
- **End user authentication and authorization:** If your API is accessing protected resources belonging to an end user (resource owner), then you probably need to apply OAuth security to your API. OAuth provides a mechanism for end-users to authenticate themselves and authorize an application to access resources on their behalf, while keeping their credentials secure and hidden from the application itself.
- **Establishing trust between the API Gateway and target application:** If your microservice or target application is running in a public cloud like public Bluemix, how can you ensure that only the API Connect Gateway is permitted to access the target microservice? If applications can target the microservice directly, then it might be possible to bypass security.

The following sections will go into more detail on each of these topics, and will describe how each type of security is applied to the inventory scenario.

4.2 Application security

When defining an API, you can require calling applications to identify themselves by passing a Client ID or a Client ID and a Client Secret. Requiring *at least a Client ID is recommended,* as this allows you to monitor usage of your API by specific applications, apply rate limits and even revoke access if an API consumer is *misbehaving* in any way. For the sample scenario, all API operations are secured with a Client ID. As discussed in the following section, some clients cannot reliably protect a Client Secret. For this reason, we will not require a Client Secret in this sample scenario.

1. To configure a Client ID security definition, navigate to the **Security Definition** section of your API. For this scenario, you need to set the clientID on the socialreviews and inventory API.
2. Select '**+**' and choose **API Key**. Fill out the section as shown in Figure 4-1 on page 67. You can specify that the caller passes the Client ID in either a header or query parameter. For our sample, we use a header as shown in Figure 4-1 on page 67.

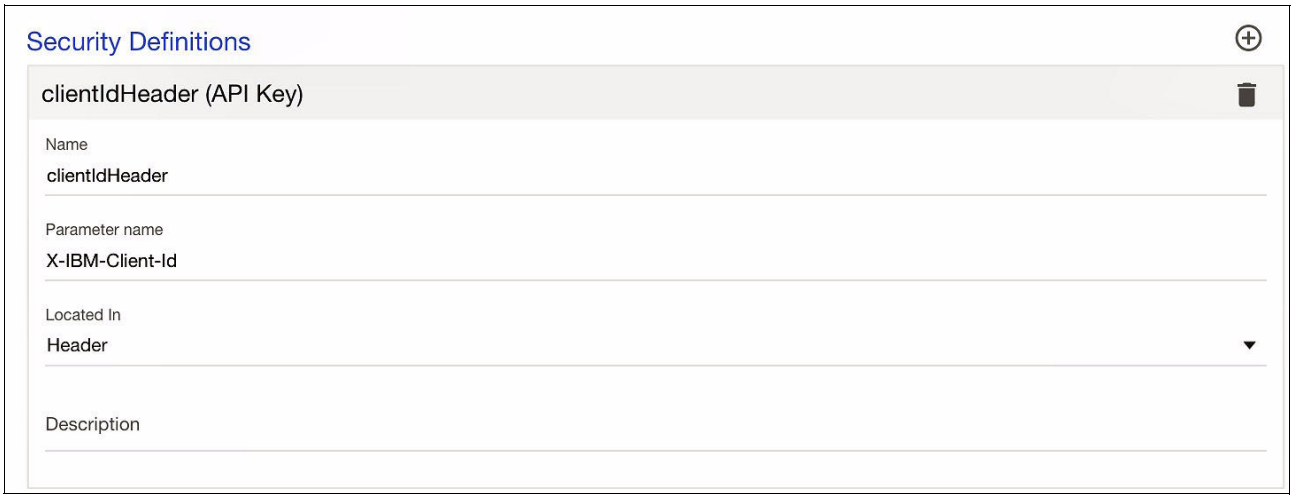

Figure 4-1 Security definitions

The following section will show you how to apply the Client ID APIKey security definition to your APIs, along with an OAuth security definition. In Chapter 5, “Consuming the APIs” on page 79, you will see how a client application, such as a mobile app, can access protected APIs using Client ID and OAuth.

4.3 OAuth Security to protect user resources

There are many use cases where an API provides access to end-user resources; for example, an API might provide access to a user's profile, documents or other assets. These assets are referred to as resources and the end-user of an application is often referred to as the resource owner. When an application uses an API to access resources on behalf of its end-user it is important that:

- The application can authenticate the end-user.
- The application doesn't gain access to the end-user's credentials.
- The end-user has an opportunity to authorize the application to access their resources.

OAuth 2.0 is a token-based authorization protocol that satisfies these requirements. By securing an API with OAuth 2.0, you can ensure that applications can securely and safely access resources that belong only to authenticated users that have authorized access to their resources. For detailed information on OAuth 2.0, refer to `https://tools.ietf.org/html/rfc6749`.

In the inventory scenario, the Social Review API allows a user to write and manage reviews. These reviews belong to the user. We will use API Connect to secure the write operations of the Social Review API with OAuth 2.0.

The first step to securing an API with OAuth 2.0 is to create a special type of API called the *OAuth Provider API*. Once the OAuth Provider API is configured, you must set up the security definition and security settings in the APIs that you want to protect.

4.3.1 Configuring the OAuth Provider API

Perform the following steps to conigure the OAuth Provider API.

1. If you haven't already, you should clone the lab material to your local environment by following the instructions in section “Clone the GitHub repository” on page 11.
2. If you haven't already, deploy the authentication utility application as described in Appendix A, “Deploying the authentication utility application” on page 121.
3. Go to the **redp5350-apiconnect-sample\OAuth** folder to edit the example **OAuth Provider API**, or you can create a new one.

> **Note:** At various points in this chapter we will make references to socialreviews and inventory folders respectively. They are organized under `redp5350-apiconnect-sample\socialreviews` and `redp5350-apiconnect-sample\inventory` folder structure.

4. The OAuth Provider API contains the authorization and token endpoints that are used in an OAuth 2.0 flow. In the API Connect Designer, navigate to APIs, select **+Add** and then choose **New** → **OAuth 2.0 Provider API**. Fill out the fields as shown in Figure 4-2:

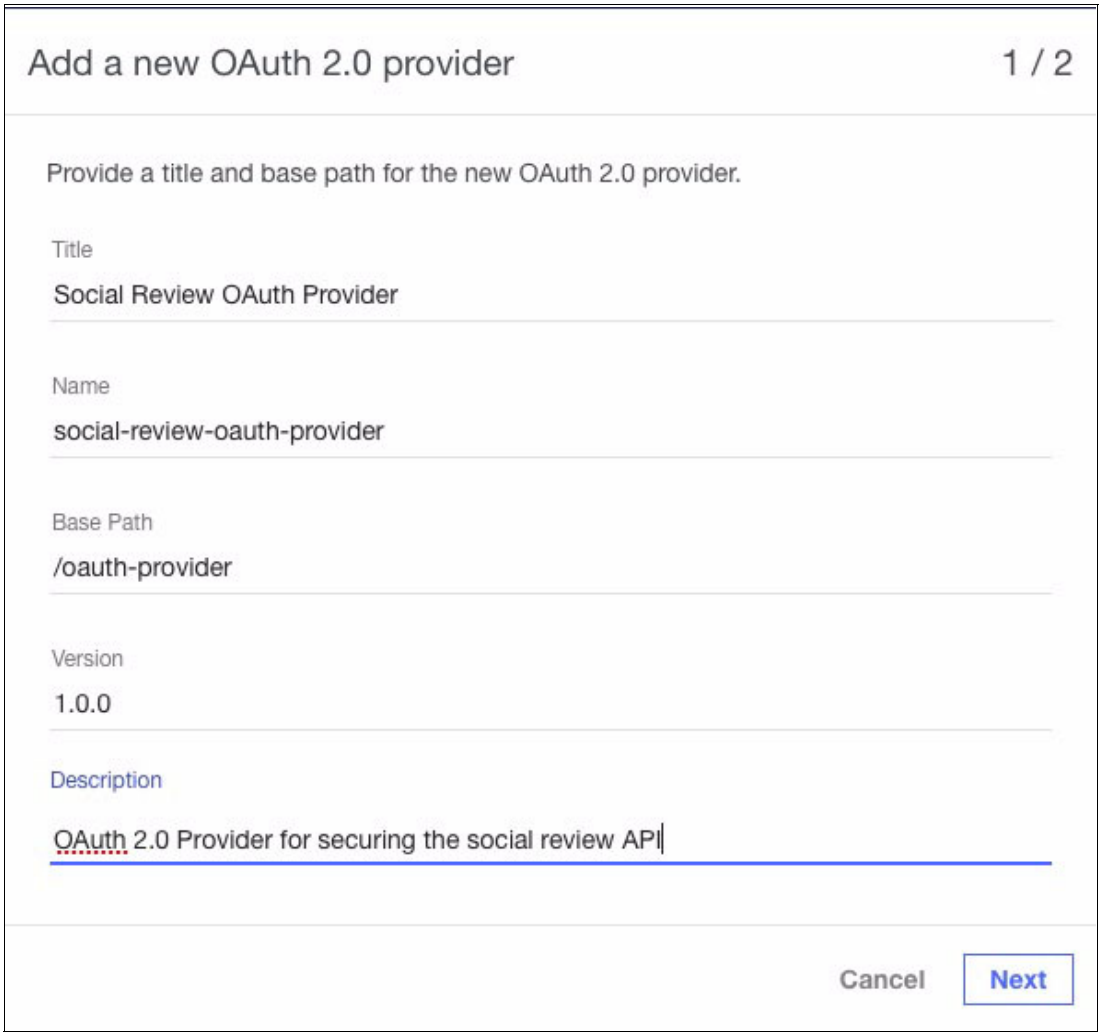

Figure 4-2 Add a new OAuth 2.0 Provider API

Note that you can use one OAuth Provider API to secure endpoints in multiple APIs; however, in the sample scenario, we are only using it to secure endpoints in the Social Review API.

5. On the next page, select **Don't add to a product** and then go ahead and push the **Add** button. Click **Save** and exit from the new API and return to your list of APIs. You should see the OAuth Provider API. Its type will be "*OAuth 2.0*".

Figure 4-3 New OAuth Provider API

Now, let's open the API again and have a closer look.

Client type

If you navigate to the OAuth 2 section, the first thing we need to decide is whether to secure the API for Public or Confidential clients. In OAuth 2.0, *client* refers to the application that invokes the API. Confidential clients are able to maintain the security of their client credentials, while public clients are not. Typically, native and browser-based applications are public, while those with a server-side component are confidential. The choice of Public or Confidential is based on the API Provider's acceptable exposure. You cannot make assumptions about the client. In this case, we will choose the Public client. This means, we will not require clients to protect or provide a client secret as part of the OAuth 2.0 flow.

Scopes

Scopes are used as labels representing the set of endpoints for which a token must be valid for, in order to grant access. In the OAuth Provider API, we list the scopes the OAuth Provider can secure. In this case, we just need one scope. Let's call it `review`. Create that scope and delete all others, so that the first two sections look like this (Figure 4-4):

Figure 4-4 Creating the scope

Grants

There are four types of grant flows to choose from. Detailed descriptions can be found in the API Connect Knowledge Center https://www.ibm.com/support/knowledgecenter/SSMNED_5.0.0/com.ibm.apic.toolkit.doc/tapim_sec_api_config_scheme_create_oauth.html and the OAuth 2.0 RFC https://tools.ietf.org/html/rfc6749. Complete flow diagrams can be found at both of these links. Below, is a brief overview of each these grants types.

Implicit

In the Implicit grant flow, the client calls the /authorize endpoint exposed by the OAuth API Provider. This redirects the end-user to authenticate and authorize the client access to their

resources. Once the end-user has completed this, a Bearer token is returned to the redirection provided by the client. The client then passes the Bearer token in an Authorization header to all protected endpoints. If the Bearer token is invalid, API Connect will reject the invocation; otherwise, the invocation proceeds.

This flow is optimized for public clients. The redirection, which is provided in an independent registration process by the client provides a level of assurance that the client is trusted. In API Connect, the redirect_uri is provided in the Developer Portal when the client is registered.

Password

In the Password flow, the client is trusted with the end-users credentials. It collects the credentials and then calls the /token endpoint exposed by the OAuth Provider API to exchange the credentials for a Bearer token. Then OAuth Provider validates the credentials and returns the Bearer token which the client must then pass in an Authorization header to all protected endpoints.

The token-based flow means that clients do not need to persist the user's credentials (unlike basic authentication where credentials must be sent on every call), but this flow should only be used if the API clients are known and can be trusted with the end user's credentials.

Application

In the Application flow, the client exchanges its credentials directly for an access token. There is no end-user authentication or authorization. This grant type is only applicable to Confidential clients and would be used for an application to access its own resources, or user resources that it is already authorized to access through some other mechanism.

Access Code

In the Access Code flow, both the /token and /authorize endpoints are used. Like the Implicit flow, the client first calls the /authorize endpoint which redirects the user to authenticate and authorize the client to access to their resources. An authorization grant is returned to the client as a query parameter of the redirect_uri that the client specified when they registered their application. The client then calls the /token endpoint, authenticating itself by passing its client credentials using basic authentication and passing the authorization grant in the body of the request. Assuming the client credentials and grant are valid, the OAuth Provider accepts the authorization grant and exchanges it for a Bearer token. The client then passes the Bearer token in the authorization header to all protected endpoints.

This flow is optimized for Confidential clients. If it is used by a Public client, then instead of authenticating the client credentials using basic authentication, the client_id is sent in the request body to the /token call, and the OAuth Provider must ensure that the grant was indeed made to the specified client_id.

4.3.2 Summary

In the first step, we selected a Public client type. This rules out the use of the Application flow which is not applicable for our scenario and can only be used by Confidential clients. For this scenario, we will use the Implicit Grant flow which is optimized for public clients. Unlike the password flow, it keeps end-user credentials protected. It provides a simplified exchange that allows the client to get the Bearer token with a single call instead of the additional call required by the Access Code flow.

In the Grants section, make sure **Implicit** is selected and deselect all other Grant types. See Figure 4-5 on page 71.

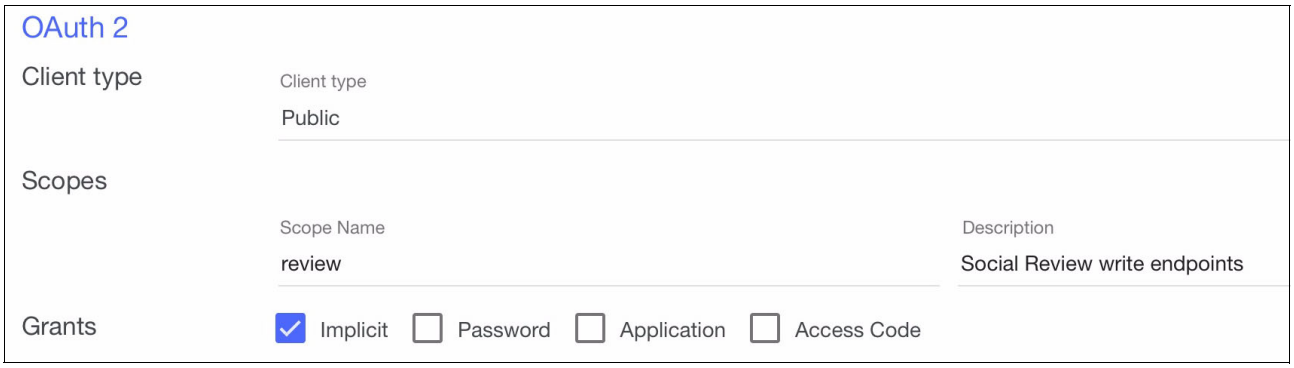

Figure 4-5 Implicit is selected

4.3.3 Identity extraction

The *Identity extraction* is where you specify the means by which the end-user's (resource owner's) credentials will be collected. Remember that for all Grant Types except *Password*, the application itself should never have access to the end-user's credentials. API Connect provides four options for Identity extraction:

Default form

The end-user is presented with a generic form that is provided out-of-the-box by API Connect.

Basic

This will use the native basic authentication support provided by your user agent. In Firefox, this would popup a dialog like the following (Figure 4-6).

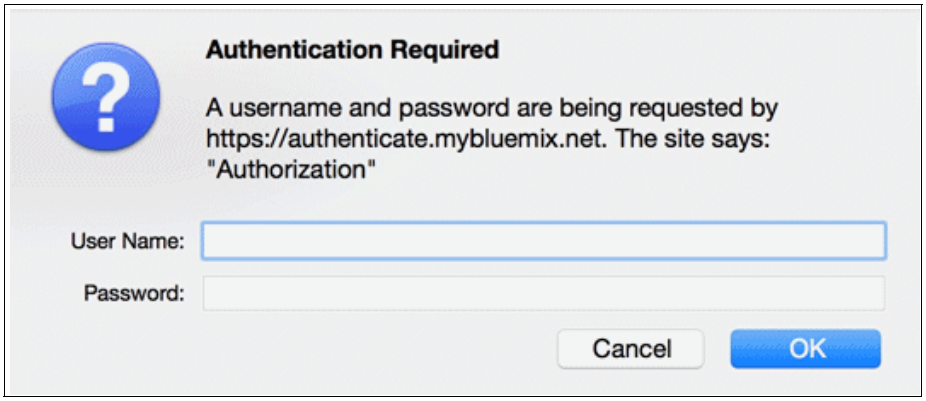

Figure 4-6 Basic authentication

Custom

You can provide a custom HTML form to API Connect. With this option, you most host the HTML form and provide the URL where it can be found.

Redirect

Use an externally hosted service for authentication and authorization. You provide API Connect with the redirect URL and then handle the authentication and authorization steps, following the protocol described in the Knowledge Center at this link:

http://www.ibm.com/support/knowledgecenter/SSFS6T/com.ibm.apic.toolkit.doc/task_apionprem_redirect_form_.html

In our scenario, we are going to provide custom authentication and authorization forms. These will be hosted on Bluemix in a simple Cloud Foundry node application. The application with these forms is located in folder `redp5350-apiconnect-sample/OAuth/authentication-app`. To deploy this application:

1. Navigate to redp5350-apiconnect-sample/OAuth/authentication-app.
2. Edit the `manifest.yml` file to give the application unique app and host names.
3. Make sure you have the Cloud Foundry CLI.
4. Type **`cf login`** to log in and answer the prompts to target the Bluemix Organization and Space to which you will deploy this application.
5. Type **`cf push`** to deploy the application to Bluemix.

Now that the authentication-app is deployed to Bluemix, fill in the URL for the login form. This should be <route>/login.html, where <route> is the route configured in Bluemix for your application. See Figure 4-7.

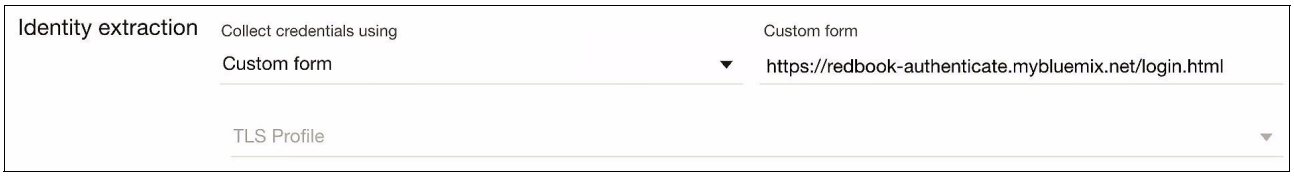

Figure 4-7 Custom authentication form

Note the TLS profile field. If you wish to set up mutual authentication to restrict access to the custom form, then you can configure a TLS profile in API Connect and a Custom Domain in Bluemix. For simplicity, we will not configure mutual authentication for this custom form.

4.3.4 Authentication

Next you must specify how the user will be authenticated. You have the following options:

1. Provide an authentication endpoint which will accept a basic authentication header and return a status code of 200 if the credentials are valid and 401 if they are invalid.
2. Define an LDAP registry and select that registry. An LDAP registry can currently only be defined in the API Manager console and not in the API Designer. Also, when using API Connect in Bluemix, the specified LDAP registry must be accessible from the internet.

For our scenario, we will use an Authentication URL. This endpoint is also implemented in the authentication-app that we previously deployed to Bluemix. At the <route>/authenticate endpoint, it checks the basic authentication header and validates against a known set of userid and passwords. The code for this can be found in gitrepo/OAuth.js.

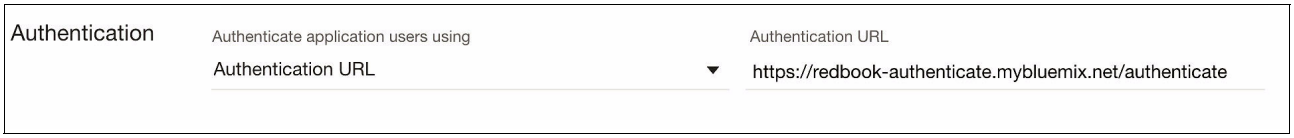

Figure 4-8 Authentication URL

Again, you can set up mutual authentication to restrict access to this authentication endpoint. For simplicity, we will not configure mutual authentication in this sample scenario.

4.3.5 Authorization

Next you must configure how the end-user will authorize the application to access their resources. There are three options:

Default Form

If you select this option, then API Connect will present a generic out-of-the-box form for the user to accept.

Custom Form

If you select this option, then similar to the custom for Identity Extraction, you must host an custom HTML form which prompts the user to authorize the application to access resources. Detailed instructions can be found here in the Knowledge Center:

http://www.ibm.com/support/knowledgecenter/SSFS6T/com.ibm.apic.toolkit.doc/task_apionprem_create_a_custom_authorization_form.html

Authenticated

If you select this option, then it assumes that if the user has authenticated, they have implicitly authorized access. Note that if you choose the redirect option for Identity Extraction, then you are responsible for both identity extraction and authorization and must select this option.

For our sample scenario, we will use a custom form. Sample XHTML can be found. This is what the form looks like (Figure 4-9):

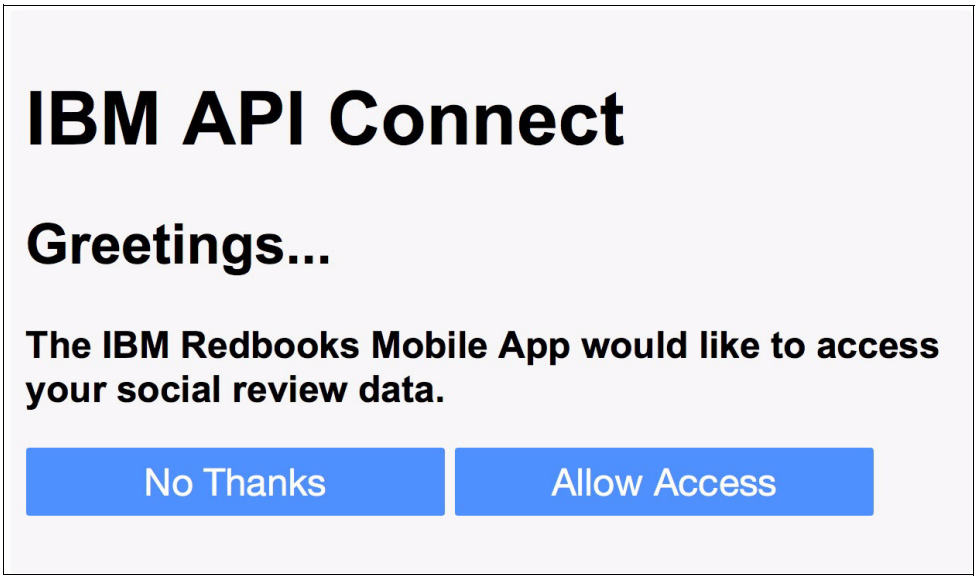

Figure 4-9 Custom form

We will again host this form on Bluemix and enter the URL into API Connect. If desired, you can set up mutual authentication to restrict access to this form.

4.3.6 Tokens

Note the *Time to live* setting of 3600 seconds. By default, the access token will expire after one hour and for the Implicit Flow, the application will need to go through the authorization process again to get a new access token. Other OAuth flows support a refresh token process

where the refresh token can be issued with the original access token. Once the access token expires, the refresh token can be exchanged for a new access token.

Leave the default setting at 3600 seconds and deselect both **Enable refresh token** and **Enable revocation URL**. Your OAuth Provider API configuration should look like this (Figure 4-10).

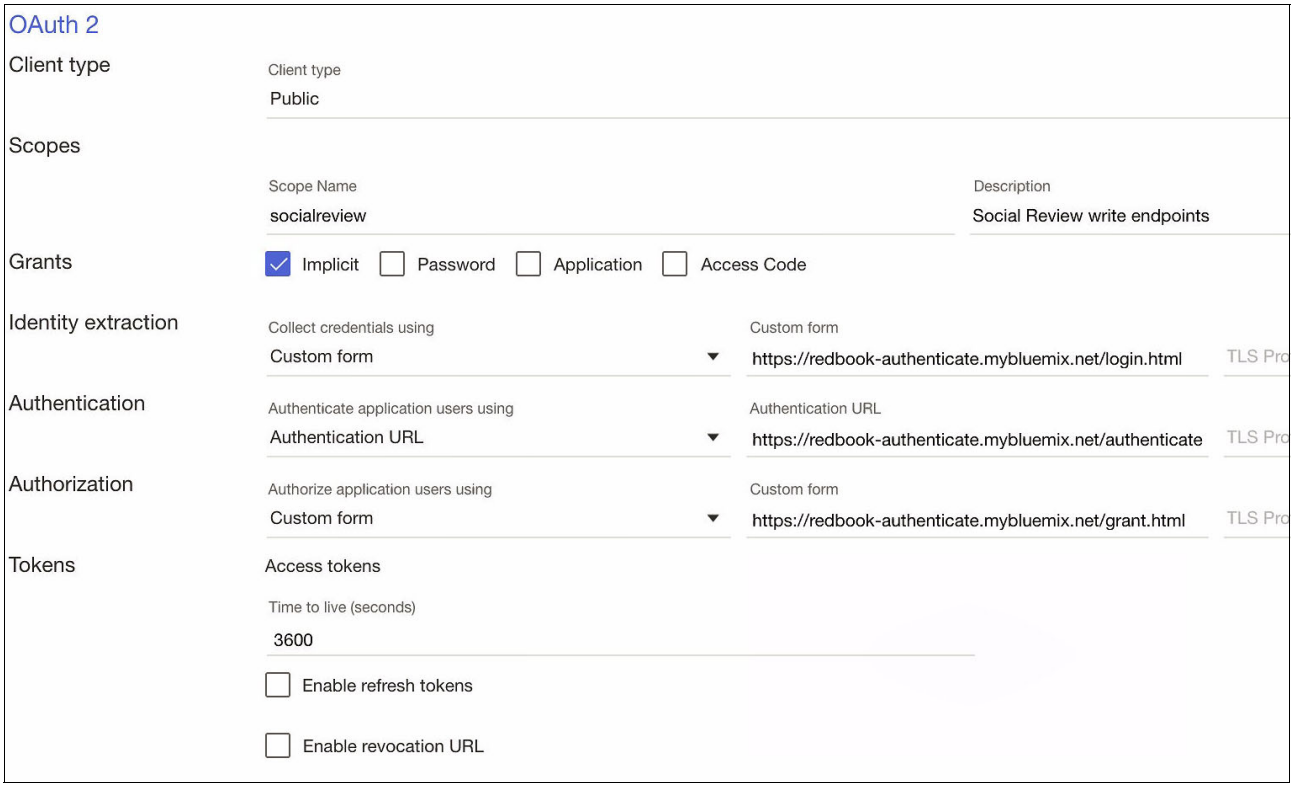

Figure 4-10 Auth Provider API configuration

Completion of OAuth Provider API

Make sure that Produces and Consumes are configured to application/json. For the rest of the configuration, accept the defaults and save the API.

4.3.7 Securing your API

Now that your OAuth Provider API is configured, you need to define the security definition on the API that you want to protect.

1. Before we get started configuring the API, we're going to need the URL for the authorization endpoint of the OAuth Provider API that we just completed. This URL is catalog specific, so let's navigate to the **Inventory** Catalog. If you are in the API Designer, you may need to open the API Manager UI.
2. Go to the **Settings** tab of the Catalog, and look for the API Endpoint Base URL. It will look something like Figure 4-11.

Figure 4-11 Base URL

The full Authorization URL will be `<Base URL>/oauth-provider/oauth2/authorize`.

3. Now that we have the authorization URL, navigate to the **Social Review API** in the Security Definitions section.
4. On the right-hand side, click the '**+**' and then select **OAuth** to create an OAuth security definition. Search below, and you should find the definition you just created. It will look like Figure 4-12.

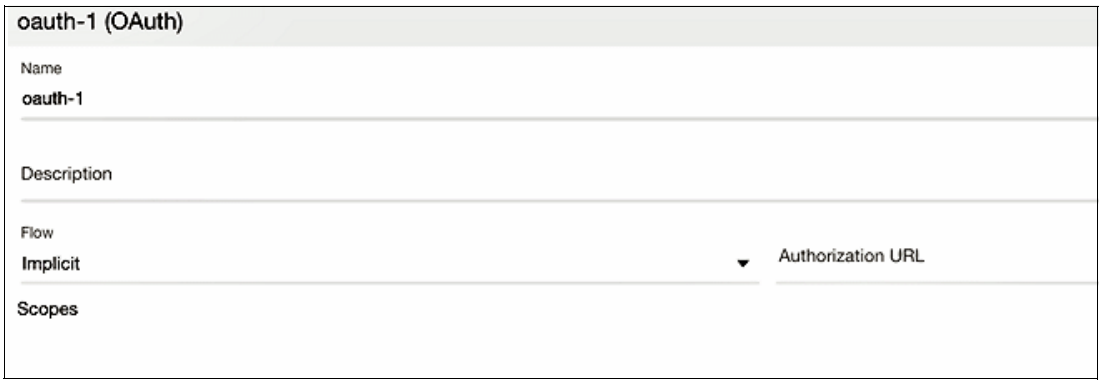

Figure 4-12 OAuth security definition

5. Give it the name `OAuth Definition` and optionally give it a Description. Make sure the Flow is set to **Implicit**.
6. You must now give it the Authorization URL provided by the OAuth Provider API that is protecting this API. This is where you put the Authorization URL that we just looked up.
7. Finally, in the Scopes section, put `socialreview`. This must match the scope we used in the OAuth Provider API. The scopes in the OAuth Provider API indicate all the scopes that it can protect. The scope(s) listed in the Security Definition list the scope(s) that the caller must specify in order to be granted access to this API. Once completed, your configuration should look similar to Figure 4-13.

Figure 4-13 OAuth Definition

8. Scroll down to the Security section for the API and make sure that your OAuth Definition is selected as shown in Figure 4-14 on page 76.

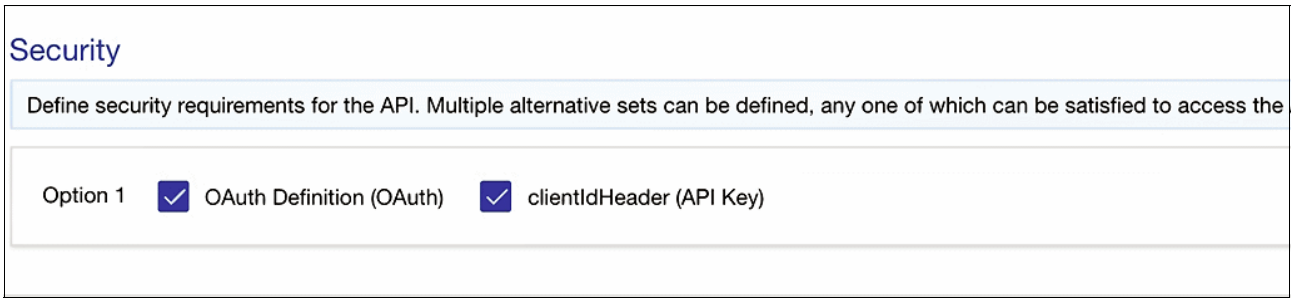

Figure 4-14 Security section

9. We don't need OAuth on every operation, just the operation where the user is creating a review. Expand each operation and look at the Security section.
10. For the all operations except the POST reviews section, deselect **Use API security definitions** and make sure only the **clientIdHeader (API Key)** box is checked. In this way, you can override the API security definition and select just the types of security you need per endpoint. Once you've made this update, you should see the following (Figure 4-15 on page 77).

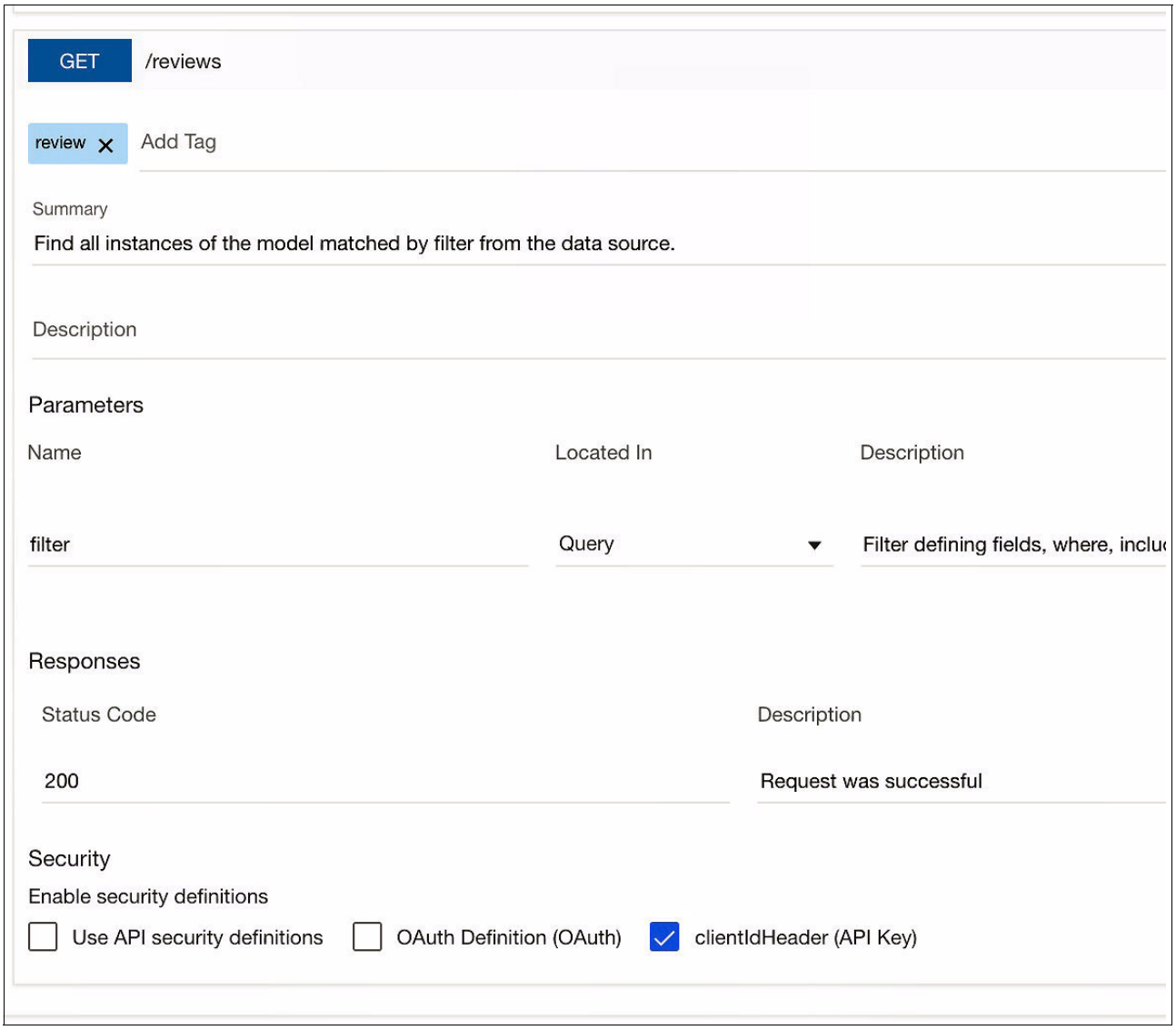

Figure 4-15 clientIdHeader (API Key) box is checked

11. For POST /reviews, you will use the API security definitions as seen in Figure 4-16.

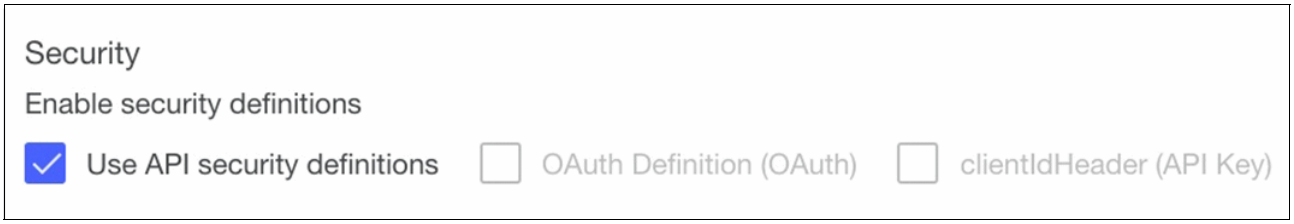

Figure 4-16 Use API security definitions selected

12. Now you can save the API. Make sure the OAuth Provider API and the Social Reviews API are both in your Product and publish to the Inventory catalog.
13. At this point follow the instructions in 3.2, “Publish APIs with apic command line tool” on page 62 to publish the APIs.

4.4 Summary

In this chapter we have discussed the importance of controlling access to your target application endpoints so that only calls from the API Connect Gateway are accepted by the target application. We have covered mutual authentication and basic authentication as techniques for protecting the target application. We have also shown how API Connect will automatically configure mutual authentication when you use the API Designer to publish LoopBack applications to Bluemix.

5

Consuming the APIs

This chapter describes the steps necessary for an application developer to consume an API.

In this chapter we provide a description on how an application developer can discover and consume the APIs provided by IBM Connect in the development of their applications. We present two different scenarios to describe the process.

- In the first scenario we present the steps needed for a Mobile developer to consume the APIs available in API Connect. In this case, you will experiment how to build an iOS application to consume the inventory and socialreview APIs.
- In the second scenario, we present the steps needed for a Web developer to consume the same APIs.

This chapter has the following sections:

5.1 Sign up to use API

In this section we will show how the developers can consume APIs produced by the API developers using the API Connect Developer Portal.

The Developer Portal enables API providers to build a customized developer portal for their application developers. It also provides the interface for API consumers to discover APIs and subscribe to a consumption plan by which the API is consumed in either the mobile or web application.

In reality, the Mobile and Web application might very well being developed by different digital agencies or teams. For simplicity, we'll assume that both Mobile and Web application are developed by the same team and they will consume the same API plan in the Developer Portal.

As an API consumer, the common path to consume an API will be:

1. Browse available APIs
2. Register an application
3. Subscribe to a plan
4. Test the API in developer Portal
5. Consume the API

Let's get started.

5.1.1 How to register an application

The first step in order to consume APIs is the API discovery. To do this, you need to log on to API Connect Developer Portal.

Note: There are different options to get the developer portal depending on how the API providers want to expose the APIs such as establishing developer community or email the developer portal invitation. In our walk through case, we'll assume that the API providers and API consumer are the same team who shares the API connect environment. Thus, the developer portal created in Chapter 1, “Introduction to API Connect and environment setup” on page 1 is known to the API consumer here.

1. To get the developer Portal URL, login to your Bluemix API Management console.
2. Navigate into the **ApicStore Catalog**. Click **Settings** tab, then click the **Portal** subtab. You should see the IBM Developer Portal URL. Copy that. Figure 5-1 on page 81 shows the Developer Portal URL the API Developer.

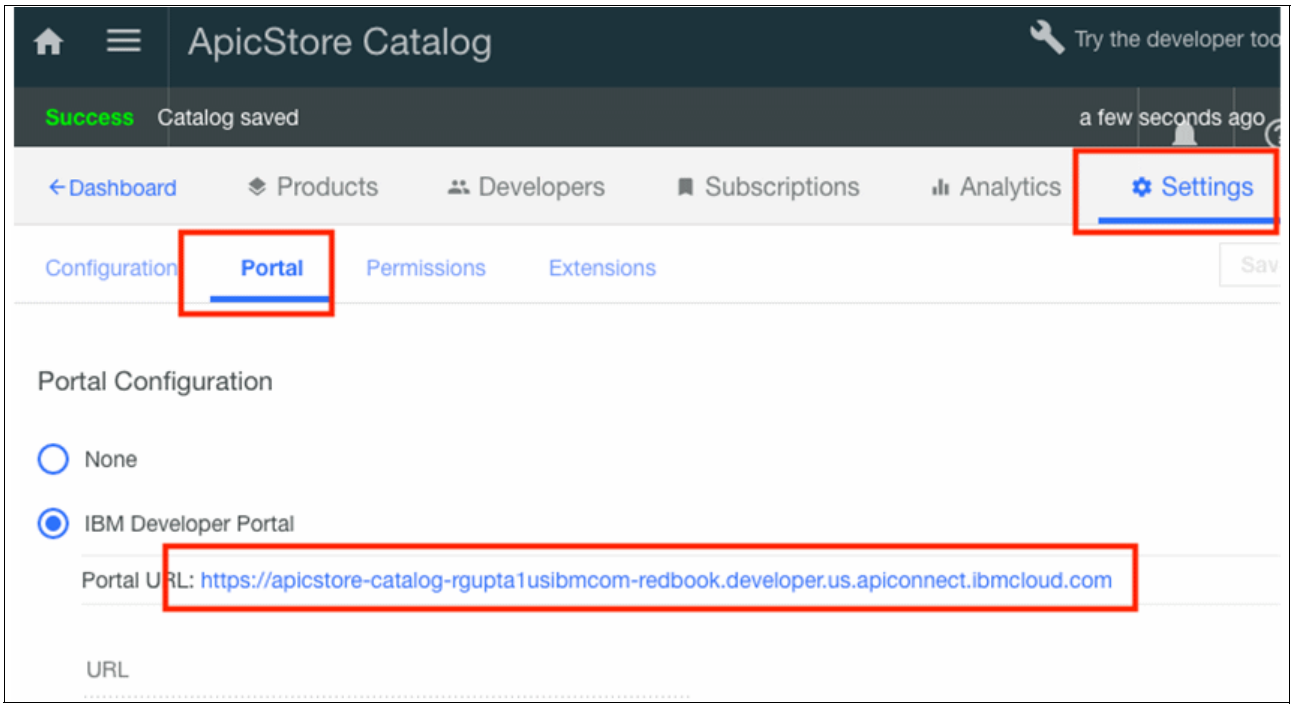

Figure 5-1 View from the API Developer, showing the Product URL he needs to send the API Developers

3. Open the Developer Portal in another browser window. You should see the portal home page with both inventory and socialreviews APIs highlighted, as shown in Figure 5-2.

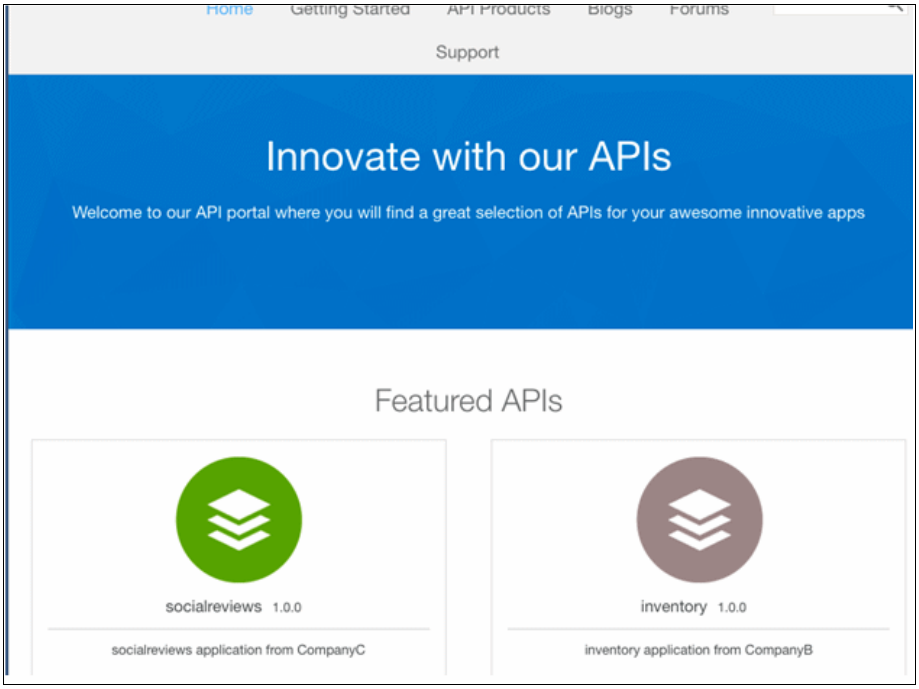

Figure 5-2 API Developer Portal main page

4. To use the API Developer Portal, the user will need to log in using the credentials on the right hand side on the top of the page. Click **Login**, and use your username and password. In this walk-through, we will assume that you don't have a username and

password, so click on the link to **Create a user account**. Figure 5-3 on page 82 shows the form you need to use to create a new user to consume the APIs.

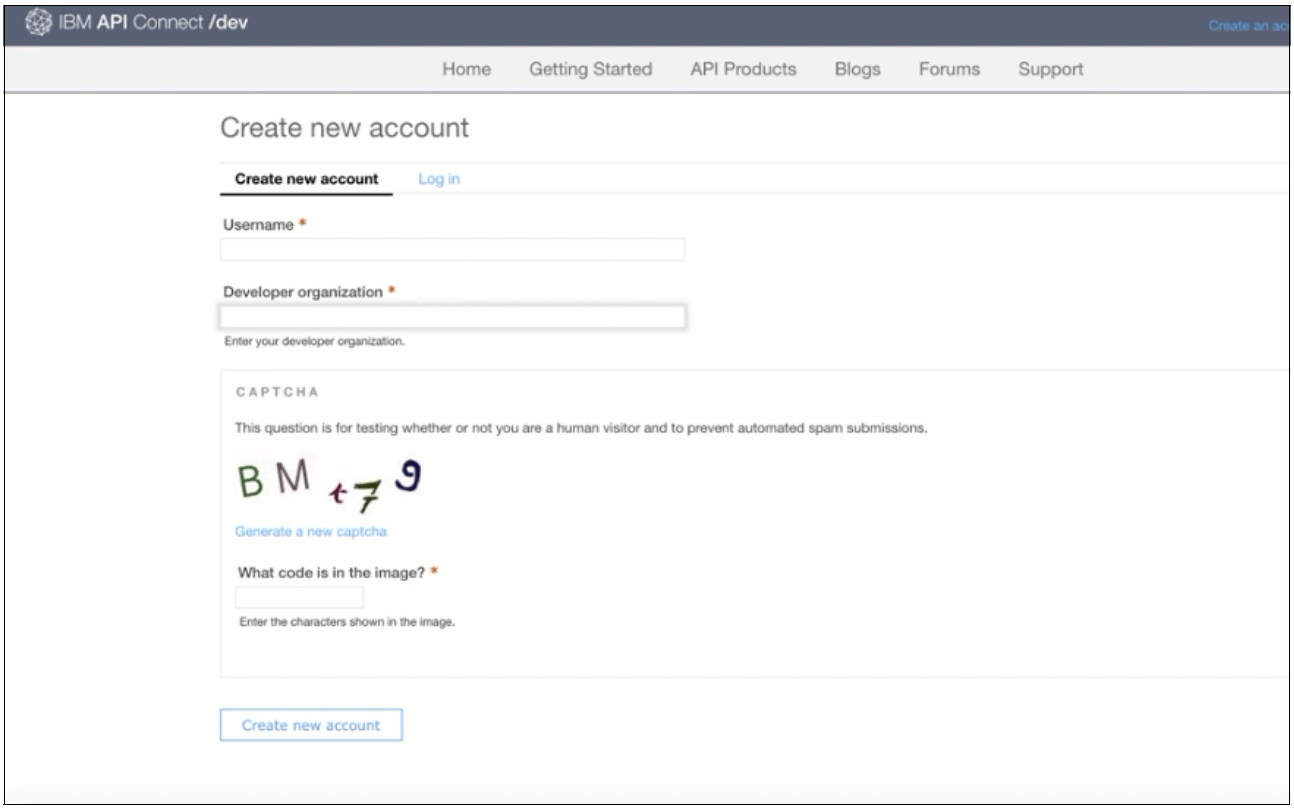

Figure 5-3 How to create a new user account at the API Developer Portal

The username will be your Bluemix login account or a valid IBM ID.

5. Enter the "ApicStore-App-Dev" as your developer organization.
6. Click **Create new account** button.
7. Then, click **Login.** Enter your IBM ID credential to login.
8. Once you log in, click on the tab **API Products** to start discover APIs you want to consume. We will select the product Inventory. In the page you can see additional details related to the Product we are selecting. You can see the version, and if the Product is online or offline. Figure 5-4 on page 83 shows the view of different products available to the App developer.

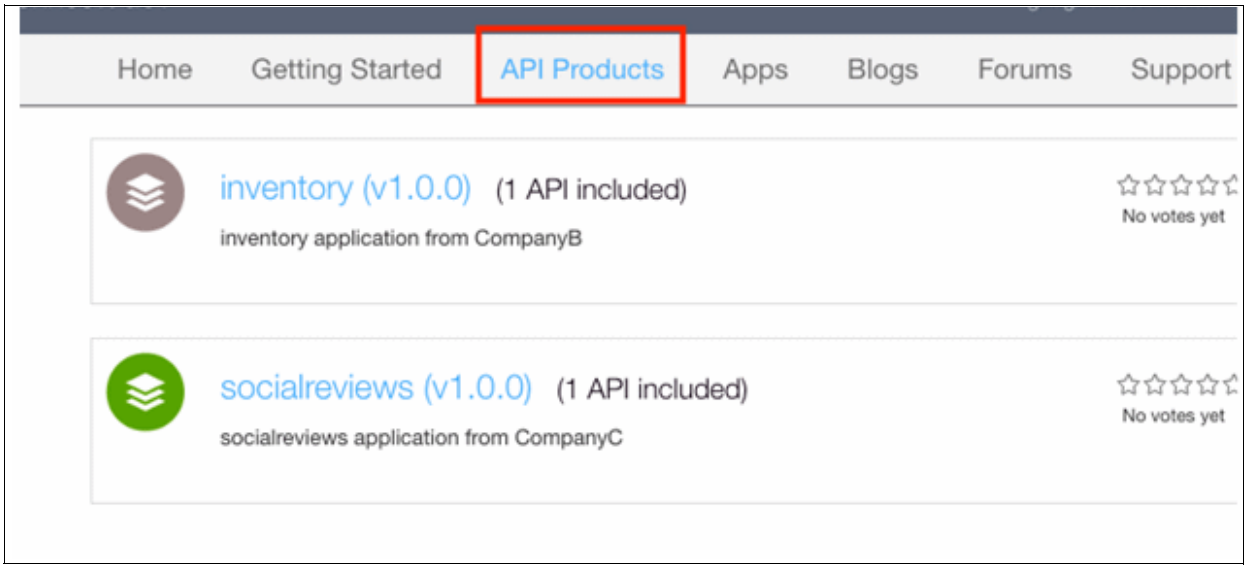

Figure 5-4 API Products page

9. Let's create a development application in order to subscribe to API. To do that, click on the menu tab **Apps** on top. Click on the link to **Register a new Application**. See Figure 5-5.

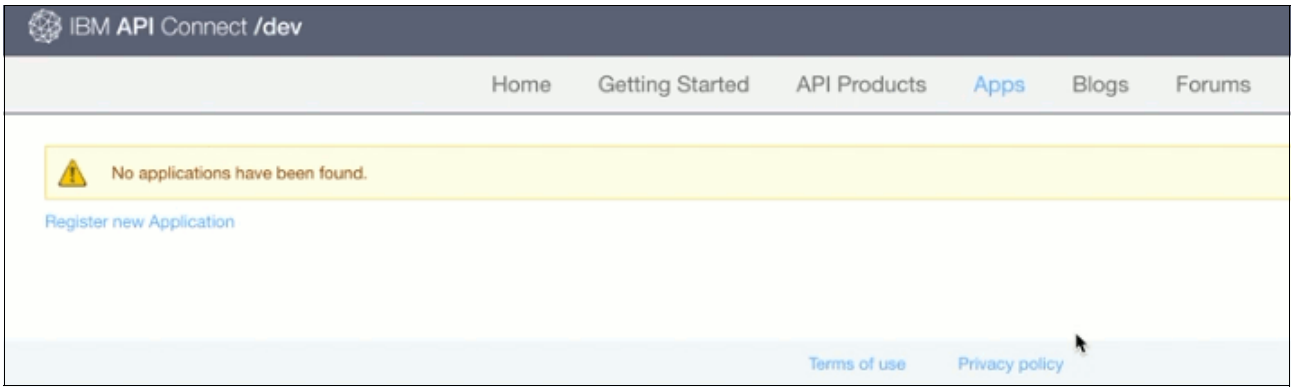

Figure 5-5 New app registration

To register your new application, you will need to fill the fields of **Title** and **Description**. Enter `org.apic://example.com` for the OAuth URI (redirection page). See Figure 6-6 on page 96. Then click **Submit**.

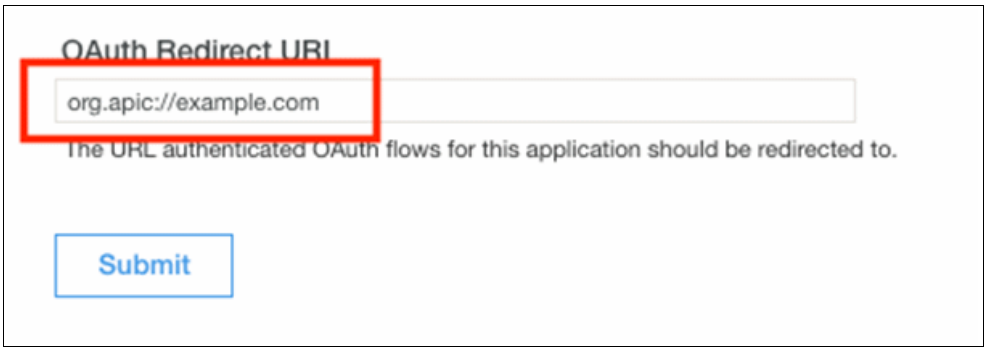

Figure 5-6 Registering a new application

10. The new application will be registered, and you will receive a screen similar to Figure 5-7. It is important that you store your **Client secret**, as you will need it to consume the API. In the same way, you need to capture and store the **Client ID**. You will need them when building your mobile and web application.

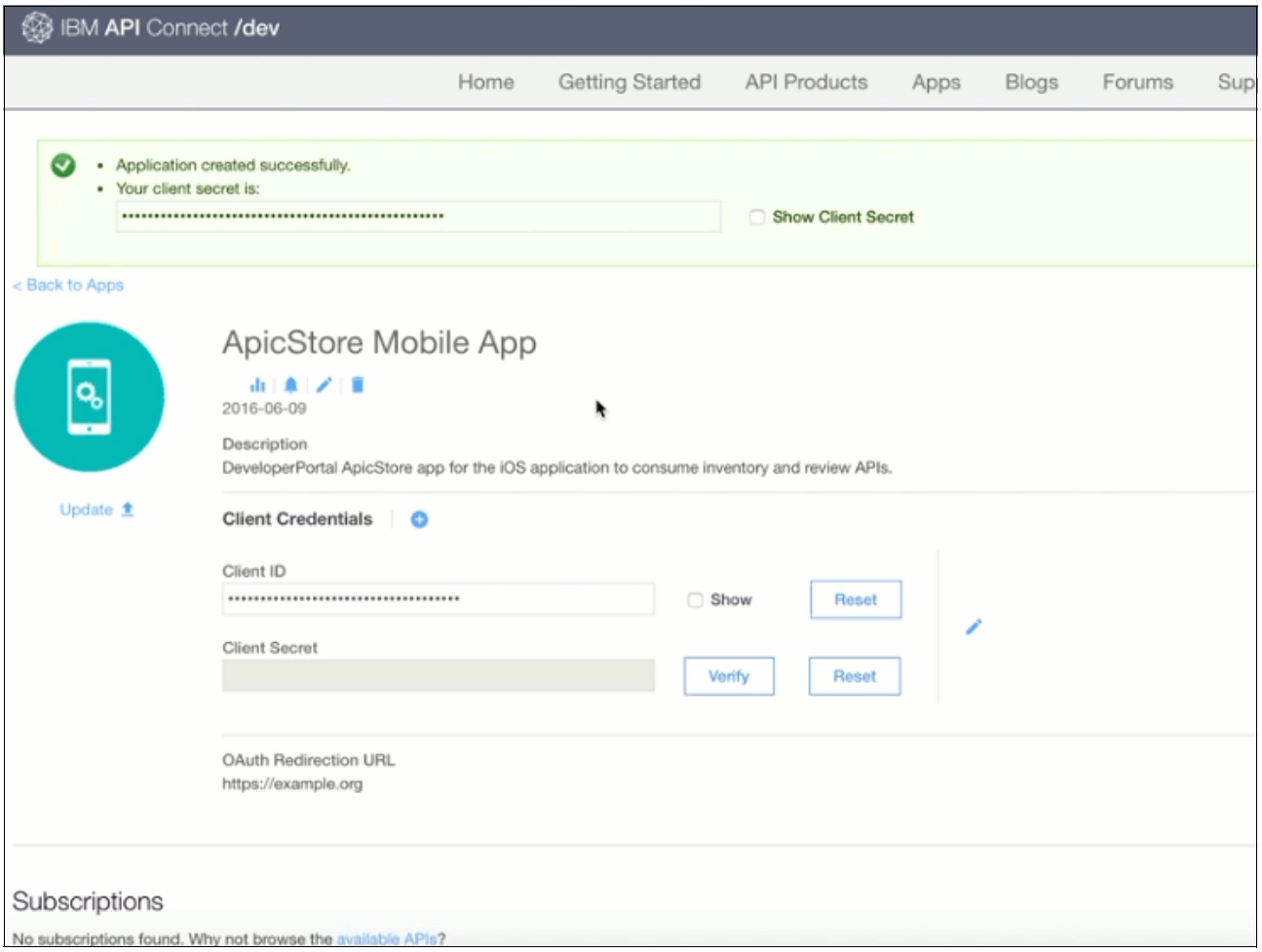

Figure 5-7 Application created using API Developer Portal

5.1.2 How to subscribe to an API plan

Once you have registered a new application with the API Developer portal, you are now ready to consume the APIs published in your new application.

1. The next step to consume the APIs is to Subscribe to the available APIs. Below your new application, you will see a link to take you to the current *available APIs*. Click on that **link**.
2. 2.You will be directed to the API product page, as shown in Figure 5-8. To subscribe to one of the APIs, click on one of the available **API products**. Figure 5-8 shows the API Product page for the sample Social Reviews API we want to subscribe to.

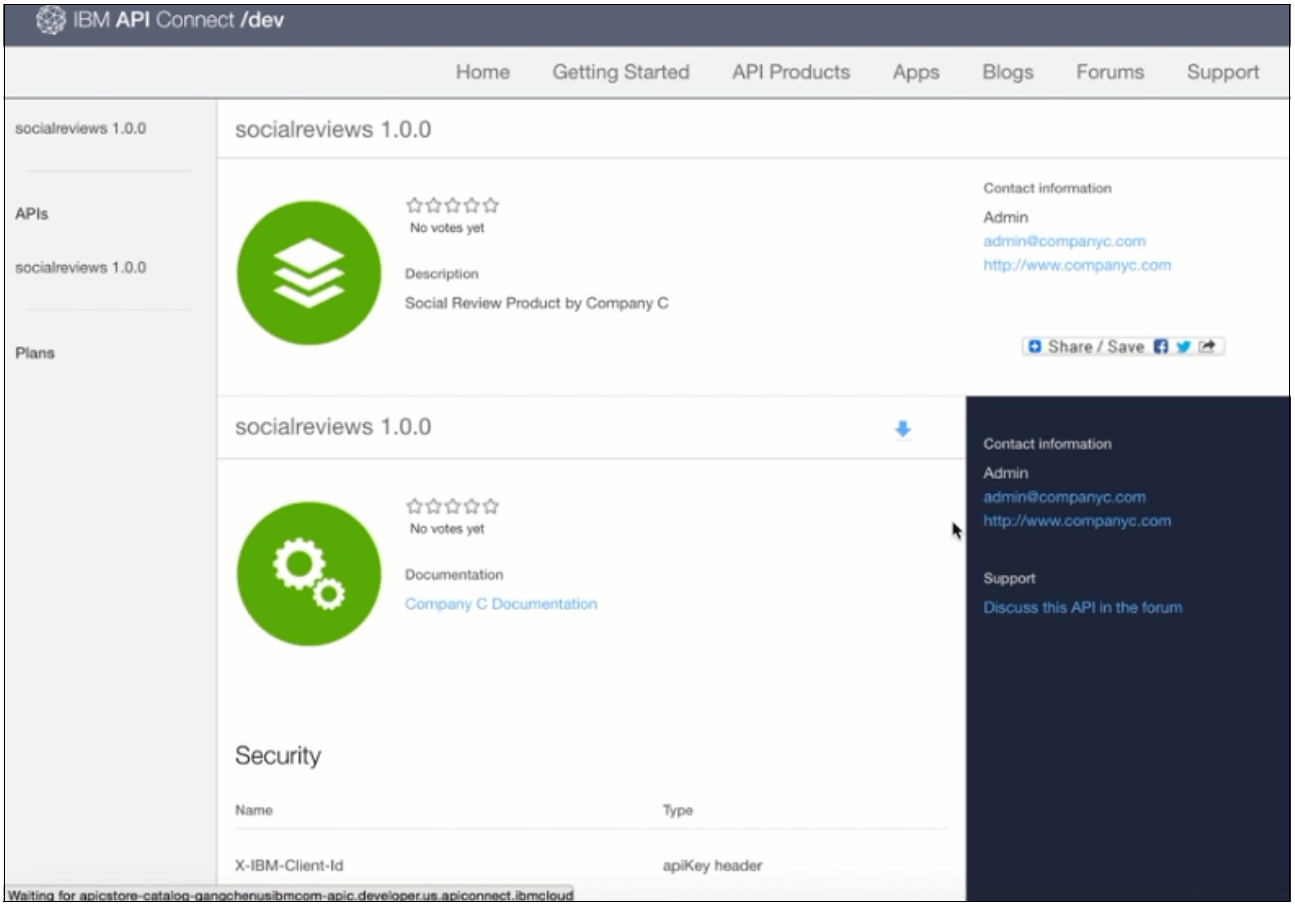

Figure 5-8 API Developer portal showing API Product page

3. To subscribe to this API, click on the link **Subscribe** in the API page, as shown in Figure 5-9.

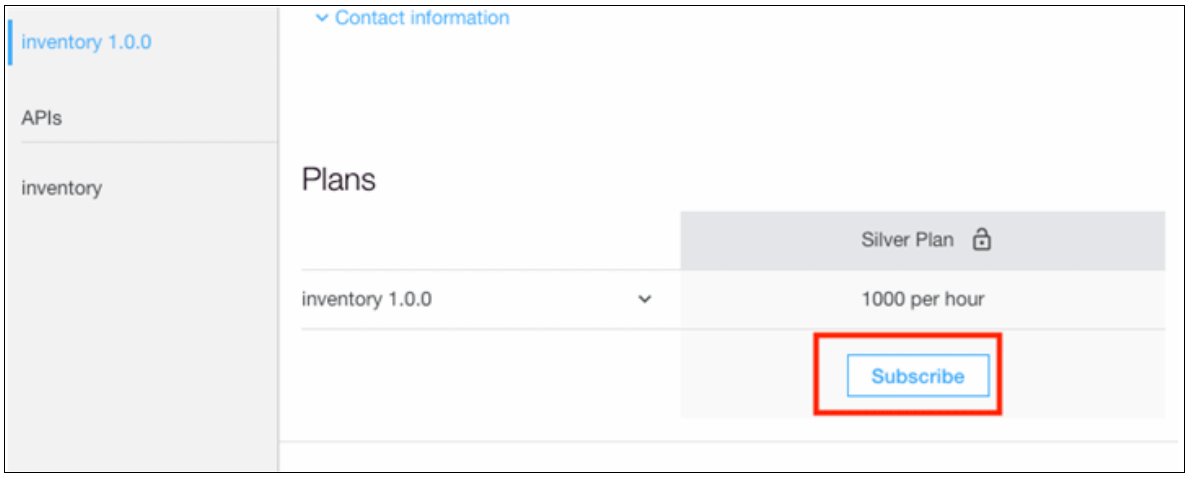

Figure 5-9 API Subscription page

4. Subscribe to the socialreview API, as well (you can choose either silver or gold plan).
5. Now navigate back to the **Apps** → **ApicStore Mobile and Web App page**. You will see both APIs are subscribed in your Application page. See Figure 5-10.

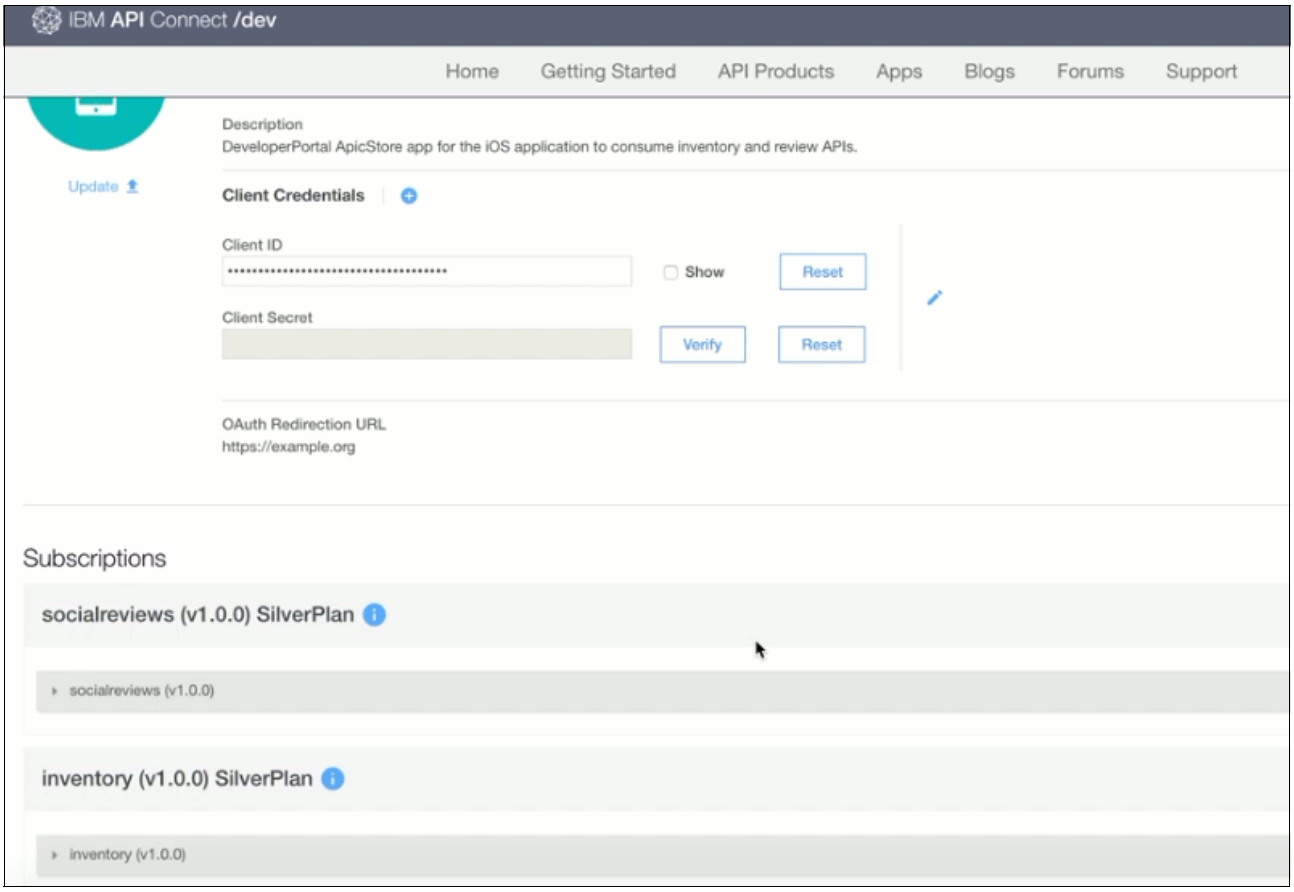

Figure 5-10 Product page showing the new subscriptions for the developer

5.1.3 How to test an API from the development portal

One of the useful features of the API Developer Portal is to test the APIs. Let's test one of the APIs subscribed.

1. Go to the API Developer Portal, and click on one of the **product APIs**. Then click on **inventory (v1.0.0)**.
2. In the left navigation menu under APIs, click on **inventory 1.0.0.** It will expand and show all available APIs you subscribed to, as shown in See Figure 5-11.

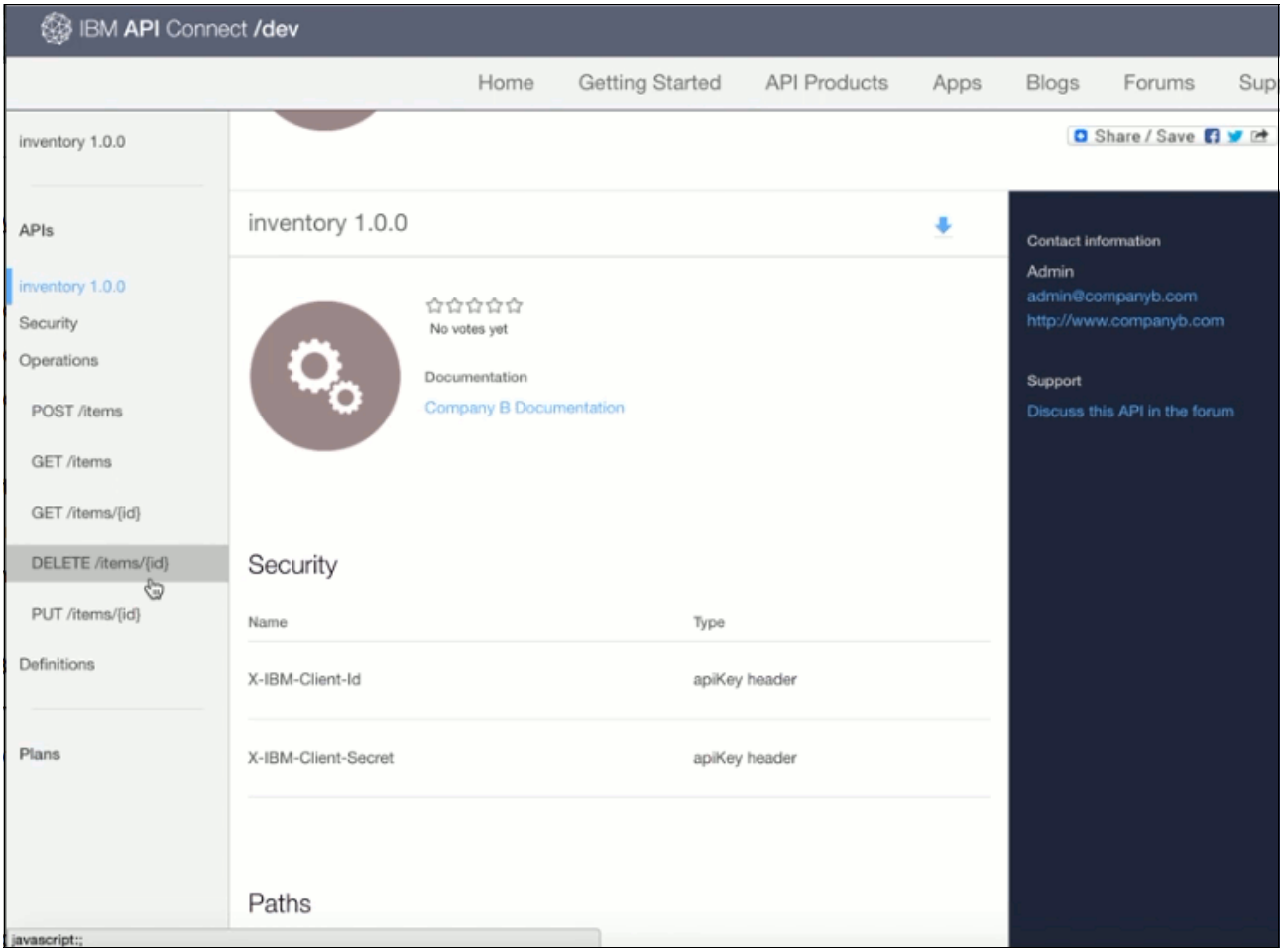

Figure 5-11 Product page with API options displayed

3. On the left menu, click on the **Operation GET /items**. This will bring up the API detail in right content pane. Figure 5-12 on page 88 shows one of the APIs displayed. On the right menu, you will see that the Developer Portal gives you many options to test the API, and sample code in different programming languages. The first tab corresponds to $cURL$. cURL is a tool available in most *NIX systems, which the developers can use to test the API access.

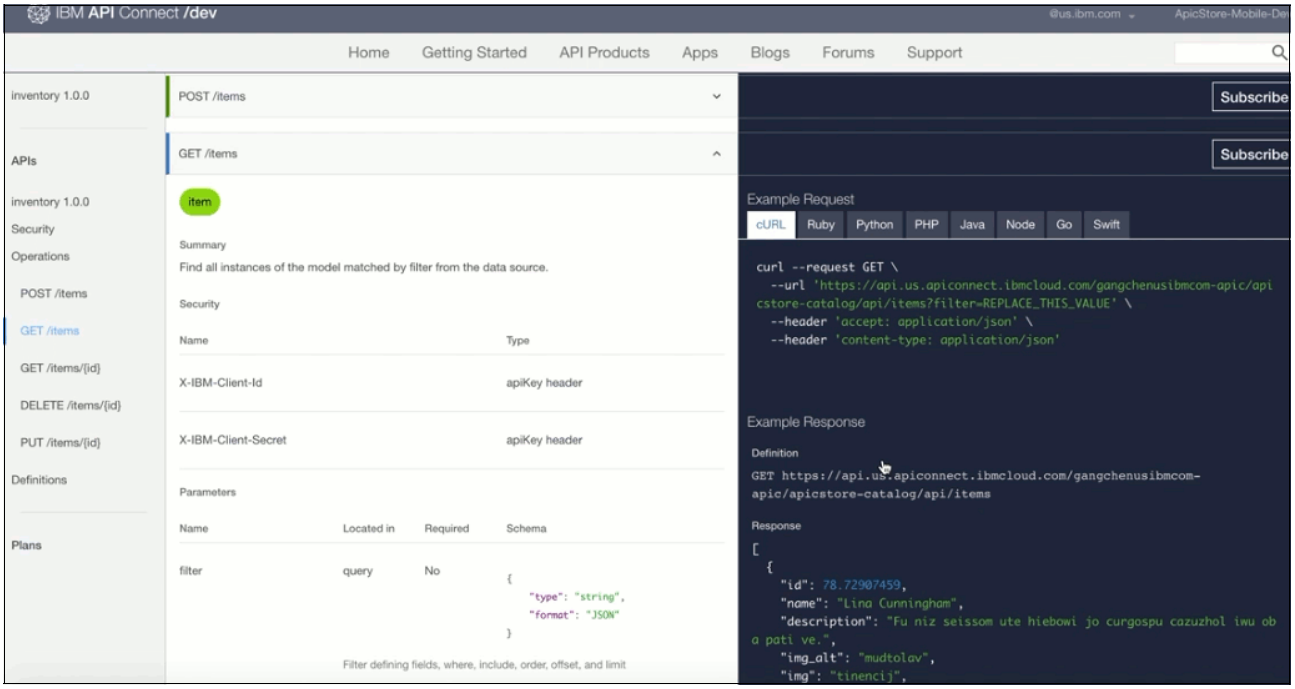

Figure 5-12 API Developer Portal tools to test API access and sample code.

4. The API Developer Portal contains multiple tools you can use to test the API and its functionality. For example, you can test the API call directly from the browser. Select one of the **API routes (GET /items)**, and on the right hand side scroll down until you see the button **Call operation**. This button will call the API end point from the browser and display the results. Figure 6-15 shows the interface to test the API end point from the browser using the button **Call operation**. See Figure 5-13 on page 89.

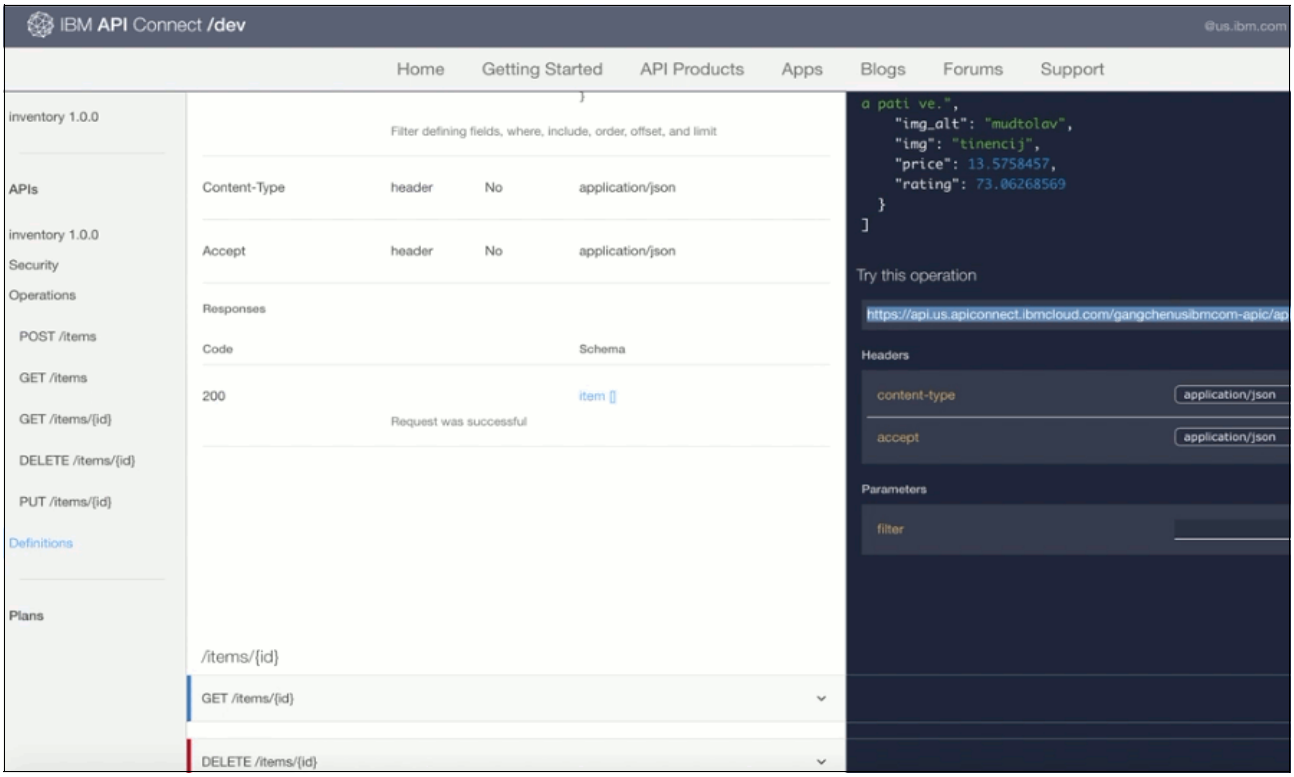

Figure 5-13 Testing the API from the API Developer Portal

5. Once you call the operation, the browser will show you the results of calling the API. Figure 5-14 on page 90 shows the results obtained after calling the API. The API call was successful. You can see in the figure that the API returned a 200 status, and the image shows the response data as well.

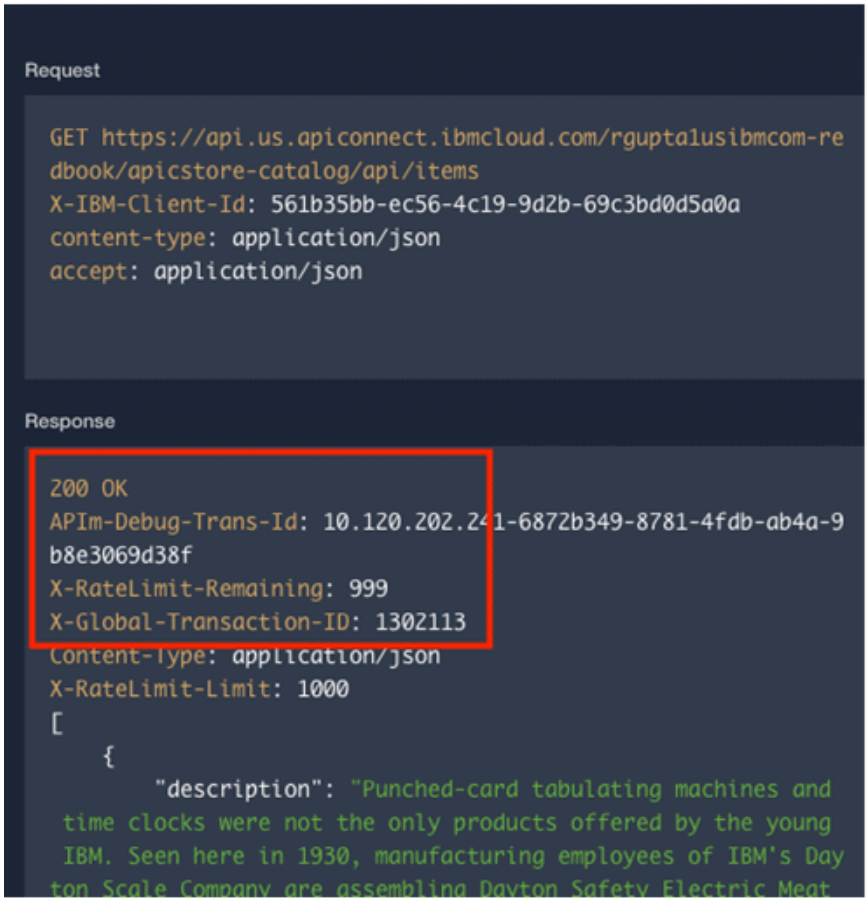

Figure 5-14 Results of the API call from the API Developer Portal

6. The API Developer portal tool also provides sample code for Ruby, Python, PHP, Java, Node, GO and Swift. You can use the code starters to integrate the API into your existing application if you are using one of those programming languages.

   Now you are ready to build the applications!

5.2 Develop the mobile iOS application

Now that we have registered the mobile application in the API Connect Developer Portal and tested the API, we can now clone the iOS application from the GitHub repository and test it locally.

Note for Mac users: You need to run this walkthrough on a Mac machine with Apple xCode development IDE installed.

If you do not already have the project cloned locally, then use the following command to do so:

Example 5-1 Git repository with code for the mobile application

```
git clone https://github.com/IBMRedbooks/redp5350-apiconnect-sample
```

Now that you have the project cloned locally, navigate to the project folder that contains the GitHub project:

Example 5-2 Location of the directory

```
/redp5350-apiconnect-sample/ApicStoreApp
```

From this location type the following to open the Xcode application:

```
open ApicStoreApp.xcodeproj
```

You can also open the iOS project by double clicking the above file from the Mac Finder.

Now that you have the iOS project opened in Xcode, let's take a brief look at the project structure. On the left hand side, you should see the following file structure in Figure 5-15. Note that the file structure may or may not be expanded after opening.

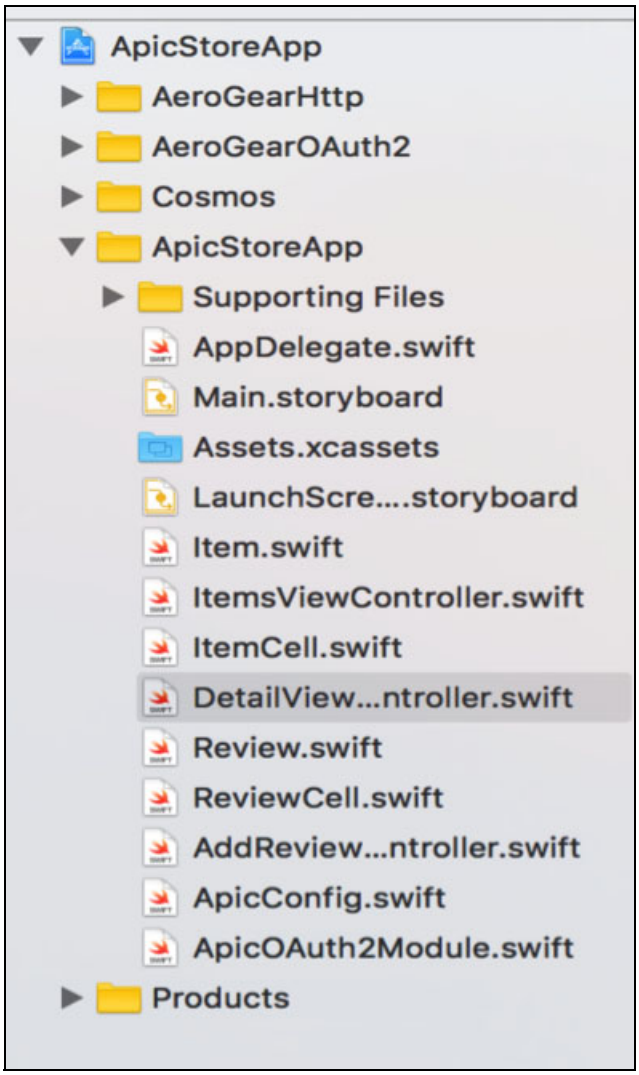

Figure 5-15 File structure

The main application resides in the ApicStoreApp folder. The AeroGearHttp and AeroGearOAuth2 folders contain third party libraries that we'll use to make our API calls using OAuth. You can also find more information on these libraries here:

https://aerogear.org/ios/

There is also a simple third party library to show the star ratings for the user reviews

https://github.com/marketplacer/Cosmos

1. Before you can run and test the iOS app, you need to specify the API endpoint configuration for your Bluemix API Connect deployment. Edit the *ApiStoreApp / Supporting Files / Config.plist* file. The Config.plist file contains all of the API endpoint URLs as well as the clientId registered earlier in Developer Portal. Any changes to the API endpoint urls can be made in this file. See Figure 5-16.

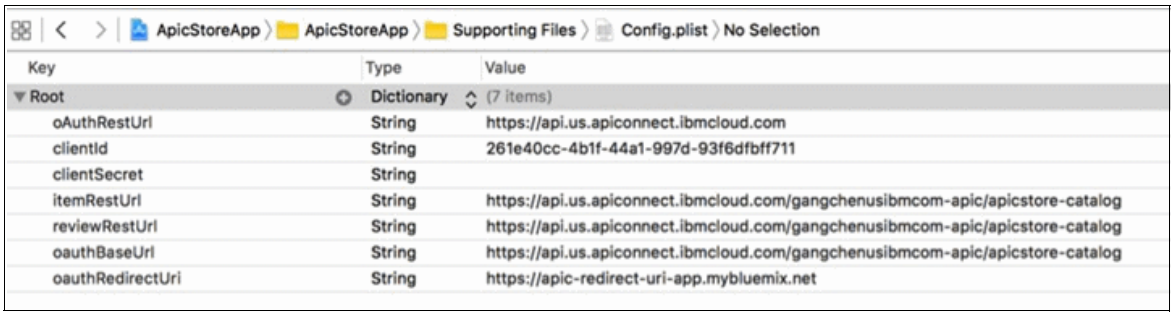

Figure 5-16 Configuration list for the mobile application

Config.plist file: The following is a description of the endpoints and constants in the Config.plist file:

- **oAuthRedirectUrl:** This is the oAuth Redirect API defined in the earlier section. It should be org.apic://example.com.
- **clientId**: This is the client Id that is obtained in the Developer Portal in the earlier section.
- **ItemRestUrl, reviewRestUrl, oAuthBaseURI**: These are the API endpoints from Developer Portal for Inventory API, review API, and OAuth API. In our case, the base URL host for all of these are the same, but in the code the URIs will be different for each call.
- **AuthRestUrl**: This is the endpoint to trigger the OAuth flow for socialreview API. The base URL is the same as above.

These four endpoints should all be the same and is actually your apic-catalog endpoint. For example:
https://api.us.apiconnect.ibmcloud.com/gangchenusibmcom-apic/apicstore-catalog

2. To run the application, click the **Play** button in the upper left corner to run the application (make sure to select either iPhone 6 or 6 Plus) as shown in Figure 5-17 on page 93.

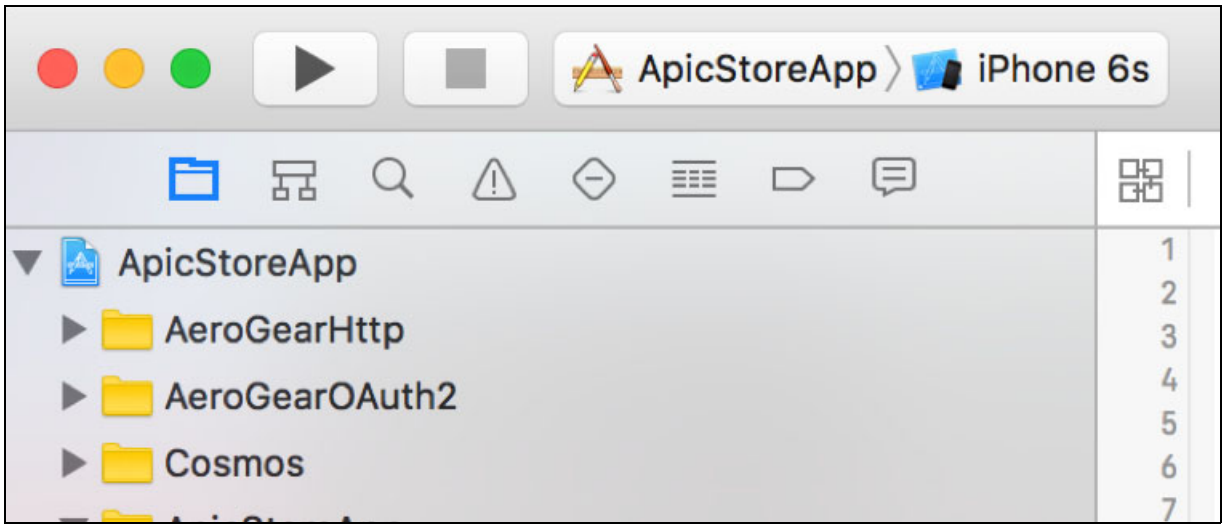

Figure 5-17 Mobile application runtime

The iOS application consists of 3 views:

- The main view, shown in Figure 5-18, shows the main inventory list.

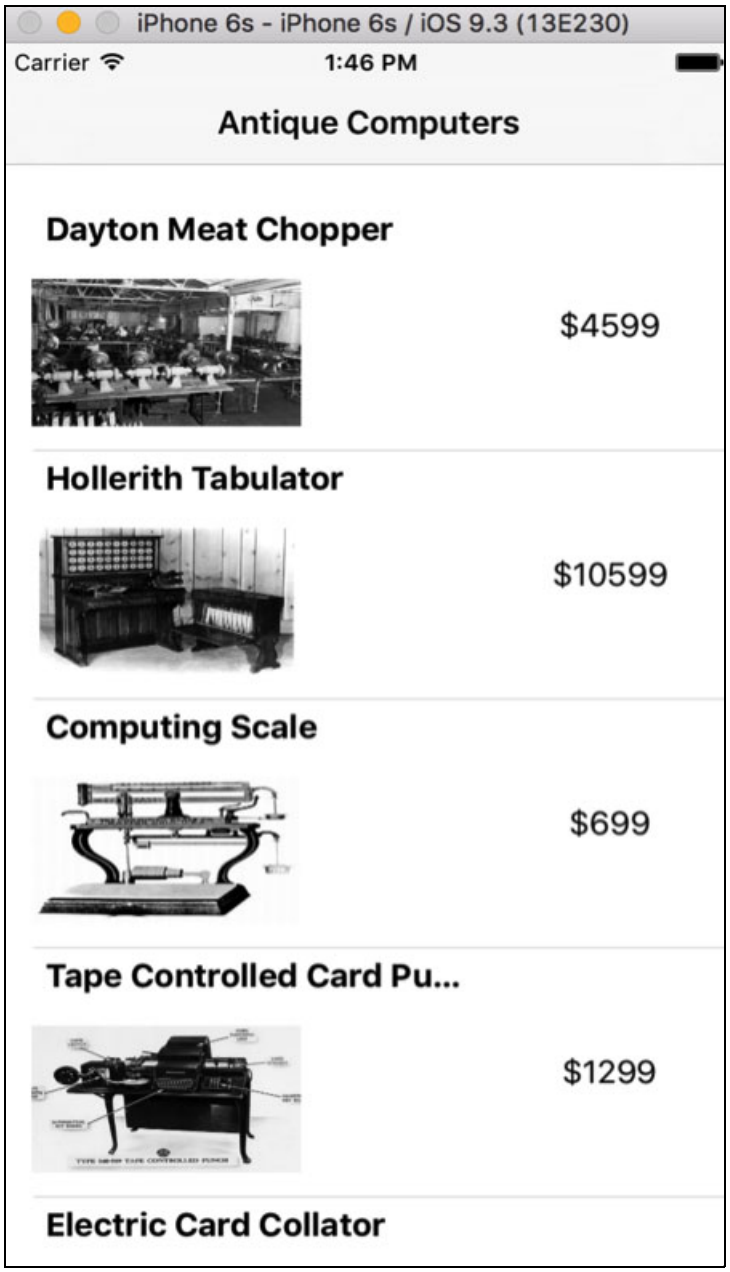

Figure 5-18 Mobile application item list

- The individual item details are shown in Figure 5-19 on page 95.

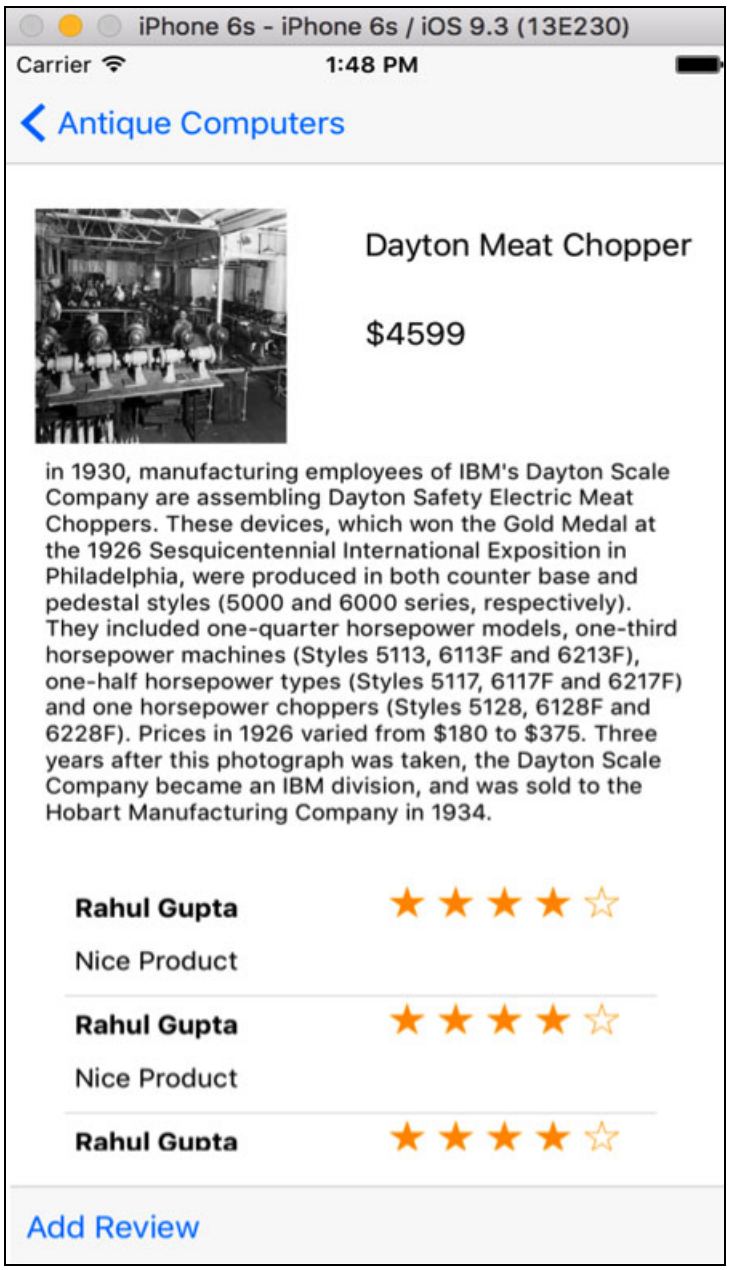

Figure 5-19 Mobile application item details

- And finally the Add Review screen is shown in Figure 5-20 on page 96.

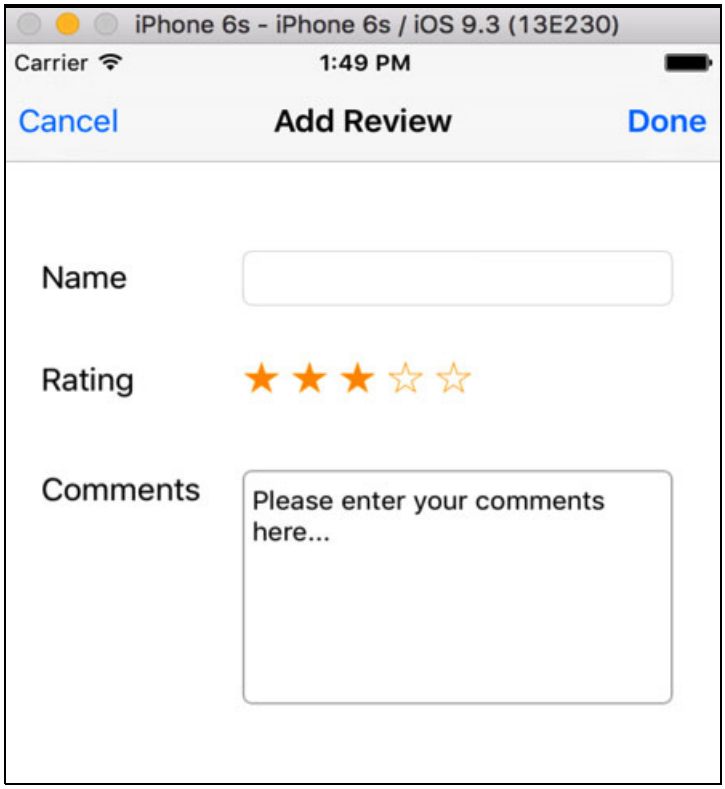

Figure 5-20 Mobile application item review detail

3. When running the application, after tapping the **Add review** button, the application will transition to the API Oauth screen with login info that retrieves the OAuth session and token that is used by the add review API. The correct credentials (username: foo, password: bar) will need to be provided to authenticate the OAuth session, otherwise the add review API will not work. Once the credentials are provided, then the session will be authenticated without having to authenticate again for the remainder of the session.
4. You can see the basic flow of the iOS application by clicking on the **Main.Storyboard** file in the file navigation menu under the ApicStoreApp folder, as shown in Figure 5-21.

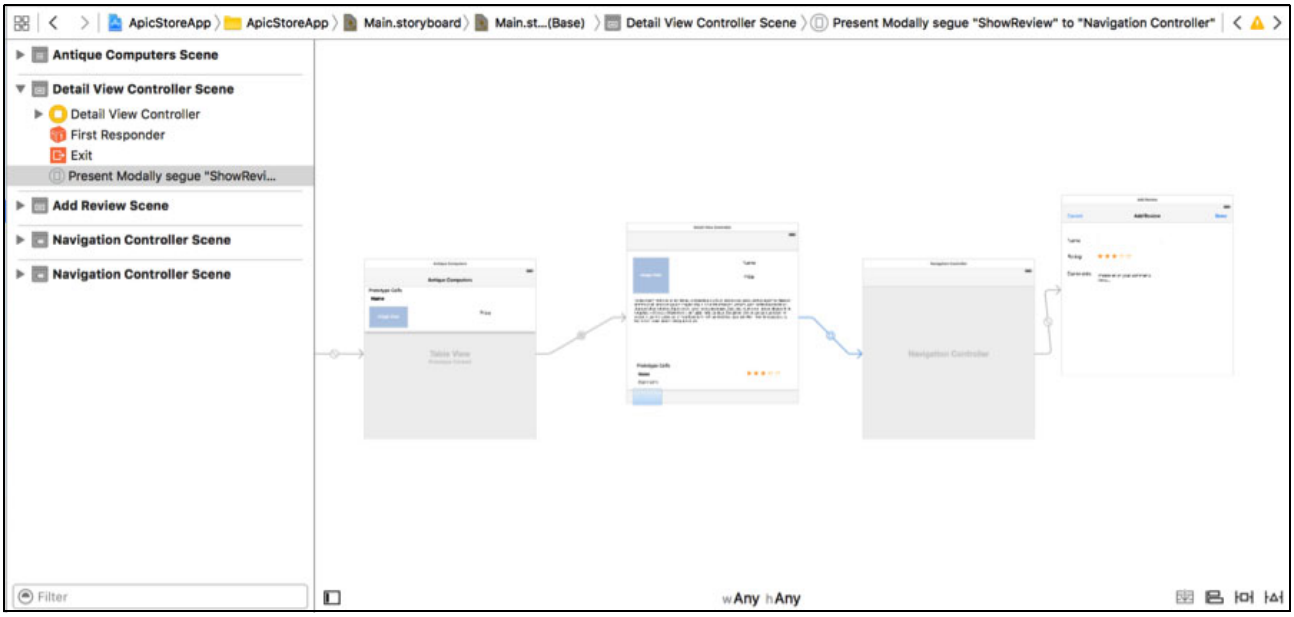

Figure 5-21 Mobile application story board

While this section will not go into the details of creating a basic iOS app in Swift, it is important to note the typical structure from an iOS application follows an MVC (model–view–controller) style framework in which a view as shown above has an associated controller with it. The three main controllers in this application are:

- **ItemsViewController.swift:** This is the first screen that loads in the app, which makes the REST call that returns the list of inventory items.
- **DetailViewController.swift:** This shows the details of the individual item selected from the home screen.
- **AddReviewController.swift:** This shows the add review screen and makes the REST call that adds a review to the associated item.

5.3 Run the sample consumer web application

In this section, we will describe how the developers can create a sample application using the APIs published by the API developer. The consumer web app provides the basic function to allow user to browse the Inventory items. The sample web app is built as a Node.js application that uses Express framework and Jade templates.

1. You need to configure the web app to point to the correct API Connect endpoints for the APIs you published earlier. Locate the folder where you downloaded the git projects. Edit the file: */redp5350-apiconnect-sample/StoreWebApp/config/default.json.*

You need to update the following 3 entries as shown in Figure 5-22:

- client_id
- host
- org

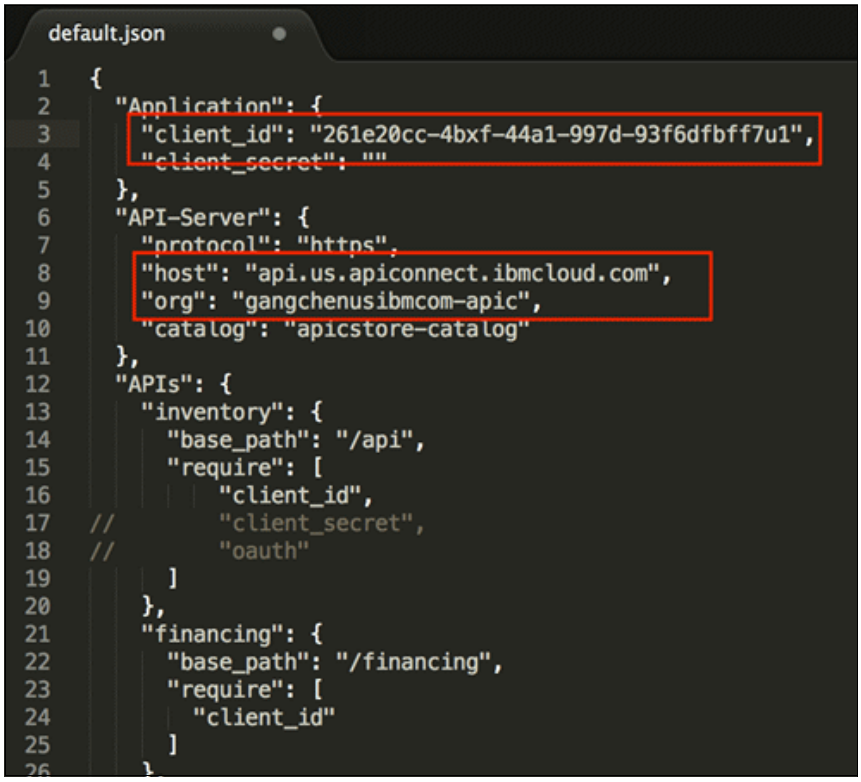

Figure 5-22 default.json file

2. Save the default.json file.
3. Navigate to your local git directory in a terminal window:

   ```
   cd $project_dir/redp5350-apiconnect-sample/StoreWebApp
   ```

4. Start the Web app by entering the **`npm start`** command. This will start the Node.js application and open a web browser with home page of the sample app, as shown in Figure 5-23 on page 99.

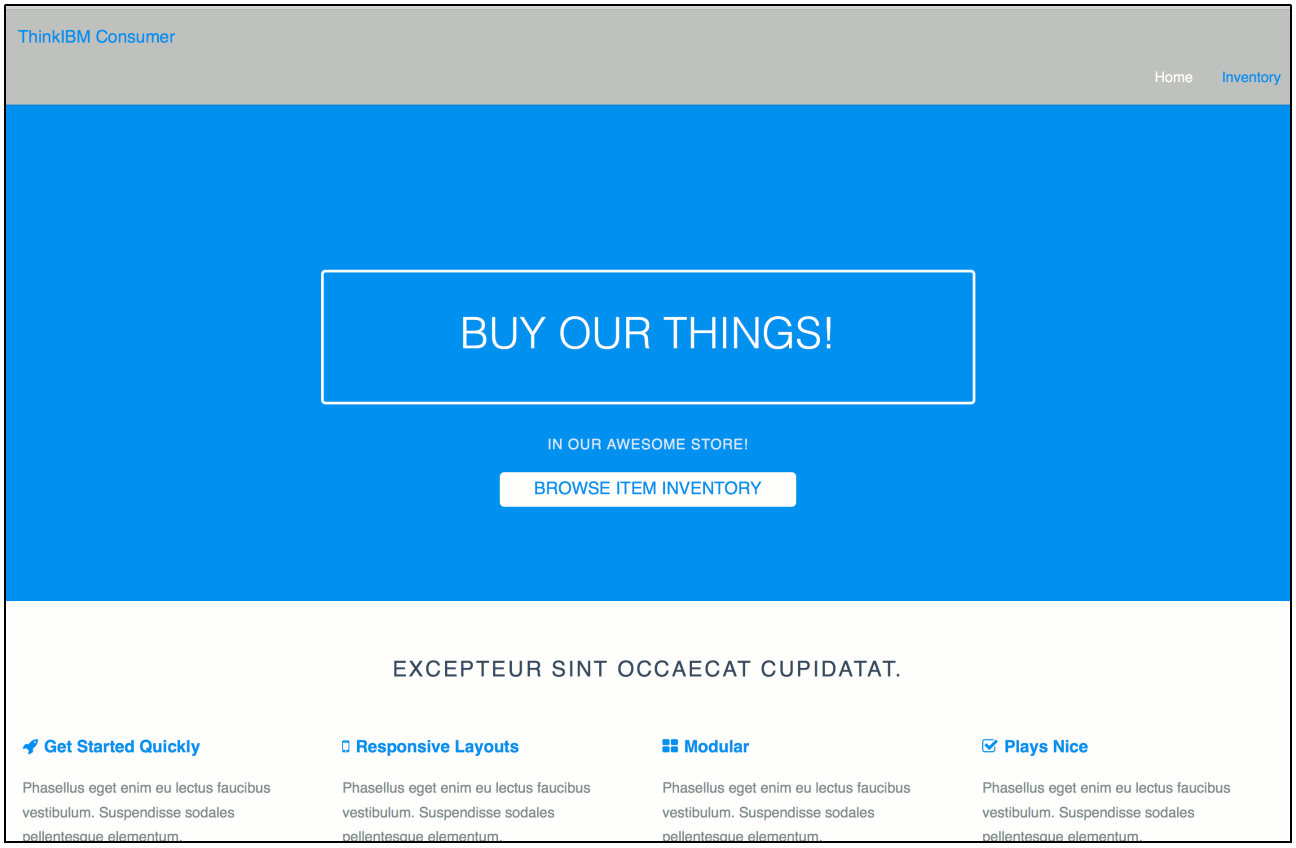

Figure 5-23 Main screen for the Inventory application

5. Click on **Browse Item Inventory**. You should see a list of items (Figure 5-24 on page 100).

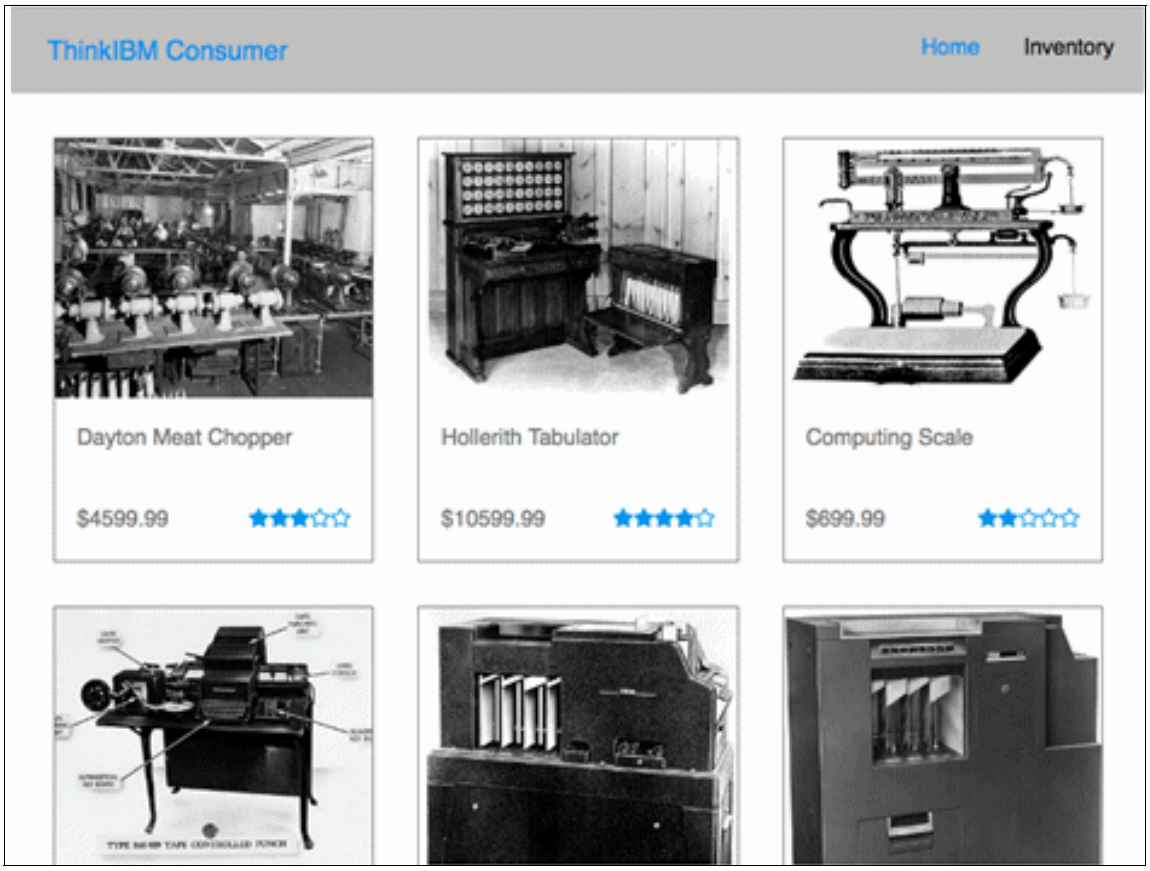

Figure 5-24 List of items

6. Clicking on any one of the items will bring you to the detail page, as shown in Figure 5-25.

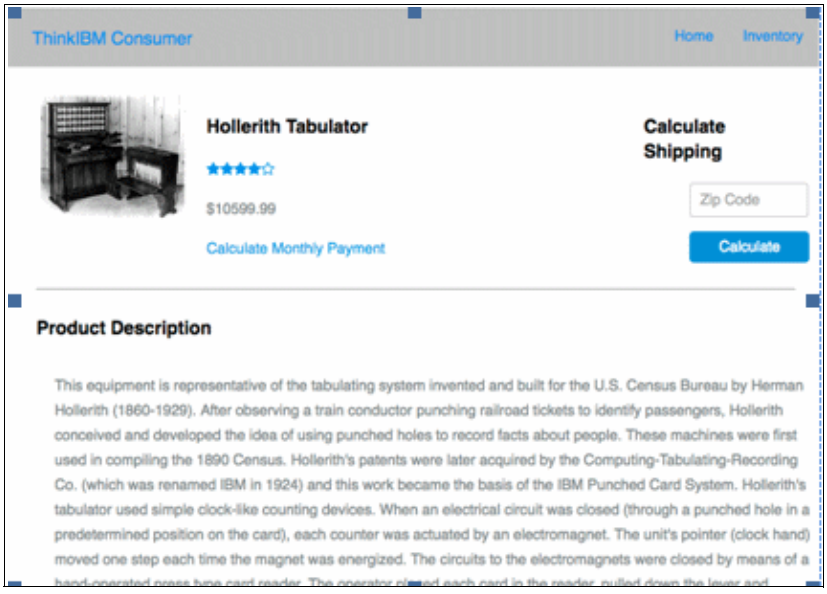

Figure 5-25 Detail page

Congratulations! You have successfully consumed the Inventory and SocialReview APIs in a mobile and web application.

5.4 Summary

In this chapter we showed how an application developer can consume the APIs provided by a third party in the development of their applications within the context of 2 different scenarios. One for mobile developers, the other for web developers.

6

Monitoring and analyzing the APIs

This chapter introduces IBM API Connect Analytics and shows you the commonly used features for creating and customizing your dashboards. We cover the following in this chapter:

6.1 Introduction to IBM API Connect Analytics

The IBM API Connect service in IBM Bluemix provides API analytics capabilities using the open source Elastic Stack (formerly known as ELK stack.) Within the Elastic Stack, Kibana is the visualization engine which allows you to create visualizations and dashboards. The API Manager console embeds Kibana for visualizing API Events. By default, API Connect logs an API Event for every invocation of an API operation.

Tip: For more information on the activity-log policy and how to disable logging, or enrich logs with header or payload information, see this Knowledge Center link:
http://www.ibm.com/support/knowledgecenter/SSFS6T/com.ibm.apic.toolkit.doc/rapim_ref_ootb_policyactivitylog.html

Using Kibana visualizations, you can explore information like which API, Products, Plans and operations are most popular, response times and success rates. Kibana visualizations in API Connect can give you insights that can impact business policy or help manage the health of the system.

API Connect users with authorized roles can access the API Connect analytics capability in the API Manager console to visualize raw API event data. Users can also create custom visualizations and dashboards, share these views and download data. In this chapter, we will show you where to find the default dashboards and visualizations that come out-of-the-box with API Connect. We will also show some examples of how to customize visualizations and dashboards and share them.

If you are new to Kibana and are looking for a more detailed introduction, check out this video:

https://www.elastic.co/webinars/kibana-101-get-started-with-visualizations

6.2 API Connect dashboards

Let's start by examining the default dashboards in API Connect. Analytics data is available in the API Manager console, so go ahead and log into API Manager.

1. Navigate to the dashboard and then select the APIcStore Catalog. All API Connect analytics are strictly scoped to catalog. This supports isolation between catalogs, so that only authorized viewers can access analytics data for a catalog.
2. Navigate to the **Analytics** tab. See Figure 6-1 on page 104.

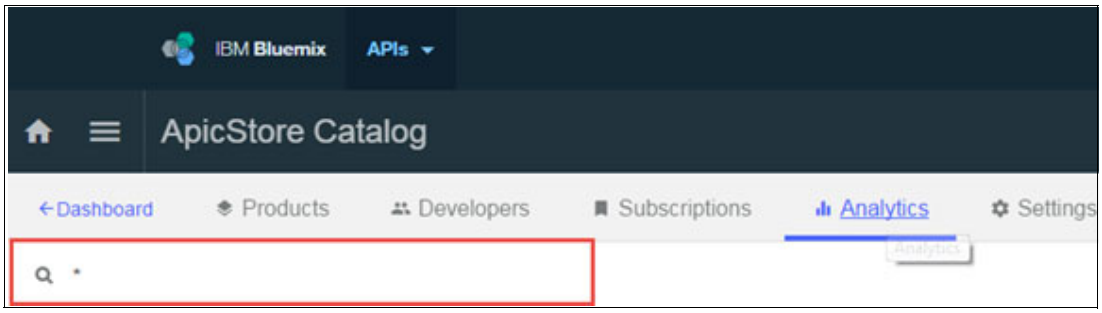

Figure 6-1 IBM API Connect Analytics dashboard search bar

Notice the dashboard configuration icons on the right as shown in Figure 6-2. Selecting these icons, you can open, create, save and share dashboards. You can also create visualizations which can be added to existing or new dashboards.

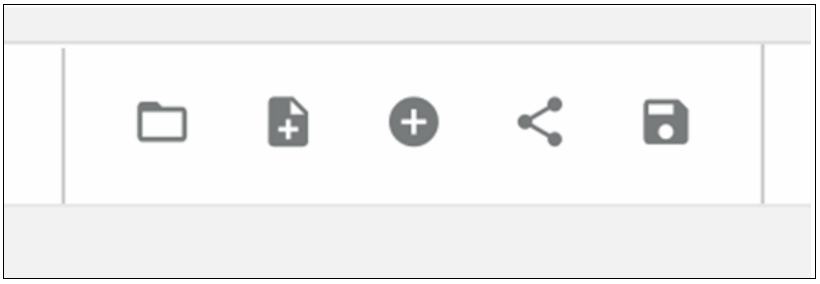

Figure 6-2 Dashboard configuration icons

3. Select the first icon (resembles a folder). This will give you a list of the out-of-the-box dashboards. See Figure 6-3. You can select any one of these dashboards to see them, but they will not be automatically filtered to a specific API, plan or product. The next sections will show you how to navigate to and explore these dashboards in API Manager and the Developer Portal.

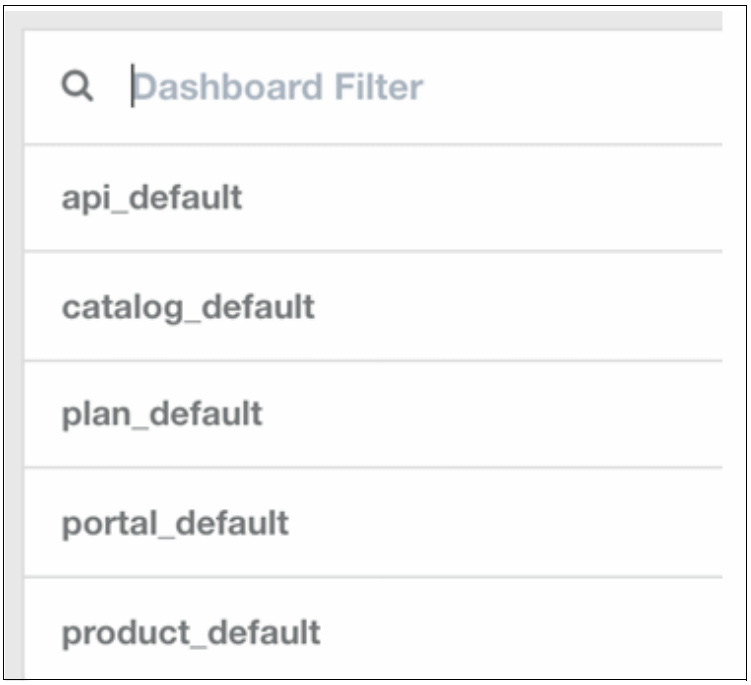

Figure 6-3 List of the out-of-the-box dashboards

6.2.1 Default catalog dashboard

Navigate to the Analytics tab. This will display the default catalog dashboard. It contains two visualizations as shown in Figure 6-4 on page 106:

- 5 Most Active Products
- 5 Most Active APIs

If you haven't made any calls yet, the visualizations will be empty:

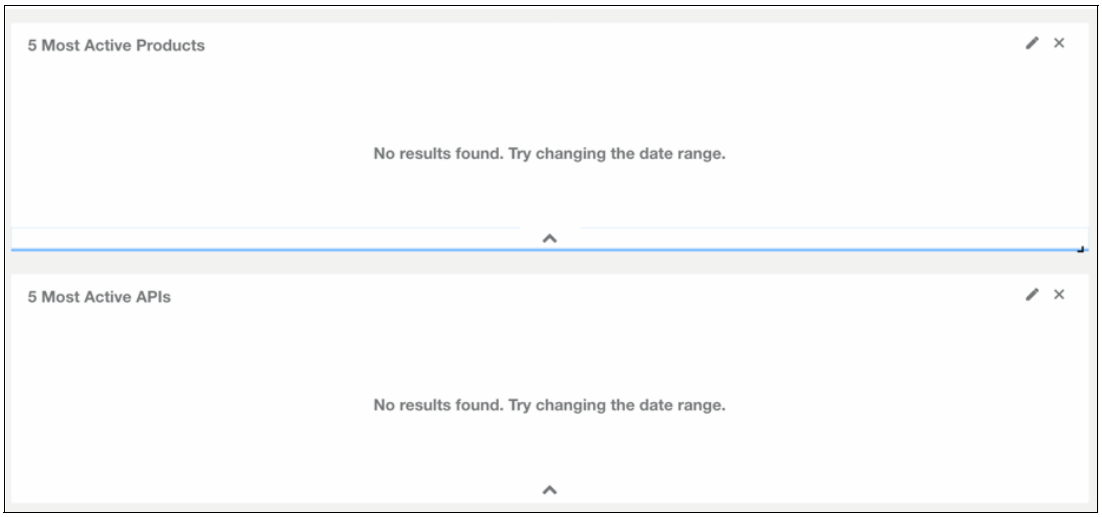

Figure 6-4 Default catalog dashboard

4. See the suggestion to change the date range. By default, the API events in the visualizations are scoped to the last 7. If you click on **Last 7 days** at the top right, this will expand to show you a set of date and time ranges. As you select these, the visualizations will be updated to reflect the new range. See Figure 6-5.

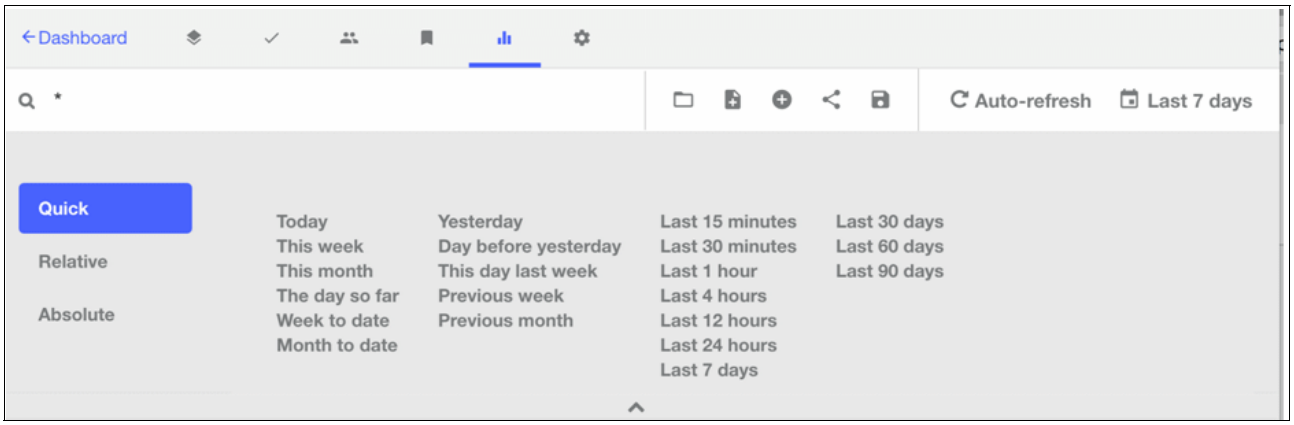

Figure 6-5 API events - Last 7 days

5. You can also select **Relative** or **Absolute** to further fine-tune the data that you want to examine. This time range selection is available in all dashboards.

6.2.2 Default product dashboard

Now let's see the default product dashboard.

1. Navigate now to the Product tab on the Catalog view. See Figure 6-6 on page 107.

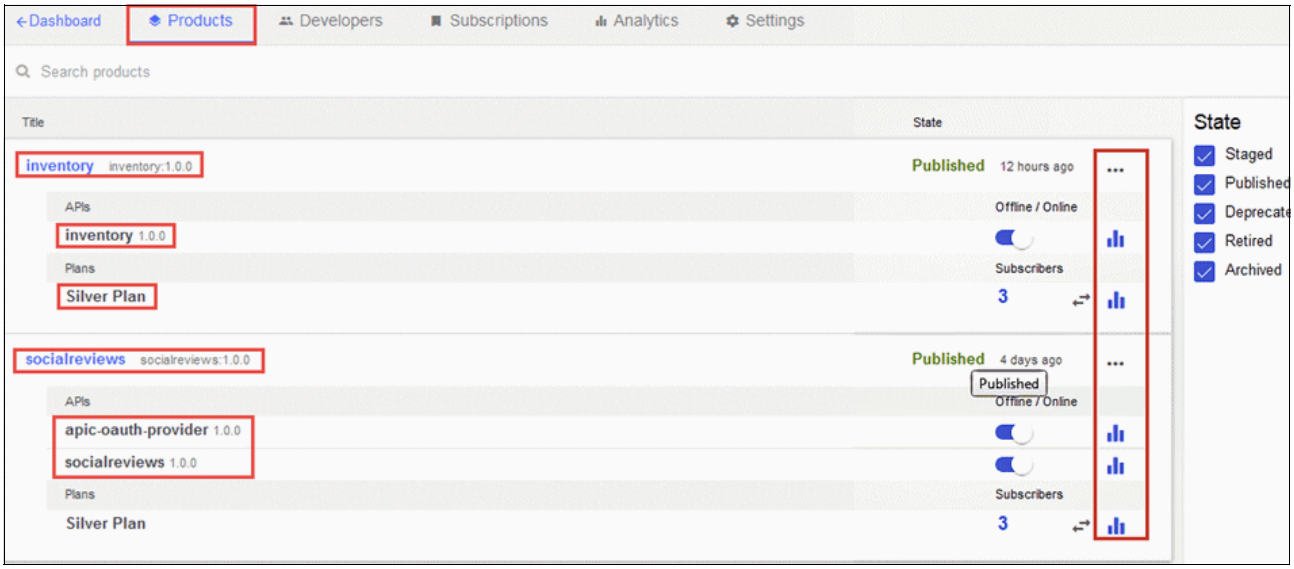

Figure 6-6 IBM API Connect Analytics access for Product, APIs and Plans scope

2. Select the ellipses ('**...**') at the top right, and then select **Product analytics**. See Figure 6-7.

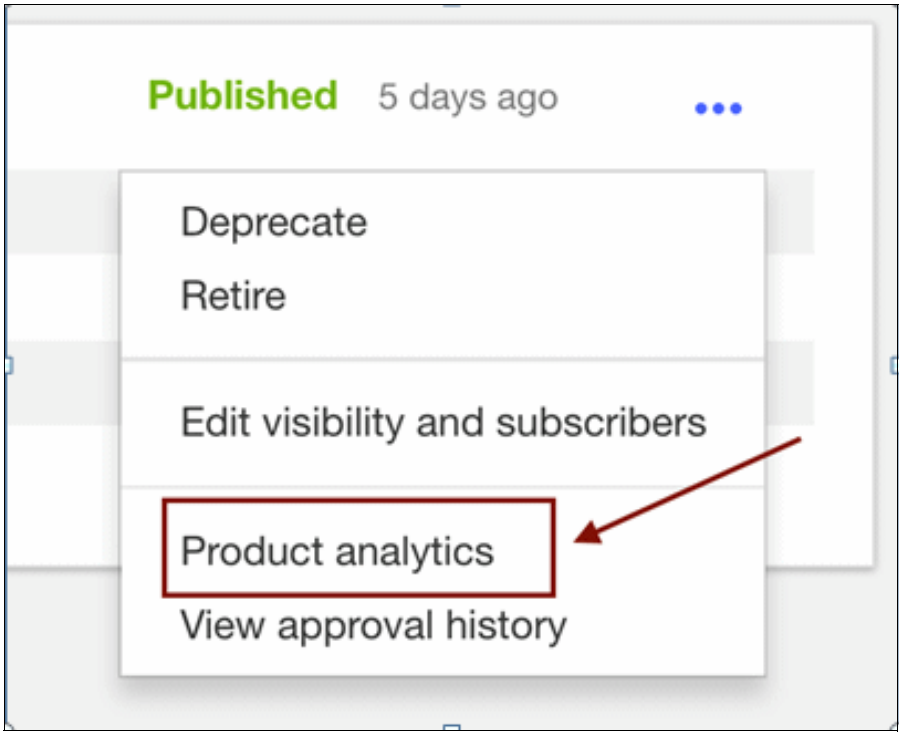

Figure 6-7 Select Product analytics

This will open the default product dashboard, scoped to the selected Product. You should see four visualizations in this dashboard:

- The count of Developer Organizations that have applications subscribed to Plans for this Product.
- The number of API calls that have been made to APIs in this Product.

– A pie chart that shows applications that are subscribed to each plan. An inner ring shows the breakdown of plans, and the outher ring shows the applications that are subscribed to the plan on the outer ring. An application might be subscribed to more than one plan. In this simple graph, there is one plan, the purple inner circle. There are five applications subscribed to that plan shown in the green, blue, pink, orange and light blue segments on the outer ring as shown in Figure 6-8. The applications are shown proportionately to the percentage of the total number of calls each one made. You can see that the app represented by green made the most calls and the app represented by light blue made the fewest.

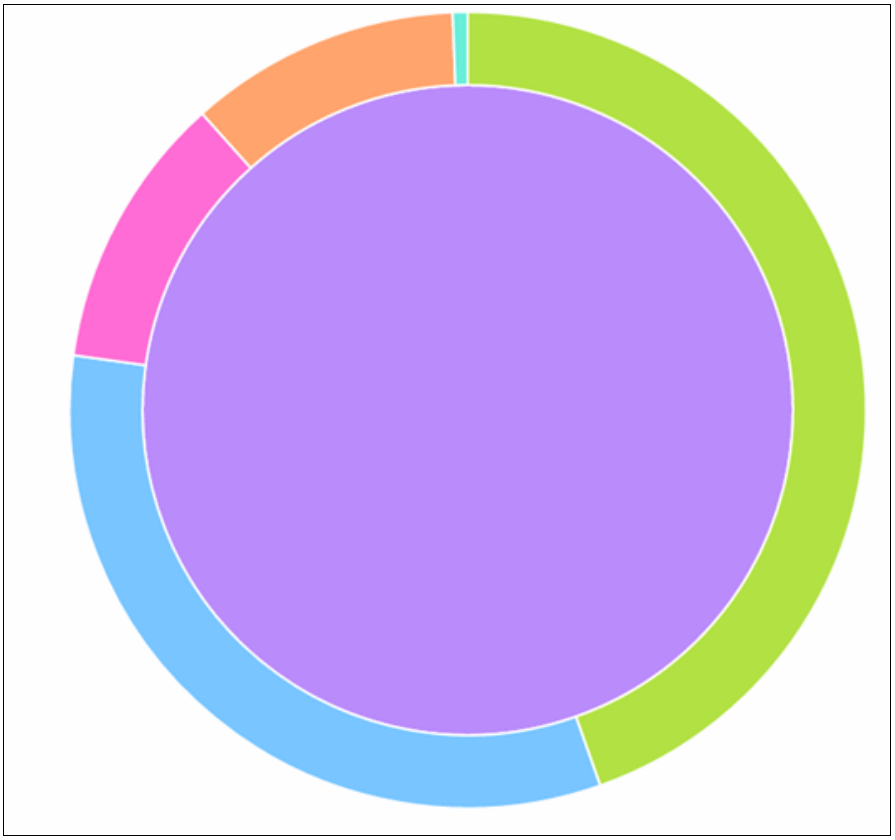

Figure 6-8 Applications that are subscribed to each plan

a. A bar chart showing the number of API calls per day.

3. You can hover over a bar to get more details as shown in Figure 6-9 on page 109.

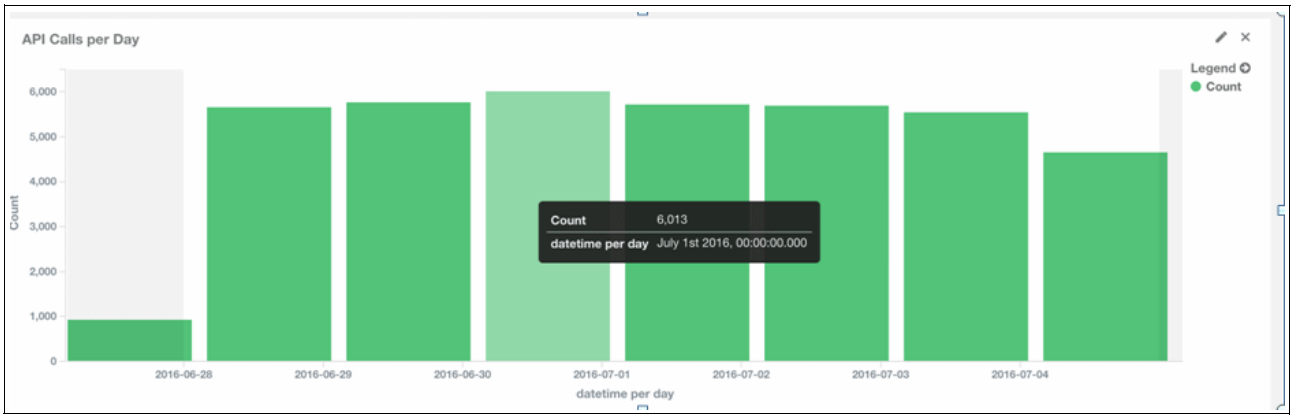

Figure 6-9 API calls per day - details view

In summary, this dashboard gives you high-level visibility into the consumption of your Product. It can answer questions like:

- How many calls are being made every day or for any selected time period?
- What plans are being subscribed to most?
- How many unique Developer Organizations have signed up?

6.2.3 Default API dashboard

Now let's see the default API dashboard.

1. Go back to the Product View on the catalog. Select the **graph icon** beside the Inventory API under the Inventory Product. See Figure 6-10.

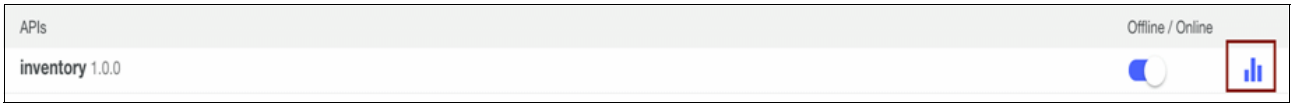

Figure 6-10 Select the graph icon

This will open up the Default API Dashboard, which has eight visualizations:

- A pie chart that shows the HTTP Status codes returned from all invocations of this API
- A stacked bar chart that shows error counts per status code per day. This view filters out status codes of 2xx to only reflect error codes.
- A visualization that shows the minimum response time for all calls to the Inventory API in the specified time period.
- A visualization that shows the average response time for all calls to the Inventory API in the specified period.
- A visualization that shows the maximum response time for all calls to the Inventory API in the specified period. See Figure 6-11 on page 110 for an example of visualizations 3,4,5. You can see that the minimum response time is 24 milliseconds, the maximum is almost 7 seconds and the average is less than a second. This might reflect differences in specific endpoints or problem that is causing occasional slow response times.

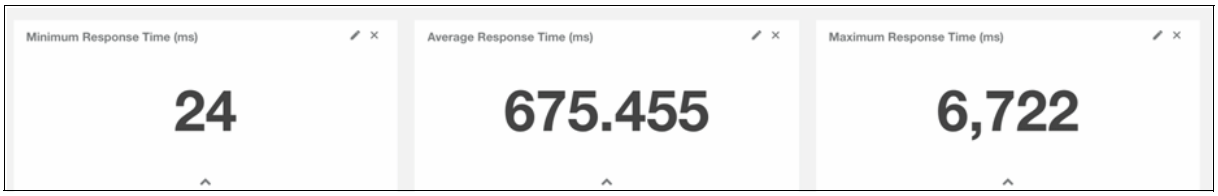

Figure 6-11 Maximum response time for all calls

- A line graph with lines for minimum, average and maximum response times over time, with time on the x-axis.
- The total number of Inventory API calls.
- A bar chart with the number of Inventory API calls per day.

This dashboard answers questions like:

- What are the success and error rates of calls to this API?
- What are the response times of calls to this API?

Answering these questions can help you understand the health and usage rates of the APIs.

6.2.4 Default plan dashboard

We will now see the default plan dashboard.

1. Select now the **analytics** icon beside Silver Plan under the Inventory Product. The default plan dashboard has three visualizations:
   - Number of API calls made by apps subscribed to the Silver plan
   - Number of applications that are subscribed to the Silver plan
   - A stacked graph that shows the max rate limit and rate limit count for each application over time. This visualization can be used to show if applications are approaching exceeding rate limits, over time. If your plan is an unlimited plan, this visualization will be empty.

This dashboard helps to understand whether a particular plan is being used and whether subscribers are approaching rate limits for the plan.

Tip: This information might be especially useful if you are monetizing your APIs.

6.2.5 Default Portal dashboard

The Portal dashboard is available to consumers in the Developer Portal. It allows API consumers to see visualizations that show response time, API success rates and data usage. To see this dashboard you must be a Developer Organization owner.

Here's an example of the Success rate graph. You can see there was a spike in usage and that the failure rate remains quite constant, although at the start there were a higher percentage of failures.

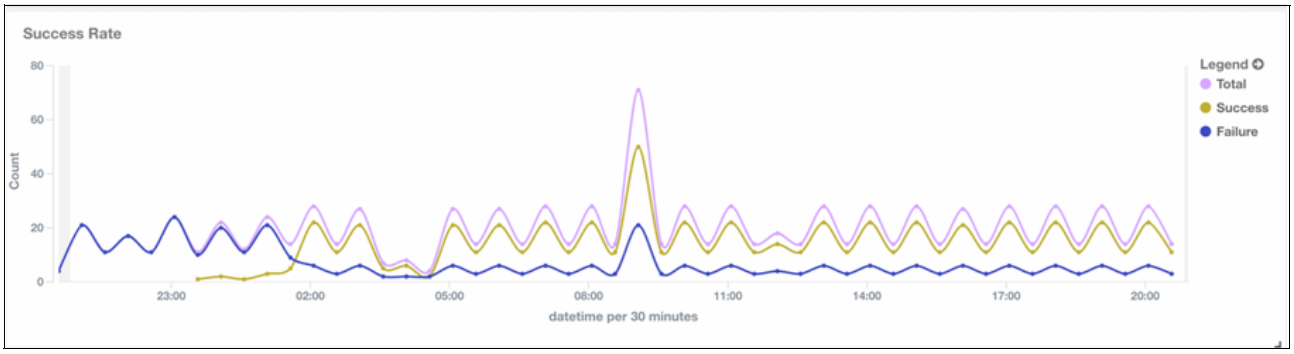

Figure 6-12 Success rate graph

For more details on Portal Analytics, see the Knowledge Center:

http://www.ibm.com/support/knowledgecenter/SSFS6T/com.ibm.apic.devportal.doc/capim_portal_analyticsparent.html

6.3 Customizing visualizations and dashboards

Now that you've explored the dashboards and visualizations, you might have noticed some data that you'd like to explore more deeply to get further insights; for example, the response times on the API dashboard had a very large range, from 24ms to almost 7 seconds. Or you might have spotted a large number of error response codes. In both cases, you might want to find out more about which specific operations were affected.

1. Let's go back to the API dashboard for the Inventory API. Navigate to the Errors visualization in the upper right and select the **pencil icon** to edit this visualization.

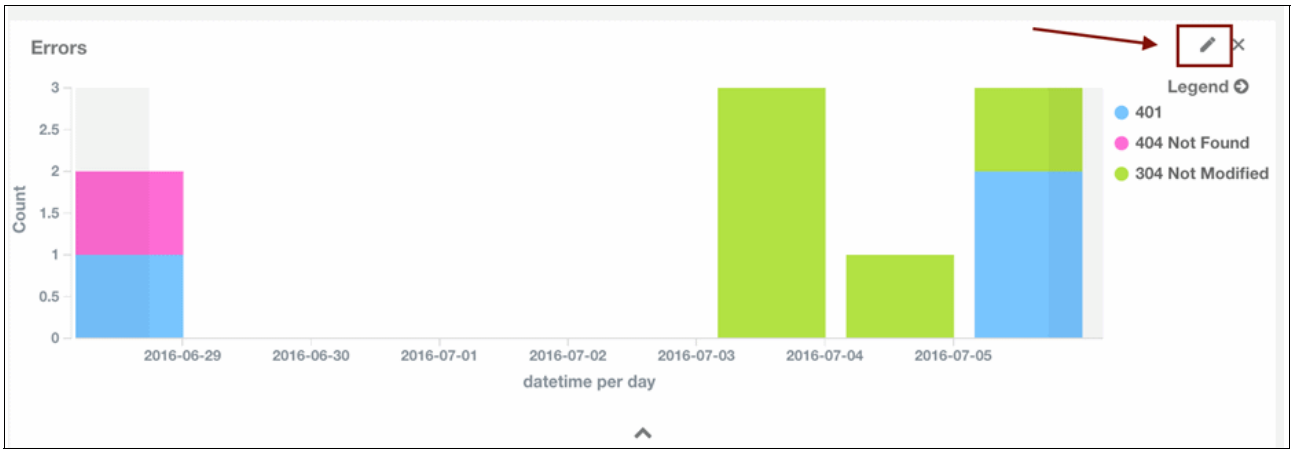

Figure 6-13 Errors visualization

2. This will open the Kibana visualization editor. You can make changes here and then use the **Play** button to check the results. You can then save this as a new visualization. See Figure 6-14 on page 112.

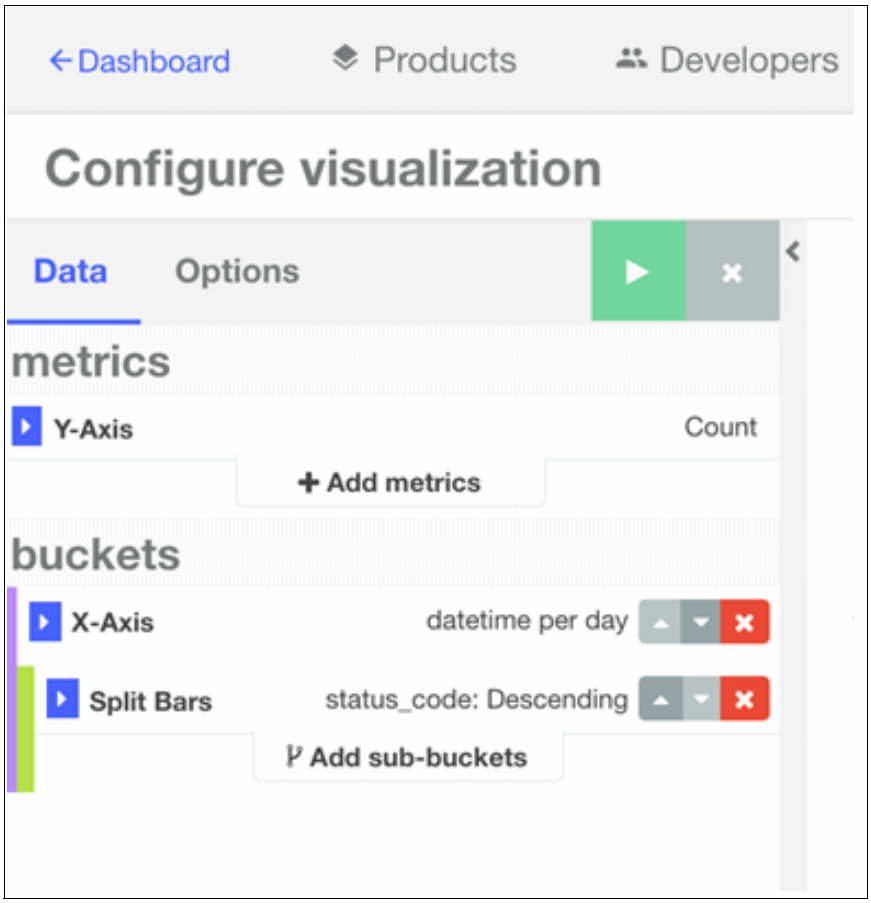

Figure 6-14 New visualization

3. Let's modify this visualization to show us error codes by individual resource. Select **Add sub-buckets**, and then choose the default **Split Chart** as shown in Figure 6-15 on page 113.

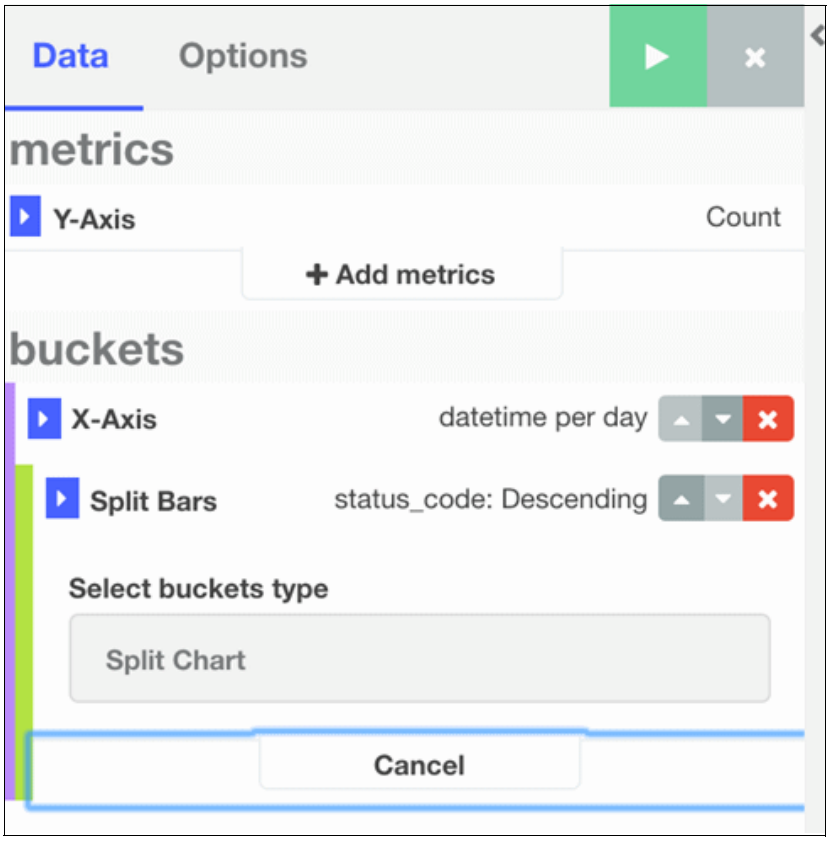

Figure 6-15 Modify the visualization

4. Now, make selections as shown in the following figure. You want to choose a Sub Aggregation that uses "Terms" where the field is "resource_id". Resource_id is the identifier for each resource or operation. Within the aggregation, you will order by metric:Count. See Figure 6-16 on page 114.

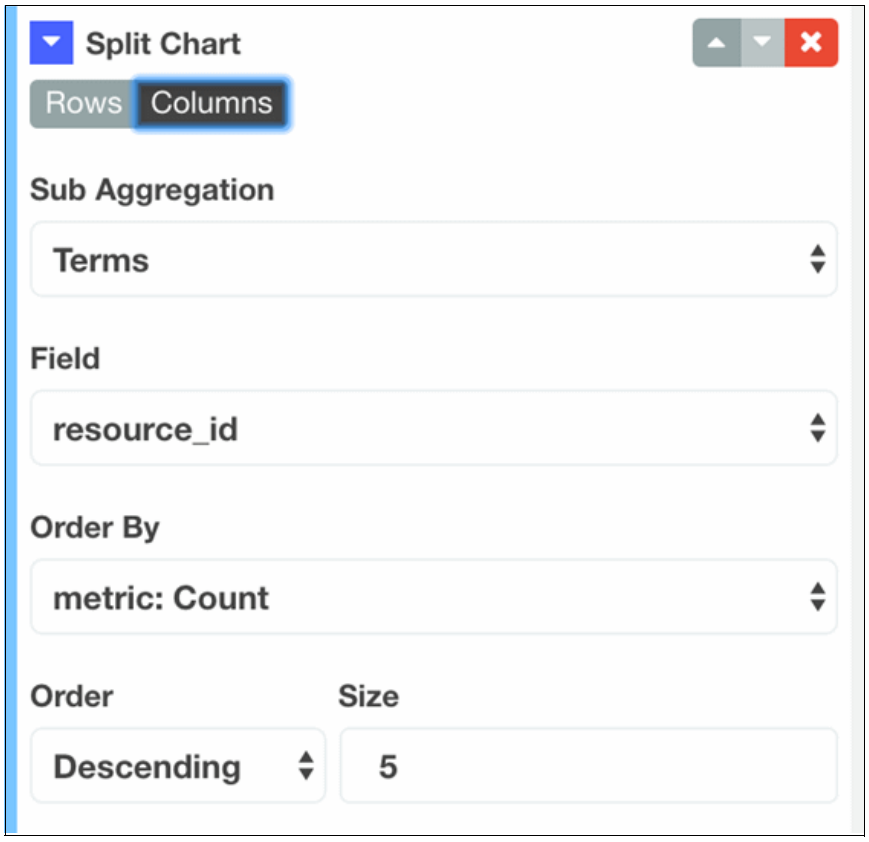

Figure 6-16 Modify the visualization

Tip: You can find a description of all API Connect terms in the Knowledge Center here: http://www.ibm.com/support/knowledgecenter/SSFS6T/com.ibm.apic.apionprem.doc/rapim_analytics_apieventrecordfields.html

5. If you click the **green Run button** at the top, you will see an example of the resulting visualization as show in Figure 6-17. What you see will depend on how many calls you have made so far and what the resulting responses were. You should see individual operations listed along the x-axis and then further divided by time. The bars should be stacked, showing the number of error response codes of each type. You might notice a mix of socialreview and inventory operations. In this editing view, the displayed output is scoped to catalog and not filtered based on where you might have launched the editor.

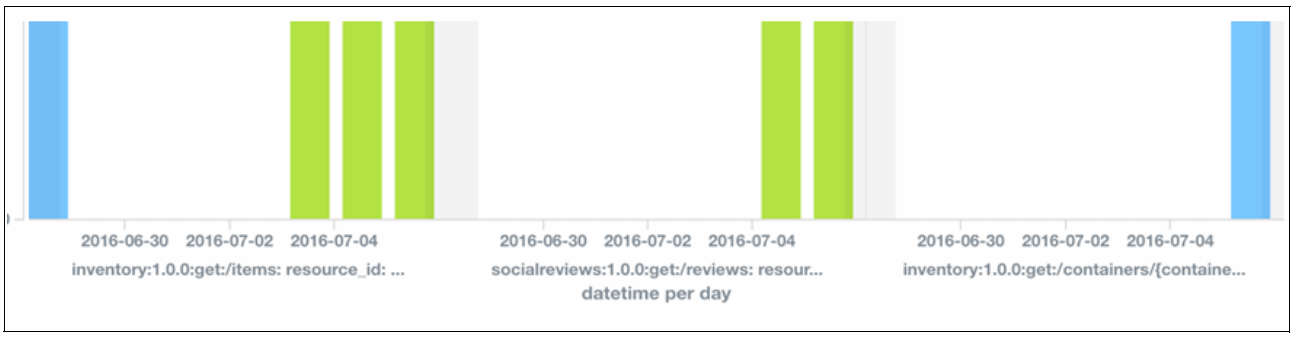

Figure 6-17 Resulting visualization

6. You may decide that you want to see non-error status codes in this view as well. In that case, go back to the editor on the left. Under X-Axis, expand **Split Bars.** Now expand **Advanced**. This is where Exclude and Include Patterns can be specified. Under JSON Input you should see {"exclude" : "2."}. This instruction is telling the visualization to exclude all response codes that start with 2; that is, it is excluding successful response. Delete this text, as Figure 6-18.

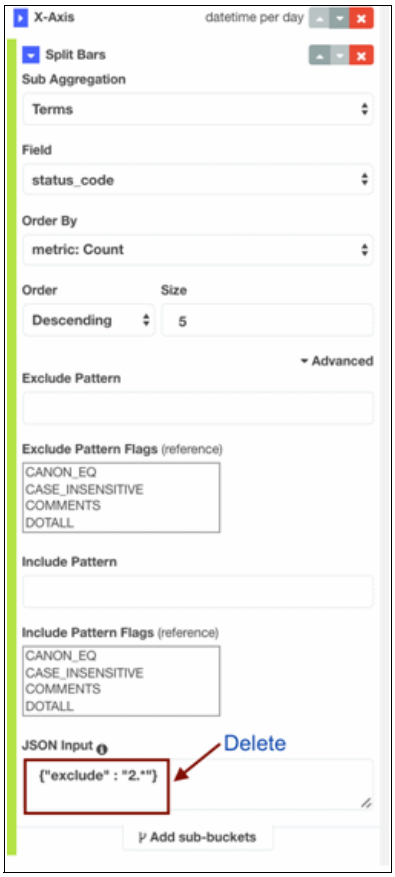

Figure 6-18 Delete the text

7. After running again, you may see more operations across the bottom. This is because those that were empty weren't displayed previously. Now you might see that some operations are running 100% successful, while others have failed 100% of the time. You can investigate to find out whether this is a usage error, a defect in the implementation of the operation a downstream outage or some other issue. See Figure 6-19 on page 116.

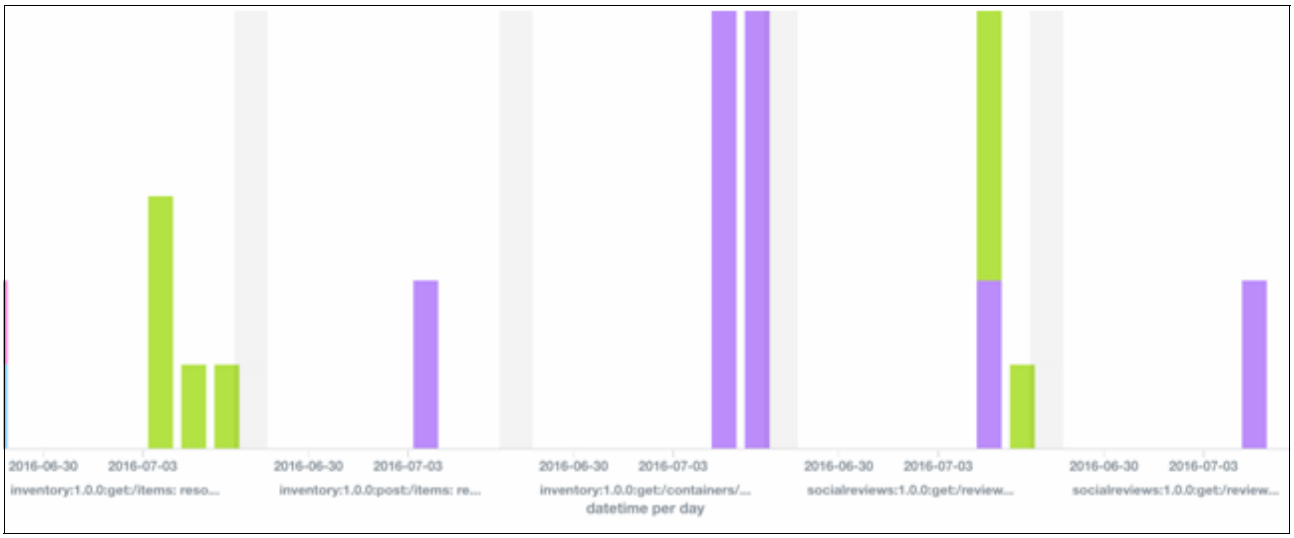

Figure 6-19 Same view with non-error status codes

8. You can also flip this graph and show the sub-aggregation on the y-axis instead of the x-axis. In this case, the graph might look like Figure 6-20, where each operation is grouped along the y-axis.

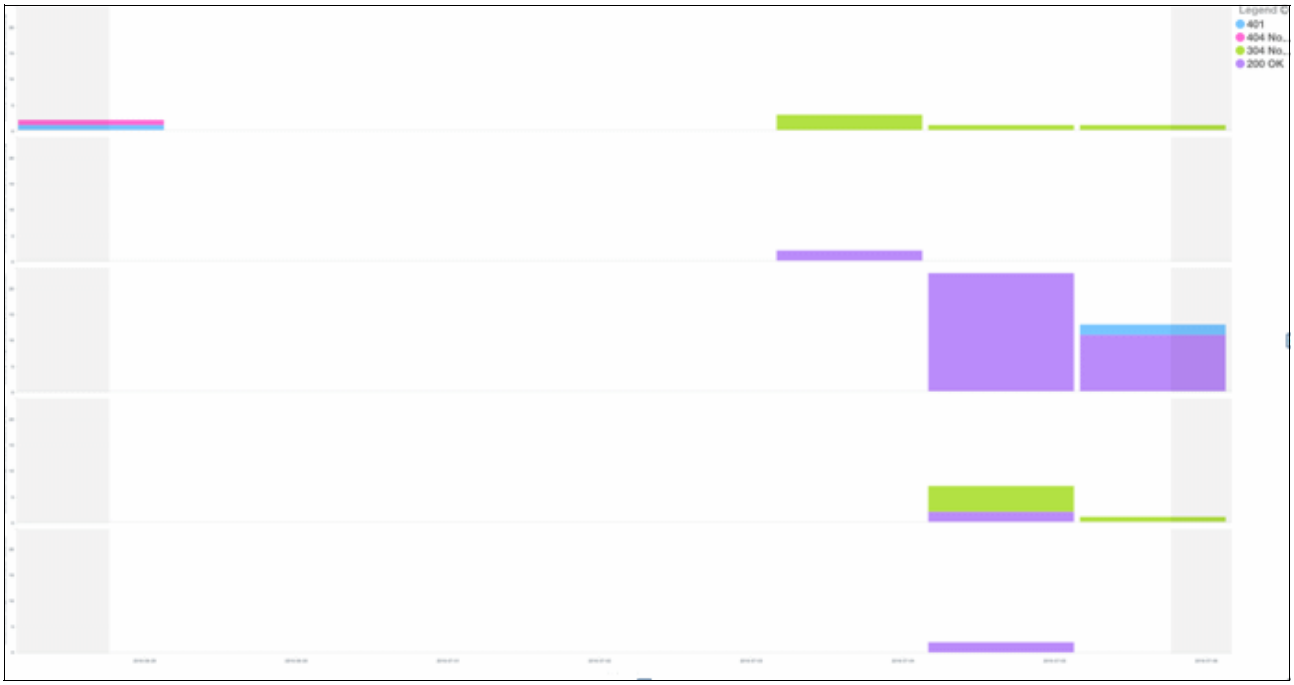

Figure 6-20 Sub-aggregation on the y-axis

9. You might now decide that this visualization looks the way you like it and is really useful. Let's save it. Click the **Save** icon on the top right of the visualization. See Figure 6-21 on page 117.

Important: Give the visualization a new name, so you don't replace the default Errors visualization.

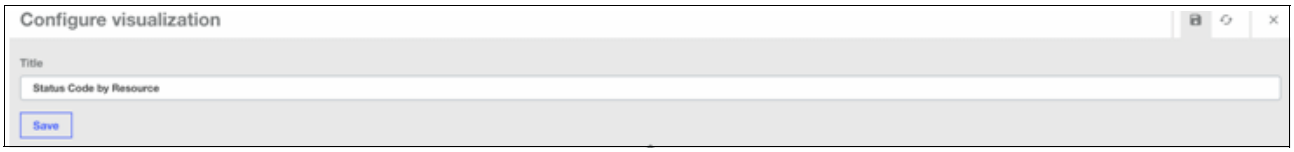

Figure 6-21 Configure visualization

10. Now click **Save** to save this visualization. Then click the '**X**' at the top right to close the editor. You're back in the Product Dashboard. Let's add our new Visualization to this Dashboard. Select the **Add Visualization** icon from the Dashboard configuration icons, as shown in Figure 6-22.

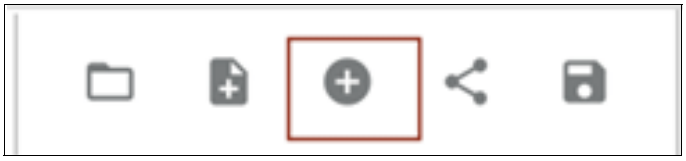

Figure 6-22 Select the Add Visualization icon

11. If you don't see your new visualization in the list, start typing the name in the Visualization Filter. See Figure 6-23.

Figure 6-23 Start typing the name in the Visualization Filter

12. Select your visualization. Scroll down and you should be able to spot your visualization on the dashboard as shown in Figure 6-24.

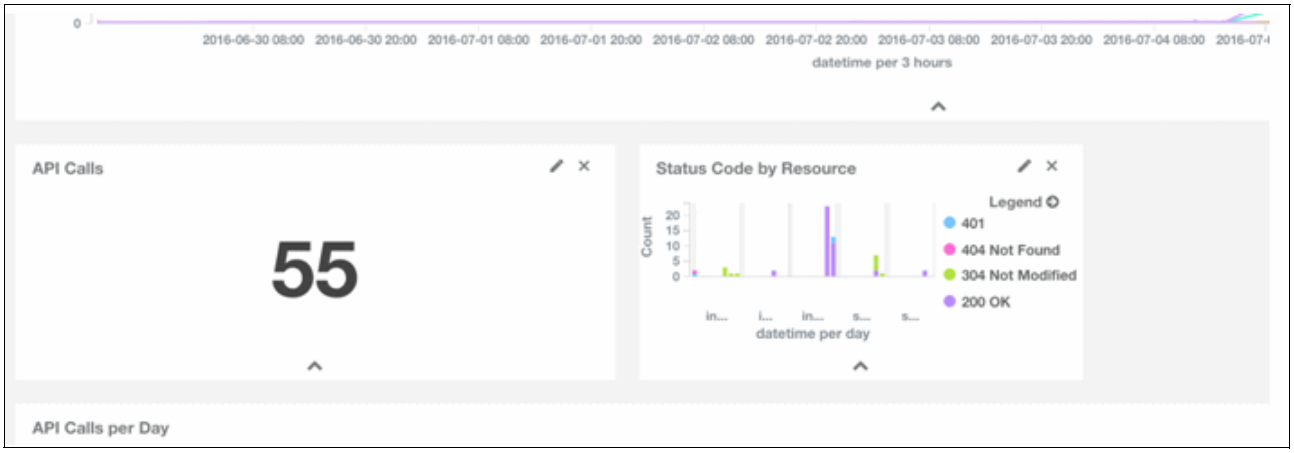

Figure 6-24 Spot your visualization on the dashboard

13. If it's not sized or located the way you want it, then you can resize and move it around by grabbing the corners and dragging and dropping. See Figure 6-25 for an example.

Figure 6-25 Resizing the visualization

If you want, you can now save the Dashboard so that it always includes this visualization. Select the save icon on the top right. You can give this Dashboard a new name, to create a brand new dashboard. Alternatively, you can save this default dashboard.

Tip: If you save the default dashboard with changes, you will not be able to reset to the original default. For this reason, it is recommended that you give the customized dashboard a new name.

6.4 Sharing dashboards

In order for a user to access visualizations, they must be authorized in two places.

1. In API Manager, navigate to **Admin**, and to the **Roles** view. A user must have a Role that is permitted to "View catalog analytics". Note that you can create a custom role that is specialized for Analytics access. See Figure 6-26 on page 119.

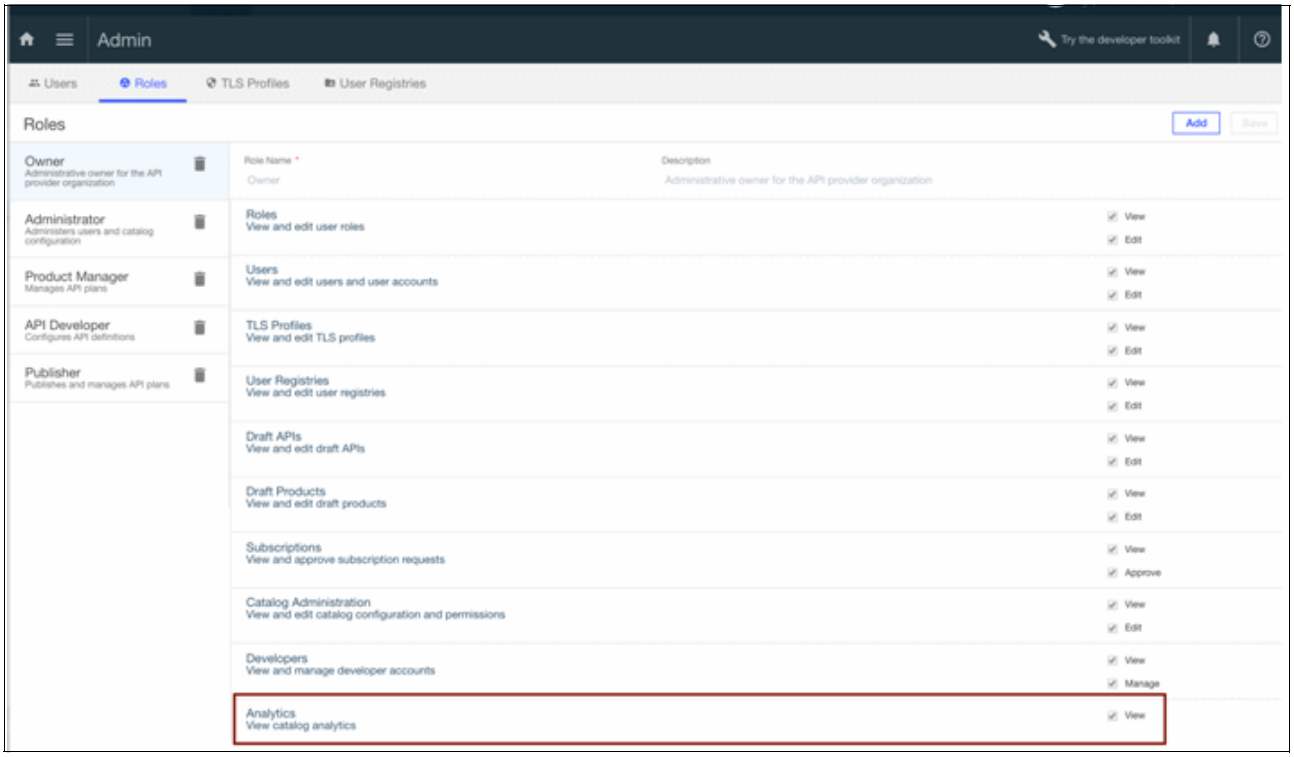

Figure 6-26 Roles view

2. Navigate now to the **ApicStore Catalog** → **Settings** → **Permissions**. See Figure 6-27.

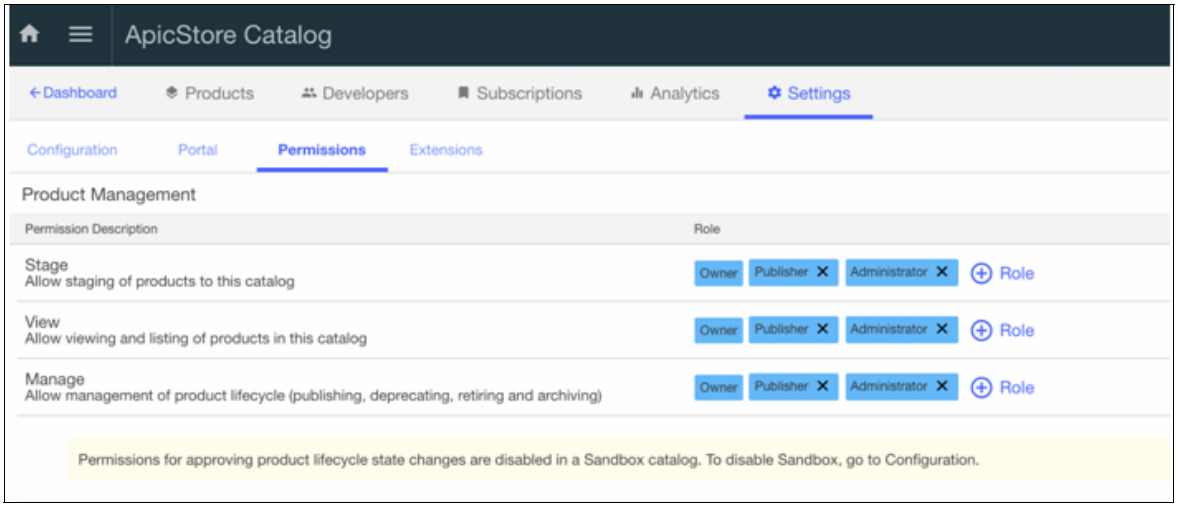

Figure 6-27 Permissions view

3. Make sure that the role of the user is listed in "View". Now that we have checked permissions, let's go to a dashboard that we want to share.
4. Click the **link icon** on the dashboard configuration. See Figure 6-28 on page 120.

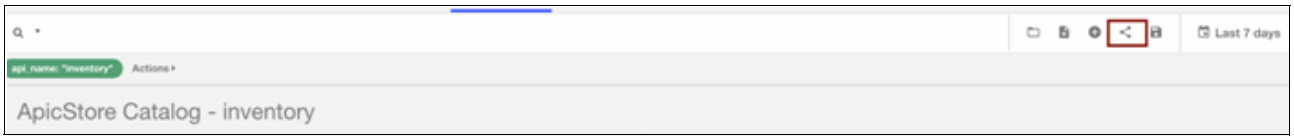

Figure 6-28 Click the link icon

5. You will see two large strings. One is an iframe that contains the dashboard. You can mash this up into a larger dashboard. Alternatively, you can copy the second string which is a link and share that. When you paste the link in a browser, you will be prompted for your API Connect credentials. If you are not authorized to view API Connect analytics, then you will not be permitted access.

6.5 Summary

In this chapter we have explored the out-of-the-box visualizations and dashboards that come with API Connect. We have discussed how these can help your business understand the usage and health of your APIs. We have shown how to customize visualizations and dashboards to study specific aspects of API usage. Finally, we have shown how dashboards can be shared with authorized users.

A

Deploying the authentication utility application

This appendix describes how to deploy the authentication utility application. This application is used by the socialreview OAuth Provider API in Chapter 4, “Securing the APIs” on page 65. In order to deploy this application, you need to the cloud foundry command line and in this appendix we provide some basic instructions on how to login and use that.

Steps for deploying the authentication utility application

Perform the following steps to deploy the authentication utility application. You can refer to following link for downloading the Cloud Foundry CLI package and other plugins: https://console.ng.bluemix.net/docs/cli/index.html#downloads.

1. Clone the `https://github.com/IBMRedbooks/redp5350-apiconnect-sample` repository to a local file system location such as D:\APIM50\Redbooks\redp5350. Refer to section "Clone the GitHub repository" on page 11 for more details. In the instructions that follows, please replace D:\APIM50\Redbooks\redp5350 with the directory that you use in your setup.
2. Go to the following directory:

   ```
   cd D:\APIM50\Redbooks\redp5350\OAuth\authentication-app.
   ```

3. Type **`ls`**. You should see the following files:

   ```
   app.js  manifest.yml  package.json  public
   ```

4. Type `bluemix api https://api.ng.bluemix.net`. You will receive the following (in this example we are using the user and org `bvperepa@us.ibm.com`.

   ```
   Setting api endpoint to https://api.ng.bluemix.net...
   OK
   API endpoint:   https://api.ng.bluemix.net (API version: 2.54.0)
   User:           bvperepa@us.ibm.com
   Org:            bvperepa@us.ibm.com
   Space:          dev
   ```

5. Run **`cf push authentication-app`**. You will see the following output:

```
Using manifest file
D:\APIM50\Redbooks\redp5350\OAuth\authentication-app\manifest.yml
...
...
Showing health and status for app authentication-app in org bvperepa@us.ibm.com
/ space dev as bvperepa@us.ibm.com...
OK

requested state: started
instances: 1/1
usage: 512M x 1 instances
urls: authentication-app.mybluemix.net, case-oauth-authenticate.mybluemix.net
last uploaded: Sat Jul 23 18:39:01 UTC 2016
stack: unknown
buildpack: SDK for Node.js(TM) (node.js-4.2.6, buildpack-v3.6-20160715-0749)

     state     since                    cpu    memory          disk details
#0   running   2016-07-23 02:39:49 PM   0.0%   67.3M of 512M   58.4M of 1G
```

These instructions and output of the commands can also be seen in Figure A-1 on page 123 and Figure A-2 on page 124.

```
D:\APIM50\Redbooks\redp5350\OAuth\authentication-app>ls
app.js  manifest.yml  package.json  public

D:\APIM50\Redbooks\redp5350\OAuth\authentication-app>bluemix api https://api.ng.
bluemix.net
Invoke 'cf api https://api.ng.bluemix.net'...

Setting api endpoint to https://api.ng.bluemix.net...
OK


API endpoint:   https://api.ng.bluemix.net (API version: 2.54.0)
User:           bvperepa@us.ibm.com
Org:            bvperepa@us.ibm.com
Space:          dev


D:\APIM50\Redbooks\redp5350\OAuth\authentication-app>cf push authentication-app
Using manifest file D:\APIM50\Redbooks\redp5350\OAuth\authentication-app\manifes
t.yml

Updating app authentication-app in org bvperepa@us.ibm.com / space dev as bvpere
pa@us.ibm.com...
OK

Creating route case-oauth-authenticate.mybluemix.net...
OK

Binding case-oauth-authenticate.mybluemix.net to authentication-app...
OK

Uploading authentication-app...
Uploading app files from: D:\APIM50\Redbooks\redp5350\OAuth\authentication-app
Uploading 8.1K, 7 files
Done uploading
OK

Stopping app authentication-app in org bvperepa@us.ibm.com / space dev as bvpere
pa@us.ibm.com...
OK

Starting app authentication-app in org bvperepa@us.ibm.com / space dev as bvpere
pa@us.ibm.com...
-----> Downloaded app package (8.0K)

-----> IBM SDK for Node.js Buildpack v3.6-20160715-0749
       Based on Cloud Foundry Node.js Buildpack v1.5.14
-----> Creating runtime environment
       NPM_CONFIG_LOGLEVEL=error
       NPM_CONFIG_PRODUCTION=true
       NODE_ENV=production
       NODE_MODULES_CACHE=true
-----> Installing binaries
       engines.node (package.json):  4.2.x
       engines.npm (package.json):   unspecified (use default)
       Resolving node version 4.2.x via 'node-version-resolver'
       Downloading and installing node 4.2.6...
       Using default npm version: 2.14.12
-----> Restoring cache
       Skipping cache restore (new runtime signature)
-----> Building dependencies
       Pruning any extraneous modules
       Installing node modules (package.json)
       basic-auth@1.0.4 node_modules/basic-auth
       body-parser@1.15.2 node_modules/body-parser
       ├── content-type@1.0.2
       ├── bytes@2.4.0
       ├── depd@1.1.0
       ├── qs@6.2.0
       ├── on-finished@2.3.0 (ee-first@1.1.1)
       ├── raw-body@2.1.7 (unpipe@1.0.0)
       ├── http-errors@1.5.0 (setprototypeof@1.0.1, inherits@2.0.1, statuses@1.3
.0)
       ├── debug@2.2.0 (ms@0.7.1)
       └── type-is@1.6.13 (media-typer@0.3.0, mime-types@2.1.11)
       express@4.13.4 node_modules/express
       ├── array-flatten@1.1.1
       ├── methods@1.1.2
       ├── content-type@1.0.2
       ├── cookie-signature@1.0.6
       ├── utils-merge@1.0.0
       ├── merge-descriptors@1.0.1
```

Figure A-1 Deploying the authentication utility application - 1

```
D:\APIM50\Redbooks\redp5350\OAuth\authentication-app>ls
app.js  manifest.yml  package.json  public

D:\APIM50\Redbooks\redp5350\OAuth\authentication-app>bluemix api https://api.ng.
bluemix.net
Invoke 'cf api https://api.ng.bluemix.net'...

Setting api endpoint to https://api.ng.bluemix.net...
OK


API endpoint:   https://api.ng.bluemix.net (API version: 2.54.0)
User:           bvperepa@us.ibm.com
Org:            bvperepa@us.ibm.com
Space:          dev


D:\APIM50\Redbooks\redp5350\OAuth\authentication-app>cf push authentication-app
Using manifest file D:\APIM50\Redbooks\redp5350\OAuth\authentication-app\manifes
t.yml

Updating app authentication-app in org bvperepa@us.ibm.com / space dev as bvpere
pa@us.ibm.com...
OK

Creating route case-oauth-authenticate.mybluemix.net...
OK

Binding case-oauth-authenticate.mybluemix.net to authentication-app...
OK

Uploading authentication-app...
Uploading app files from: D:\APIM50\Redbooks\redp5350\OAuth\authentication-app
Uploading 8.1K, 7 files
Done uploading
OK

Stopping app authentication-app in org bvperepa@us.ibm.com / space dev as bvpere
pa@us.ibm.com...
OK

Starting app authentication-app in org bvperepa@us.ibm.com / space dev as bvpere
pa@us.ibm.com...
-----> Downloaded app package (8.0K)

-----> IBM SDK for Node.js Buildpack v3.6-20160715-0749
       Based on Cloud Foundry Node.js Buildpack v1.5.14
-----> Creating runtime environment
       NPM_CONFIG_LOGLEVEL=error
       NPM_CONFIG_PRODUCTION=true
       NODE_ENV=production
       NODE_MODULES_CACHE=true
-----> Installing binaries
       engines.node (package.json):  4.2.x
       engines.npm (package.json):   unspecified (use default)
       Resolving node version 4.2.x via 'node-version-resolver'
       Downloading and installing node 4.2.6...
       Using default npm version: 2.14.12
-----> Restoring cache
       Skipping cache restore (new runtime signature)
-----> Building dependencies
       Pruning any extraneous modules
       Installing node modules (package.json)
       basic-auth@1.0.4 node_modules/basic-auth
       body-parser@1.15.2 node_modules/body-parser
       ├── content-type@1.0.2
       ├── bytes@2.4.0
       ├── depd@1.1.0
       ├── qs@6.2.0
       ├── on-finished@2.3.0 (ee-first@1.1.1)
       ├── raw-body@2.1.7 (unpipe@1.0.0)
       ├── http-errors@1.5.0 (setprototypeof@1.0.1, inherits@2.0.1, statuses@1.3
.0)
       ├── debug@2.2.0 (ms@0.7.1)
       └── type-is@1.6.13 (media-typer@0.3.0, mime-types@2.1.11)
       express@4.13.4 node_modules/express
       ├── array-flatten@1.1.1
       ├── methods@1.1.2
       ├── content-type@1.0.2
       ├── cookie-signature@1.0.6
       ├── utils-merge@1.0.0
       ├── merge-descriptors@1.0.1
```

Figure A-2 Deploying the authentication utility application - 2

Figure A-3 Deploying the authentication utility application - 3

6. Note that Figure A-3 on page 125 shows the URL to test the application. Now we will use this link to test the application. Type the following in a browser window:

http://authentication-app.mybluemix.net/login.html. You should see the login screen in Figure A-4.

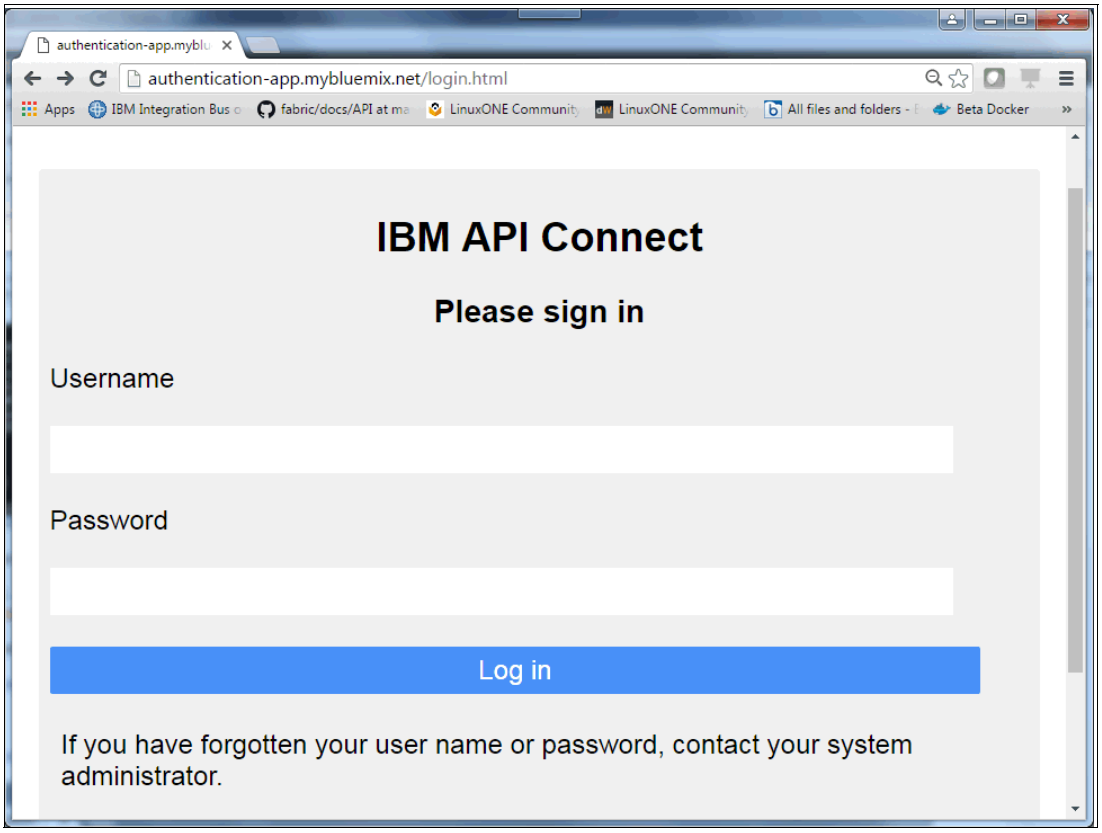

Figure A-4 Login screen

7. You can also test the deployed application. Type http://authentication-app.mybluemix.net/grant.html in a browser. You should see the following screen as shown in Figure A-5 on page 127.

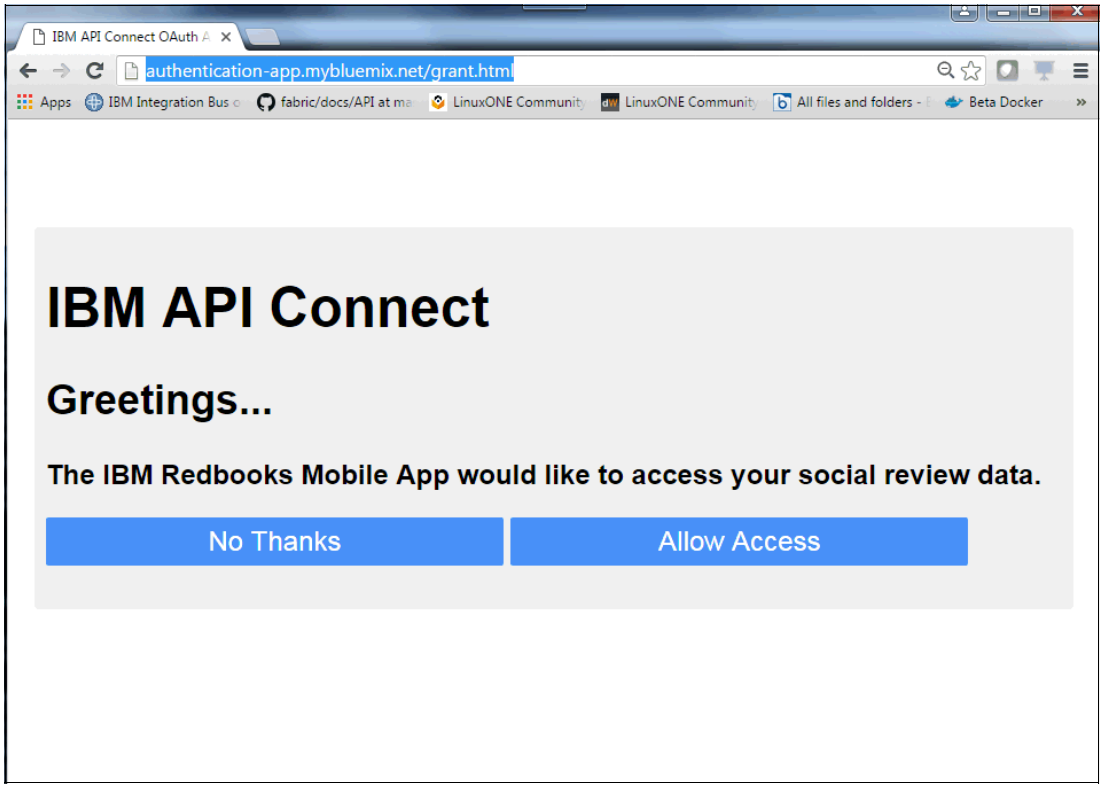

Figure A-5 Testing the deployed application

This shows that you have successfully deployed the authentication utility application.

B

Additional material

This paper refers to additional material that can be downloaded from the Internet as described in the following sections.

Locating the Web material

The Web material associated with this paper is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser at:

`ftp://www.redbooks.ibm.com/redbooks/REDP5350`

Alternatively, you can go to the IBM Redbooks website at:

`ibm.com/redbooks`

Select the **Additional materials** and open the directory that corresponds with the IBM Redpaper form number, REDP5350.

Using the Web material

The additional Web material that accompanies this paper includes the following files:

File name	*Description*
REDP5350.zip	Zipped code for the sample scenario

System requirements for downloading the Web material

The Web material requires the following system configuration:

Hard disk space: 10 MB minimum
Operating System: Windows or Linux

Downloading and extracting the Web material

Create a subdirectory (folder) on your workstation, and extract the contents of the Web material .zip file into this folder.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this paper.

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- *Getting Started with IBM API Connect: Concepts, Architecture and Strategy Guide*, REDP-5349
- *Hybrid Cloud Data and API Integration: Integrate Your Enterprise and Cloud with Bluemix Integration Services*, SG24-8277
- *Hybrid Cloud Event Integration: Integrate Your Enterprise and Cloud with Bluemix Integration Services*, SG24-8281

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Other publications

These publications are also relevant as further information sources:

- *????full title???????,* xxxx-xxxx
- *????full title???????,* xxxx-xxxx
- *????full title???????,* xxxx-xxxx

Online resources

These websites are also relevant as further information sources:

- IBM API Connect V5.0 IBM Knowledge Center:

  https://www.ibm.com/support/knowledgecenter/SSMNED_5.0.0/mapfiles/getting_started.html

- Description2

  http://????????.???.???/

- Description3

  http://????????.???.???/

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

REDP-5350-00

ISBN DocISBN

Printed in U.S.A.