

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





International Technical Support Organization

**Getting Started with IBM API Connect: Scenarios
Guide**

September 2016

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

First Edition (September 2016)

This edition applies to IBM API Connect Version 5.0.

© Copyright International Business Machines Corporation 2016. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

| | |
|---|------|
| Notices | vii |
| Trademarks | viii |
| IBM Redbooks promotions | ix |
| Preface | xi |
| Authors | xi |
| Now you can become a published author, too! | xiv |
| Comments welcome | xiv |
| Stay connected to IBM Redbooks | xv |
| Chapter 1. Introduction to IBM API Connect and environment setup | 1 |
| 1.1 Introduction to IBM API Connect | 2 |
| 1.1.1 Developer Toolkit | 2 |
| 1.1.2 API Manager Console | 2 |
| 1.1.3 Developer Portal | 3 |
| 1.2 IBM API Connect sample scenario | 3 |
| 1.3 Environment setup | 4 |
| 1.3.1 Provisioning the IBM API Connect service in Bluemix | 4 |
| 1.3.2 Installing the IBM API Connect Developer Toolkit | 9 |
| 1.3.3 Downloading the applications from GitHub | 10 |
| Chapter 2. Developing and deploying LoopBack applications and APIs | 13 |
| 2.1 What is LoopBack? | 14 |
| 2.1.1 IBM API Connect and LoopBack | 14 |
| 2.1.2 LoopBack core concepts | 14 |
| 2.2 IBM API Connect Developer Toolkit | 16 |
| 2.2.1 Installing IBM API Connect Developer Toolkit | 16 |
| 2.2.2 IBM API Connect Developer Toolkit command line interface | 16 |
| 2.2.3 IBM API Connect Developer Toolkit API Designer | 17 |
| 2.2.4 Products in the API Designer | 18 |
| 2.3 Sample application scenario | 19 |
| 2.4 Develop LoopBack applications and APIs | 20 |
| 2.4.1 Inventory LoopBack application from Company B | 20 |
| 2.4.2 Social Reviews LoopBack application from Company C | 29 |
| 2.5 Starting and testing the LoopBack applications locally | 38 |
| 2.5.1 Starting the inventory application | 38 |
| 2.5.2 Testing the inventory application | 38 |
| 2.5.3 Stopping the inventory application | 39 |
| 2.5.4 Starting the socialreviews application | 39 |
| 2.5.5 Testing the socialreviews application | 40 |
| 2.5.6 Stopping the socialreviews application | 40 |
| 2.6 Testing the APIs | 41 |
| 2.6.1 Testing the inventory application API | 41 |
| 2.6.2 Testing the socialreviews application API | 43 |
| 2.7 Update the LoopBack application gateway | 44 |
| 2.7.1 Updating the inventory LoopBack application gateway | 45 |
| 2.7.2 Updating the socialreviews LoopBack application gateway | 45 |
| 2.8 Deploying LoopBack applications to IBM Bluemix | 46 |

| | | |
|-----------------------------|---|------------|
| 2.8.1 | Configuring the Bluemix environment | 46 |
| 2.8.2 | Deploying inventory application in Bluemix | 47 |
| 2.8.3 | Deploying socialreviews application in Bluemix | 49 |
| 2.9 | Summary | 53 |
| Chapter 3. | Managing the APIs | 55 |
| 3.1 | Publishing APIs by using API Designer user interface | 56 |
| 3.1.1 | Understanding the API lifecycle | 56 |
| 3.2 | Publishing APIs by using the apic command line tool | 61 |
| 3.3 | Summary | 63 |
| Chapter 4. | Securing the APIs | 65 |
| 4.1 | Overview | 66 |
| 4.2 | Application security | 66 |
| 4.3 | OAuth Security to protect user resources | 67 |
| 4.3.1 | Configuring the OAuth Provider API | 67 |
| 4.3.2 | Summary | 70 |
| 4.3.3 | Identity extraction | 71 |
| 4.3.4 | Authentication | 72 |
| 4.3.5 | Authorization | 73 |
| 4.3.6 | Tokens | 74 |
| 4.3.7 | Securing your API | 74 |
| 4.4 | Summary | 77 |
| Chapter 5. | Using the APIs | 79 |
| 5.1 | Sign up to use API | 80 |
| 5.1.1 | Registering an application | 80 |
| 5.1.2 | Subscribing to an API plan | 84 |
| 5.1.3 | Testing an API from the development portal | 86 |
| 5.2 | Developing the mobile iOS application | 89 |
| 5.3 | Running the sample consumer web application | 97 |
| 5.4 | Summary | 100 |
| Chapter 6. | Monitoring and analyzing the APIs | 101 |
| 6.1 | Introduction to IBM API Connect Analytics | 102 |
| 6.2 | IBM API Connect dashboards | 102 |
| 6.2.1 | Default catalog dashboard | 103 |
| 6.2.2 | Default product dashboard | 105 |
| 6.2.3 | Default API dashboard | 107 |
| 6.2.4 | Default plan dashboard | 108 |
| 6.2.5 | Default portal dashboard | 108 |
| 6.3 | Customizing visualizations and dashboards | 109 |
| 6.4 | Sharing dashboards | 115 |
| 6.5 | Summary | 117 |
| Appendix A. | Deploying the authentication utility application | 119 |
| | Deploying the authentication utility application | 120 |
| Appendix B. | Additional material | 127 |
| | Locating the Web material | 127 |
| | Using the Web material | 127 |
| | System requirements for downloading the Web material | 127 |
| | Downloading and extracting the Web material | 128 |
| Related publications | | 129 |

IBM Redbooks 129

Online resource 129

Help from IBM 129

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.

Bluemix®
DataPower®
IBM®
IBM API Connect™

IBM Watson™
Rational®
Redbooks®
Redpaper™

Redbooks (logo) ®
Smarter Cities®

The following terms are trademarks of other companies:

LoopBack, StrongLoop, and the StrongLoop logo are trademarks of StrongLoop, Inc., an IBM Company.

ITIL is a Registered Trade Mark of AXELOS Limited.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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. By using these capabilities, the full lifecycle of APIs for on-premises and cloud environments can be managed.

This IBM Redpaper™ publication describes practical scenarios that show the IBM API Connect™ capabilities for managing the full API life cycle, creating, running, securing, and managing the APIs. This Redpaper publication is targeted to users of an IBM API Connect based API strategy, developers, IT architects, and technical evangelists.

If you are not familiar with APIs or IBM API Connect, we suggest that you read the Redpaper publication *Getting Started with IBM API Connect: Concepts and Architecture Guide*, REDP-5349, before reading this publication.

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. He is an automation enthusiast and usability guru, and is 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 and Delivery and Testing. He focuses on integration testing in the API Economy through service virtualization. Before joining IBM, Alex served as the Senior DevOps Architect at TIAA-CREF for 4 years. Before that role, 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 is a data scientist with a strong background in Natural Language Processing and Cognitive computing and 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 previously supported the Communications sector, developing solutions for the Telecommunications and Media and Entertainment industries. During this time, he developed multiple solutions, including Audience Insights, IBM Smarter Cities® solution in collaboration with AT&T, IBM 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 over 10 years of Enterprise development and consulting experience in several different industries. He focuses on the MobileFirst platform and IBM API Connect.



Gang Chen is an IBM Executive IT Specialist from IBM Cloud Architecture and Solution Engineering team. He is a leader 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. Before 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's degree in Software Management from Middle East Technical University. His areas of expertise include Change and Configuration, Requirement, Quality, and API Management. He works as an IBM Rational® consultant for various clients in the 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 also worked as a Services Architect with various IBM customers in North America. He has authored several IBM Redbooks® publications about messaging, mobile, and cloud computing. He is a recognized inventor by the 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 12 product certificates, such as IBM Worklight, Oracle Java Business Component, IBM SOA Associate, and Oracle Java Programmer. She also received many awards for her achievements, such as 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 and is the author of many technical articles. 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 about this topic was not possible without the generous support and guidance from key members in the IBM API Connect product and development organizations, and many others at IBM.

The authors team want 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>



Introduction to IBM API Connect and environment setup

This chapter provides an overview of the sample scenario that we use to demonstrate the tasks of an API Provider and an API Consumer. The scenario shows how IBM 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 IBM API Connect. Instructions also are provided for setting up your local environment and the IBM API Connect service in Bluemix so that you can follow along and attempt the scenario that is described in this book.

This chapter includes the following topics:

- ▶ 1.1, “Introduction to IBM API Connect” on page 2
- ▶ 1.2, “IBM API Connect sample scenario” on page 3
- ▶ 1.3, “Environment setup” on page 4

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. IBM 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 IBM API Connect as a Bluemix service.

Note: For more information about IBM API Connect, see the IBM Redpaper *Getting Started with IBM API Connect: Concepts, Architecture and Strategy Guide*, REDP-5349.

The IBM API Connect Bluemix service features the following interfaces:

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

These interfaces are described next.

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 by using the Micro Gateway. The Micro Gateway is a Node.js-based gateway that is built on StrongLoop® technology. It provides a subset of the policies that are available in the IBM DataPower® Gateway. After APIs are designed and tested, they can be published to an API Manager catalog by using the Designer console or the command line interface (CLI).

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 more information about LoopBack, see Chapter 2, “Developing and deploying LoopBack applications and APIs” on page 13.

This paper describes a scenario that uses LoopBack to implement a REST API. If you have REST or WSDL assets, you can use the API Designer to configure your APIs to directly connect to those assets.

1.1.2 API Manager Console

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

- ▶ 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 with which consumers can browse and subscribe to APIs. The Developer Portal provides access to documentation, sample code for API invocations, and live “Try It” capability. Consumers acquire API credentials by registering applications in the Developer Portal. They can also monitor API usage.

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

<https://ibm.biz/Bdrgee>

1.2 IBM API Connect sample scenario

This paper uses a sample application to walk through an end-to-end API lifecycle that uses IBM API Connect. The scenario includes the following tasks:

1. Implement an API as a Node.js application that uses LoopBack.
2. Secure the API.
3. Test the API.
4. Deploy the API implementation (application) to Bluemix.
5. Publish the 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 starts the API.

Bob (the user) downloads a mobile application that is called IBM Redbooks Mobile App A. This application is built by Digital Agency Company A. The application lists 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 is interested in “Punched-card tabulating machines.” He taps the item and the Mobile application shows the detail about this item. The IBM Redbooks Mobile Application A pulls the item and 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 that use this website share his interests.

Bob liked what he saw. To share some thoughts about Social Sharing Provider Company C, he taps the button to write and share a comment. The Mobile application starts the OAuth flow that is provided by Social Sharing Provider Company C. Bob grants IBM Redbooks Mobile Application A access to his Social Sharing Provider Company C account. Then, he posts some comments that are associated with this item.

The high-level architecture of this scenario is shown in Figure 1-1.

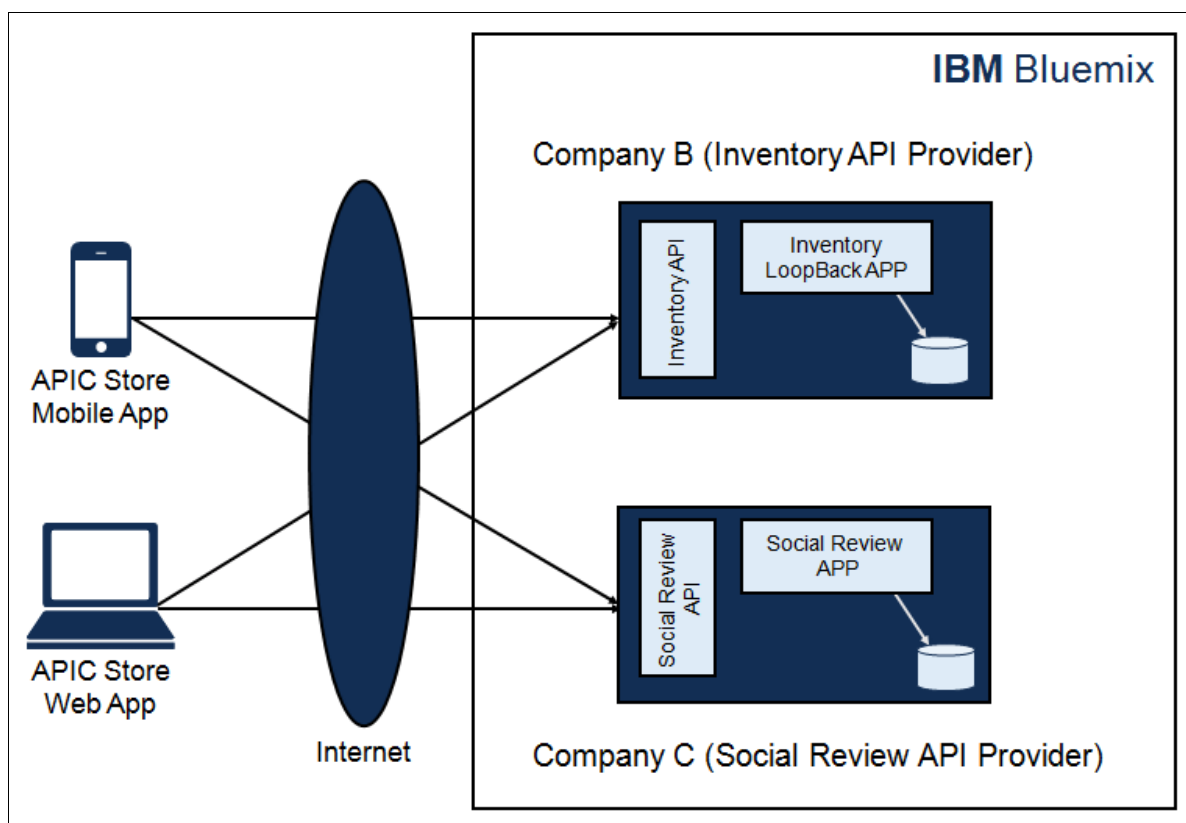


Figure 1-1 High-level architecture of the sample scenario

1.3 Environment setup

Before a LoopBack application and API can be created, the environment must be set up by provisioning IBM API Connect on Bluemix and installing the Developer Toolkit locally.

1.3.1 Provisioning the IBM API Connect service in Bluemix

To provision IBM API Connect, you must have a Bluemix account. Log in to your Bluemix account or register for a new Bluemix account at the following website:

<https://bluemix.net/registration>

After you are logged in, create a space for hosting the sample application that is created with LoopBack and for provisioning the IBM API Connect service to manage the API.

Creating space in Bluemix organization

Complete the following steps to create a space in Bluemix organization:

1. Click **Bluemix account** in top-right corner.
2. Create a space.
3. Create a space with name "IBM Redbooks".

Creating an IBM API Connect service

Complete the following steps to create an IBM API Connect service:

1. Click **Console** and select **APIs**, as shown in Figure 1-2.

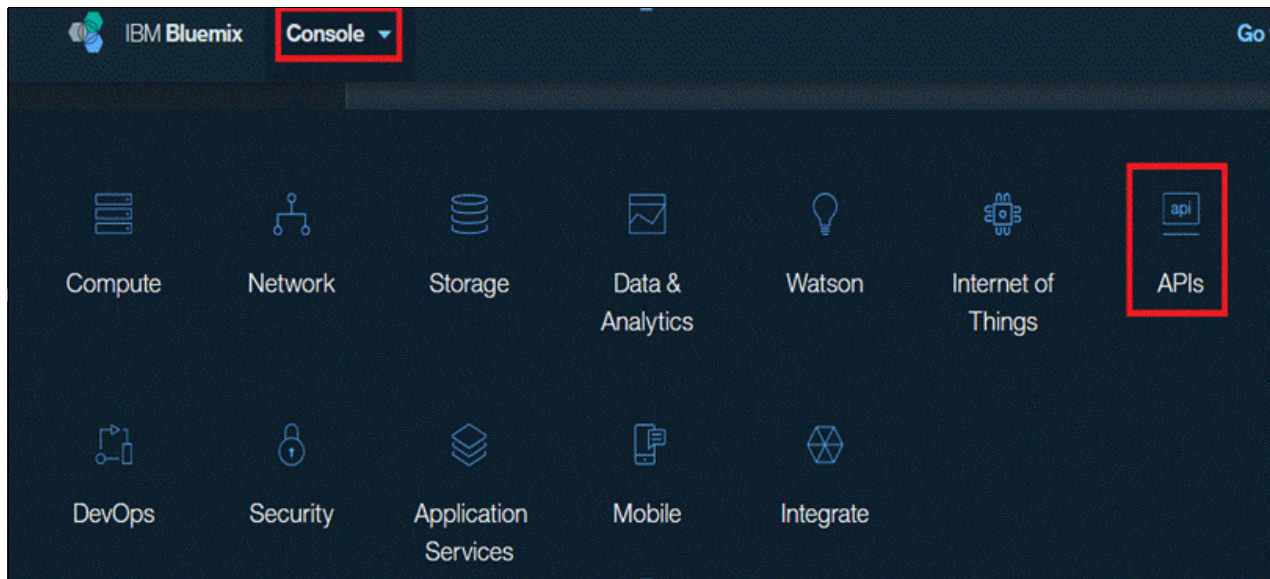


Figure 1-2 Selecting API

2. Select **API Connect**, as shown in Figure 1-3.

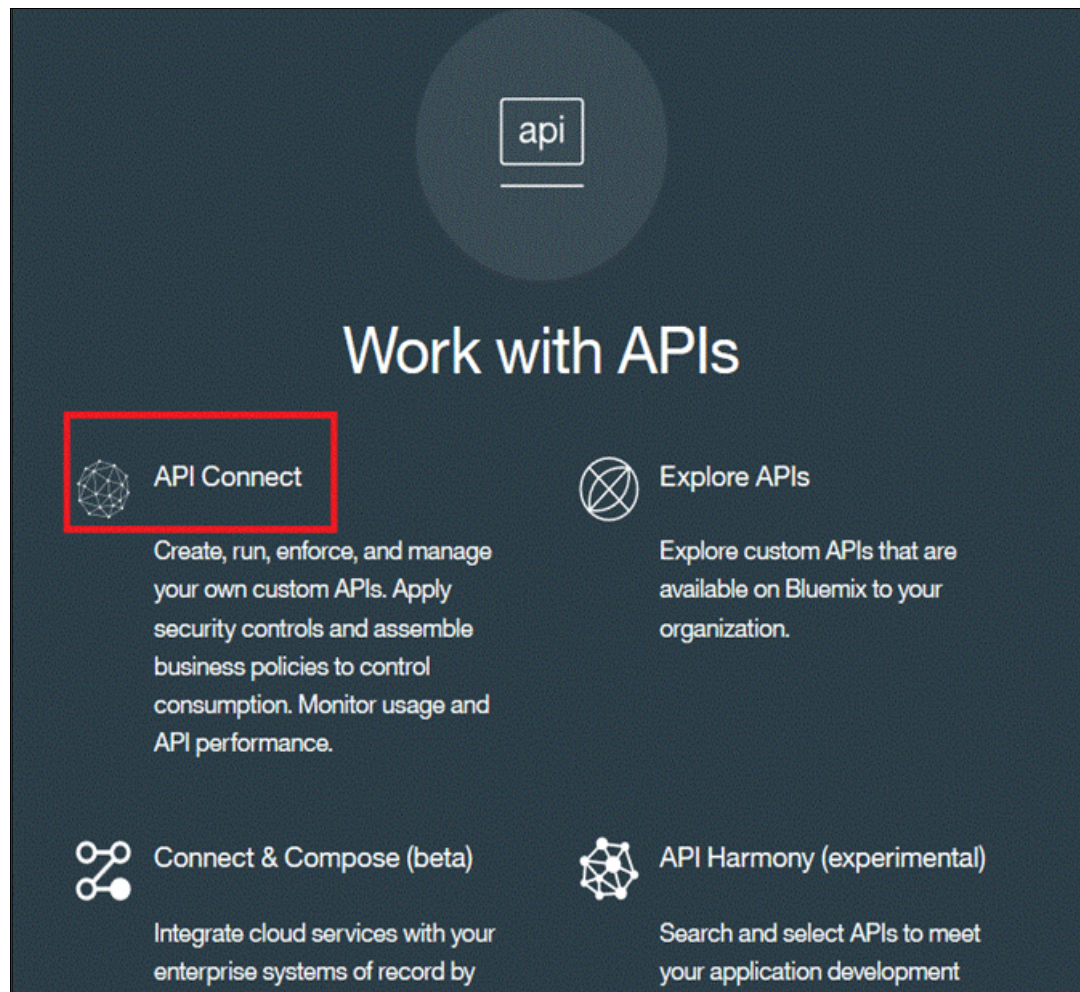


Figure 1-3 Selecting API Connect

3. On the next page, click **Create**.
4. Select **Essentials Plan** and click **Create**.
5. Click **Launch API Manager** (see Figure 1-4).

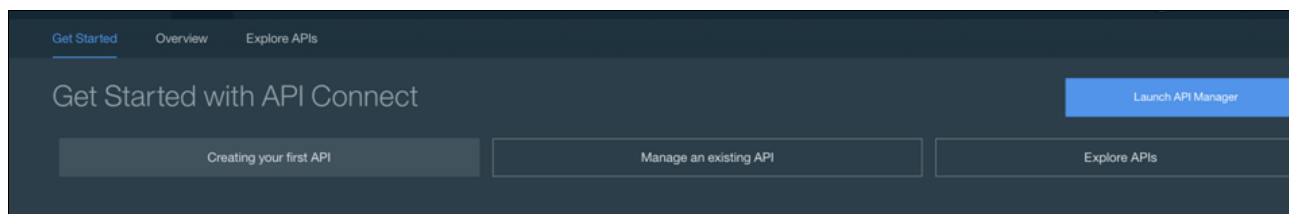


Figure 1-4 Get Started with API Connect panel

6. If prompted for your credentials, enter your Bluemix account User ID and Password.
We 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. APIs can be packaged by multiple products, as shown in Figure 1-5.

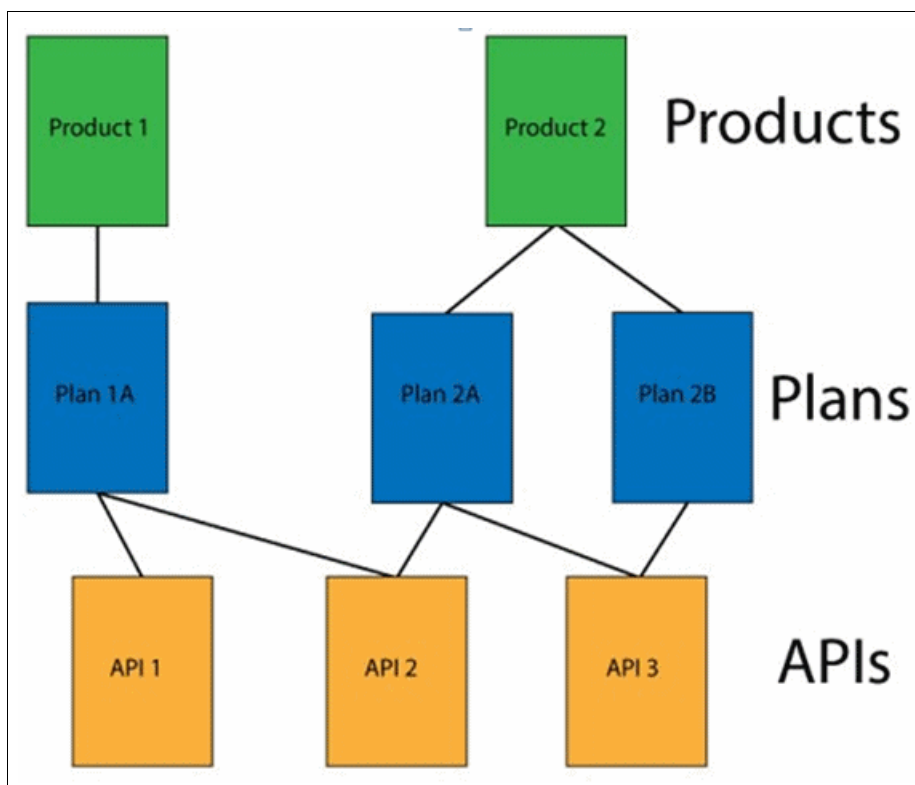


Figure 1-5 Products, plans, and APIs

When you publish an API Product to a catalog, that API Product becomes available on the Developer Portal that is 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 a user signs up or is invited by the API Provider to a Developer Portal, a 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 can be subscribed to the public to users who are not authenticated or signed into the Developer Portal. An example of the use of Developer Organizations is having a different Developer Organization for each Business Partner. This configuration allows you to control which API Products each Business Partner can see and subscribe to. It also isolates Business Partners from each other in terms of registered applications and usage data.

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

7. After starting the API Manager, browse to the API Connect Dashboard and select **Add Catalog** at the top left, as shown in Figure 1-6 on page 8. You notice that a *Sandbox Catalog* was automatically generated for you, which is what we want. We are going to create a catalog for our scenario.

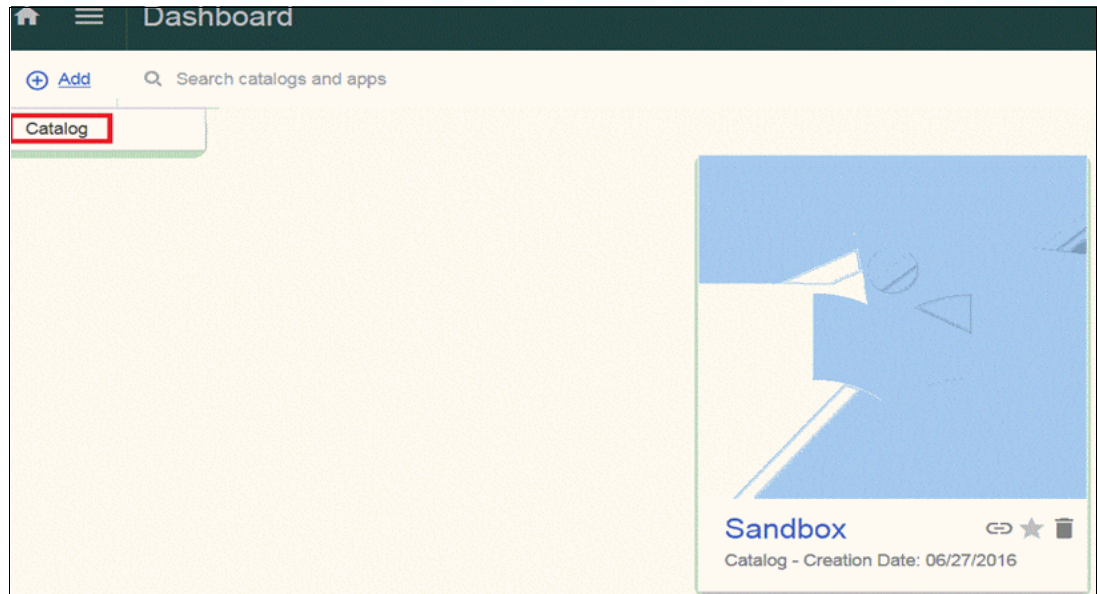


Figure 1-6 Create a catalog

8. Name the catalog ApicStore Catalog and click **Add**, as shown in Figure 1-7.

Figure 1-7 Adding a catalog

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

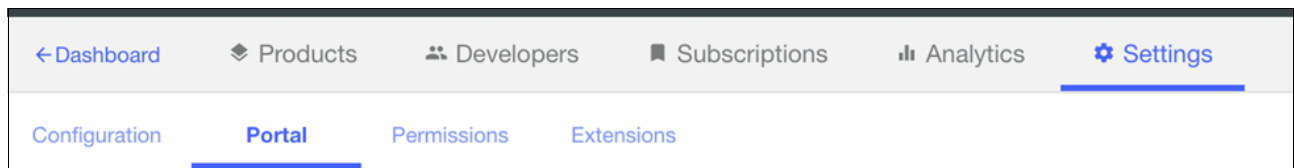


Figure 1-8 Settings and Portal tabs

10. To set up a Portal that your consumers can use to explore your APIs, select the **IBM Developer Portal** option, which provisions a Portal for you. You receive a message as shown in Figure 1-9.

Thank you for enabling the IBM Developer Portal. Please note that due to the time taken for the DNS definition to propagate around the Internet it may take up to 2 hours for your Portal to become accessible

Figure 1-9 Provisioning a Portal

After the new Developer Portal is created and active, you receive an email.

Tip: Ensure that you note 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 authenticate with their IBM ID (Bluemix account) credentials. Leave the remaining defaults as well, as shown in Figure 1-10. Click **Save** after making these changes.

User Registration and Invitation

User Registry

IBM ID ▼

☒ Developers can invite collaborators and assign the following roles:

- ☐ Viewer
Viewers can only view applications and application activity.
- ☐ App Developer
App developers can create and edit applications, manage client keys and subscribe to plans.
- ☒ Viewer and App Developer

☒ Self-service onboarding

Figure 1-10 User Registration

1.3.2 Installing the IBM API Connect Developer Toolkit

Now that you set up the IBM API Connect service, you must 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 the API Designer user interface and a CLI that developers use to develop APIs and LoopBack applications and publish them to the IBM API Connect run time.

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 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.

The IBM API Connect Developer Toolkit can be installed by using the **npm** command or from the management server in IBM API Connect cloud. For more information about installing the toolkit, see this website:

<https://ibm.biz/BdrE2P>

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

1.3.3 Downloading the applications from GitHub

The LoopBack, security, web application, and mobile applications that are developed throughout this Redpaper publication are provided in the public GitHub repository for as-is use. The different projects in this repository are shown in Figure 1-11.

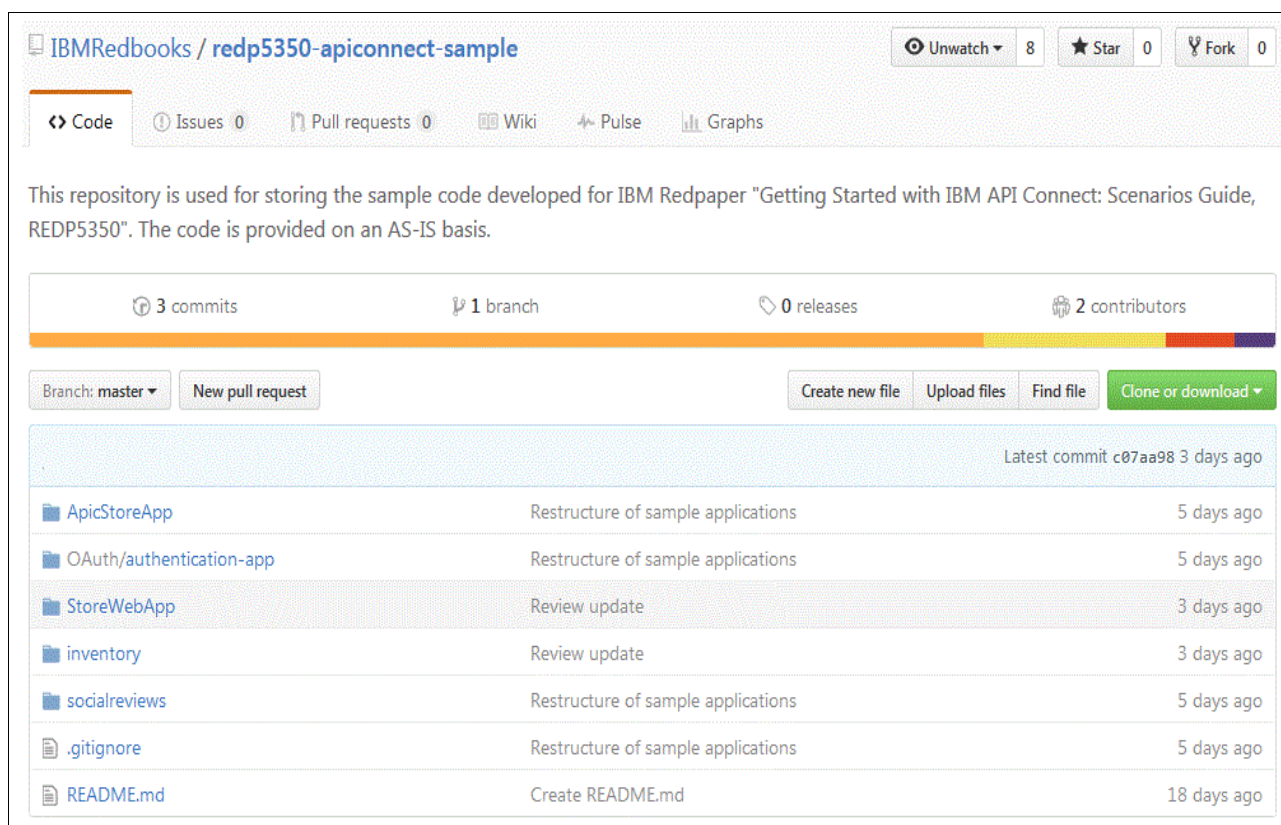


Figure 1-11 GitHub repository for IBM Redpaper REDP5350

This repository can be accessed by using the following GitHub URL:

<https://github.com/IBMRedbooks/redp5350-apiconnect-sample>

The different projects in the GitHub repository are listed in Table 1-1.

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 application for accessing the items are reviewed through APIs |

Cloning the GitHub repository

Use the `git clone` command to clone the GitHub repositories, as shown in Example 1-1.

Example 1-1 Cloning the Redpaper publication 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 the Git client if it is not installed before cloning the code repository. For more information about installing the client, see this website:

<https://git-scm.com/downloads>



Developing and deploying LoopBack applications and APIs

This chapter describes 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 topics:

- ▶ 2.1, “What is LoopBack?” on page 14
- ▶ 2.2, “IBM API Connect Developer Toolkit” on page 16
- ▶ 2.3, “Sample application scenario” on page 19
- ▶ 2.4, “Develop LoopBack applications and APIs” on page 20
- ▶ 2.5, “Starting and testing the LoopBack applications locally” on page 38
- ▶ 2.6, “Testing the APIs” on page 41
- ▶ 2.7, “Update the LoopBack application gateway” on page 44
- ▶ 2.8, “Deploying LoopBack applications to IBM Bluemix” on page 46
- ▶ 2.9, “Summary” on page 53

2.1 What is LoopBack?

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

- ▶ 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 applications by 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 run time for API implementation. IBM API Connect has the Loopback framework built-in.

For more information, see 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 make them available as usable APIs.

2.1.2 LoopBack core concepts

We introduce LoopBack core concepts in this section.

LoopBack models

LoopBack models are at the heart of LoopBack application and represent backend data sources, such as databases or other back end services (REST, SOAP, and so on). LoopBack models are JavaScript objects with Node and REST APIs.

A key powerful feature of LoopBack is that when a developer defines a model, it automatically includes a predefined REST API with a full set of create, read, update, and delete operations. Every LoopBack application features a set of predefined built-in models, such as User, Role, and Application, so a developer does not have to create these common models from scratch. Developers can define their own custom models that are specific to their application.

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

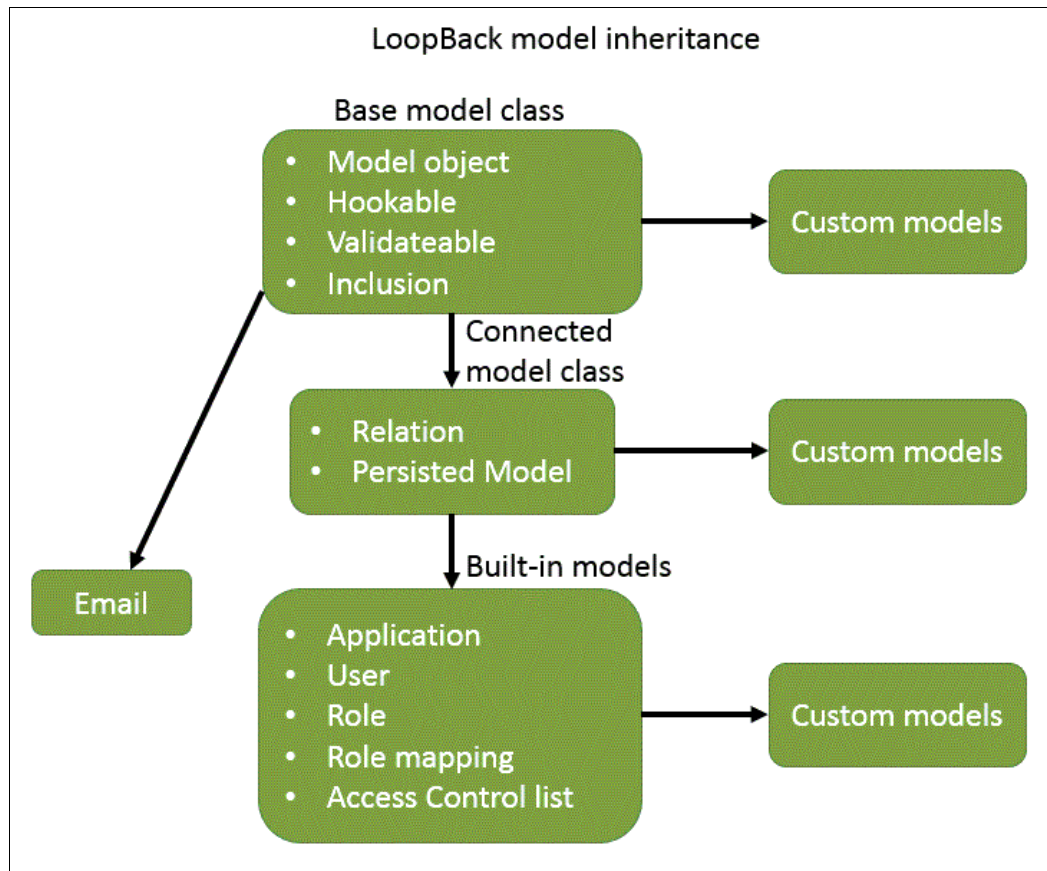


Figure 2-1 LoopBack model inheritance

Application logic

Developers can add custom application logic by using the following 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, which is similar to Express middleware framework.

For more information about the Express framework, see this website:

<http://expressjs.com/>

Middleware phases

Middleware refers to functions that are when HTTP requests are made to REST endpoints. Because 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. The use of phases helps to avoid ordering issues that can occur with standard Express middleware.

Data sources and connectors

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

Data sources are backed by connectors that then communicate directly with the database or other back-end service. Applications do not use connectors directly; rather, they go through data sources by using the data source and PersistedModel APIs.

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

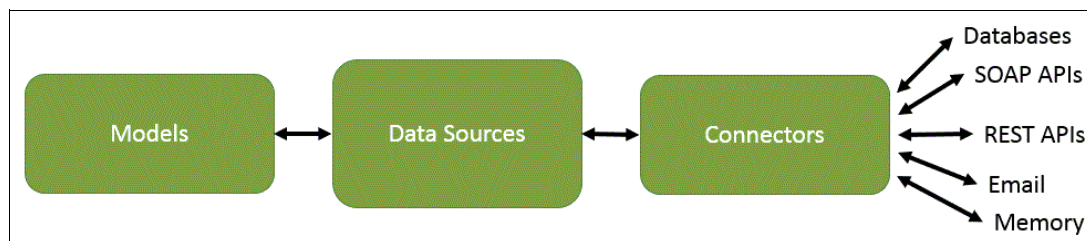


Figure 2-2 LoopBack data sources and connectors

2.2 IBM API Connect Developer Toolkit

The IBM API Connect Developer Toolkit is described in this section.

2.2.1 Installing IBM API Connect Developer Toolkit

To get started with IBM API Connect, the developer toolkit should be installed. For more information about the steps for installing IBM API Connect Developer Toolkit, see 1.3.2, “Installing the IBM API Connect Developer Toolkit” on page 9.

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

2.2.2 IBM API Connect Developer Toolkit command line interface

The IBM API Connect Developer Toolkit features a command line interface (CLI) that is named **apic** to augment the toolset that developers use to create and test APIs to be run, managed, and secured by IBM API Connect. The **apic** command line set also provides capability to support devops oriented engineering tasks (for example, continuous integration and delivery).

The various CLI options for the **apic** command are shown in Example 2-1.

Example 2-1 Command options for apic

```
>apic -h
```

```
Usage: apic COMMAND OPTIONS
```

```
Options
```

```
-h, --help      command usage
-v, --version   toolkit version
```

```
Commands (type apic COMMAND -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 of the **apic** commands use a **command:action** syntax (for example, **apic apps:publish**). For the most popular commands, the command or action portion is optional to simplify usage, as shown in the following example:

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

All of these 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 that is provided by the toolkit to support most of the capability that is 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 that is shown in Example 2-2 on page 17 is run from the command line, the designer tool opens in a web browser. The API Designer tool is shown in Figure 2-3.

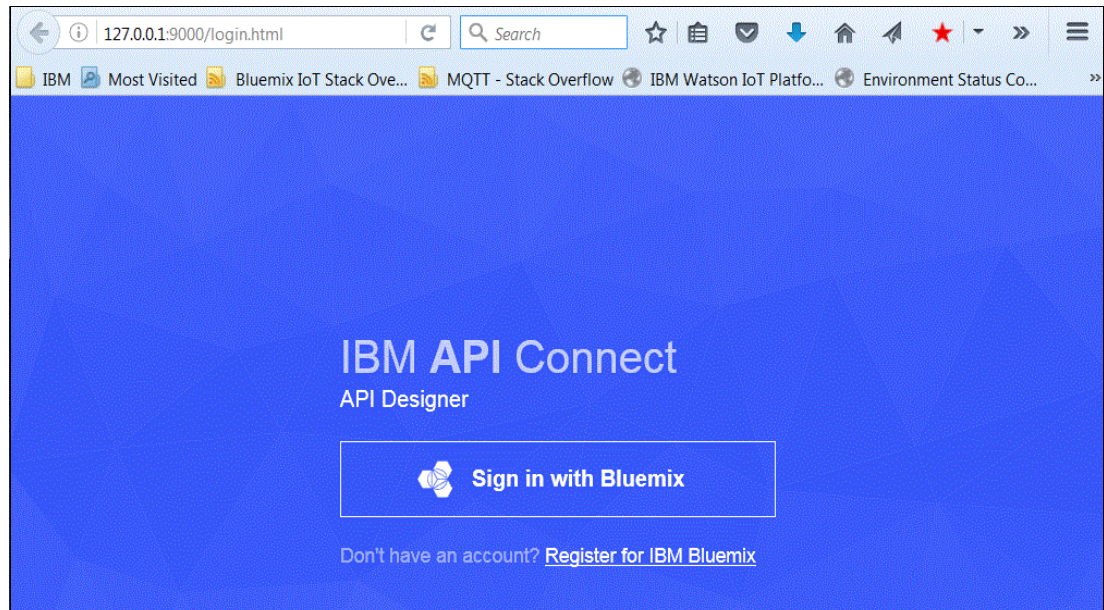


Figure 2-3 API Designer tool

Use the IBM Bluemix login credentials to log in to the designer tool. After you are logged in, you can start developing products and APIs. The designer tool to add an API product for a company is shown in Figure 2-4.

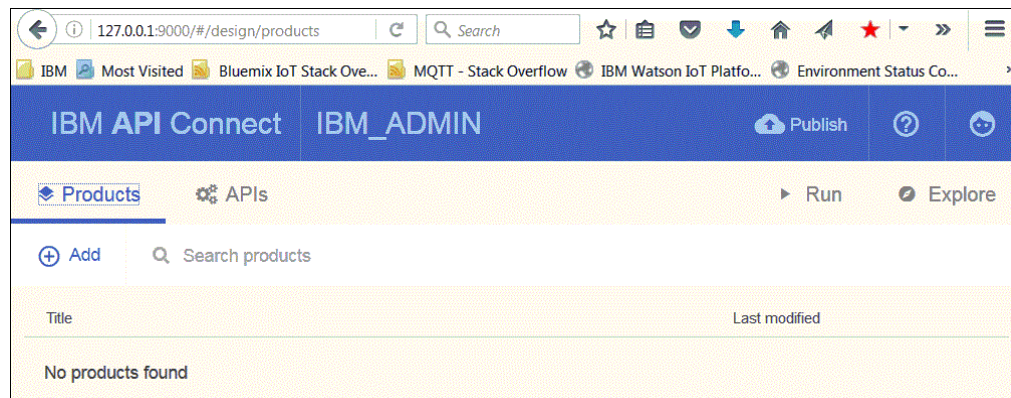


Figure 2-4 Adding an API product for a company

2.2.4 Products in the API Designer

In IBM API Connect, Plans and APIs are grouped 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. The API Manager is used to manage the lifecycle of the Product. Although Plans belong to only one Product, they can possess different APIs to other Plans within the same Product. They also can share APIs with Plans from any Product.

The relation between Products, Plans, and APIs is shown in Figure 2-5. Products provide a method by which a company can group APIs into a package that is intended for a particular use. Products also 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. Also, 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.

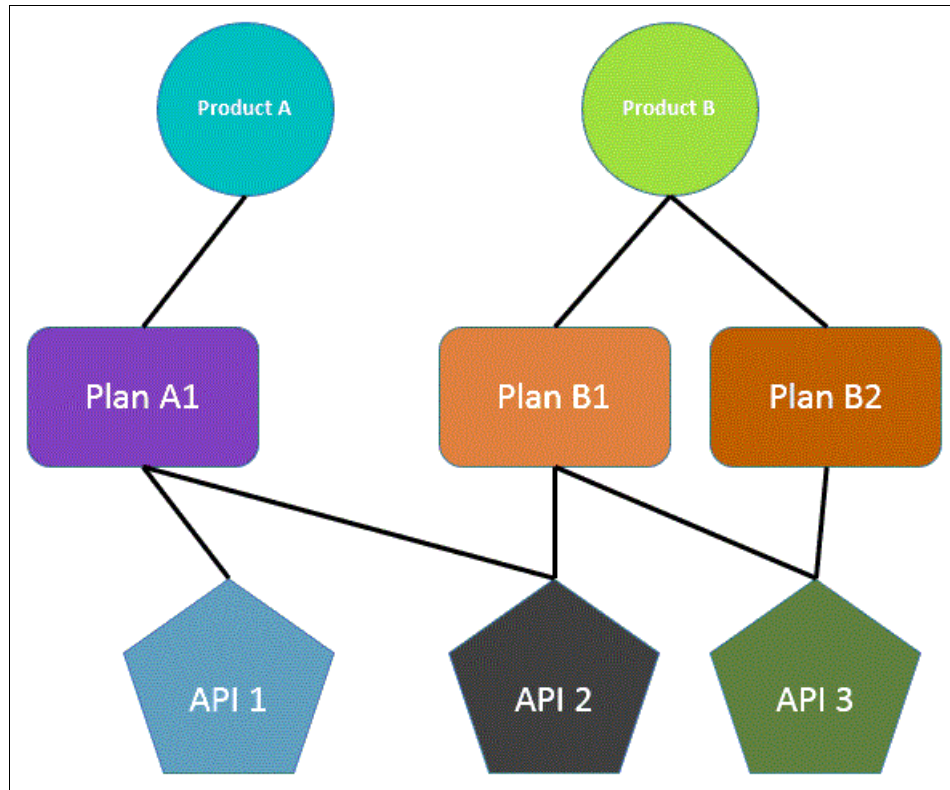


Figure 2-5 Products plans and APIs in API Designer

2.3 Sample application scenario

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

Bob browses through the list and inventory and interested in “Punched-card tabulating machines”. He taps the item and the Mobile application shows the detail about this item. The IBM Redbooks Mobile App A pulls the Item or 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 who are using this website share his interests).

Next, we review the process that is used to build the LoopBack Inventory application that was developed by E-Commerce Company B and the SocialReviews application that was developed by social sharing provider Company C.

The Inventory and Social Reviews applications and APIs are shown in Figure 2-6.

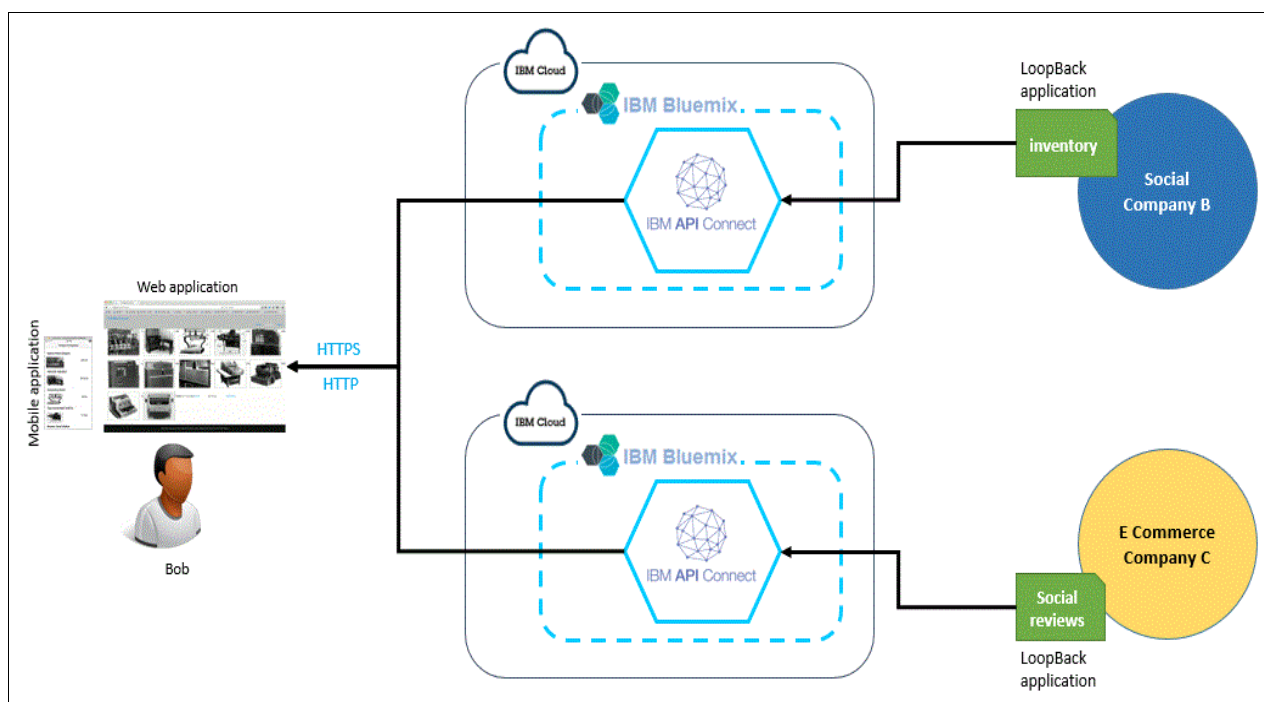


Figure 2-6 LoopBack application that is developed by Company B and Company C

2.4 Develop LoopBack applications and APIs

This section describes the step-by-step process that is used to develop the APIs 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 includes the following LoopBack models:

- ▶ Item: This model provides functionality to work with the individual items
- ▶ Container: This model provides functionality to work with images of individual items

Note: You can skip the walk-through of creating the LoopBack applications and can clone the inventory and socialreviews application from GitHub by using the following git repository and can continue with 2.5, “Starting and testing the LoopBack applications locally” on page 38:

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

For more information about cloning the Github project repository, see 1.3.3, “Downloading the applications from GitHub” on page 10.

Use the `npm install` command from CLI to install the dependencies of the LoopBack project. This command is run from the CLI where the LoopBack project package.json file is found.

Creating a LoopBack project

Complete the following steps to create a LoopBack project:

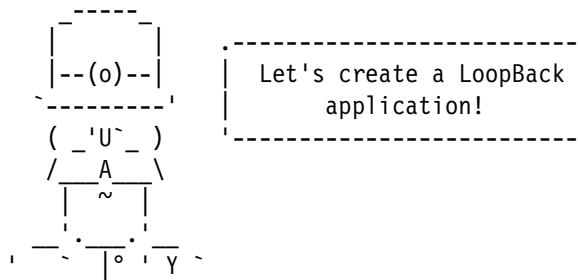
1. Create a directory with the name APIConnect from the command line.
2. Change the directory to APIConnect.
3. Create the LoopBack application by running the **api** command, as shown in the following example:

```
apic loopback -n <applicationname>
```

Example 2-3 shows steps to create the inventory application from Company B.

Example 2-3 Inventory LoopBack application from Company B

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



```
? 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 data sources)
```

```
? 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
```

4. Enter both the application and directory names as inventory.
5. Choose an empty server application.
6. Choose the LoopBack version 3.x.
7. From command line, change the work directory to inventory.

Creating data source and models

A LoopBack model represents data in backend systems, such as databases, and by default includes Node and REST APIs. Functionality for validation rules and business logic can be added to the LoopBack models. Inventory application has two models (item and container). LoopBack models can manipulate data via the data source object.

A data source enables a model to access and modify data in backend system, such as a relational database. Data sources encapsulate business logic to exchange data between models and various backend systems, such as relational databases, REST APIs, SOAP web services, and storage services. Data sources often provide create, retrieve, update, and delete (CRUD) functions.

Item model

Item model represents data that is in backend for storing the inventory of computer items from Company B. The item model also creates the REST API for working with item data. The properties of item model are listed in Table 2-1.

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 |
| 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. Complete the following steps to configure the in-memory data source for item model:

1. On the command line, change the directory to APIConnect/inventory.
2. Create a directory with the name db.
3. Create a file inside the db directory with name memdb.json.

4. Return to the directory inventory.
5. Run command that is shown in Example 2-4 to create a data source that is called memdb.

Example 2-4 Create memdb data source for item model

```
C:\APIConnect\inventory>apic create --type data source
? 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
```

6. Enter the data source name as memdb and select the **in-memory** data source.
7. Leave the window.localStorage field blank and press Enter.
8. Provide full path to file as db/memdb.json.

This process creates the in-memory data source for item model. Complete the following steps to create the item model by using the user interface of the API Connect Toolkit:

1. To start the Designer toolkit, use the command that is shown in Example 2-5.

Example 2-5 Start API Connect Toolkit

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

The Designer toolkit application opens in browser.

2. Log in by using your IBM Bluemix credentials, as described in 2.2.3, “IBM API Connect Developer Toolkit API Designer” on page 17.

Note: The API Connect Toolkit Designer tool also can be accessed by using the following URL:

<http://127.0.0.1:9000/#/design/apis>

3. Click the **Models** tab.
4. Click **Add** to create a model.
5. Create a LoopBack model that features the name `item` and then, click **New**, as shown in Figure 2-7.



The screenshot shows a web-based dialog titled "New LoopBack Model". It has a single text input field labeled "Name" which contains the word "item". Below the input field, there are two buttons: "Cancel" and "New". The "New" button is highlighted with a red rectangular border. The "Name" input field is also highlighted with a red rectangular border.

Figure 2-7 Creating a model name item

- Renew the properties in this model that reference the properties that are listed in Table 2-1 on page 22. Figure 2-8 shows the properties that were created in the item model.

| Properties | | | | | | |
|-------------------------------------|---------------|--------------------------|--------|-------------------------------------|--------------------------|------------------------|
| Required | Property Name | Is Array | Type | ID | Index | Description (optional) |
| <input checked="" type="checkbox"/> | id | <input type="checkbox"/> | number | <input checked="" type="checkbox"/> | <input type="checkbox"/> | Item ID |
| <input checked="" type="checkbox"/> | name | <input type="checkbox"/> | string | <input type="checkbox"/> | <input type="checkbox"/> | Item name |
| <input checked="" type="checkbox"/> | description | <input type="checkbox"/> | string | <input type="checkbox"/> | <input type="checkbox"/> | Item description |
| <input checked="" type="checkbox"/> | price | <input type="checkbox"/> | number | <input type="checkbox"/> | <input type="checkbox"/> | Item price |
| <input type="checkbox"/> | rating | <input type="checkbox"/> | number | <input type="checkbox"/> | <input type="checkbox"/> | Item rating |
| <input checked="" type="checkbox"/> | img_alt | <input type="checkbox"/> | string | <input type="checkbox"/> | <input type="checkbox"/> | Item image title |
| <input checked="" type="checkbox"/> | img | <input type="checkbox"/> | string | <input type="checkbox"/> | <input type="checkbox"/> | Item image location |

Figure 2-8 Properties of item model

- In the item model, set the data source to use the in-memory data source memdb, as shown in Figure 2-9.

IBM API Connect

inventory

[← All Models](#)

Name

item

Plural

Base Model

PersistedModel

Data Source

memdb

☒ Public

☐ Strict

Figure 2-9 Setting the item model to use the in-memory data source

- Click **Save** (top-right corner) to save the changes to the item model.

Container model

The container model is used for storing item image files locally and then making them available through the REST API. This model uses the storage connector and the loopback-component-storage module. For more information about using the storage connector, see the following website:

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

Complete the following steps to implement the storage connector data source and configure the container model:

- Change the directory to APIConnect/inventory/server.
- Create a directory that is called "storage".

3. Change the directory to storage (which was created in the previous step).
4. Create directory items under storage.
5. Copy all of 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. For more information about downloading the item images, see Appendix B, “Additional material” on page 127.

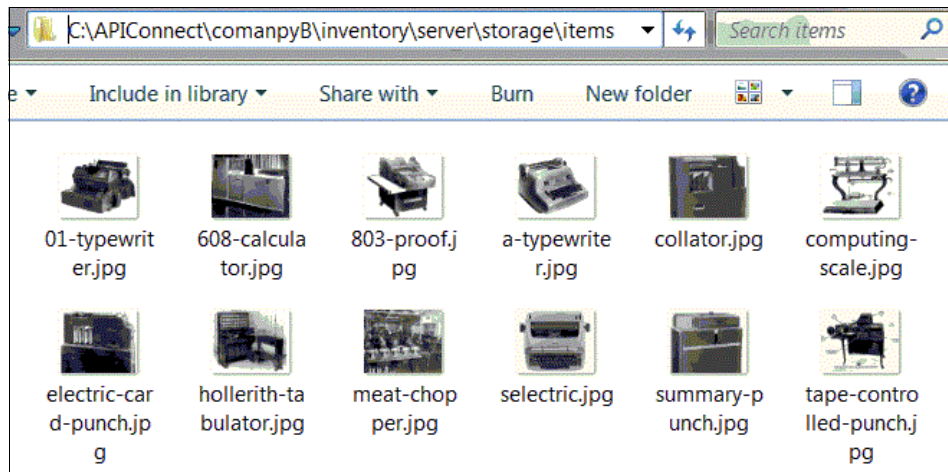


Figure 2-10 Item images in storage for container model

6. Change the directory to APIConnect/inventory.
7. Create data source storage by using the command that is shown in Example 2-6.

Example 2-6 Creating storage data source for container model

```
c:\APIConnect\inventory>apic create --type data source
? 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 data source JSON file to update the provider and root properties of the data source. Edit the datasources.json file in the server folder as shown in Example 2-7.

Example 2-7 Update data sources.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"
  }
}

```

-
9. Create the container model by using the command that is shown in Example 2-8. Use the connector as loopback-component-storage. The use of the **npm** command installs this module in node_modules.

Example 2-8 Creating 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

Configuring the inventory product

Complete the following steps to configure the inventory product through the API Designer toolkit:

1. Click the **Products** tab in the designer (the inventory product is shown in Figure 2-11). Configure the plans and provider details of inventory product by clicking the product.

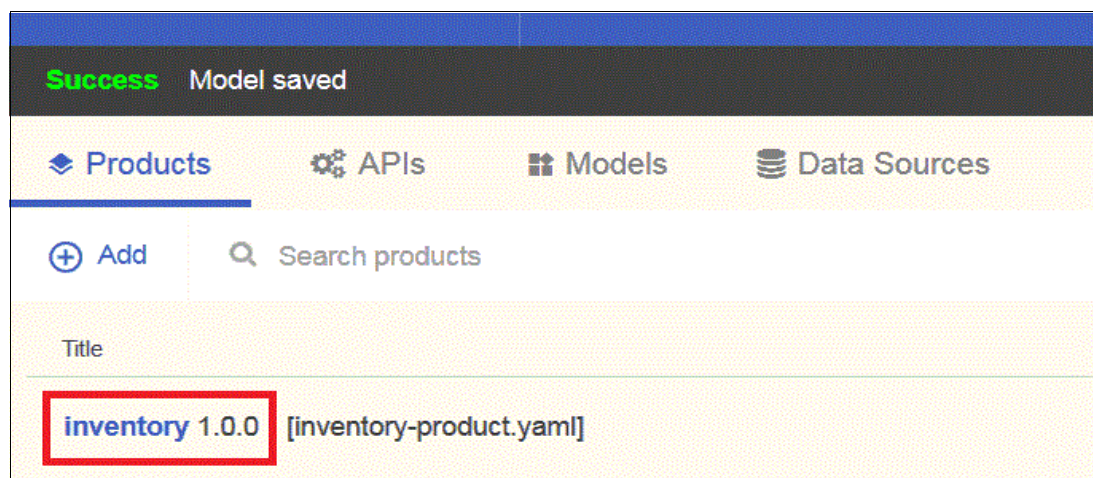


Figure 2-11 Inventory product

2. Configure the inventory product by using the property values that are 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 Company B |
| | | |
| 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 | Company B terms of service |
| | | |
| Visibility | Visible to | Public |
| | Subscribable by | Authenticated users |
| | | |

| Inventory product design details | | |
|----------------------------------|-------------|---------------------------------------|
| API | | inventory 1.0.0 |
| | | |
| Plans | Silver-Plan | 1000 invokes/hr (Hard Limit Enforced) |
| | Gold-Plan | unlimited access |

3. Save the product configuration after the changes are completed.

Configuring inventory APIs

Inventory application has two models: Item and container. Both models make available the related APIs under the inventory API. We use the API that is listed in Table 2-3 in the application. Other default APIs for these models can be deleted.

Table 2-3 Item and container model APIs

| Model | Path | Operation | Description |
|-----------|---|-----------|---|
| Item | /items | POST | Create an 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 |

Complete the following steps:

1. Browse to the inventory API by clicking **inventory 1.0.0**.
2. Disable `clientSecretHeader` in the Security section, as shown in Figure 2-12.

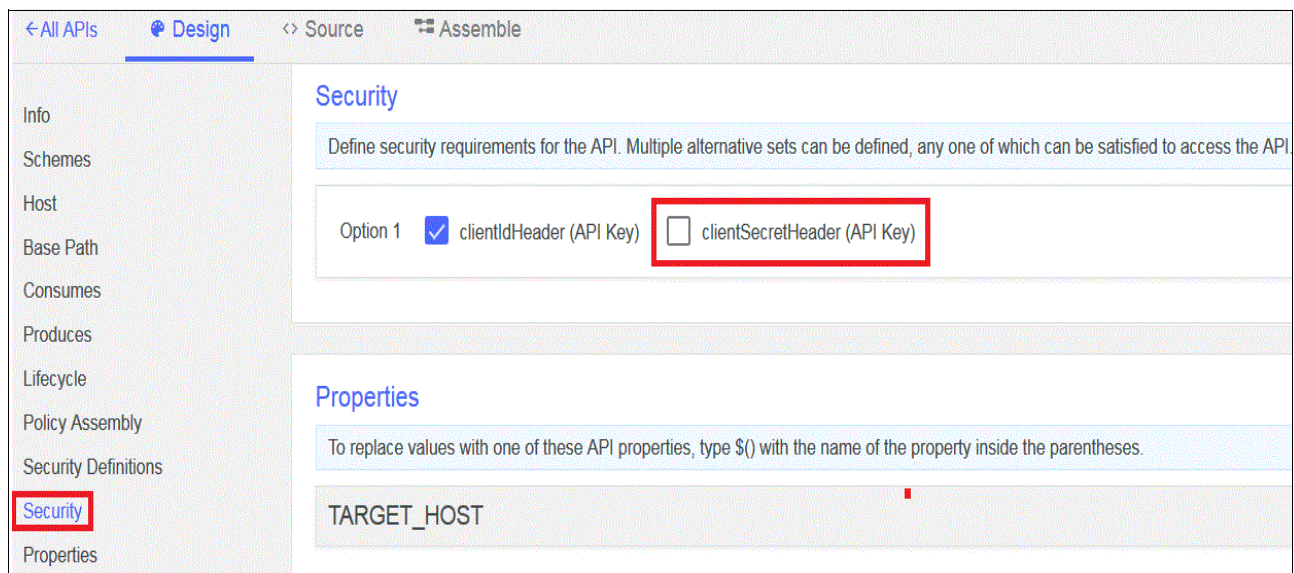


Figure 2-12 Disable `clientSecretHeader` in inventory API

3. Create a property that is named TARGET_HOST, as shown in Figure 2-13. This property indicates to the assembly where the application is deployed. This property must be changed before the deployment to Bluemix.

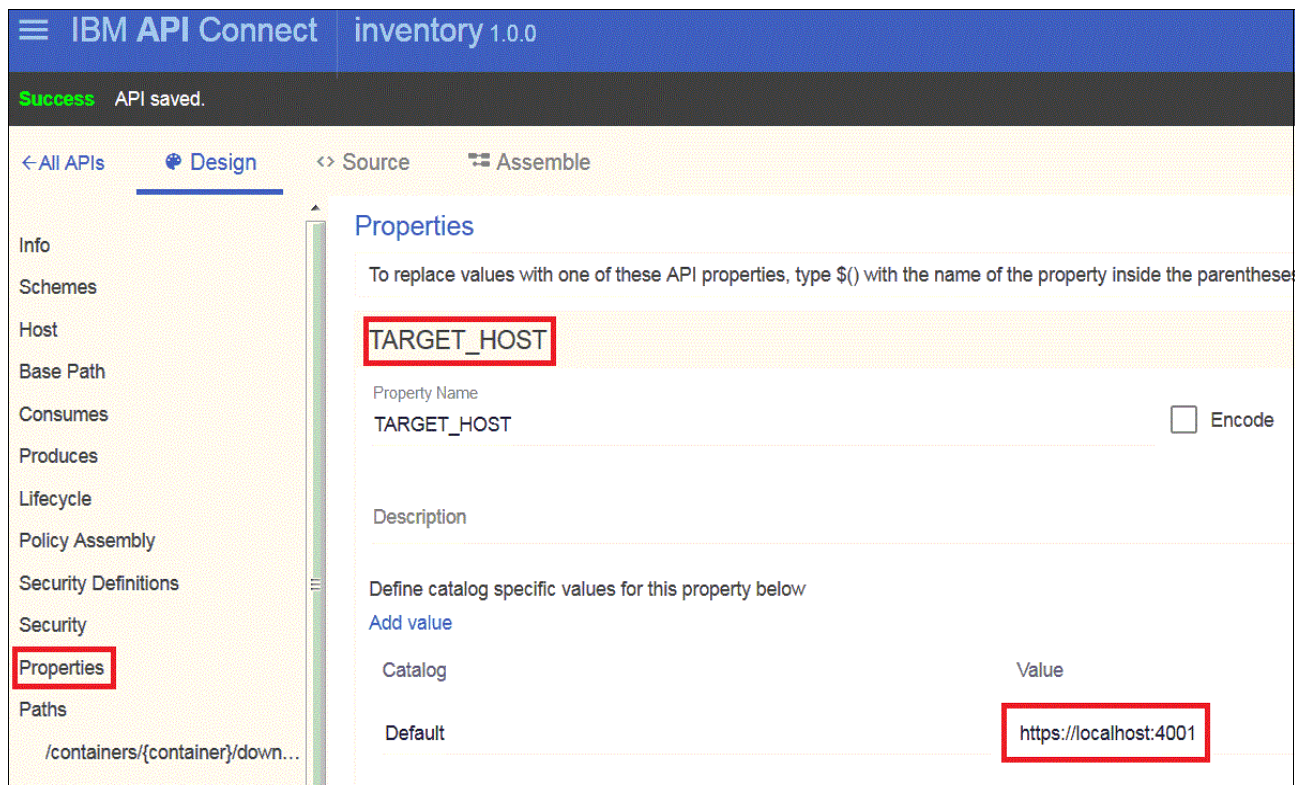


Figure 2-13 TARGET_HOST property for API assembly

The process to create the inventory LoopBack application from Company B is now complete.

4. Stop the APIConnect development toolkit from the command line by pressing Ctrl+C.

2.4.2 Social Reviews LoopBack application from Company C

Company C provides socialreviews product and features socialreviews APIs. The socialreviews API includes one LoopBack model that is named “review”. The review model provides functionality to work with individual reviews for an item.

Creating a LoopBack project

Complete the following steps to create a LoopBack project:

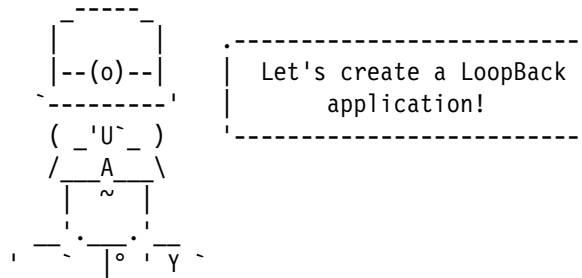
1. Change the directory to APIConnect.
2. Create the LoopBack application by using the following API command:

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

The steps that are used to create the socialreviews application from Company C are shown in Example 2-9 on page 30.

Example 2-9 SocialReviews LoopBack application from Company C

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



```
? 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 data sources)
? 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
```

3. Enter both the application and directory names as `socialreviews`.
4. Choose an empty server application.
5. Choose the LoopBack version 3.x.
6. Change the directory to `socialreviews` after the LoopBack application is created.

Creating data source and models

A LoopBack model represents data in backend systems, such as databases, and by default includes Node and REST APIs. Other functionality for validation rules and business logic can be added to the LoopBack models. The `socialreviews` application includes one model that is named “review”. LoopBack models can manipulate data by using the data source object. Attaching a data source to a model adds instance methods and static methods to the model.

Review model

The review model represents data in backend for storing the reviews from Company C. The review model also creates the REST API for working with reviews data. The properties of the review model are listed in Table 2-4.

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 an in-memory database. Data for the in-memory database is persisted in flat file on the local storage of the application. Complete the following steps to configure the in-memory data source for the review model:

1. On the command line, change the directory to `APIConnect/socialreviews`.
2. Create a directory that is named `db`.
3. Create a file inside the `db` directory that is named `memdb.json`.
4. Return to the `APIConnect/socialreviews` directory.
5. Run the command that is shown in Example 2-10 to create a data source that is named `memdb`.

Example 2-10 Creating `memdb` data source for review model

```
C:\APIConnect\socialreviews>apic create --type data source
? 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

6. Name the data source memdb and select the **in-memory** data source.
7. Leave the window.location field empty and press Enter.
8. Provide the full path to the file as db/memdb.json.

The in-memory data source for the review model is now created. Complete the following steps to create the review model by using the user interface of API Connect Toolkit:

1. Use the command that is shown in Example 2-11 to start the designer toolkit.

Example 2-11 Starting API Connect toolkit

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

The designer toolkit application opens in browser.

2. Log in by using the IBM Bluemix credentials as described in 2.2.3, “IBM API Connect Developer Toolkit API Designer” on page 17.

Note: The API Connect Toolkit designer tool also can be accessed by using the following URL:

<http://127.0.0.1:9000/#/design/apis>

3. Click the **Models** tab in the user interface.
4. Click **Add** to create a model.
5. Create a LoopBack model that is named review and then, click **New**.
6. Add new properties in this model that reference the review model properties that are listed in Table 2-4 on page 31. The properties that are created in review model are shown in Figure 2-14.







| Properties | | | | | | | |
|-------------------------------------|----------------|--------------------------|--------|--------------------------|--------------------------|------------------------|---|
| Required | Property Name | Is Array | Type | ID | Index | Description (optional) | |
| <input type="checkbox"/> | comment | <input type="checkbox"/> | string | <input type="checkbox"/> | <input type="checkbox"/> | reviewer comment |  |
| <input type="checkbox"/> | date | <input type="checkbox"/> | date | <input type="checkbox"/> | <input type="checkbox"/> | review date |  |
| <input checked="" type="checkbox"/> | itemId | <input type="checkbox"/> | number | <input type="checkbox"/> | <input type="checkbox"/> | Item ID |  |
| <input type="checkbox"/> | rating | <input type="checkbox"/> | number | <input type="checkbox"/> | <input type="checkbox"/> | reviewer item rating |  |
| <input type="checkbox"/> | reviewer_email | <input type="checkbox"/> | string | <input type="checkbox"/> | <input type="checkbox"/> | reviewers_email |  |
| <input type="checkbox"/> | reviewer_name | <input type="checkbox"/> | string | <input type="checkbox"/> | <input type="checkbox"/> | reviewers_name |  |

Figure 2-14 Properties of review model

7. In the review model, set the data source to use the in-memory data source memdb, as shown in Figure 2-15.

IBM API Connect | socialreviews

Success Model saved

[← All Models](#)

| | |
|--|----------------|
| Name | review |
| Plural | |
| Base Model | PersistedModel |
| Data Source | memdb |
| <input checked="" type="checkbox"/> Public | |

Figure 2-15 Setting the review model to use the in-memory data source

8. Click **Save** (top-right corner) to save the changes.

Configuring the socialreviews product

Complete the following steps to configure the socialreviews product through the API Designer toolkit:

1. Click the **Products** tab in the API Designer. (The socialreviews product is shown in Figure 2-16). Configure the plans and provider details of socialreviews product by clicking the product.

Success Product saved.

[Products](#) [APIs](#) [Models](#) [Data Sources](#)

[+ Add](#)

| | |
|----------------------------|------------------------------|
| Title | |
| socialreviews 1.0.0 | [socialreviews-product.yaml] |

Figure 2-16 Socialreviews product

2. Configure the socialreviews product by using the property values that are listed in Table 2-5.

Table 2-5 Socialreviews product design details

| Socialreviews product design details | | |
|--------------------------------------|------------------|--|
| Info | Title | socialreviews |
| | Name | socialreviews |
| | Version | 1.0.0 |
| | Description | socialreviews application from Company C |
| | | |
| Contact | Name | Admin |
| | Email | admin@companyc.com |
| | URL | http://www.companyc.com |
| | | |
| License | Name | Company C license |
| | URL | http://www.companyc.com |
| Terms of Service | Terms of Service | Company C 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 all of the changes are made.

Configuring socialreview APIs

The Socialreview application includes one LoopBack model that is named “review”. This application uses the APIs that are listed in Table 2-6. The other default APIs can be deleted.

Table 2-6 Review model APIs

| Model | Path | Operation | Description |
|--------|--------------------|-----------|---|
| review | /reviews | GET | Get all the reviews for an item by filter |
| review | /reviews | POST | Create a 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 the /reviews path are created by the designer tool. Complete the following steps to create the remaining three APIs:

1. Browse to the socialreviews API by clicking **socialreviews 1.0.0**.
2. Disable `clientSecretHeader` in the Security section, as shown in Figure 2-17.

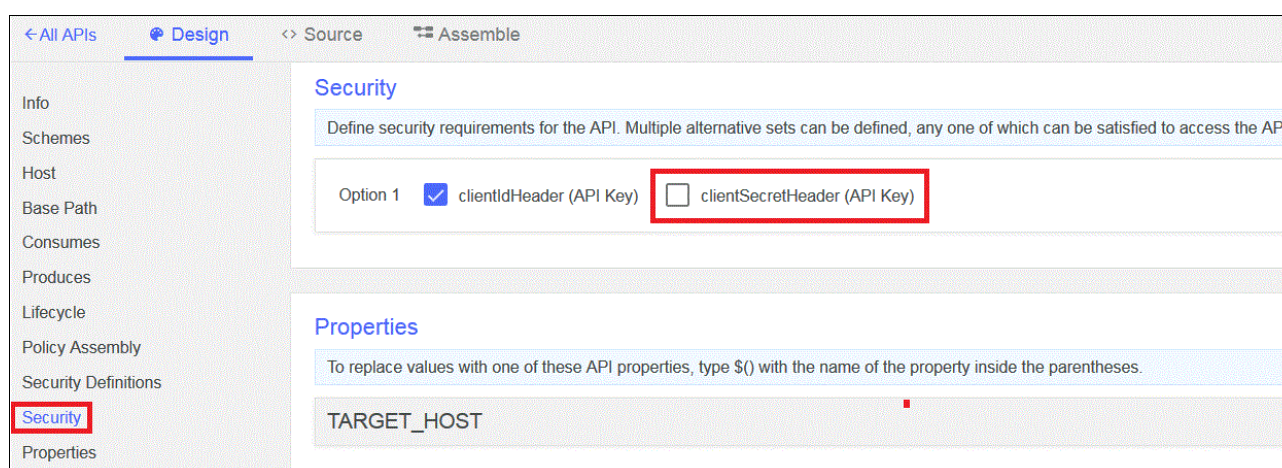


Figure 2-17 Disabling `clientSecretHeader` in `socialreviews` API

Creating the reviews average rating path

Complete the following steps to create the reviews average rating path:

1. In `socialreviews 1.0.0`, API click **Path** → **Add new Path**.
2. Enter the path name `/reviews/avgrating`.
3. Expand the default GET method that is created for this path.
4. Update the summary of GET method by entering: Find average rating of an item.
5. Update the operation ID of GET method with `review.avgrating`.
6. Click **Save**.
7. Define a remote hook to implement the logic to calculate the average rating.

Note: A *remote method* is a static method of a model that is made available over a custom REST endpoint. Use a remote method to perform operations that are not provided by LoopBack's standard model REST APIs. For more information, see this website:

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

A *remote hook* enables a user to run a function before or after a remote method is called by a client. For more information, see this website:

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

To add a remote hook for average rating, add the methods that are shown in Example 2-12 inside the `review.json` file that is inside the LoopBack application. The file is in `APIConnect/socialreviews/common/models/review.js`.

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

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

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

8. Save the review.js file

Creating the reviews counter path

Complete the following steps to create the reviews counter path:

1. In socialreviews 1.0.0 API, add new path by clicking **Path** → **Add New Path**.
2. Enter the path name /reviews/counter.
3. Expand the default GET method that is created for this path.
4. Update the summary of the GET method by entering: Find the count of reviews for an item.
5. Update the operation ID of GET method with review.counter.
6. Click **Save**.
7. Define a remote hook to implement the logic to calculate the average rating.

To add a remote hook for review counter, add the methods that are shown in Example 2-13 inside the review.js file of the 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.

Creating the reviews oauth path

This utility API is used to trigger the OAuth flow. For more information about this API, see Chapter 5, “Using the APIs” on page 79. Complete the following steps to create this API:

1. In socialreviews 1.0.0 API click, **Path** → **Add new Path**.
2. Enter the path name as /reviews/oauth.
3. Expand the default GET method that is created for this path.
4. Update the summary of the GET method with OAuth trigger API.
5. Update the operation ID of GET method with review.oauth.
6. Click **Save**.
7. A remote hook must be defined to implement a logic to send an empty response for the OAuth trigger. To add a remote hook for review OAuth, add the methods that are shown in Example 2-14 on page 38 inside the review.js file of LoopBack application.

Example 2-14 Remote method for OAuth trigger

```
Review.oauthtrigger = function(cb) {  
    cb(null, '');  
};  
  
Review.remoteMethod('oauthtrigger',  
    {  
        http: {path: '/oauth', verb: 'get'},  
    }  
);
```

8. Save the review.js file.
9. Create a property that is named TARGET_HOST. This property indicates to the assembly where the application is deployed. This property must be changed before the deployment to Bluemix.
10. The LoopBack application is now created for Company B and Company C. Stop the API Connect Developer Toolkit by pressing Ctrl+C.

2.5 Starting and testing the LoopBack applications locally

In this section, we describe how to start and test the LoopBack applications locally.

2.5.1 Starting the inventory application

To start the inventory application from the command line, browse to the inventory directory. Use the **apic start** command that is shown in Example 2-15 to start the inventory application for local testing. After the application is started, two node processes are created, as shown in Example 2-15.

Example 2-15 Starting the inventory application

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

2.5.2 Testing the inventory application

The tests that are described in this section are quick tests that are used to confirm if the APIs are reachable. For more information about testing APIs, see Chapter 5, “Using the APIs” on page 79.

Testing the API to list all items in inventory

To test the API to list all items in inventory, paste the following URL in a browser window, as shown in Figure 2-18 on page 39:

GET <http://localhost:4001/api/items>

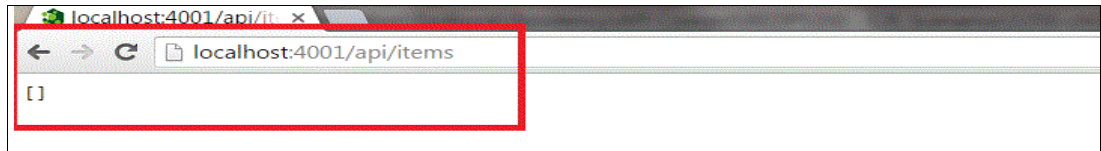


Figure 2-18 Test the GET Items inventory API

This test confirms that the inventory application is running. It also generates the empty list of items because the memory database is empty, as shown in Figure 2-18.

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

To test the API to get the item image file through the container API, paste the following URL in a browser:

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

This test confirms that the inventory application is running. It also renders the item's image file from the local storage, as shown in Figure 2-19.

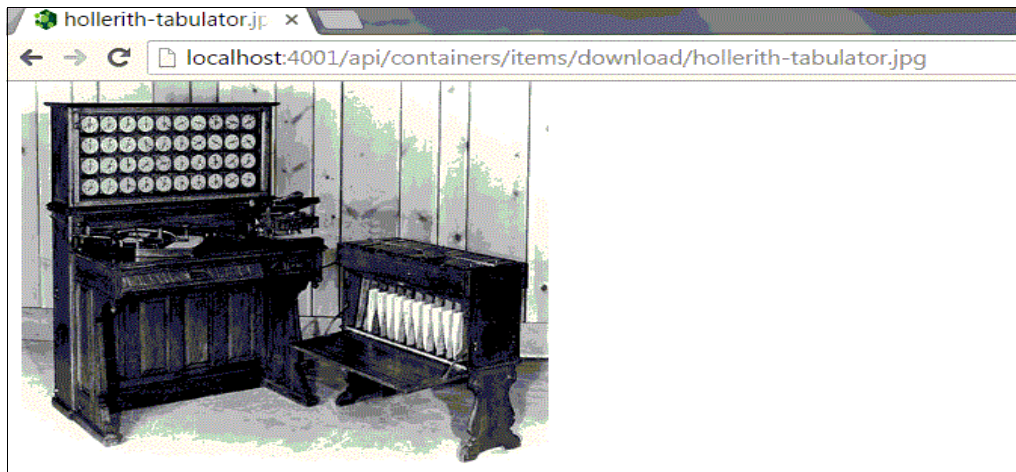


Figure 2-19 Test the download image container API

2.5.3 Stopping the inventory application

Use the **apic stop** command to stop the inventory application, as shown in Example 2-16.

Example 2-16 Stopping 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 Starting the socialreviews application

To start the socialreviews application from the IBM API Connect designer, start the socialreview designer by 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.



Figure 2-20 Starting the socialreviews application

2.5.5 Testing the socialreviews application

The tests that are described in this section are quick tests that are used to confirm if the APIs are reachable. For more information about testing APIs, see Chapter 5, “Using the APIs” on page 79.

Testing the API for the count of reviews for an item

Paste the following URL in a browser:

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

A JSON message is returned with the count of reviews for ItemId 13412. The count is zero because there are no reviews in the in-memory database, as shown in Example 2-17.

Example 2-17 Review count for an item

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

Testing the API for the average rating for an item

To test the API for the average rating for an item, paste the following URL in a browser:

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

A JSON message is returned with the average rating for itemId 13412. The average rating for this item is zero because there are no reviews in the in-memory database, as shown in Example 2-18.

Example 2-18 Average rating for an item

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

2.5.6 Stopping the socialreviews application

To stop the socialreviews application from the IBM API Connect designer, click the **Stop** button, as shown in Figure 2-21 on page 41. The loopback application and micro gateway node processes are stopped.

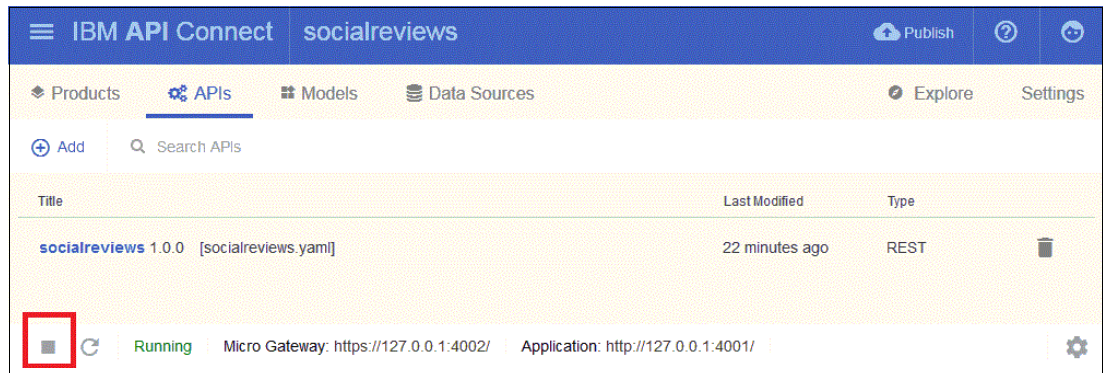


Figure 2-21 Stop the socialreviews application

2.6 Testing the APIs

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

2.6.1 Testing the inventory application API

The inventory application features six REST APIs that are made available through the LoopBack application. These APIs are listed in Table 2-3 on page 28. Use the **apic start** command from the inventory directory to start the application locally.

Creating items for the inventory application

Create 12 items by using HTTP REST tool. To create items, use the following REST URL:

POST <http://localhost:4001/api/items>

The use of the HTTP REST tools to start the POST call for the /items API is shown in Figure 2-22.



Figure 2-22 Creating item 13401

Note: A pre-populated memdb.json file is available for download from the IBM Redbooks FTP server. For more information, see Appendix B, “Additional material” on page 127.

You can download the `memdb.json` file from IBM Redbooks FTP server and replace the existing `memdb.json` with the downloaded file. Restart the LoopBack application after the new `memdb.json` file is copied. After all of the items are created, 12 items are in the in-memory database.

Getting all items for the inventory application

You can acquire the 12 items that were created by using the `GET /items` API. Use the following URL to return all of the items in the inventory LoopBack application:

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

Getting an individual item for the inventory application

You can GET an individual item by using the `GET /items/{id}` API. The following URL returns the details of item 13401, as shown in Figure 2-23:

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

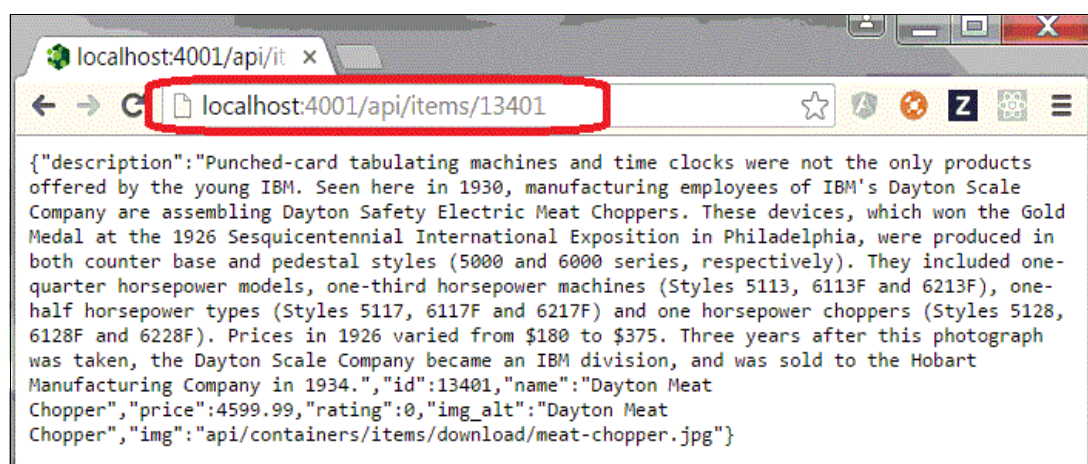


Figure 2-23 Getting details for Item 13401 from inventory application

Updating an item

You can update elements of an individual item by using the `PUT /items/{id}` API. The payload consists of a JSON element to be updated. Example 2-19 shows how to update the rating for item 13401.

Example 2-19 Update the rating of item 13401

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

Note: Ensure that the HTTP request includes the content type header that is set to `application/json`, as shown in the following example:

`Content-type: application/json`

Deleting an item

An item can be deleted from the in-memory database by using the `DELETE /items/{id}` API. The following URL for a DELETE REST call deletes item 13401 from the in-memory database:

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

Accessing an image file by using the container API

To access an image file by using the container API, paste the following URL in a browser:

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

This test confirms that the inventory application is running. It also renders the item's image file from the local storage, as shown in Figure 2-19 on page 39. Use the **apic stop** command from the inventory directory to stop the application.

2.6.2 Testing the socialreviews application API

The socialreview application features five REST APIs that are made available through the LoopBack application. These APIs are listed in Table 2-6 on page 35. Use the **apic start** command from the socialreviews directory to start the application locally.

Creating a review for an item

Create reviews by using the HTTP REST tools. To create the reviews, use the following REST URL:

POST <http://localhost:4001/api/reviews>

Creating a review for an item by using the socialreview application is shown in Figure 2-24.

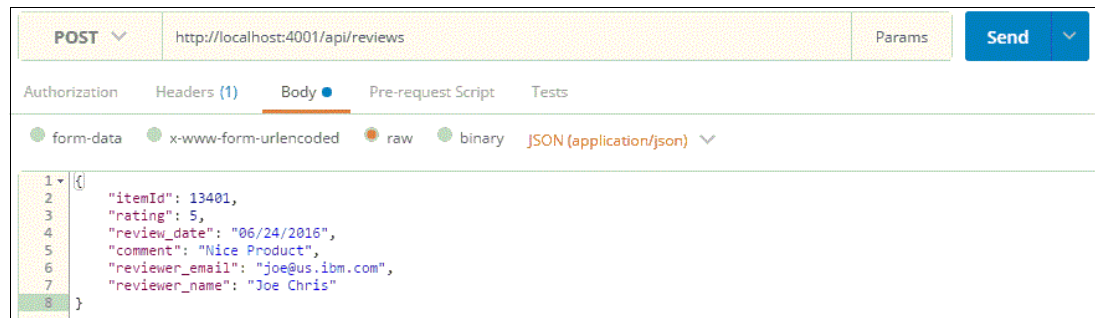


Figure 2-24 Create a review for item 13401

The payload that is used for creating reviews is shown in Example 2-20.

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 includes the content type header that is set to application/json, as shown in the following example:

Content-type: application/json

Getting reviews for an item

You can get reviews for an item by applying a filter on the GET /reviews REST API. The use of the following URL provides all of the reviews for item 13401:

GET [http://localhost:4001/api/reviews?filter={"where":{"itemId":13401}}](http://localhost:4001/api/reviews?filter={)

For more information about the LoopBack WHERE filter, see this website:

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

Getting reviews count for an item

Paste the following URL in a browser:

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

A JSON message is returned with the count of reviews for ItemId 13401. The count is one because there is only one review in the in-memory database, as shown in Example 2-21.

Example 2-21 Review count for item 13401

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

Getting the average rating for an item

To get the average rating for an item, paste the following URL in a browser:

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

A JSON message is returned with the average rating for ItemId 13401. The average rating for this item is five because there is only one review in the in-memory database, as shown in Example 2-22.

Example 2-22 Average rating for an item 13401

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

Use the **apic stop** command from the 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, which helps to test the API locally. When you publish during testing, you make your Product available through the Micro Gateway that is hosted on an IBM 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 an IBM API Connect Collective, as do LoopBack applications. The DataPower Gateway can run on either a physical or virtual DataPower appliance.

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 an IBM API Connect Collective and features a limited number of policies available to it.

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

2.7.1 Updating the inventory LoopBack application gateway

Complete the following steps to update the inventory LoopBack application gateway:

1. In the inventory API that uses the Designer Toolkit, browse the Assemble tab, as shown in Figure 2-25.

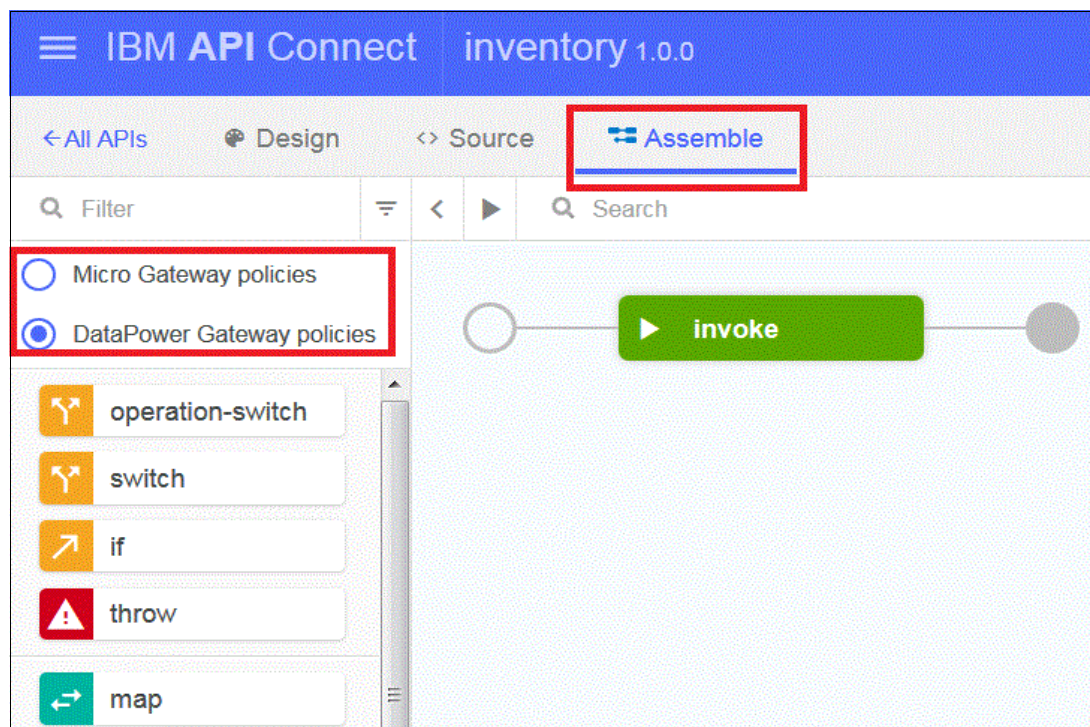


Figure 2-25 Updating inventory API to use DataPower Gateway

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

2.7.2 Updating the socialreviews LoopBack application gateway

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

1. In the socialreviews API that uses the Designer Toolkit, click the **Assemble** tab, as shown in Figure 2-26 on page 46.

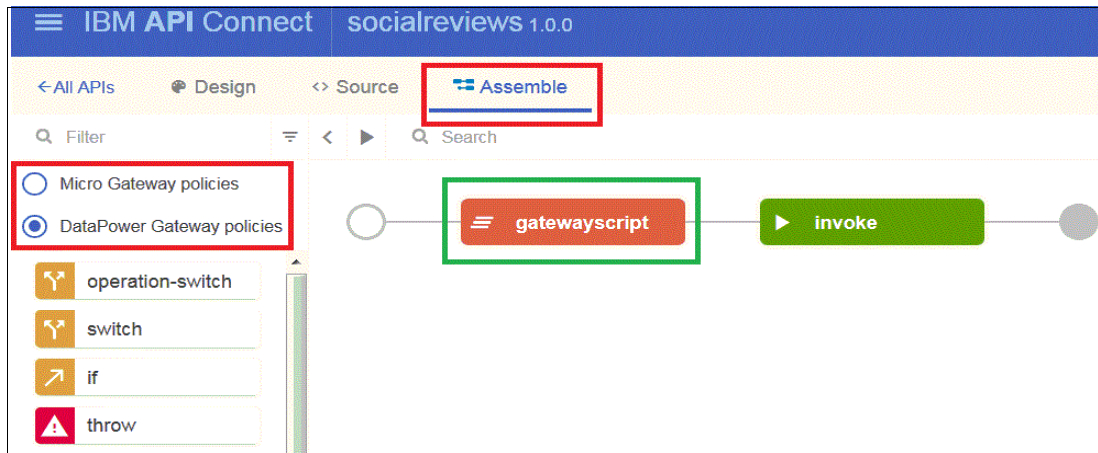


Figure 2-26 Updating socialreviewsAPI to use DataPower Gateway and Gateway script node

2. Drag the **gatewayscript** node from the node menu that is in front of the Assembly node.
3. Update the gateway script node with script that is shown in Example 2-23. This script helps to use the filters.

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 start node URL property as `$(TARGET_HOST)$(request.path)?$(filterQuery)` to use the TARGET_HOST property.
5. Save the changes.

2.8 Deploying LoopBack applications to IBM Bluemix

In this section, we describe the process of deploying the inventory and socialreviews LoopBack application in the Bluemix environment.

2.8.1 Configuring the Bluemix environment

For more information about setting up the Bluemix environment, see 1.3, “Environment setup” on page 4.

2.8.2 Deploying inventory application in Bluemix

Complete the following steps to deploy the inventory application by using command line:

1. From the command line, move to the directory of the inventory application.
2. Log in by using the **apic login** command, as shown in Example 2-24.

Example 2-24 Logging in to your apic account by 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 in to us.apiconnect.ibmcloud.com successfully
```

3. Set the publish configuration variable by 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 Publishing configuration for inventory application

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

Note: The command includes the following syntax:

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

You must replace the {bluemixOrg} and {bluemixSpace} variables with their own values.

This information also can be captured from by using the Catalog Identifier button that is in the catalog that was created in Bluemix.

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

Example 2-26 Deploying 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. After the application is deployed, it is available under the Cloud Foundry application in the Bluemix dashboard, as shown in Figure 2-27.

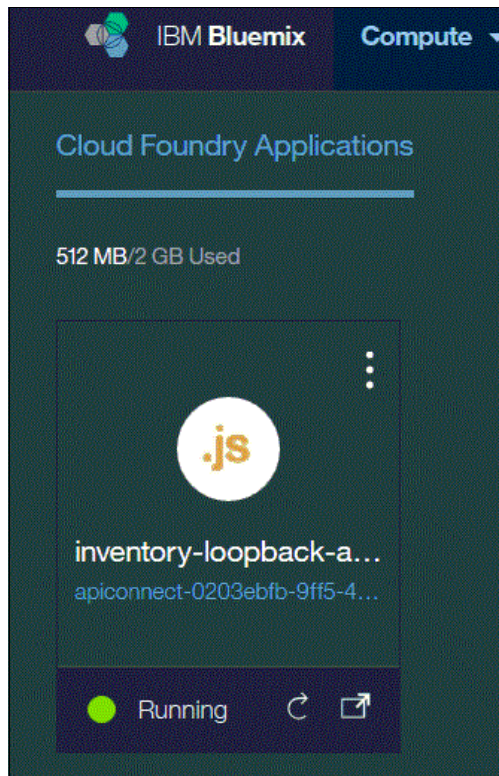


Figure 2-27 Cloud Foundry inventory application in Bluemix

6. The inventory application also is now shown in the API connect service in Bluemix, as shown in Figure 2-28.

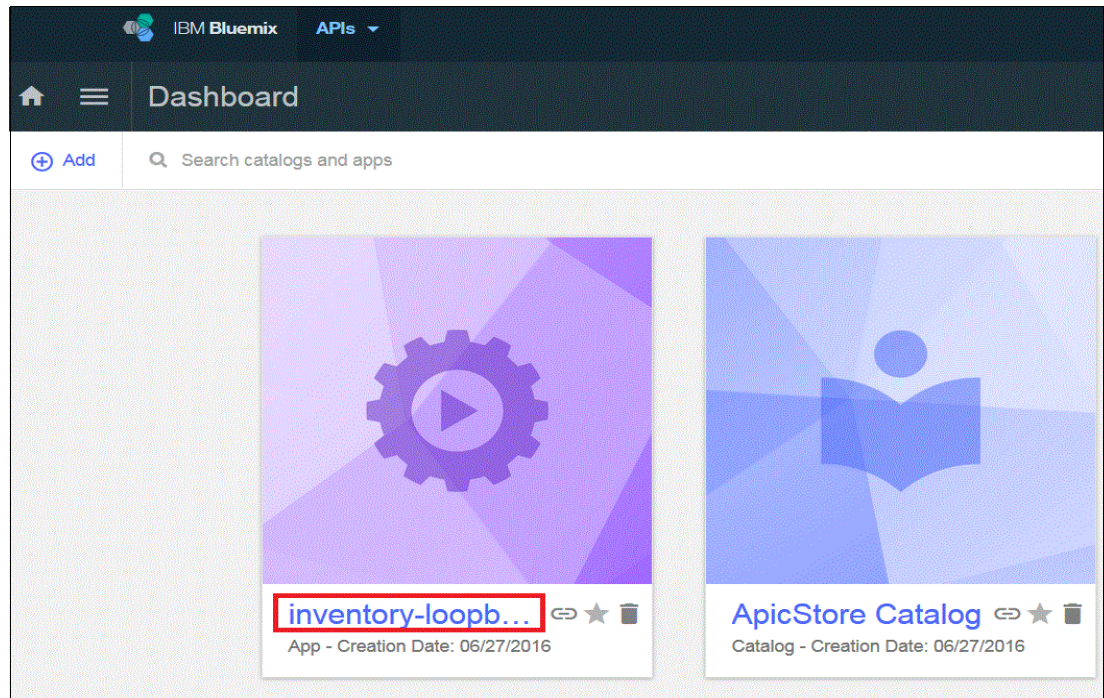


Figure 2-28 Inventory loopBack application available in IBM API Connect service

Note: At this time, the TARGET_HOST property that was created in the Inventory LoopBack application should be updated with the following Inventory application target URL from the publish command:

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

2.8.3 Deploying socialreviews application in Bluemix

Complete the following steps to deploy the Loopback application by using API Connect Designer UI:

1. Open the API Connect designer toolkit for socialreviews application by using the **apic edit** command.
2. Click **Publish** and then, click **Add and Manage Targets** to add the target for deployment, as shown in Figure 2-29 on page 50.

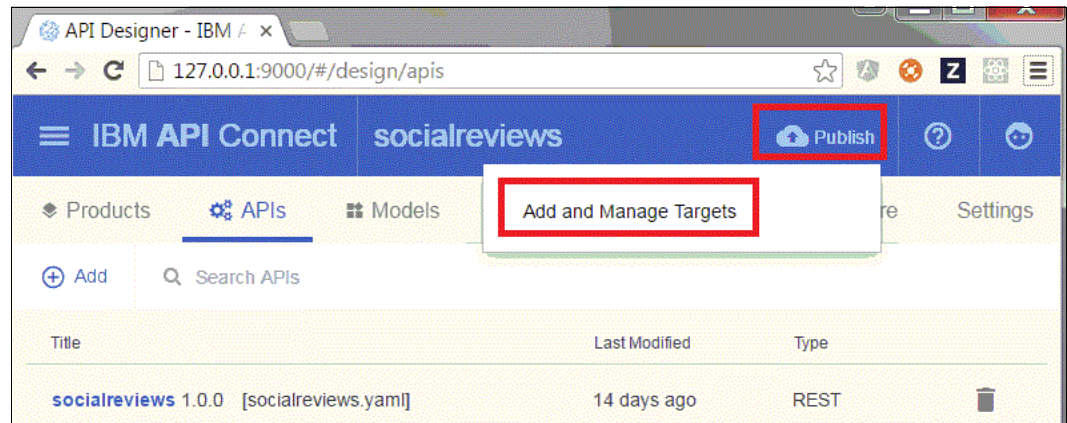


Figure 2-29 Adding and managing targets for deployment

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

 The screenshot shows a dialog box titled 'Select an organization and catalog'. It contains two dropdown menus. The first dropdown is labeled 'Region' and has 'US South' selected. The second dropdown is labeled 'Organization' and has 'rgupta1@us.ibm.com (redbook)' selected. Below these dropdowns is a search bar with the text 'Search'. Below the search bar is a list of options: 'None', 'Sandbox', and 'ApicStore Catalog'. The 'ApicStore Catalog' option is highlighted with a blue background and is also enclosed in a red box. At the bottom of the dialog box, there are three buttons: 'Back', 'Cancel', and 'Next'. The 'Next' button is highlighted with a red box.

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 51.

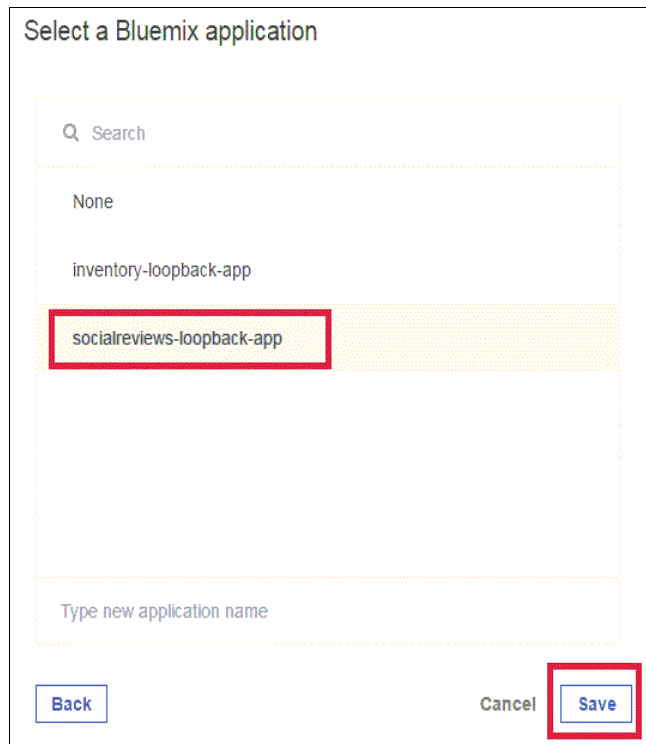


Figure 2-31 Save the publish configuration

6. To publish the socialreviews application, click **Publish** and select the publish configuration that was created in the previous steps, as shown in Figure 2-32.

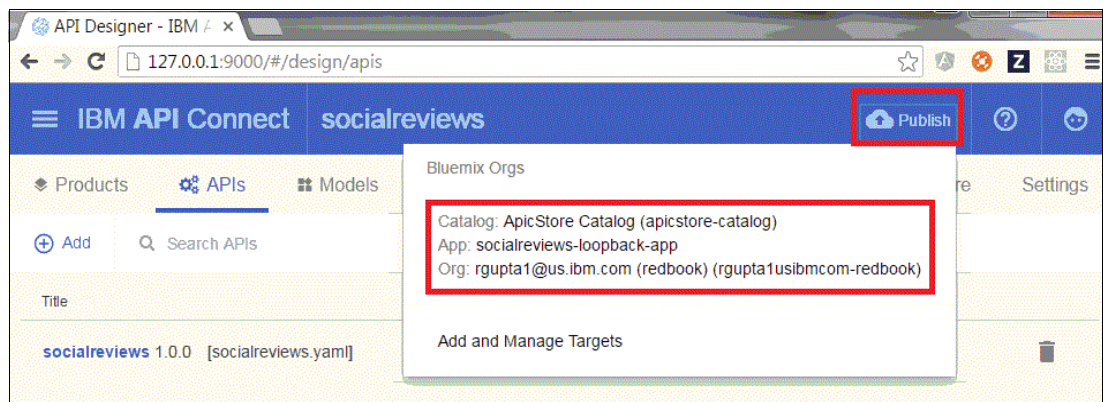


Figure 2-32 Select the publish target

7. Select **Publish**, as shown in Figure 2-33. Then, publish the socialreviews application to Bluemix.



Figure 2-33 Publishing the application to Bluemix

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

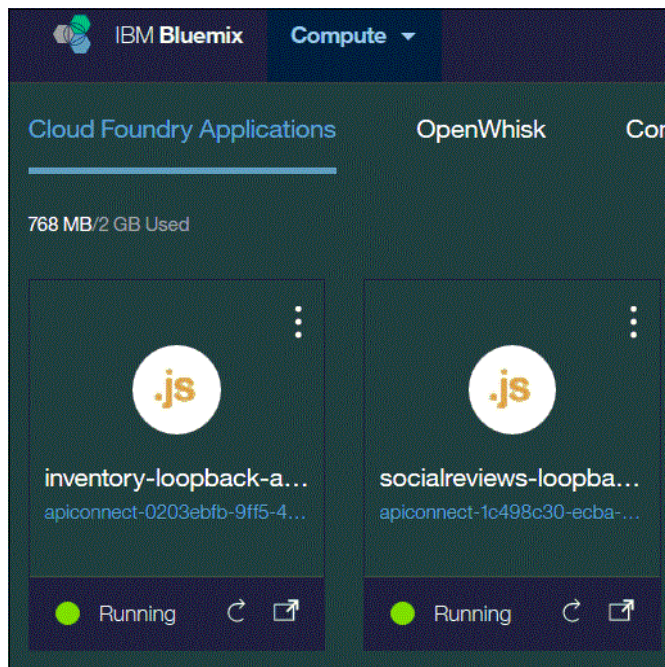


Figure 2-34 Socialreviews application deployed in Bluemix

Note: At this time, update the TARGET_HOST property that was created in socialreviews LoopBack application with the socialreviews application target URL from the deployed Bluemix Node.js application.

The URL for this application is available from the deployed Node.js application in Bluemix.

2.9 Summary

This chapter described the LoopBack applications. It also provided step-by-step processes to create inventory and socialreview LoopBack applications and local testing and application deployments in Bluemix.



Managing the APIs

In this chapter, we describe how to manage the APIs that were developed in Chapter 2, “Developing and deploying LoopBack applications and APIs” on page 13. IBM API Connect provides the following key API management capabilities:

- ▶ 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 learned about creating the catalog in Chapter 1, “Introduction to IBM API Connect and environment setup” on page 1. The following topics are described in this publication:

- ▶ Security in Chapter 4, “Securing the APIs” on page 65
- ▶ Developer Organization in Chapter 5, “Using the APIs” on page 79
- ▶ API use in Chapter 6, “Monitoring and analyzing the APIs” on page 101

In this chapter, we focus on publishing APIs so developers can write applications to use the inventory and social review functions.

This chapter includes the following topics:

- ▶ 3.1, “Publishing APIs by using API Designer user interface” on page 56
- ▶ 3.2, “Publishing APIs by using the apic command line tool” on page 61
- ▶ 3.3, “Summary” on page 63

3.1 Publishing APIs by using API Designer user interface

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

To make your API available for others to use, the following prerequisites must be met:

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

The second prerequisite is managed as part of the API Product lifecycle. You manage via the API product the different stages of the APIs you developed. The possible states of APIs in IBM API Connect are shown in Figure 3-1.

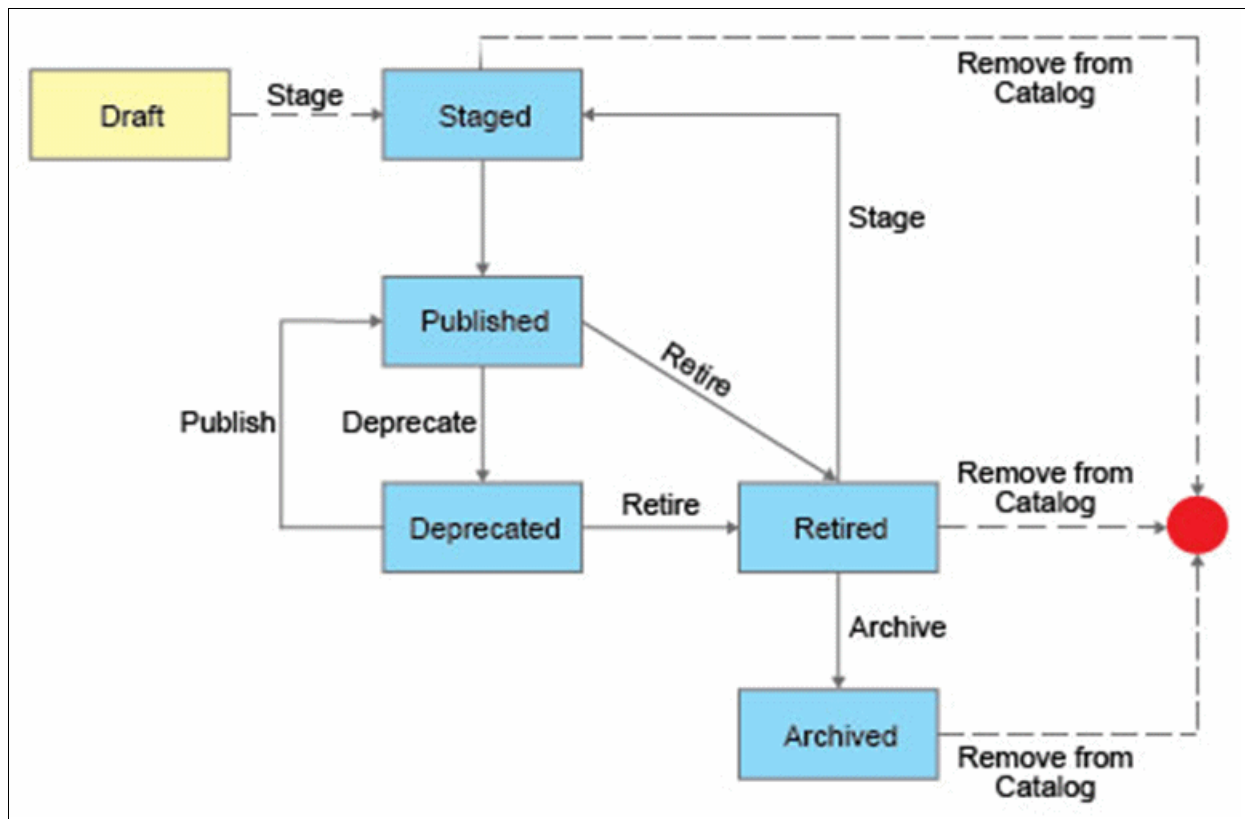


Figure 3-1 Life cycle state diagram for product versions

3.1.1 Understanding the API lifecycle

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

The following API lifecycle states are available:

► STAGED

In this state, the API is not yet live but is a click away from being published. It is not visible on the developer portal. Although this state can be bypassed and an API can be put directly into the PUBLISHED state, the STAGED state can be advantageous if you deploy the underlying application that provides the API and publishing the API at once.

Assume that the application does not deploy successfully. Then, the API cannot be used from the developer portal. Putting the API into the STAGED state first provides the opportunity to ensure that the service is available before the API is published, especially if multiple services and the corresponding APIs are rolled out at once.

► PUBLISHED

In this state, the API is live and available from the developer portal for use. New applications can subscribe to use the API.

► DEPRECATED

In this state, the API is still live and available for the applications that are using it, but new subscriptions are not allowed. This state indicates that the API is close to being retired and it is time for the applications that are using it to upgrade to a new version of the API or a different API. This state is useful if a large subscription base for this API exists unless you plan to eventually retire the API.

► RETIRED

In this state, the API is no longer available for any application to use. Any application that is using it loses access to the API. After the API is retired, it cannot be republished; instead, but it can be restaged and it receives a second life.

► ARCHIVED

After the API is archived, it cannot ever be used again. It can also be deleted, at which point the API's life ends.

Note: When describing the API lifecycle, *API* refers to a specific version of the API. From the API management perspective, Inventory 1.0.0 and Inventory 1.0.1 are two different APIs with their own lifecycles. For example, when Inventory 1.0.0 is in production, Inventory 1.0.1 can be staged. Also, 1.0.0 and 1.0.1 can be in production and available to developers to use simultaneously.

Complete the following steps to package an API 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. Inspect the directory structure of the application (under `redp5350-apiconnect-sample/inventory` folder), as shown in Figure 3-2 on page 58.

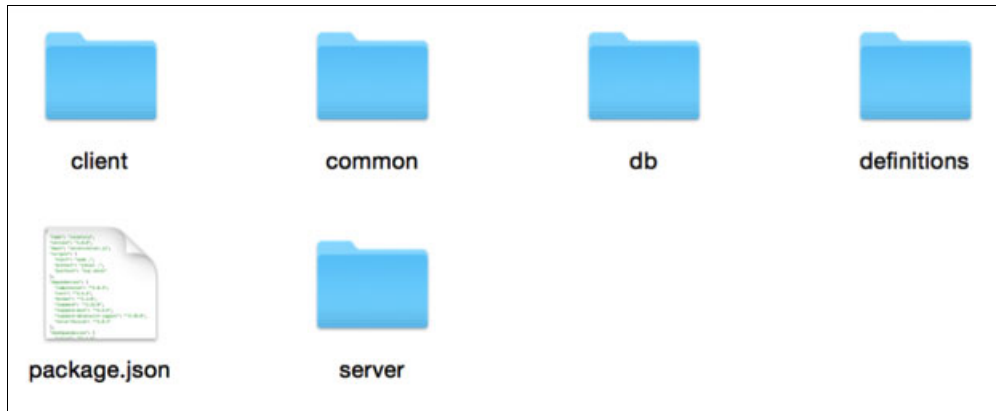


Figure 3-2 Directory structure of our application

2. Browse to the Definitions directory. Inside this directory are two .YAML files, as shown in Figure 3-3.

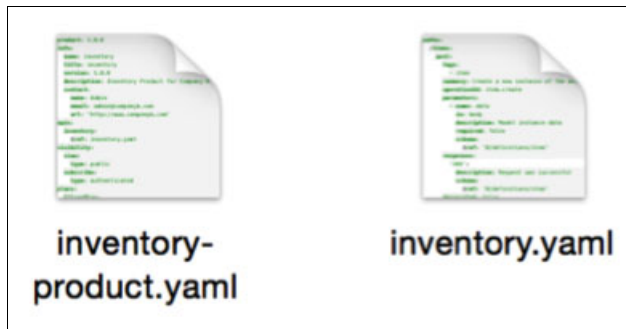


Figure 3-3 Two .yaml files

These files were generated by the API Connect Designer. The `inventory.yaml` file is the Swagger definition of the API that is provided by the Inventory application. The second file, `inventory-product.yaml`, is the product file, as shown in Figure 3-4.

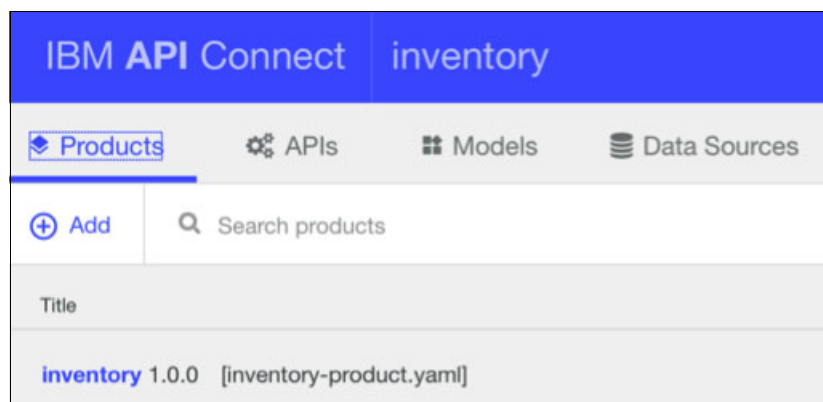


Figure 3-4 Products menu

3. You start the API Designer by browsing to the Inventory directory and running the **apic edit** command.
4. In API Designer console, sign in with your Bluemix ID.

5. In 2.8.3, “Deploying socialreviews application in Bluemix” on page 49, you configured the API Designer deployment target. Click **Publish** and select the **Bluemix target** that you created earlier. Figure 3-5 shows that target was created.

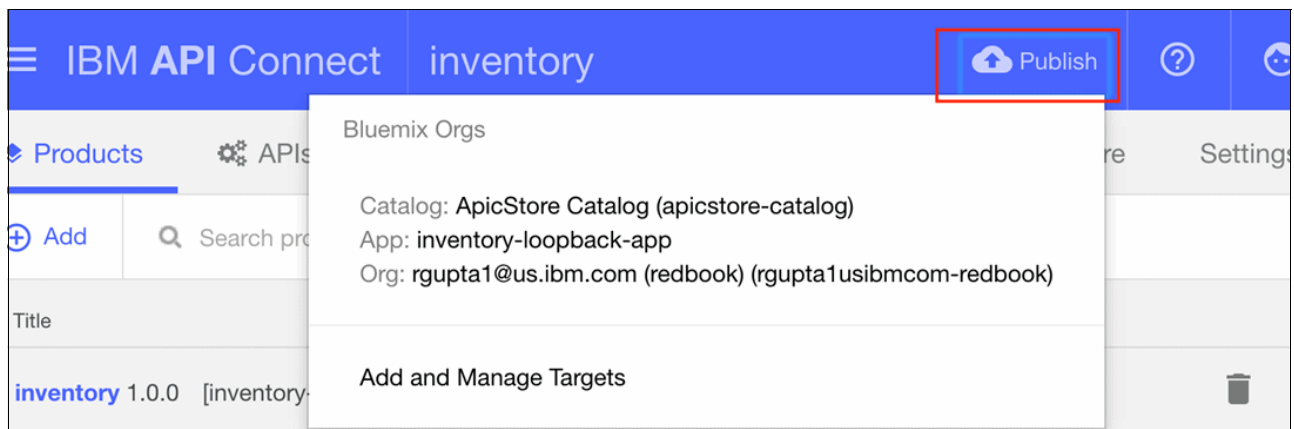


Figure 3-5 Publish target was created

6. In the Publish window, check select the **Stage and Publish products** option. Then, select the **Stage Only** and **Select specific products** options.
7. Select **Inventory** to publish, as shown in Figure 3-6. Click **Publish**.

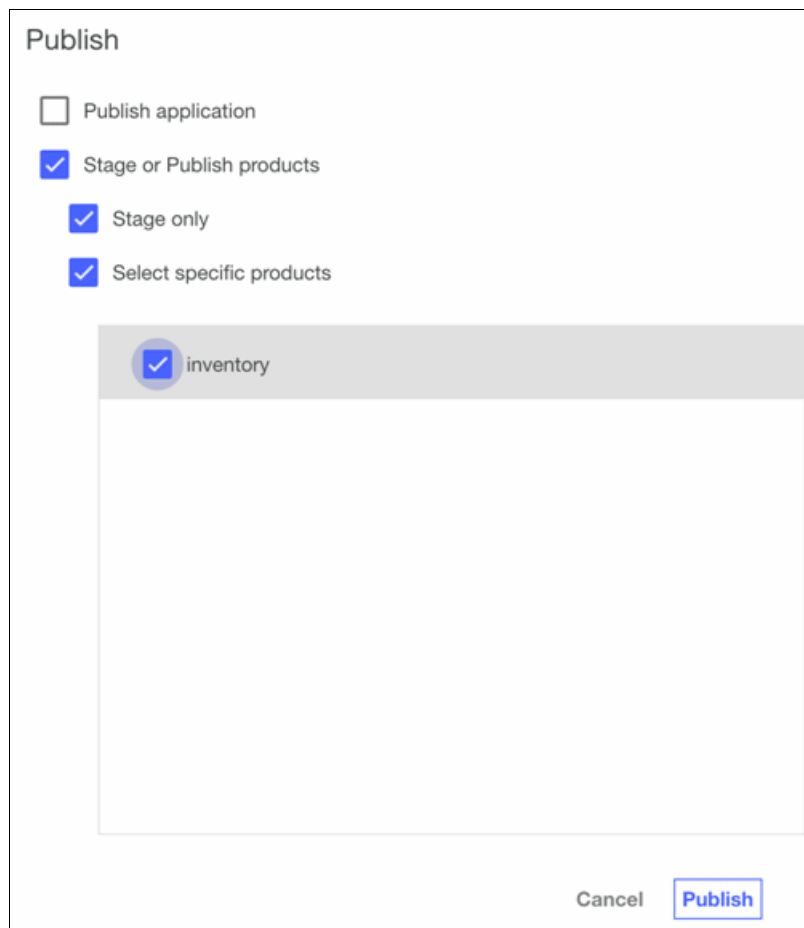


Figure 3-6 Selecting Publish

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

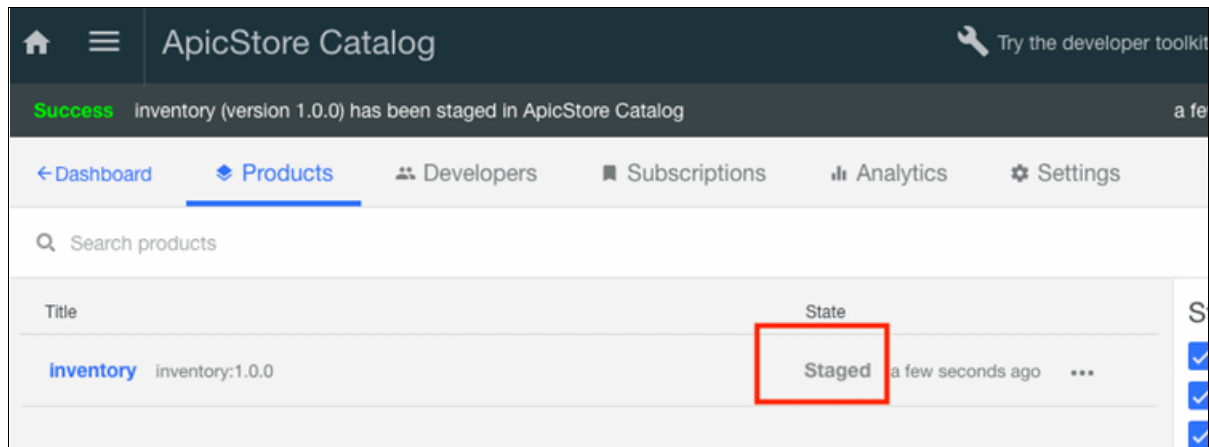


Figure 3-7 Staged state

9. Now, click the action “...” that is next to the State column to open the Action menu. Click **Publish**, as shown in Figure 3-8. The API product state is changed from Staged to Published.

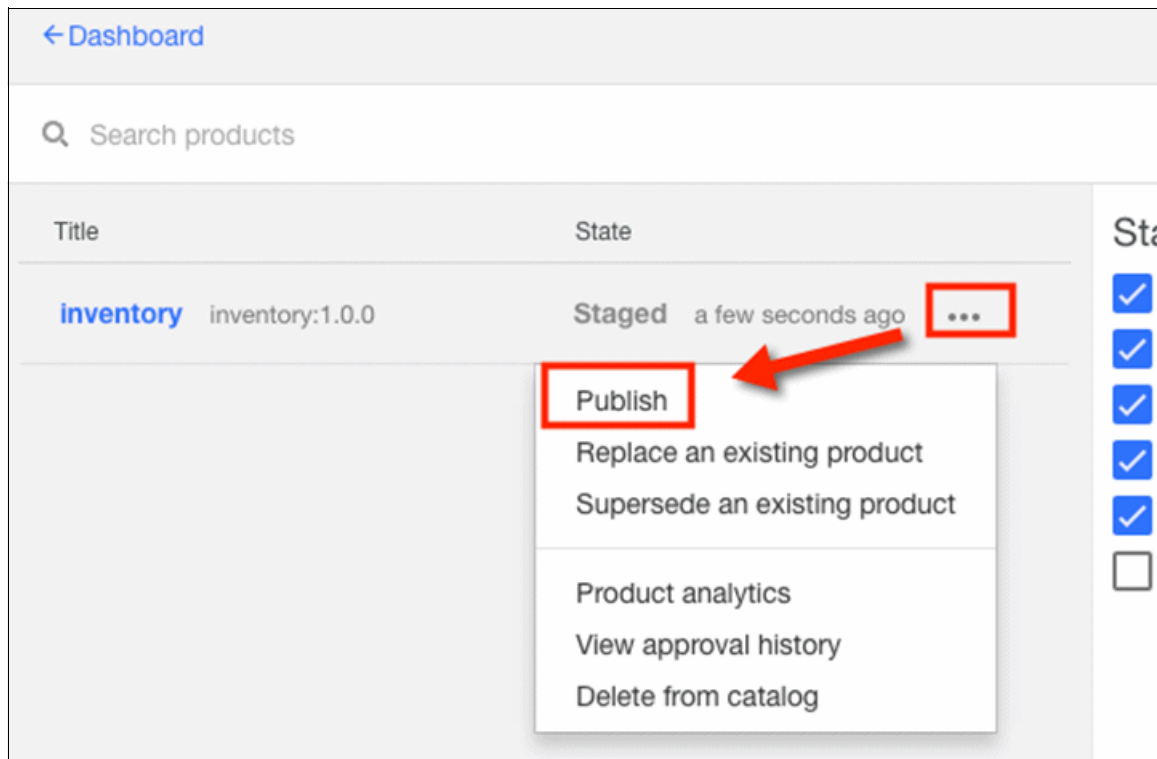


Figure 3-8 Select Publish

10. In the Edit visibility and subscribers section, keep the default settings and click **Publish**, as shown in Figure 3-9 on page 61.

inventory

Visible to: ⓘ

Public (Developer Porta ▾)

All developers will be able to see this product

Subscribable by: ⓘ

Authenticated (Develop ▾)

All authenticated developers in consumer organizations who have signed up for this developer portal can see this product

Cancel

Publish

Figure 3-9 Publishing the inventory product

The Inventory API version 1.0.0 is now ready to be used. From the dashboard, you see that the inventory product state changed to Published.

3.2 Publishing APIs by using the apic command line tool

Assuming that you are working for Company C, it is time to publish the social review APIs. In this section, we take a different approach to publishing APIs. Instead of using the API Designer User Interface, we describe how to publish API and products by using the apic command line. Complete the following steps:

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

Example 3-1 Logging in to your apic account by 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 in to us.apiconnect.ibmcloud.com successfully
```

3. You publish APIs and Products to ApicStore catalog. Therefore, you set the publish configuration variable by using the **apic config** command to the target catalog, as shown in the following example:

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/rguptalusibmcom-redbook/
catalogs/apicstore-catalog
```

Command syntax: This command features the following syntax:

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

You must replace the {bluemixOrg} and {bluemixSpace} with the values that are used in your environment.

4. Before you publish the API, validate your product and API definition by using the 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. Push the socialreview Product to the Bluemix API Connect by using the following command:

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

You should see the success message that is shown in Example 3-3.

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. Start the Bluemix API management console. Browse to the Dashboard from the top navigation pane. Click **ApicStore Catalog**. You see Inventory and socialreviews products in the Published state in Bluemix.

Both API products are now published to Bluemix API connect environment and can be used.

3.3 Summary

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



Securing the APIs

In this chapter, we describe the application security options that are available in IBM API Connect. We also describe how each type of security is applied to the sample scenario that is used in this publication.

Note: The chapter provides more background material about security and the justification for the level of application security and user authentication that we selected for the scenario. This information is interspersed with the configuration steps for the scenario.

This chapter includes the following topics:

- ▶ 4.1, “Overview” on page 66
- ▶ 4.2, “Application security” on page 66
- ▶ 4.3, “OAuth Security to protect user resources” on page 67
- ▶ 4.4, “Summary” on page 77

4.1 Overview

An API Provider must consider the following types of security that is 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 Client Secret? Client ID and secret can be thought of as user ID 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.

- ▶ User authentication and authorization

If your API is accessing protected resources that belong to a user (resource owner), you likely must apply OAuth security to your API. OAuth provides a mechanism for users to authenticate themselves and authorize an application to access resources on their behalf, while keeping their credentials secure and hidden from the application.

- ▶ Establishing trust between the API Gateway and target application

If your microservice or target application is running in a public cloud, such as public Bluemix, you face the issue of how to ensure that only the IBM API Connect Gateway is permitted to access the target microservice. If applications can target the microservice directly, it might be possible to bypass security.

In the following sections, we describe each of these issues how each type of security is applied to the inventory scenario.

4.2 Application security

When an API is defined, 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. By mandating this requirement, you can monitor the use of your API by specific applications, apply rate limits, and revoke access if an API user is misbehaving in any way.

For our sample scenario, all API operations are secured with a Client ID. As described in 4.3, “OAuth Security to protect user resources” on page 67, some clients cannot reliably protect a Client Secret. For this reason, we do not require a Client Secret in this sample scenario.

Complete the following steps to configure a Client ID security definition:

1. Browse to the **Security Definition** section of your API. (For this scenario, you must set the client ID on the socialreviews and inventory API.)
2. Select + and choose **API Key**. Enter the required information in the section, as shown in Figure 4-1 on page 67. You can specify that the caller passes the Client ID in a header or query parameter. For our sample, we use a header.

Security Definitions

+

clientIdHeader (API Key)

Name

clientIdHeader

Parameter name

X-IBM-Client-Id

Located In

Header

▼

Description

Figure 4-1 Security definitions

In the next section, we describe how to apply the Client ID APIKey security definition to your APIs, along with an OAuth security definition. For more information about how a client application, such as a mobile app, can access protected APIs by using Client ID and OAuth, see Chapter 5, “Using the APIs” on page 79.

4.3 OAuth Security to protect user resources

There are many use cases in which an API provides access to 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 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 user, it is important that the following requirements are met:

- ▶ The application can authenticate the user.
- ▶ The application does not gain access to the users credentials.
- ▶ The 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 more information about OAuth 2.0, see this website:

<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. IBM API Connect is used 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 that is named *OAuth Provider API*. After 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

Complete the following steps to configure the OAuth Provider API:

1. If you have not yet done so, clone the lab material to your local environment by following the instructions that are provided in “Cloning the GitHub repository” on page 11.

2. If you have not done so yet, deploy the authentication utility application as described in Appendix A, “Deploying the authentication utility application” on page 119.
3. Browse to the redp5350-apiconnect-sample\OAuth folder to edit the example OAuth Provider API, or you can create an API.

Note: In this chapter, we make references to socialreviews and inventory folders. These folders are organized under the following folder structures:

- ▶ redp5350-apiconnect-sample\socialreviews
- ▶ redp5350-apiconnect-sample\inventory

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, click **New** → **OAuth 2.0 Provider API**. Complete the fields as shown in Figure 4-2.

Add a new OAuth 2.0 provider 1 / 2

Provide a title and base path for the new OAuth 2.0 provider.

Title
Social Review OAuth Provider

Name
social-review-oauth-provider

Base Path
/oauth-provider

Version
1.0.0

Description
OAuth 2.0 Provider for securing the social review API

Cancel Next

Figure 4-2 Adding a new OAuth 2.0 Provider API

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

5. On the next page, select **Don't add to a product** and then, click **Add**.
6. Click **Save**, exit from the new API, and return to your list of APIs. You see the OAuth Provider API is now listed. Its type is OAuth 2.0, as shown in Figure 4-3.

| Title | Last Modified | Type | | Type |
|---|----------------|-----------|-----|---|
| Social Review OAuth Provider 1.0.0 | 23 minutes ago | OAuth 2.0 | ★ 🗑 | <input checked="" type="checkbox"/> REST <input checked="" type="checkbox"/> SOAP <input checked="" type="checkbox"/> OAuth 2.0 |

Figure 4-3 New OAuth Provider API

Next, we reopen the API and examine it more closely.

Client type

In the OAuth 2 section, the first thing that must be decided is whether to secure the API for Public or Confidential clients. In OAuth 2.0, *client* refers to the application that starts the API. Confidential clients can maintain the security of their client credentials, while public clients are not. Typically, native and browser-based applications are public, while those applications 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 choose the Public client. This choice means that clients are not required to protect or provide a client secret as part of the OAuth 2.0 flow.

Scopes

Scopes are used as labels that represent the set of endpoints for which a token must be valid to grant access. In the OAuth Provider API, we list the scopes that the OAuth Provider can secure. In this case, one scope is needed and it is named *review*. Create that scope and delete all others so that the first two sections are as shown in Figure 4-4.

OAuth 2

Client type

Client type

Public

Scopes

Scope Name

review

Description

Social Review write endpoints

Figure 4-4 Creating the scope

Grants

There are four types of grant flows available. For more information and complete flow diagrams, see the following resources:

- ▶ IBM API Connect IBM Knowledge Center:
<https://ibm.biz/BdrVUL>
- ▶ OAuth 2.0 RFC:
<https://tools.ietf.org/html/rfc6749>

These grant types are described next.

Implicit

In the Implicit grant flow, the client calls the `/authorize` endpoint that is made available by the OAuth API Provider. This call redirects the user to authenticate and authorize the client access to their resources. After the user completes this task, 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, IBM API Connect rejects 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 IBM 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 user's credentials. It collects the credentials and then calls the `/token` endpoint that is made available 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). However, this flow is used only if the API clients are known and can be trusted with the user's credentials.

Application

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

Access Code

In the Access Code flow, the `/token` and `/authorize` endpoints are used. As with 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, which authenticates by passing its client credentials by 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.

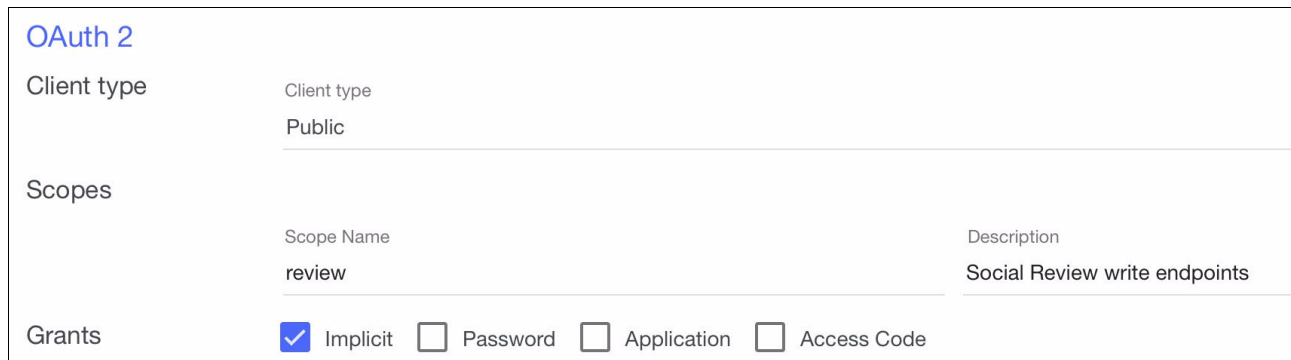
This flow is optimized for Confidential clients. If it is used by a Public client, the `client_id` is sent in the request body to the `/token` call instead of authenticating the client credentials by using basic authentication. The OAuth Provider must ensure that the grant was made to the specified `client_id`.

4.3.2 Summary

In the first step, we selected a Public client type. This selection eliminates the use of the Application flow, which is not applicable for our scenario and can be used by Confidential clients only.

For this scenario, we use the Implicit Grant flow, which is optimized for public clients. Unlike the password flow, it keeps user credentials protected. It also provides a simplified exchange that allows the client to receive the Bearer token with a single call instead of the extra call that is required by the Access Code flow.

In the Grants section, ensure that **Implicit** is selected and clear all other Grant types, as shown in Figure 4-5.



The screenshot shows the 'OAuth 2' configuration page. Under the 'Client type' section, 'Public' is selected. In the 'Scopes' section, a table lists a scope named 'review' with the description 'Social Review write endpoints'. In the 'Grants' section, four options are listed: 'Implicit' (checked with a blue checkmark), 'Password' (unchecked), 'Application' (unchecked), and 'Access Code' (unchecked).

Figure 4-5 Implicit option is selected

4.3.3 Identity extraction

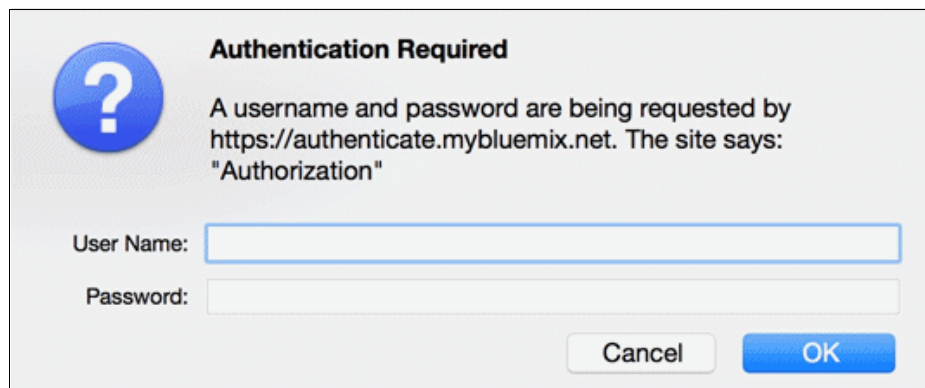
The *Identity extraction* is where you specify the means by which the user's (resource owner's) credentials are collected. For all Grant Types except *Password*, the application never accesses the user's credentials. IBM API Connect provides the following options for Identity extraction:

- Default form

The user is presented with a generic form that is provided by IBM API Connect.

- Basic

In this option, the native basic authentication support that is provided by your user agent is used. In Firefox, a pop-up panel as shown in Figure 4-6 opens.



The screenshot shows a Firefox 'Authentication Required' dialog box. It features a blue question mark icon and the text: 'Authentication Required. A username and password are being requested by https://authenticate.mybluemix.net. The site says: "Authorization"'. Below this, there are input fields for 'User Name:' and 'Password:'. At the bottom right, there are 'Cancel' and 'OK' buttons.

Figure 4-6 Basic authentication

- Custom

You can provide a custom HTML form to IBM API Connect. With this option, you must 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 IBM API Connect with the redirect URL and handle the authentication and authorization steps by following the protocol that is described in the following IBM Knowledge Center website:

<https://ibm.biz/BdrV5y>

In our scenario, we provide custom authentication and authorization forms. These forms are hosted on Bluemix in a simple Cloud Foundry node application. The application with these forms is in the `redp5350-apiconnect-sample/0Auth/authentication-app` folder. Complete the following steps to deploy this application:

1. Browse to `redp5350-apiconnect-sample/0Auth/authentication-app`.
2. Edit the `manifest.yml` file to give the application unique application and host names.
3. Ensure that you have the Cloud Foundry CLI.
4. Use the `cf login` command to log in, Answer the prompts to target the Bluemix Organization and Space to which you are deploying this application.
5. Use the `cf push` command to deploy the application to Bluemix.

Now that the authentication-app is deployed to Bluemix, enter the URL for the login form. The format for this URL is `<route>/login.html`, where `<route>` is the route that is configured in Bluemix for your application, as shown in Figure 4-7.

| | | |
|---------------------|---------------------------|---|
| Identity extraction | Collect credentials using | Custom form |
| | Custom form | ▼ https://redbook-authenticate.mybluemix.net/login.html |
| | TLS Profile | ▼ |

Figure 4-7 Custom authentication form

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

4.3.4 Authentication

Next, you must specify how the user is to be authenticated. You have the following options:

- Provide an authentication endpoint that accepts a basic authentication header. The endpoint returns a status code of 200 if the credentials are valid and 401 if they are invalid.
- Define an LDAP registry and select that registry. An LDAP registry can be defined in the API Manager console only and not in the API Designer. Also, when IBM API Connect is used in Bluemix, the specified LDAP registry must be accessible from the internet.

For our scenario, we use an Authentication URL, as shown in Figure 4-8 on page 73. This endpoint is also implemented in the authentication-app that we deployed to Bluemix. At the `<route>/authenticate` endpoint, the basic authentication header is checked and is validated against a known set of user IDs and passwords. The code for this endpoint can be found in `gitrepo/0Auth.js`.

| | | |
|----------------|--------------------------------------|---|
| Authentication | Authenticate application users using | Authentication URL |
| | Authentication URL | ▼ https://redbook-authenticate.mybluemix.net/authenticate |

Figure 4-8 Authentication URL

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

4.3.5 Authorization

Next, you must configure how the user authorizes the application to access their resources. The following options are available:

- ▶ **Default Form**
If you select this option, IBM API Connect presents a generic form for the user to accept.
- ▶ **Custom Form**
If you select this option, you must host a custom HTML form that prompts the user to authorize the application to access resources (similar to the custom for Identity Extraction). For more information, see this website:
<https://ibm.biz/BdrV591>
- ▶ **Authenticated**
If you select this option, it is assumed that the user implicitly authorized access if they authenticated. If you choose the redirect option for Identity Extraction, you are responsible for identity extraction and authorization and must select this option.

For our sample scenario, we use a custom form. Sample XHTML can be found in the demo package that was published on GitHub. For more information, see 1.3.3, “Downloading the applications from GitHub” on page 10. The custom form is shown in Figure 4-9.

IBM API Connect

Greetings...

The IBM Redbooks Mobile App would like to access your social review data.

No Thanks

Allow Access

Figure 4-9 Custom form

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

4.3.6 Tokens

In Figure 4-10, the *Time to live* setting is 3600 seconds. By default, the access token expires after 1 hour. For the Implicit Flow, the application must go complete the authorization process again to get a new access token.

The screenshot shows the 'OAuth 2' configuration page. The 'Client type' is set to 'Public'. Under 'Scopes', there is a table with 'Scope Name' as 'socialreview' and 'Description' as 'Social Review write endpoints'. In the 'Grants' section, 'Implicit' is selected with a checked checkbox, while 'Password', 'Application', and 'Access Code' are unchecked. The 'Identity extraction' section shows 'Collect credentials using' set to 'Custom form' with a dropdown arrow, and the URL 'https://redbook-authenticate.mybluemix.net/login.html' is displayed next to a 'TLS Profile' link. The 'Authentication' section shows 'Authenticate application users using' set to 'Authentication URL' with a dropdown arrow, and the URL 'https://redbook-authenticate.mybluemix.net/authenticate' is displayed next to a 'TLS Profile' link. The 'Authorization' section shows 'Authorize application users using' set to 'Custom form' with a dropdown arrow, and the URL 'https://redbook-authenticate.mybluemix.net/grant.html' is displayed next to a 'TLS Profile' link. In the 'Tokens' section, 'Access tokens' are configured with 'Time to live (seconds)' set to '3600'. There are two unchecked checkboxes: 'Enable refresh tokens' and 'Enable revocation URL'.

Figure 4-10 Auth Provider API configuration

Other OAuth flows support a refresh token process in which the refresh token can be issued with the original access token. After the access token expires, the refresh token can be exchanged for a new access token.

Leave the default setting at 3600 seconds and clear the **Enable refresh token** and **Enable revocation URL** options. Your OAuth Provider API configuration resembles that which is shown in Figure 4-10.

Completing OAuth Provider API

Ensure 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 must define the security definition on the API that you want to protect. Complete the following steps:

1. Before the API is configured, the URL for the authorization endpoint of the OAuth Provider API that we just completed is needed. Because this URL is catalog-specific, browse to the **Inventory** catalog. If you are in the API Designer, you might need to open the API Manager UI.

2. Go to the **Settings** tab of the catalog, and look for the API Endpoint Base URL, which resembles what is shown in Figure 4-11.

API Endpoint

Base URL: <https://api.us.apiconnect.ibmcloud.com/asariyusibmcom-redbooks/inventory>

Figure 4-11 Base URL

The full Authorization URL uses the <Base URL>/oauth-provider/oauth2/authorize format.

3. Browse to the **Social Review API** in the Security Definitions section.
4. On the right side, click **+** and then, select **OAuth** to create an OAuth security definition. After a search, you see the definition that you created, as shown in Figure 4-12.

oauth-1 (OAuth)

Name
oauth-1

Description

Flow
Implicit

Scopes

Authorization URL

Figure 4-12 OAuth security definition

5. Name the definition OAuth Definition and enter a description (optional). Ensure that the Flow is set to **Implicit**.
6. Enter the Authorization URL that was provided by the OAuth Provider API that is protecting this API. It is here that you enter the Authorization URL that we just looked up.
7. In the Scopes section, enter socialreview, which must match the scope that we used in the OAuth Provider API. The scopes in the OAuth Provider API indicate all of the scopes that it can protect. The scopes that are listed in the Security Definition list the scopes that the caller must specify to be granted access to this API. After this process is completed, your configuration resembles that which is shown in Figure 4-13.

OAuth Definition (OAuth)

Name
OAuth Definition

Description
OAuth security definition for Social Review API

Flow
Implicit

Scopes

Authorization URL
<https://api.us.apiconnect.ibmcloud.com/ilene-api/test/oauth-provider/oauth2/authorize>

Scope Name
review

Description
Allow user to pose comments/reviews to the Social Review API

Figure 4-13 OAuth Definition

8. Scroll down to the Security section for the API and ensure that your OAuth Definition is selected, as shown in Figure 4-14.

The figure shows a 'Security' section with a subtitle 'Define security requirements for the API. Multiple alternative sets can be defined, any one of which can be satisfied to access the API.' Below this, there is a list of options. 'Option 1' is selected, and it contains two checked items: 'OAuth Definition (OAuth)' and 'clientIdHeader (API Key)'.

| Security | |
|--|---|
| Define security requirements for the API. Multiple alternative sets can be defined, any one of which can be satisfied to access the API. | |
| Option 1 | <input checked="" type="checkbox"/> OAuth Definition (OAuth) <input checked="" type="checkbox"/> clientIdHeader (API Key) |

Figure 4-14 Security section

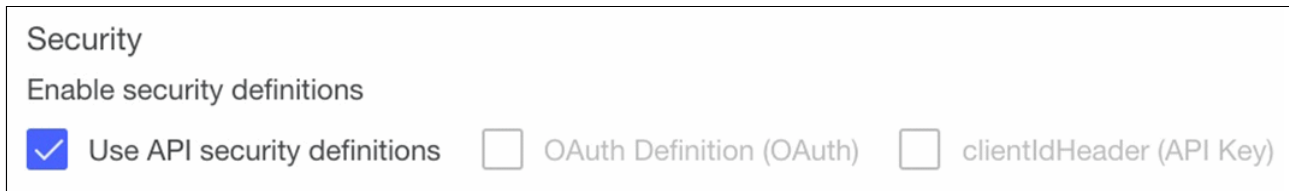
9. OAuth is not needed on every operation; instead, it is needed only on the operation where the user is creating a review. Expand each operation and review the Security section.
10. For all of the operations (except the POST reviews section), clear **Use API security definitions** and ensure only **clientIdHeader (API Key)** is selected. In this way, you can override the API security definition and select only the types of security you need per endpoint. After this update is made, you see the form that is shown in Figure 4-15.

The figure shows the configuration for the GET /reviews endpoint. It includes a 'review' tag, a summary, a description, parameters, responses, and a security section. In the security section, 'Use API security definitions' is unchecked, 'OAuth Definition (OAuth)' is unchecked, and 'clientIdHeader (API Key)' is checked.

| GET /reviews | | |
|--|---|--|
| review × Add Tag | | |
| Summary Find all instances of the model matched by filter from the data source. | | |
| Description | | |
| Parameters | | |
| Name | Located In | Description |
| filter | Query | Filter defining fields, where, includ |
| Responses | | |
| Status Code | Description | |
| 200 | Request was successful | |
| Security | | |
| Enable security definitions | | |
| <input type="checkbox"/> Use API security definitions | <input type="checkbox"/> OAuth Definition (OAuth) | <input checked="" type="checkbox"/> clientIdHeader (API Key) |

Figure 4-15 Selecting the clientIdHeader (API Key) definition

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



Security

Enable security definitions

☒ Use API security definitions ☐ OAuth Definition (OAuth) ☐ clientIdHeader (API Key)

Figure 4-16 Use API security definitions selected

12. Save the API. Ensure that the OAuth Provider API and the Social Reviews API are both in your Product and publish to the Inventory catalog.

13. Follow the instructions that are described in 3.2, “Publishing APIs by using the apic command line tool” on page 61 to publish the APIs.

4.4 Summary

In this chapter, we described the importance of controlling access to your target application endpoints so that only calls from the IBM API Connect Gateway are accepted by the target application. We also described mutual authentication and basic authentication as techniques for protecting the target application.

We have shown how IBM API Connect automatically configures mutual authentication when you use the API Designer to publish LoopBack applications to Bluemix.



Using the APIs

This chapter describes the steps that are used by an application developer to use an API.

In this chapter, we provide a description on how an application developer can discover and use the APIs that are provided by IBM Connect in the development of their applications. We present the following scenarios to describe the process:

- ▶ In the first scenario, the steps that are needed for a Mobile developer to use the APIs that are available in IBM API Connect are described. In this case, you experiment with how to build an iOS application to use the inventory and socialreview APIs.
- ▶ In the second scenario, we describe the steps that are needed for a Web developer to use the same APIs.

This chapter includes the following sections:

- ▶ 5.1, “Sign up to use API” on page 80
- ▶ 5.2, “Developing the mobile iOS application” on page 89
- ▶ 5.3, “Running the sample consumer web application” on page 97
- ▶ 5.4, “Summary” on page 100

5.1 Sign up to use API

In this section, we describe how the developers can use APIs that are produced by the API developers by using the IBM 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 users to discover APIs and subscribe to a usage plan by which the API is used in the mobile or web application.

In reality, the mobile and web application might be developed by different digital agencies or teams. For simplicity, we assume that the mobile and web application are developed by the same team and use the same API plan in the Developer Portal.

As an API user, the following common process is completed to use an API:

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

We describe the second, third, fourth steps of this process next. The first and last steps are outside the scope of this scenario.

5.1.1 Registering an application

The first step to use APIs is to register the API. To complete this step, you must log on to IBM API Connect Developer Portal.

Note: There are different options to get the developer portal. The option that is used depends on how the API providers want to make available the APIs, such as establishing developer community or emailing the developer portal invitation.

In our scenario, we assume that the API providers and API user are the same team who shares the API connect environment. Thus, the developer portal that was created in Chapter 1, “Introduction to IBM API Connect and environment setup” on page 1 is known to the API user.

Complete the following steps:

1. To get the developer Portal URL, log in to your Bluemix API Management console.
2. Browse to the **ApicStore Catalog**. Click the **Settings** tab. Then, click the **Portal** subtab. You see the IBM Developer Portal URL, as shown in Figure 5-1 on page 81. Copy that URL.

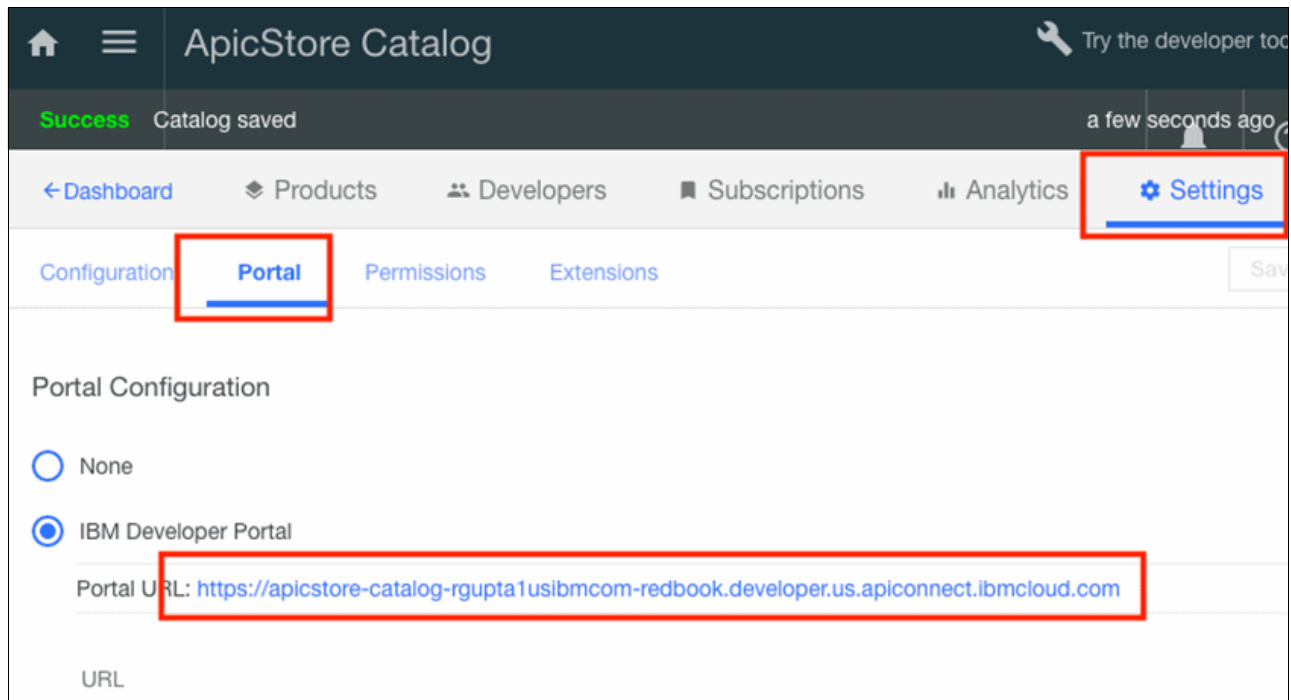


Figure 5-1 Product URL that is needed to send the API developers

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

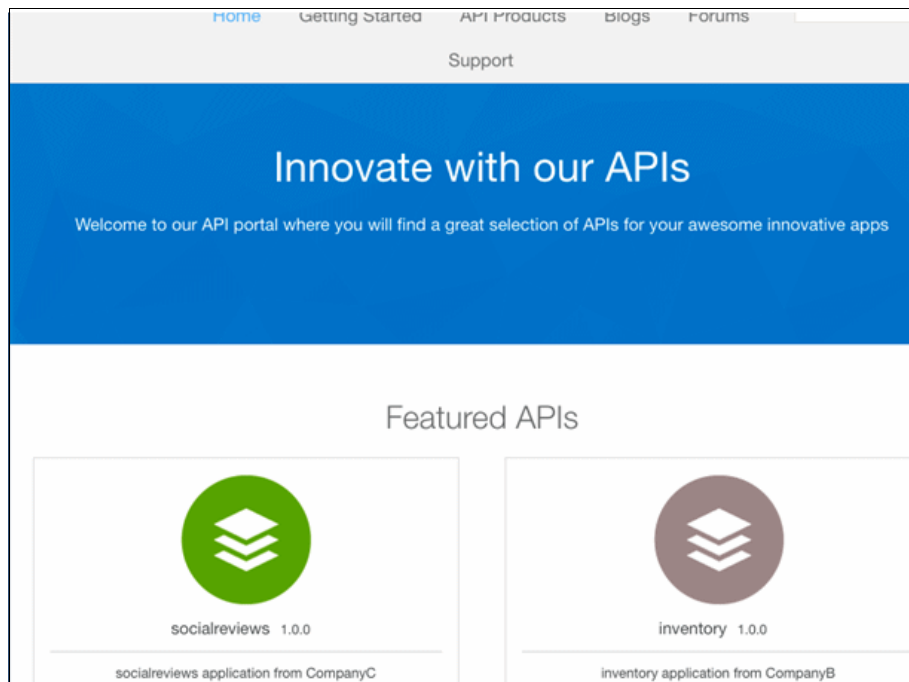


Figure 5-2 API Developer Portal main page

4. To use the API Developer Portal, log in on the right side at the top of the page. If you have an IBM ID, click **Login** and enter your username and password. Otherwise, we assume that you do not have a username and password for the purposes of this scenario. Therefore, click **Create a user account**. The form that is used to create a user is shown in Figure 5-3. The username is your Bluemix login account or a valid IBM ID.

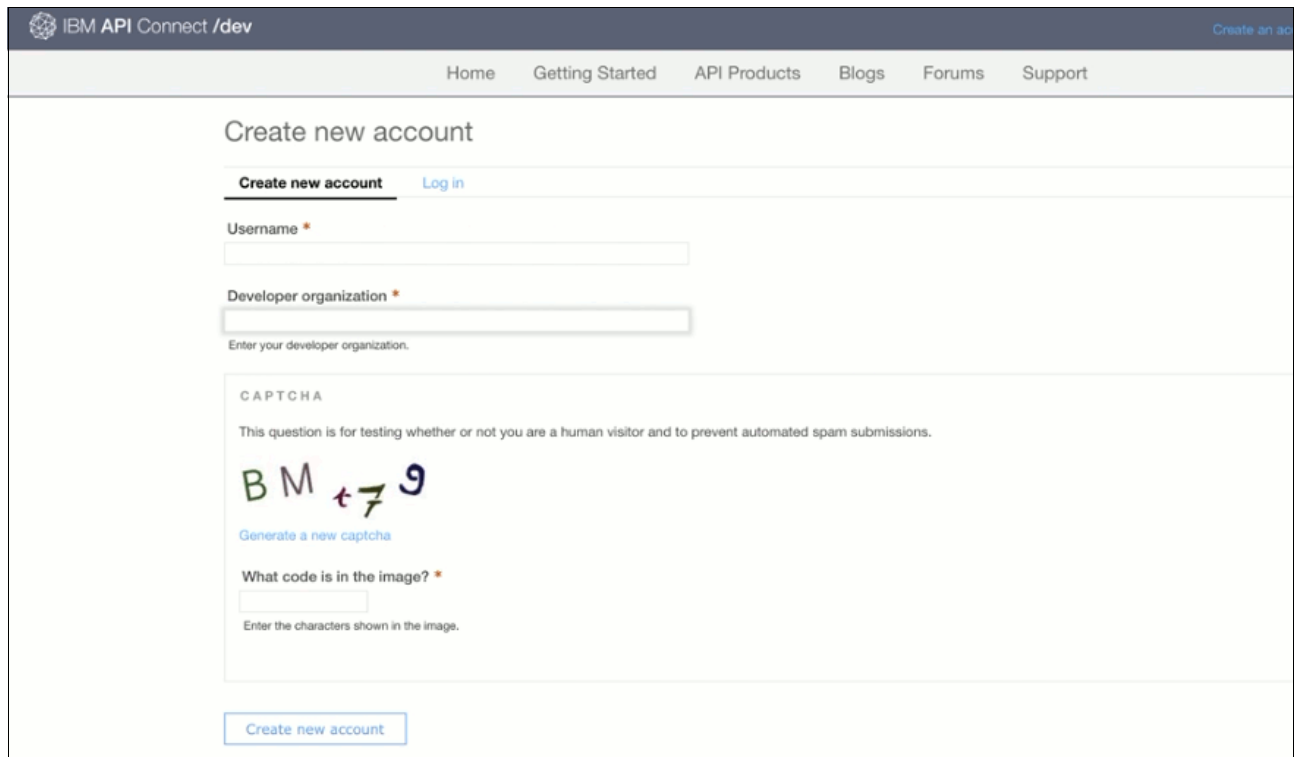
The screenshot shows the 'Create new account' page of the IBM API Connect Developer Portal. The page has a dark blue header with the IBM logo and 'IBM API Connect /dev' on the left, and a 'Create an account' link on the right. Below the header is a navigation bar with links: Home, Getting Started, API Products, Blogs, Forums, and Support. The main content area is titled 'Create new account' and contains two tabs: 'Create new account' (active) and 'Log in'. The 'Create new account' tab has two input fields: 'Username *' and 'Developer organization *'. Below the 'Developer organization *' field is a hint: 'Enter your developer organization.' Below these fields is a CAPTCHA section. The CAPTCHA section has the title 'CAPTCHA' and a description: 'This question is for testing whether or not you are a human visitor and to prevent automated spam submissions.' The CAPTCHA image shows the characters 'BM' followed by a red arrow pointing to a '7' and a '9'. Below the image is a link: 'Generate a new captcha'. Below the CAPTCHA section is a label: 'What code is in the image? *' and an input field. Below the input field is a hint: 'Enter the characters shown in the image.' At the bottom of the form is a 'Create new account' button.

Figure 5-3 Creating a user account at the API Developer Portal

5. Enter ApicStore-App-Dev as your developer organization.
6. Click **Create new account**.
7. Click **Login**. Enter your IBM ID credential to log in.
8. After you log in, click the **API Products** tab to discover the APIs you want to use. In our scenario, we select the product Inventory. More information is available on the page that is related to the Product we are selecting. You can see the version and if the product is online or offline. The view of different products that are available to the App developer is shown in Figure 5-4 on page 83.

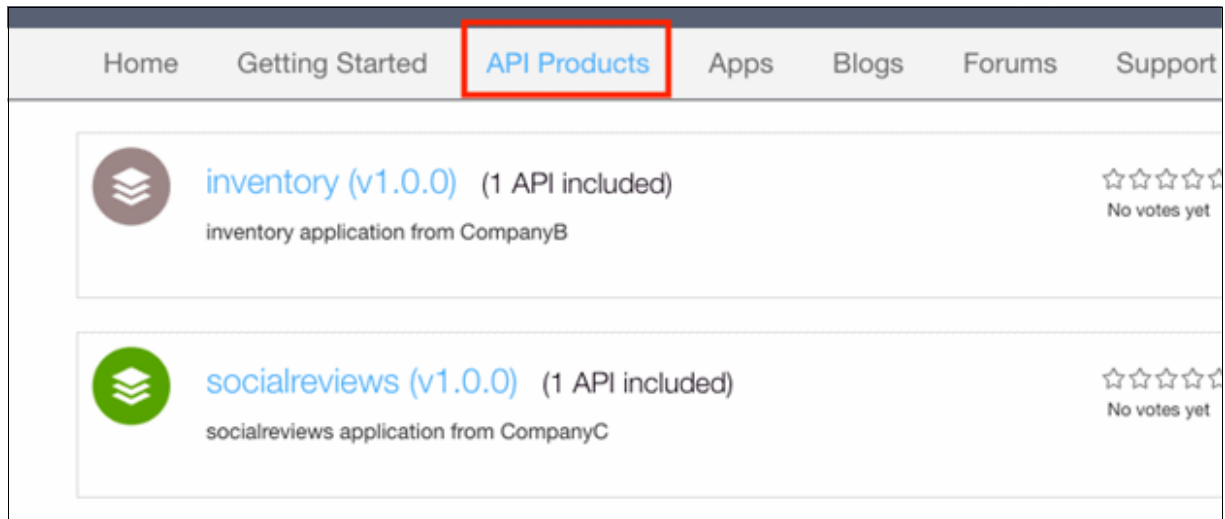


Figure 5-4 API Products page

We now must create a development application to subscribe to API. Complete the following steps:

1. Click the **Apps** tab at the top of the window.
2. Click **Register a new Application**, as shown in Figure 5-5.

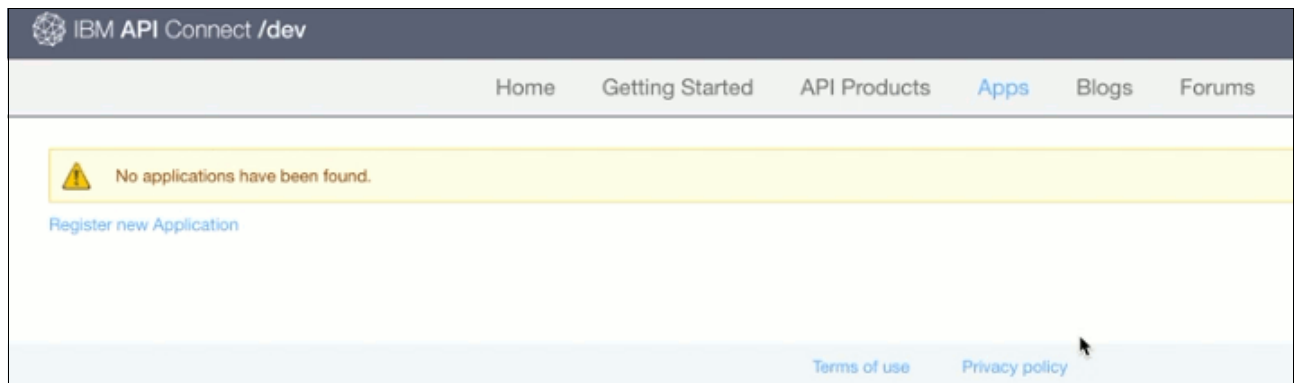


Figure 5-5 New app registration

3. To register your new application, you must complete the Title and Description fields. Enter `org.apic://example.com` for the OAuth URI (redirection page), as shown in Figure 5-6.

Figure 5-6 Registering a new application

4. Click **Submit**.

The new application is registered, and a window that is similar the window that is shown in Figure 5-7 opens.

Note: Store your Client secret because this information is needed to use the API. You also must store the Client ID. You this information when your mobile and web applications are built.

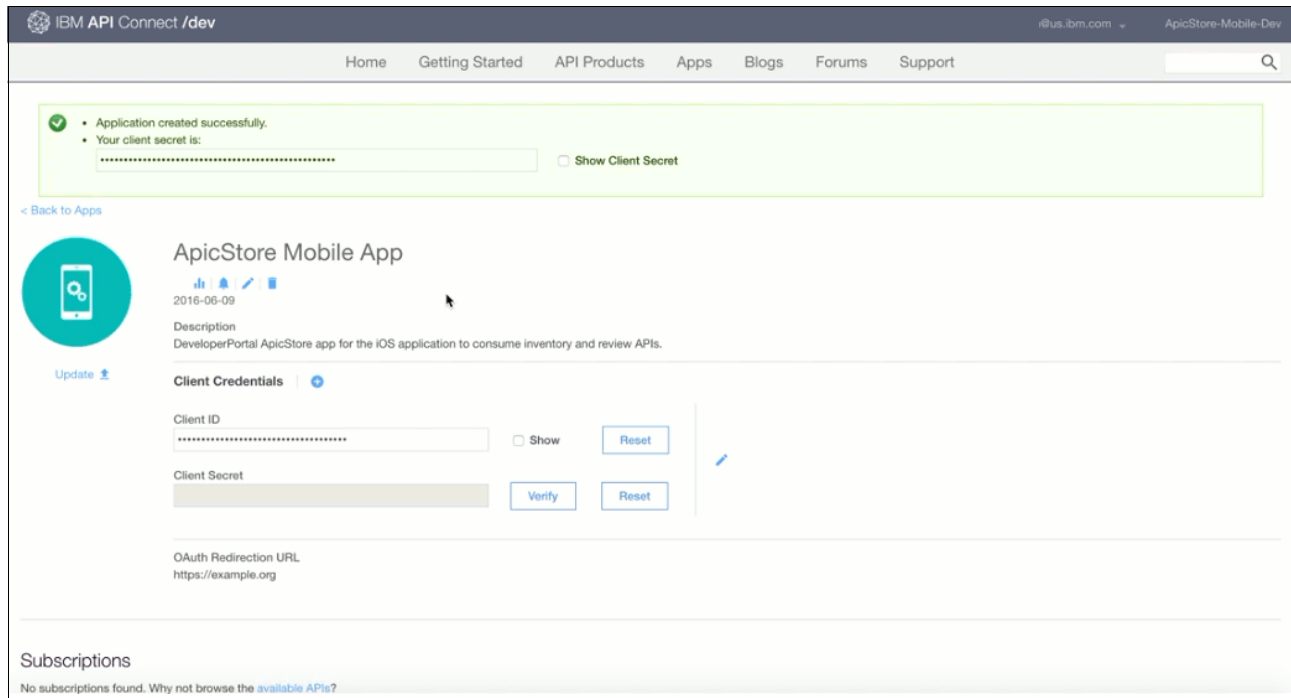


Figure 5-7 Application that is created by using API Developer Portal

5.1.2 Subscribing to an API plan

After you register a new application with the API Developer portal, you are now ready to use the APIs that are published in your new application. Complete the following steps:

1. Subscribe to the available APIs. Click the link that is beneath your new application. Clicking this link takes you to the available APIs.
2. The API product page opens, as shown in Figure 5-8 on page 85. To subscribe to an API, click one of the **API products**. Figure 5-8 on page 85 shows the API Product page for the sample Social Reviews API to which we want to subscribe.

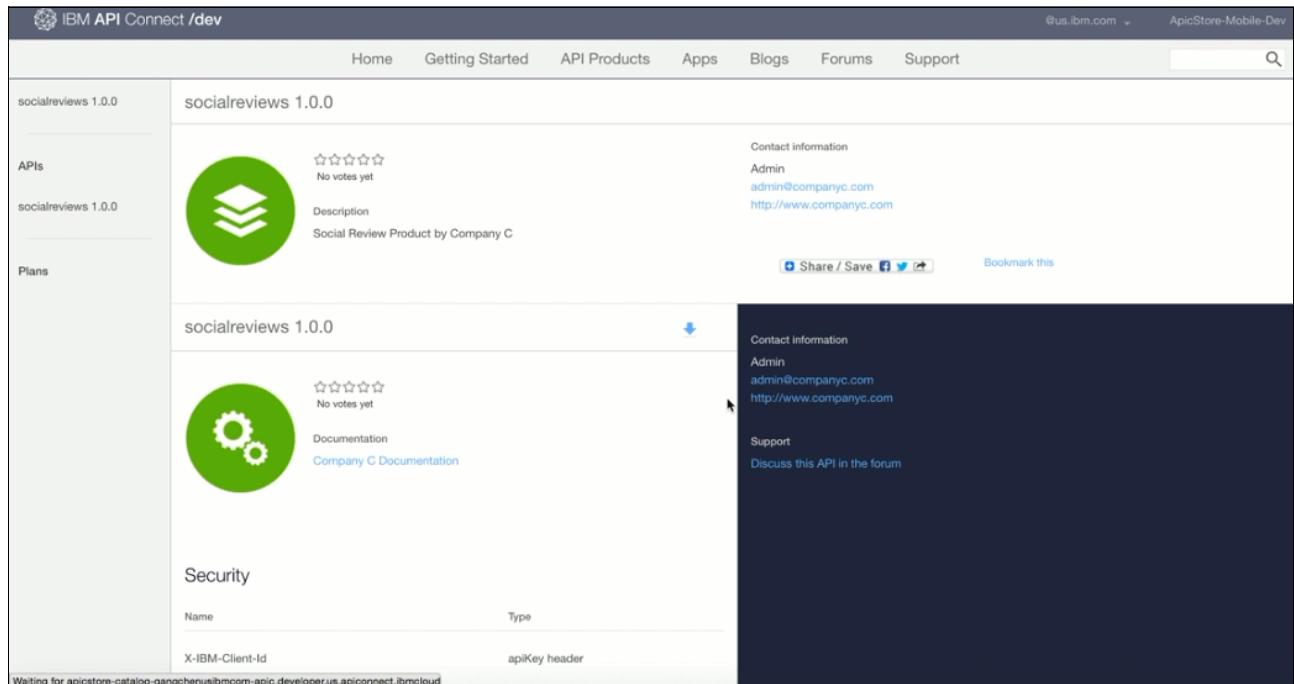


Figure 5-8 API Developer portal showing API Product page

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

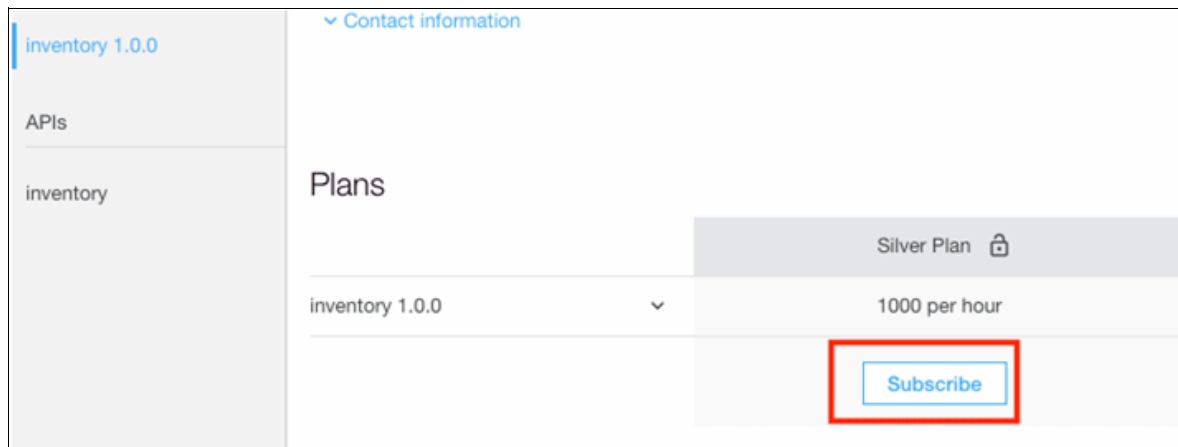


Figure 5-9 API Subscription page

4. Subscribe to the socialreview API as well (choose the silver or gold plan).

5. Click **Apps** → **ApicStore Mobile and Web App page**. You see that both APIs are subscribed in your Application page, as shown in Figure 5-10.

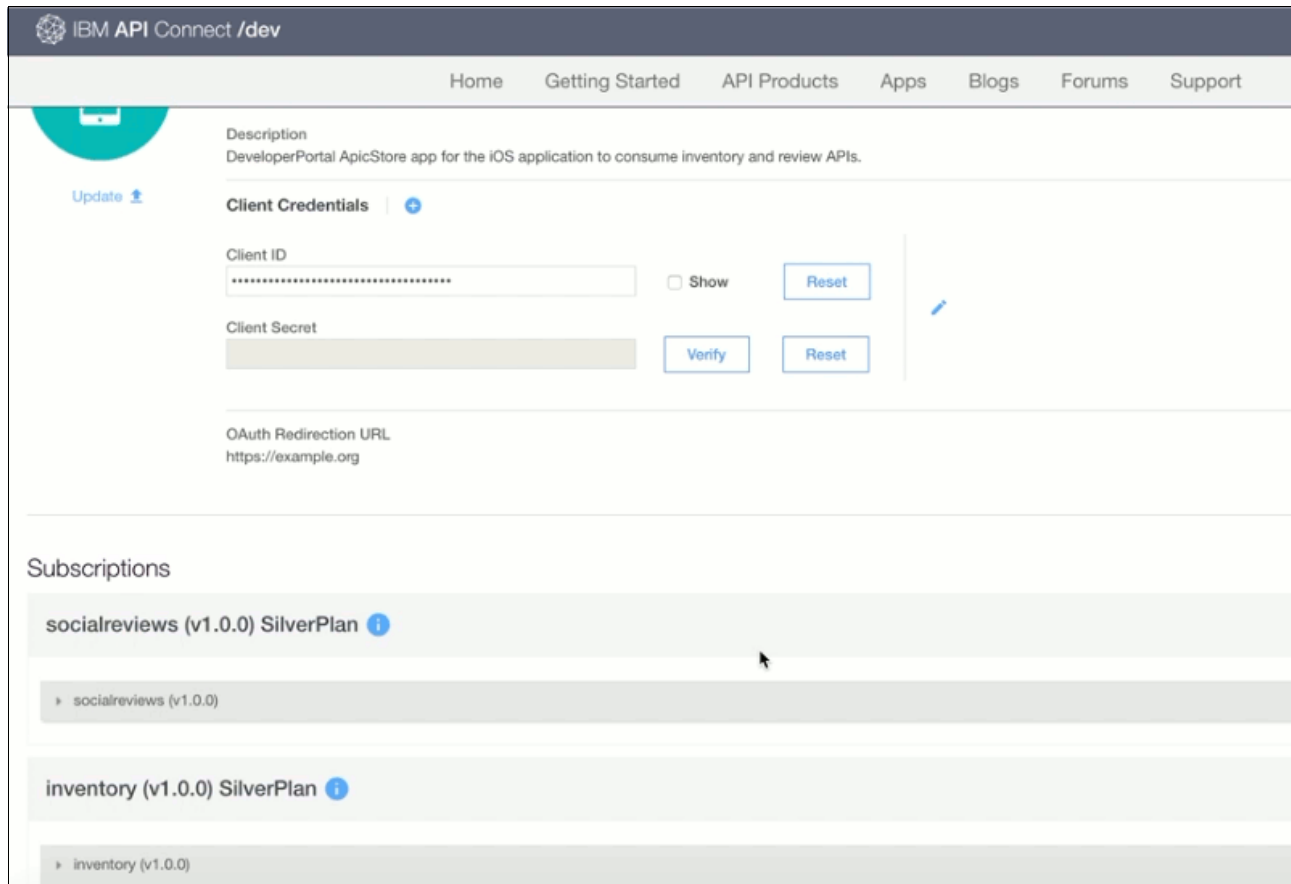


Figure 5-10 Product page showing the new subscriptions for the developer

5.1.3 Testing an API from the development portal

One of the useful features of the API Developer Portal is to test the APIs. Complete the following steps to test one of the APIs to which we subscribed:

1. Go to the API Developer Portal and click one of the **product APIs**. Then, click **inventory (v1.0.0)**.
2. In the left navigation menu under APIs, click **inventory 1.0.0**. The menu expands and shows all APIs to which you subscribed, as shown in Figure 5-11 on page 87.

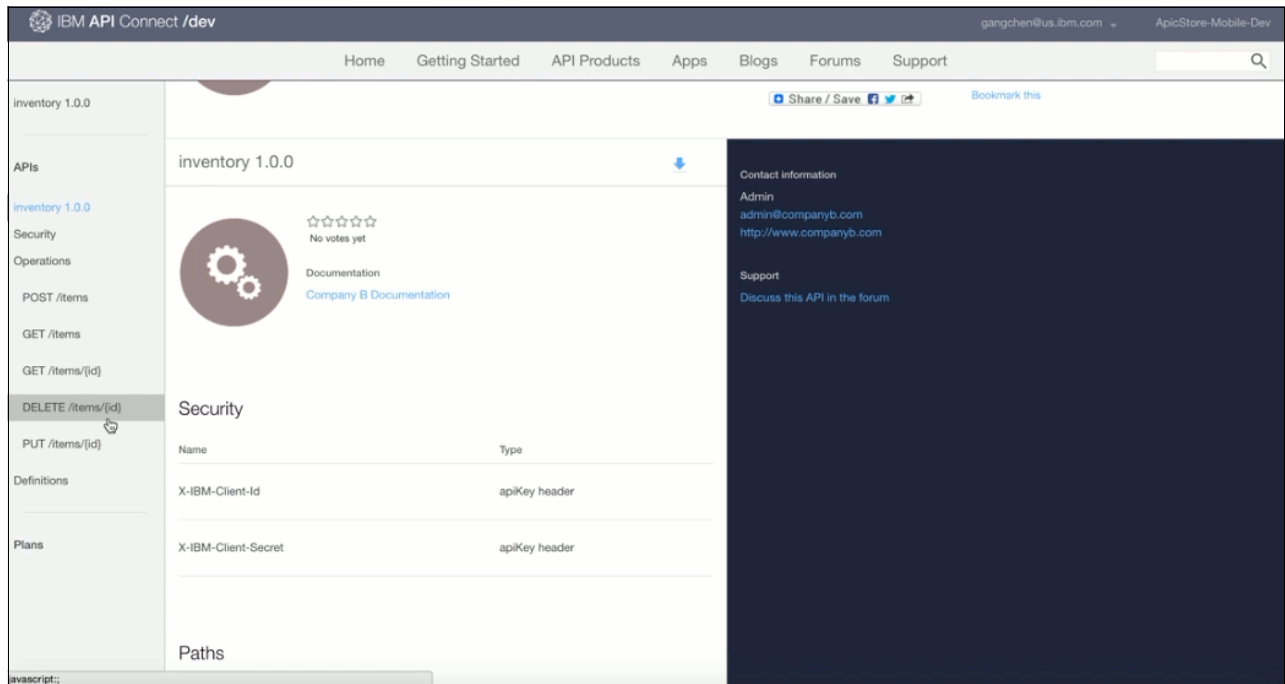


Figure 5-11 Product page with API options displayed

3. In the left menu, click **Operation GET /items**. The API detail is shown in right side pane. Figure 5-12 shows one of the APIs displayed. In the right menu, you 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 that is available in most *NIX systems, which the developers can use to test the API access.

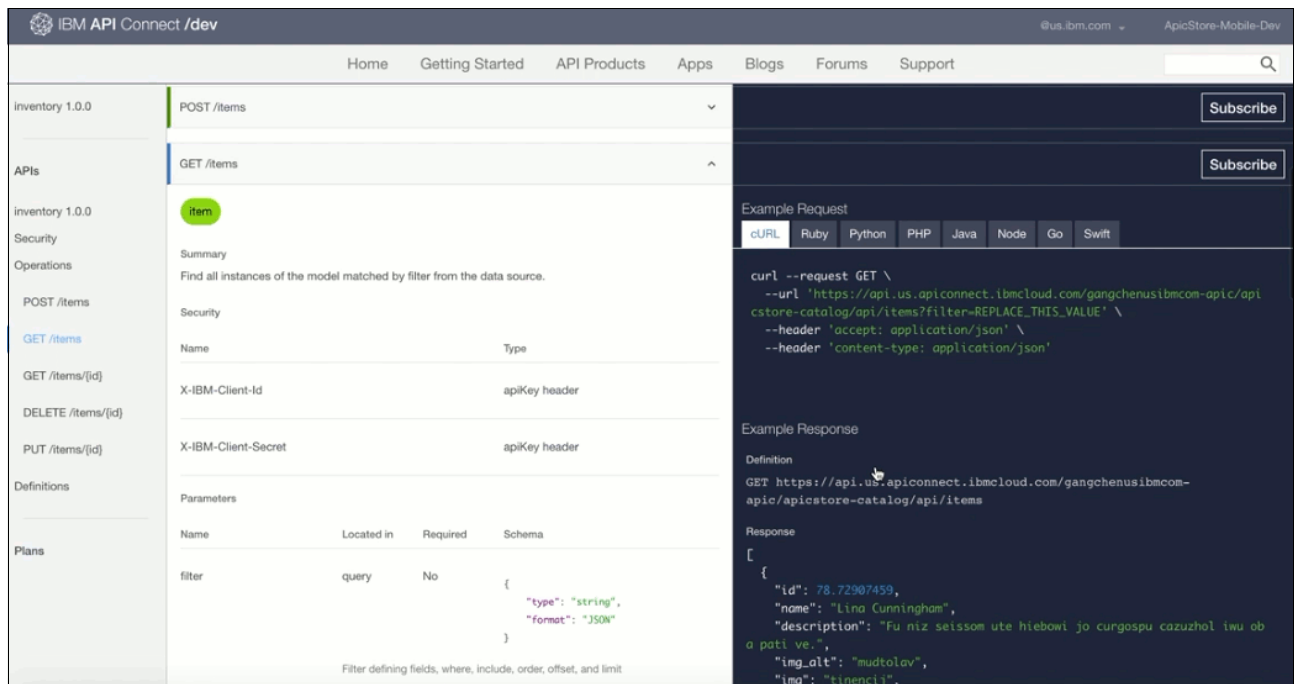


Figure 5-12 API Developer Portal tools to test API access and sample code

- The API Developer Portal contains multiple tools that can be used 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). On the right side, scroll down until you see the Call operation option. By using this option, the API endpoint is called from the browser and the results are displayed. The interface that is used to test the API endpoint from the browser by clicking **Call operation** is shown in Figure 5-13.

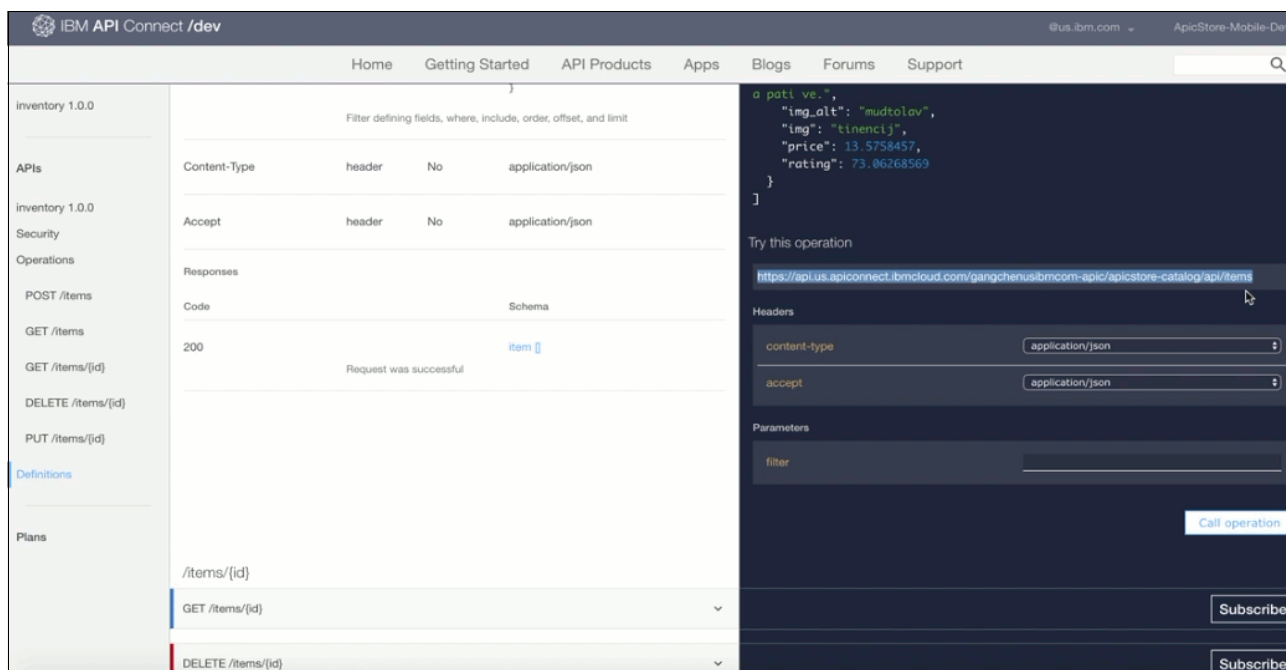


Figure 5-13 Testing the API from the API Developer Portal

- After you call the operation, the browser shows you the results of calling the API. Figure 5-14 on page 89 shows the results that were 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 also shows the response data.

```
Request

GET https://api.us.apiconnect.ibmcloud.com/rguptalusibmcom-redbook/apicstore-catalog/api/items
X-IBM-Client-Id: 561b35bb-ec56-4c19-9d2b-69c3bd0d5a0a
content-type: application/json
accept: application/json


Response

200 OK
APIm-Debug-Trans-Id: 10.120.202.241-6872b349-8781-4fdb-ab4a-9b8e3069d38f
X-RateLimit-Remaining: 999
X-Global-Transaction-ID: 1302113
Content-type: application/json
X-RateLimit-Limit: 1000
[
  {
    "description": "Punched-card tabulating machines and time clocks were not the only products offered by the young IBM. Seen here in 1930, manufacturing employees of IBM's Dayton Scale Company are assembling Dayton Safety Electric Meat
```

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

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 application if you are using one of those programming languages.

Now you are ready to build the applications.

5.2 Developing the mobile iOS application

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

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

Use the command that is shown in Example 5-1 to locally clone the project if it was not yet cloned.

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

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

Now that the project is cloned locally, browse to the project folder that contains the GitHub project, as shown in Example 5-2.

Example 5-2 Location of the directory

/redp5350-apiconnect-sample/ApicStoreApp

From this location, enter `open ApicStoreApp.xcodeproj` to open the Xcode application.

You can also open the iOS project by double-clicking the `open ApicStoreApp.xcodeproj` file from the Mac Finder.

Now that the iOS project is opened in Xcode, we take a brief look at the project structure. On the left side, you see the file structure that is shown in Figure 5-15. (This file structure might not expand after it is opened.)

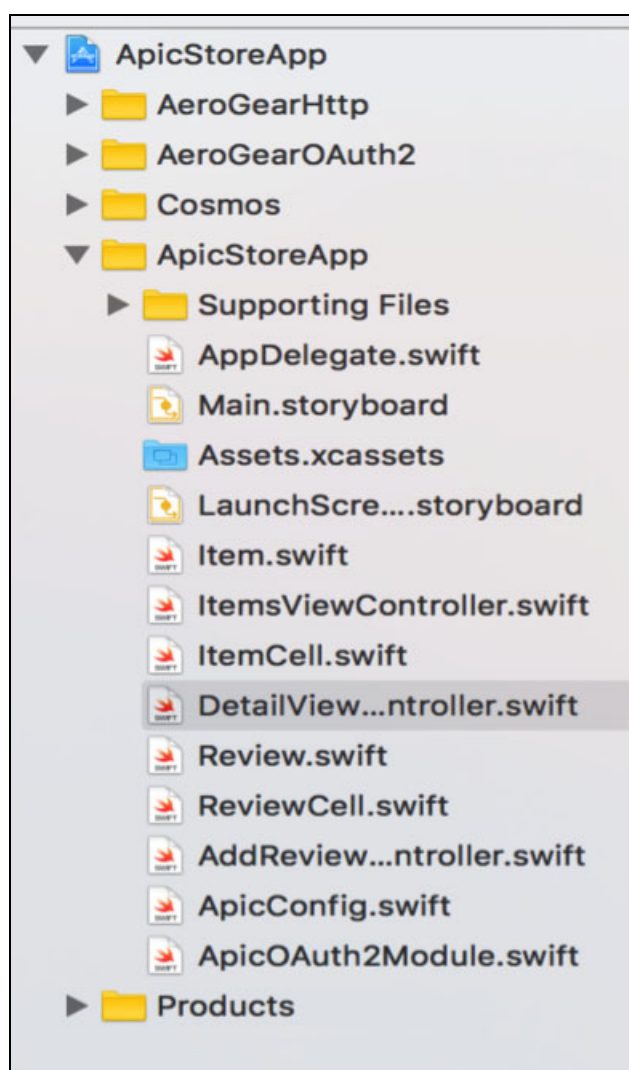


Figure 5-15 File structure

The main application is in the ApicStoreApp folder. The AeroGearHttp and AeroGearOAuth2 folders contain third-party libraries that we use to make our API calls by using OAuth. For more information about these libraries, see this website:

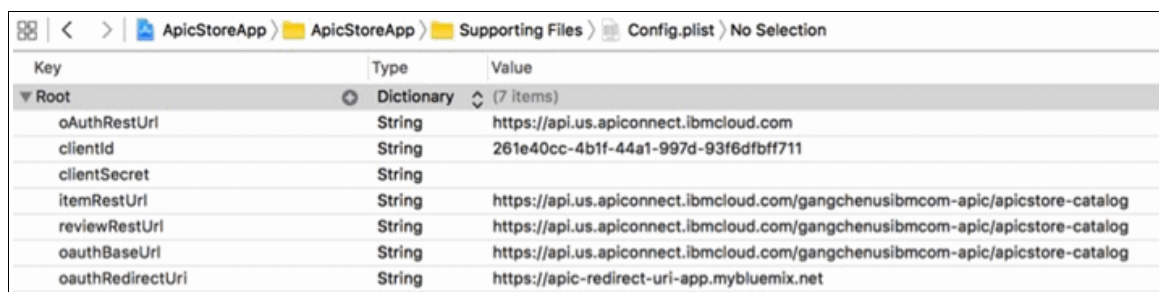
<https://aerogear.org/ios/>

A simple third-party library also is available to show the star ratings for the user reviews. For more information, see this website:

<https://github.com/marketplacer/Cosmos>

Complete the following steps to run and test the iOS application:

1. Specify the API endpoint configuration for your Bluemix IBM API Connect deployment by editing the ApiStoreApp/Supporting Files/Config.plist file. The Config.plist file contains all of the API endpoint URLs and the client ID that was registered in the Developer Portal. Any changes to the API endpoint URLs can be made in this file, as shown in Figure 5-16.



| Key | Type | Value |
|------------------|----------------------|--|
| Root | Dictionary (7 items) | |
| oAuthRestUrl | String | https://api.us.apiconnect.ibmcloud.com |
| clientId | String | 261e40cc-4b1f-44a1-997d-93f6dfbf711 |
| clientSecret | String | |
| itemRestUrl | String | https://api.us.apiconnect.ibmcloud.com/gangchenusibmcom-apic/apicstore-catalog |
| reviewRestUrl | String | https://api.us.apiconnect.ibmcloud.com/gangchenusibmcom-apic/apicstore-catalog |
| oauthBaseUrl | String | https://api.us.apiconnect.ibmcloud.com/gangchenusibmcom-apic/apicstore-catalog |
| oauthRedirectUrl | String | https://apic-redirect-uri-app.mybluemix.net |

Figure 5-16 Configuration list for the mobile application

Config.plist file: The following endpoints and constants are part of the Config.plist file:

- ▶ `oAuthRedirectUrl`: This OAuth Redirect API URL was defined in step 3 in 5.1.1, “Registering an application” on page 80. It should be `org.apic://example.com`.
- ▶ `clientId`: This client ID that is obtained in the Developer Portal.
- ▶ `ItemRestUrl`, `reviewRestUrl`, `oAuthBaseUrl`: These API endpoints are from the Developer Portal for Inventory API, review API, and OAuth API. In our case, the base URL host for all of these are the same; however, the URIs in the code are different for each call.
- ▶ `AuthRestUrl`: This endpoint triggers the OAuth flow for socialreview API. The base URL also is `org.apic://example.com`.

These four endpoints should all be the same and are your apic-catalog endpoint. For more information, see the following example:

<https://api.us.apiconnect.ibmcloud.com/gangchenusibmcom-apic/apicstore-catalog>

2. To run the application, click **Play** (in the upper left corner) to run the application (ensure that iPhone 6 or 6 Plus is selected), as shown in Figure 5-17.

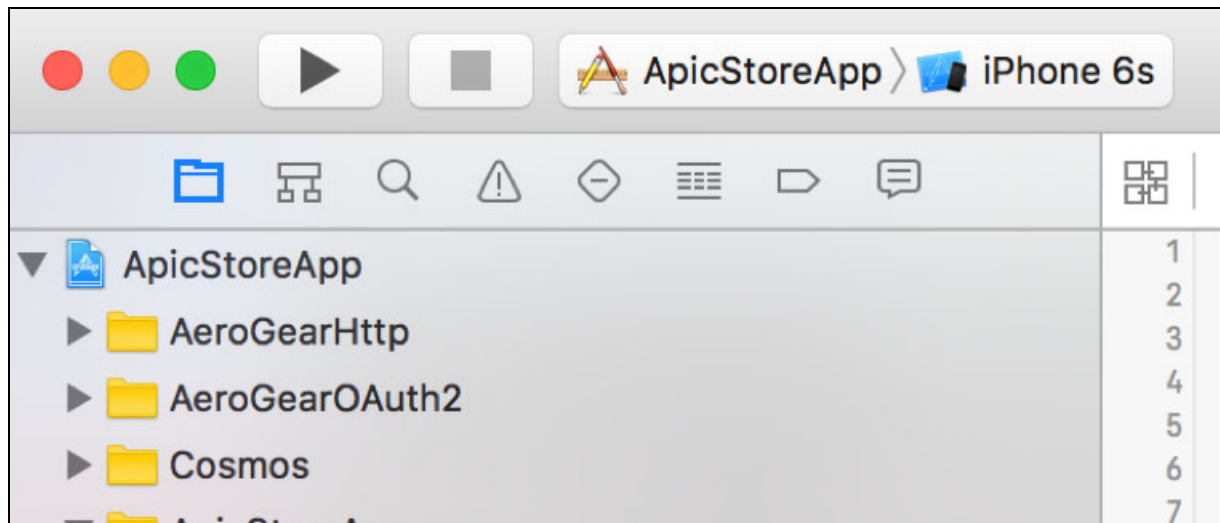


Figure 5-17 Mobile application run time

The iOS application consists of the following views:

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

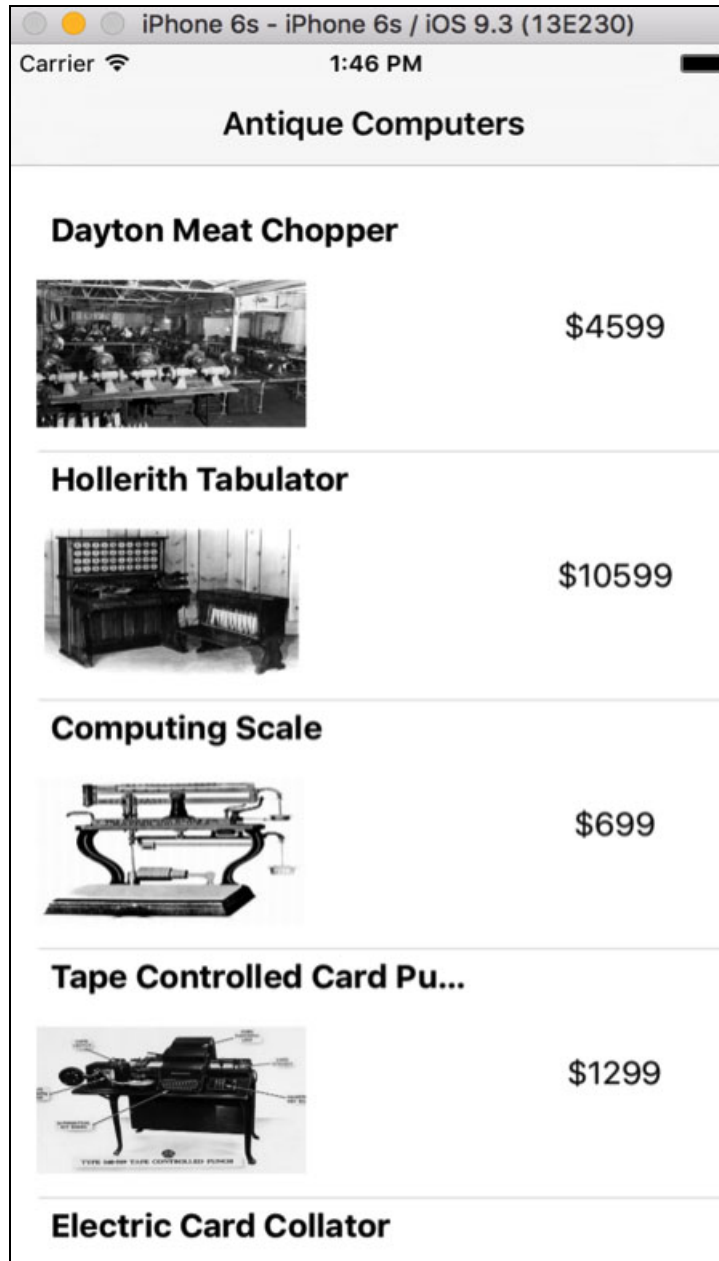


Figure 5-18 Mobile application item list

- The individual item details are shown in the Product detail menu (which is accessed by clicking **Dayton Meat Chopper**), as shown in Figure 5-19.

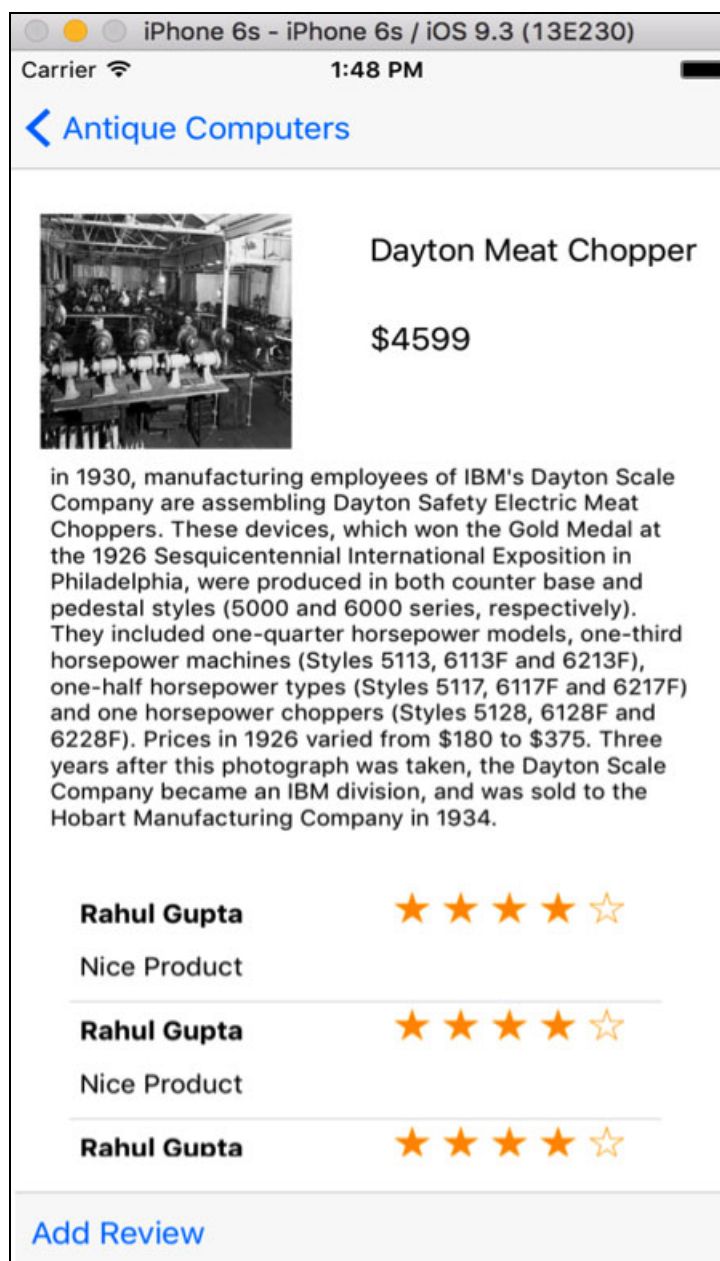
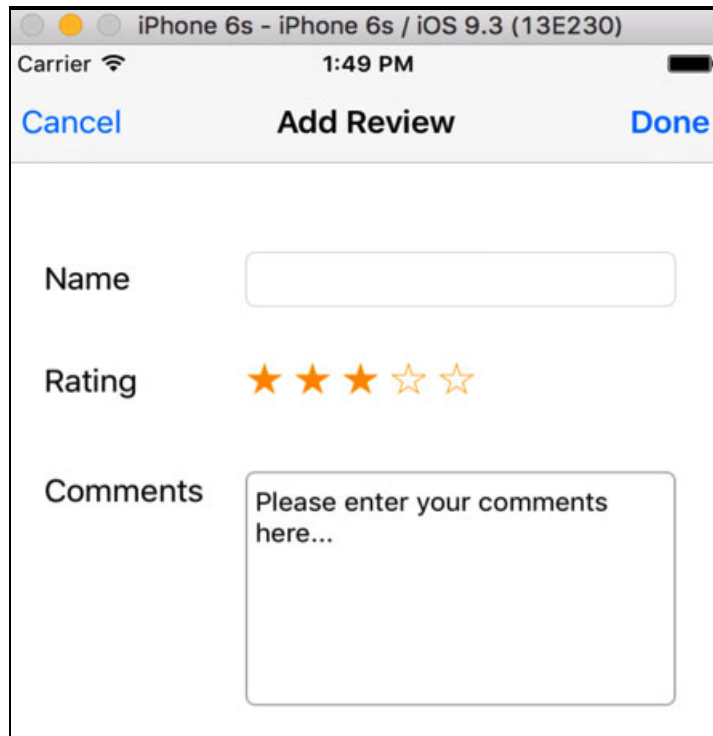


Figure 5-19 Mobile application item details

- The Add Review window is shown in Figure 5-20.



The screenshot shows the 'Add Review' window on an iPhone 6s. The status bar at the top indicates 'Carrier', signal strength, '1:49 PM', and battery level. The window title bar contains 'Cancel', 'Add Review', and 'Done' buttons. The main content area has three sections: 'Name' with a text input field, 'Rating' with five star icons (three filled, two empty), and 'Comments' with a text area containing the placeholder text 'Please enter your comments here...'.

Figure 5-20 Mobile application item review detail

3. When the application is running, tap **Add review** button and the application transitions to the API OAuth window with login information that retrieves the OAuth session and token that is used by the add review API.

The correct credentials (username: foo, password: bar) must be entered to authenticate the OAuth session; otherwise, the add review API does not work. After the credentials are provided, the session is 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 the **Main.Storyboard** file in the file navigation menu under the ApicStoreApp folder, as shown in Figure 5-21.

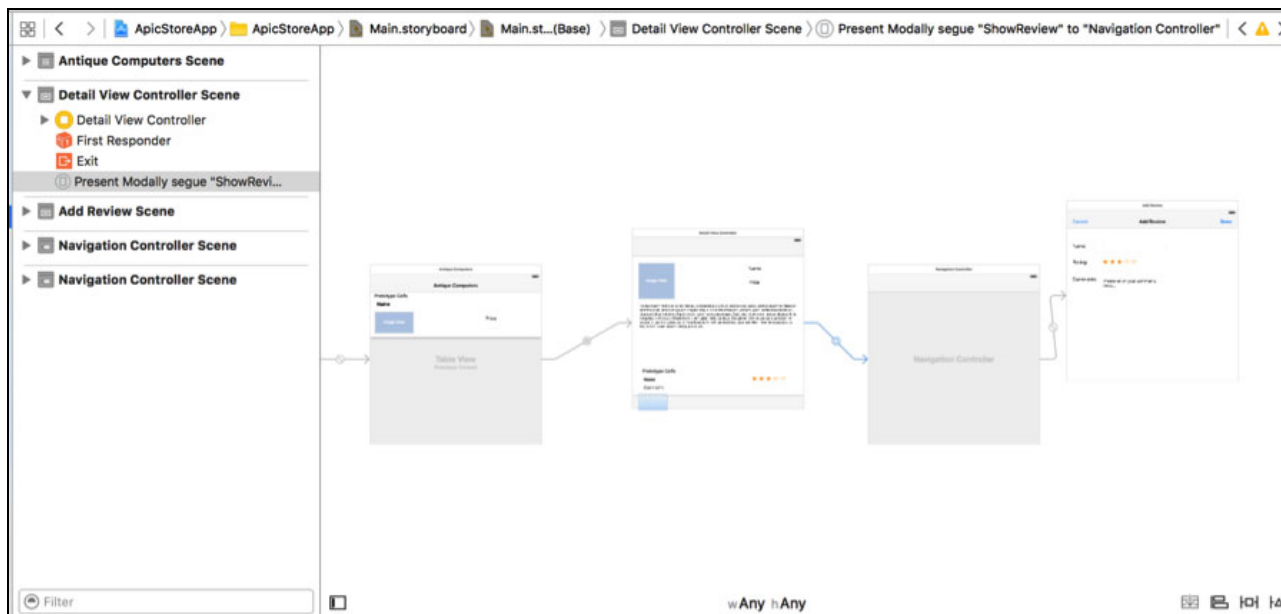


Figure 5-21 Mobile application storyboard

Although this section does not describe the process that used to create a basic iOS application in Swift, the typical structure from an iOS application follows a model–view–controller (MVC) style framework in which a view (as shown in Figure 5-21) includes an associated controller. The following main controllers are featured in this application:

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

5.3 Running the sample consumer web application

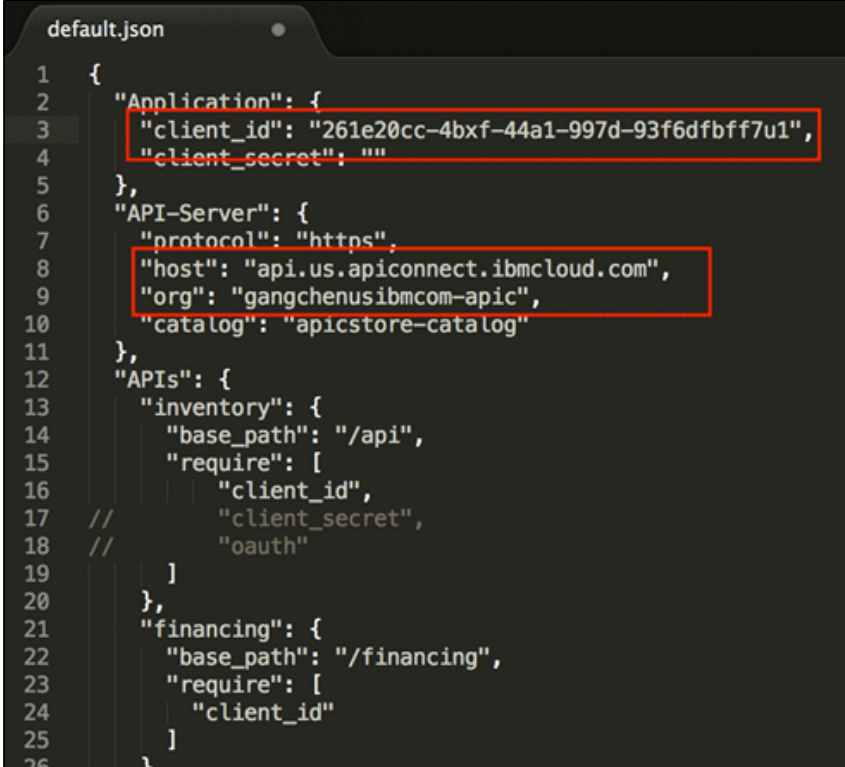
In this section, we describe how the developers can create a sample application by using the APIs that are published by the API developer. The consumer web application provides the basic function to allow user to browse the Inventory items. The sample web application is built as a Node.js application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

Complete the following steps:

1. You must configure the web application to point to the correct IBM API Connect endpoints for the APIs that you published. Locate the folder where you downloaded the git projects. Edit the `/redp5350-apiconnect-sample/StoreWebApp/config/default.json` file.

You must update the following entries, as shown in Figure 5-22:

- `client_id`
- `host`
- `org`



```
1  {
2    "Application": {
3      "client_id": "261e20cc-4bxf-44a1-997d-93f6dfbff7u1",
4      "client_secret": ""
5    },
6    "API-Server": {
7      "protocol": "https",
8      "host": "api.us.apiconnect.ibmcloud.com",
9      "org": "gangchenusibmcom-apic",
10     "catalog": "apicstore-catalog"
11   },
12   "APIs": {
13     "inventory": {
14       "base_path": "/api",
15       "require": [
16         "client_id",
17         "client_secret",
18         "oauth"
19       ]
20     },
21     "financing": {
22       "base_path": "/financing",
23       "require": [
24         "client_id"
25       ]
26     }
27   }
28 }
```

Figure 5-22 `default.json` file

2. Save the `default.json` file.
3. Browse to the following local git directory:

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

4. Start the Web application by running the **npm start** command. The Node.js application starts and web browser opens with home page of the sample application, as shown in Figure 5-23.

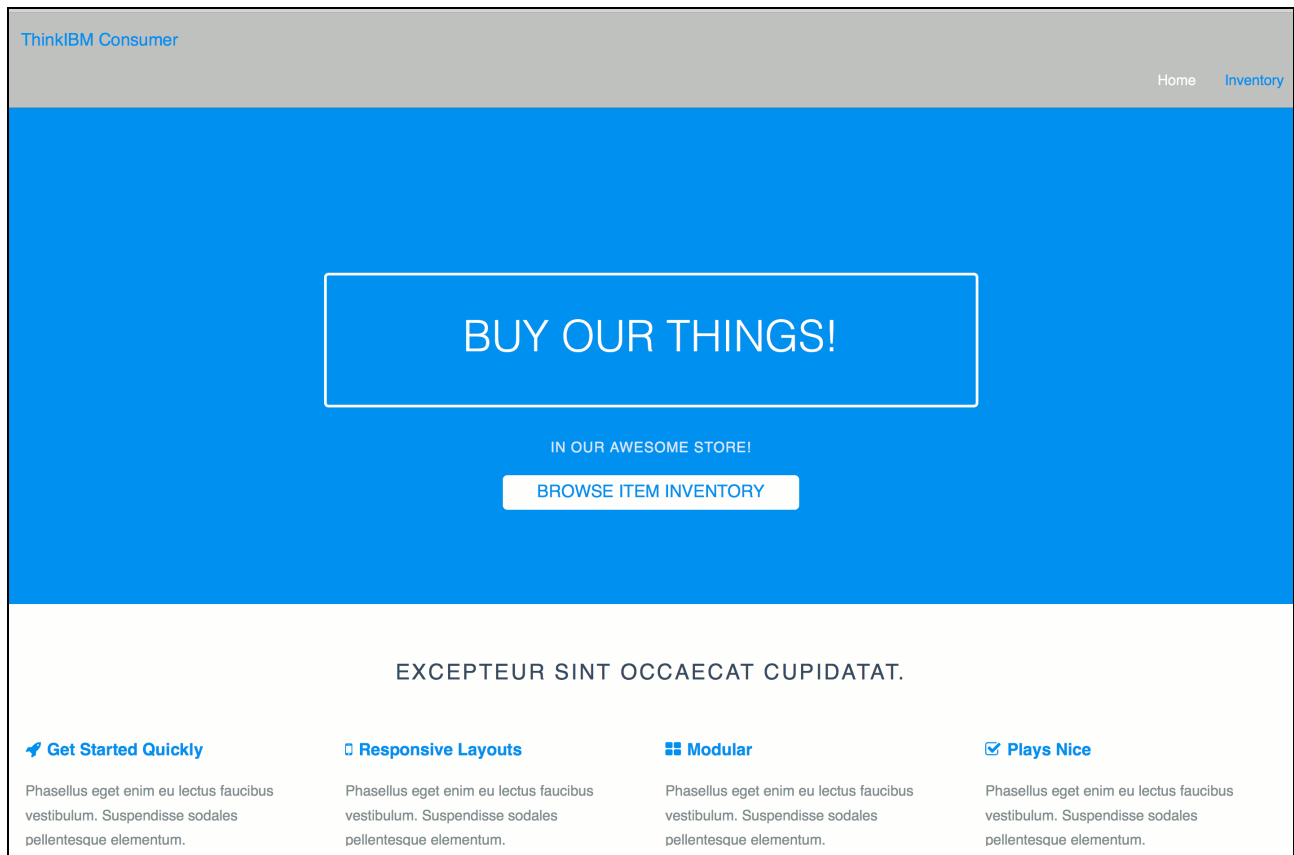


Figure 5-23 Main window for the Inventory application

5. Click **Browse Item Inventory**. You see a list of items, as shown in Figure 5-24.

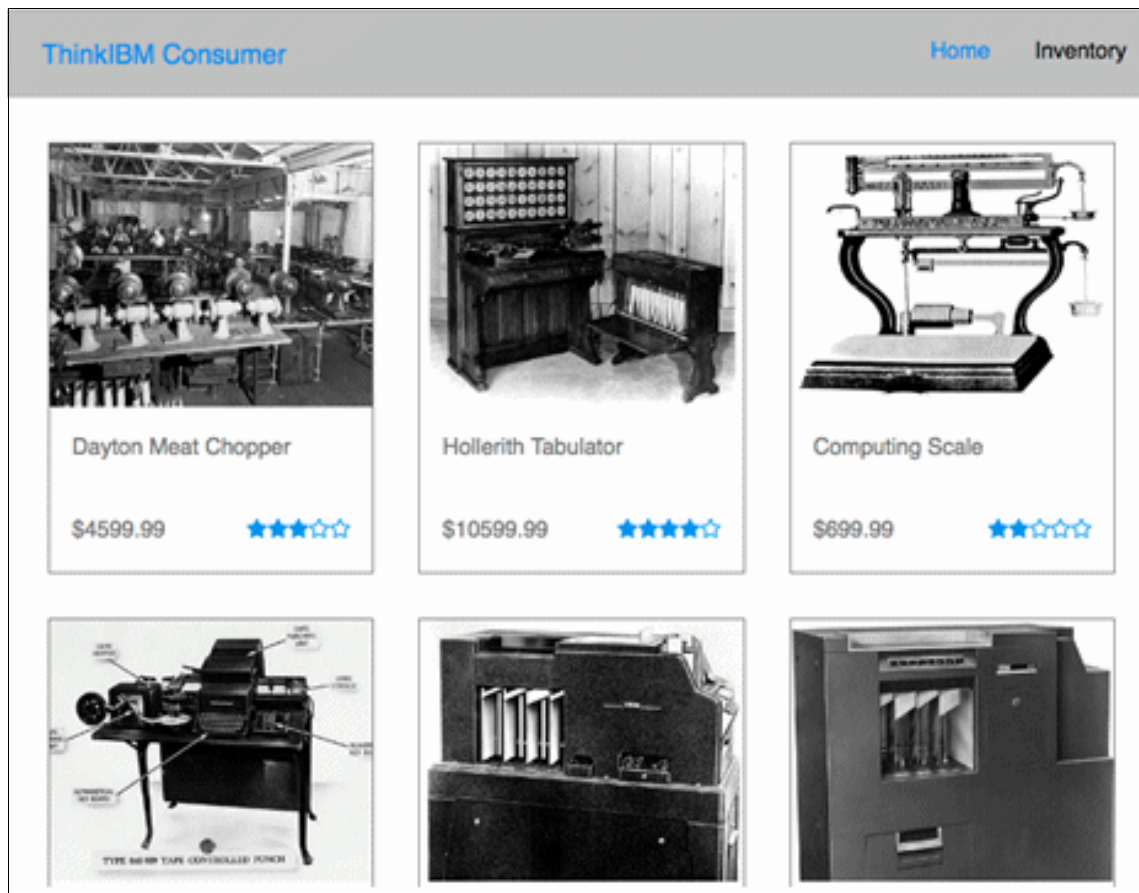


Figure 5-24 List of items

6. Clicking any of the items brings you to the Product Description page, as shown in Figure 5-25.

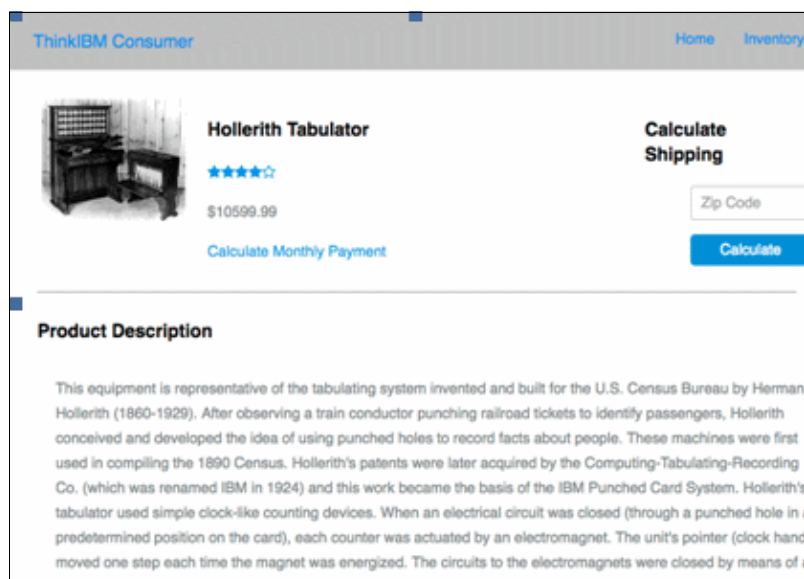


Figure 5-25 Product Description page

You now successfully used the Inventory and SocialReview APIs in a mobile and web application.

5.4 Summary

In this chapter, we showed how an application developer can use the APIs that are provided by a third party in the development of their applications within the context of two scenarios: One for mobile developers and the other for web developers.



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.

This chapter includes the following topics:

- ▶ 6.1, “Introduction to IBM API Connect Analytics” on page 102
- ▶ 6.2, “IBM API Connect dashboards” on page 102
- ▶ 6.3, “Customizing visualizations and dashboards” on page 109
- ▶ 6.4, “Sharing dashboards” on page 115
- ▶ 6.5, “Summary” on page 117

6.1 Introduction to IBM API Connect Analytics

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

Tip: For more information about the activity-log policy and how to disable logging, or enrich logs with header or payload information, see the following IBM Knowledge Center website:

<https://ibm.biz/BdrAH3>

By using Kibana visualizations, you can explore information, such as which API, Products, Plans, and operations are most popular, response times, and success rates. Kibana visualizations in IBM API Connect can provide insights that can affect business policy or help manage the health of the system.

IBM API Connect users with authorized roles can access the IBM 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 show you where to find the default dashboards and visualizations that are included with IBM API Connect. We also show some examples of how to customize visualizations and dashboards and share them.

For more information about Kibana, see this website:

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

6.2 IBM API Connect dashboards

In this section, we review the default dashboards in IBM API Connect. Analytics data is available in the API Manager console. Complete the following steps to review the dashboards:

1. Log in to API Manager.
2. Browse to the dashboard and then select **APicStore Catalog**. All IBM API Connect analytics are strictly scoped to a catalog. This configuration supports isolation between catalogs so that only authorized viewers can access analytics data for a catalog.
3. Browse to the **Analytics** tab, as shown in Figure 6-1.

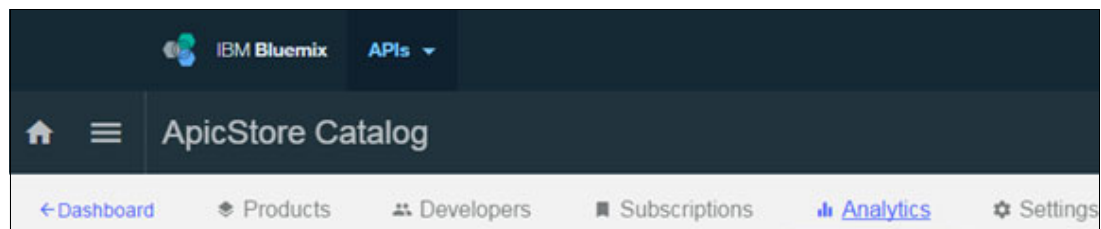


Figure 6-1 IBM API Connect Analytics dashboard search bar

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

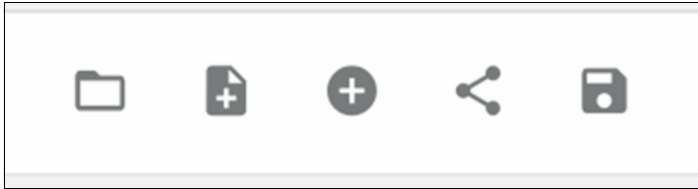


Figure 6-2 Dashboard configuration icons

4. Select the first icon, which resembles a folder. A list of the included dashboards opens, as shown in Figure 6-3. You can select any of these dashboards to review them, but they are not automatically filtered to a specific API, plan, or product.

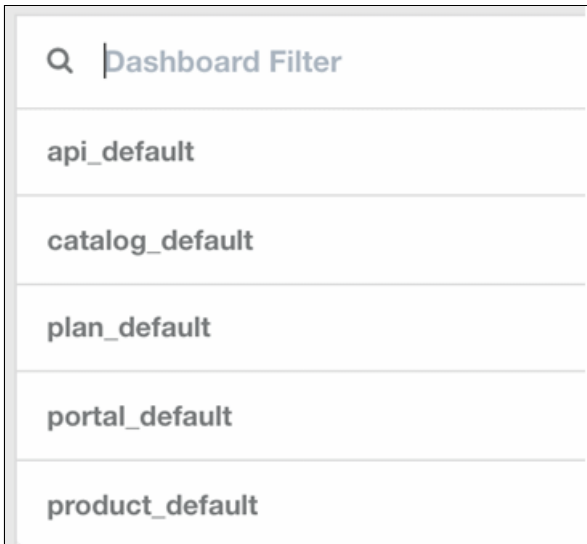


Figure 6-3 List of included dashboards

In the following sections, we show you how to browse and explore these dashboards in API Manager and the Developer Portal.

6.2.1 Default catalog dashboard

Browse to the Analytics tab. The default catalog dashboard opens. It includes the following visualizations (as shown in Figure 6-4 on page 104):

- ▶ 5 Most Active Products
- ▶ 5 Most Active APIs

If calls were not made, the visualizations are empty.

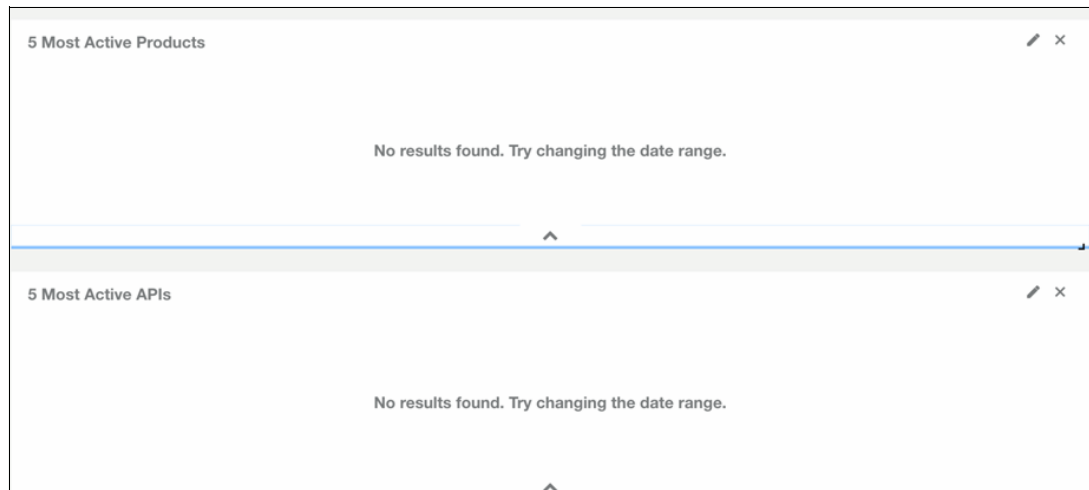


Figure 6-4 Default catalog dashboard

5. See the suggestion to change the date range. By default, the API events in the visualizations are scoped to the last seven days. If you click **Last 7 days** at the upper right, you see a set of date and time ranges. As you select these ranges, the visualizations are updated to reflect the new range, as shown in Figure 6-5.

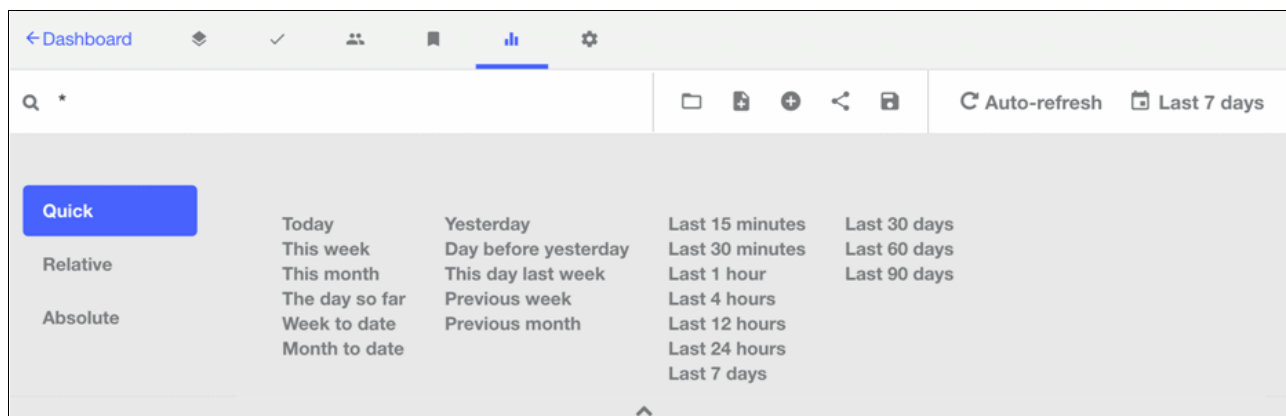


Figure 6-5 API events of last seven days

6. 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

Complete the following steps to review the default product dashboard:

1. Browse to the Product tab in the Catalog view, as shown in Figure 6-6.

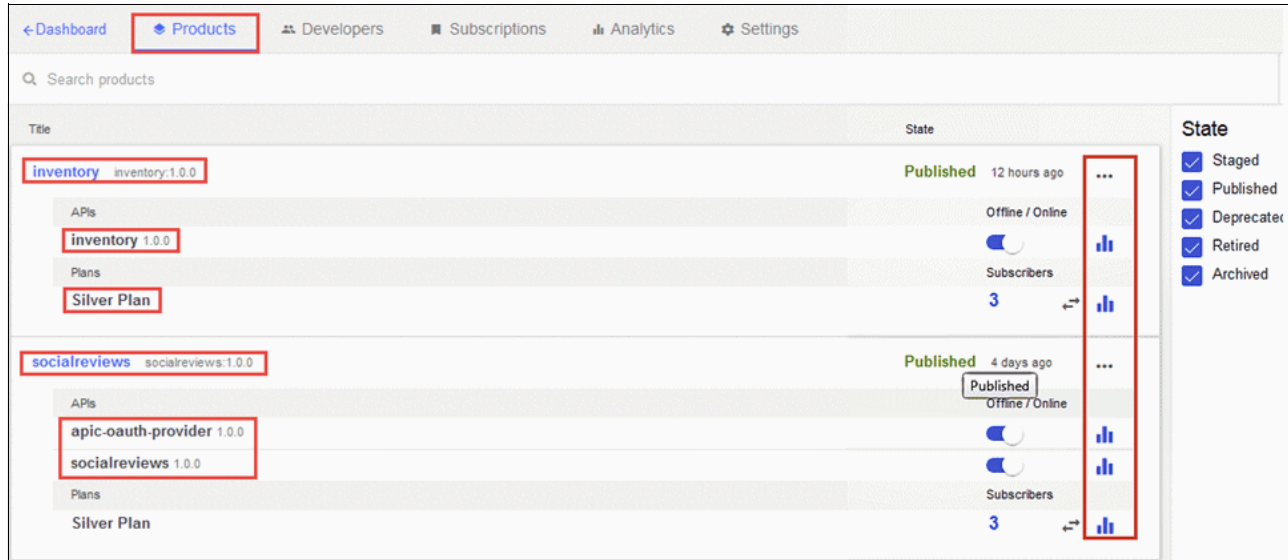


Figure 6-6 IBM API Connect Analytics access for Product, APIs, and Plans scope

2. Select the ellipses (...) at the upper right. Then, select **Product analytics**, as shown in Figure 6-7.

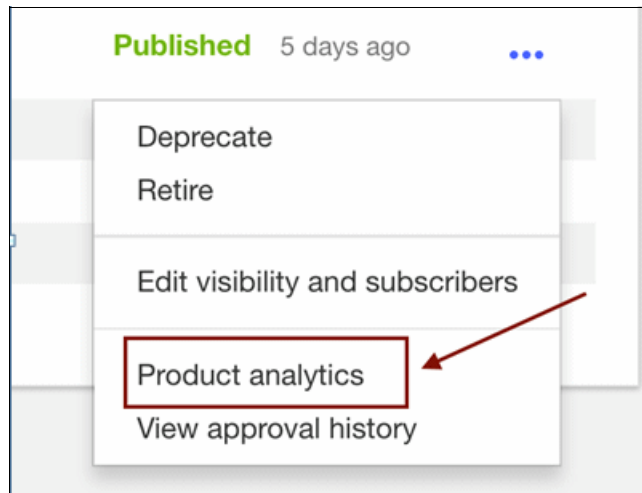


Figure 6-7 Select Product analytics

The default product dashboard opens and is scoped to the selected Product. The following visualizations are shown in this dashboard:

- The count of Developer Organizations that feature applications that are subscribed to Plans for this Product.
- The number of API calls that were 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 other 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 that are subscribed to that plan, (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 application that is represented by green made the most calls and the application that is represented by light blue made the fewest calls.

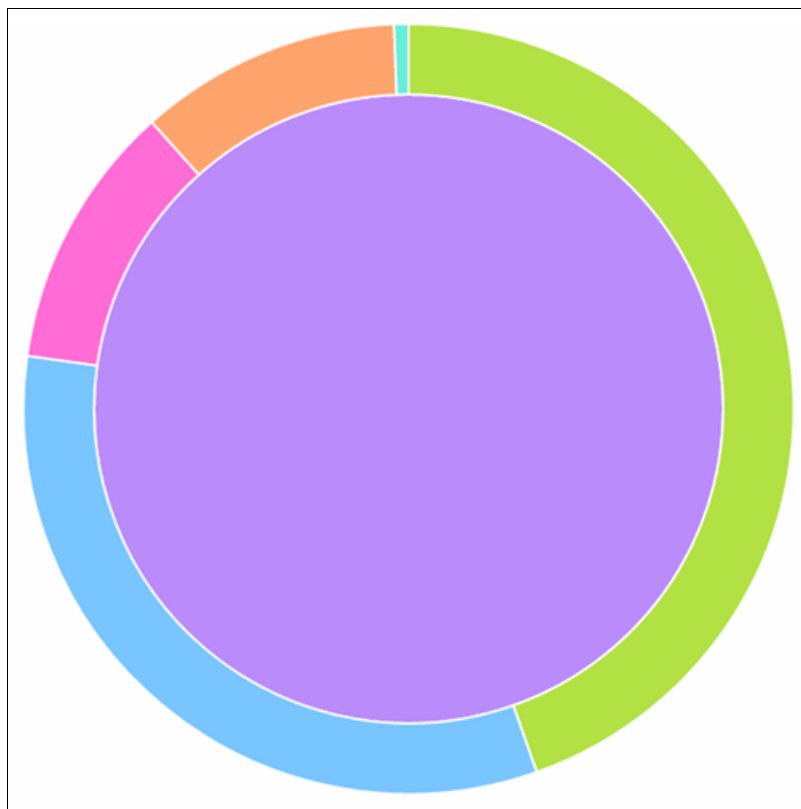


Figure 6-8 Applications that are subscribed to each plan

- A bar chart that shows the number of API calls per day.

3. Hover over a bar to see more information, as shown in Figure 6-9.

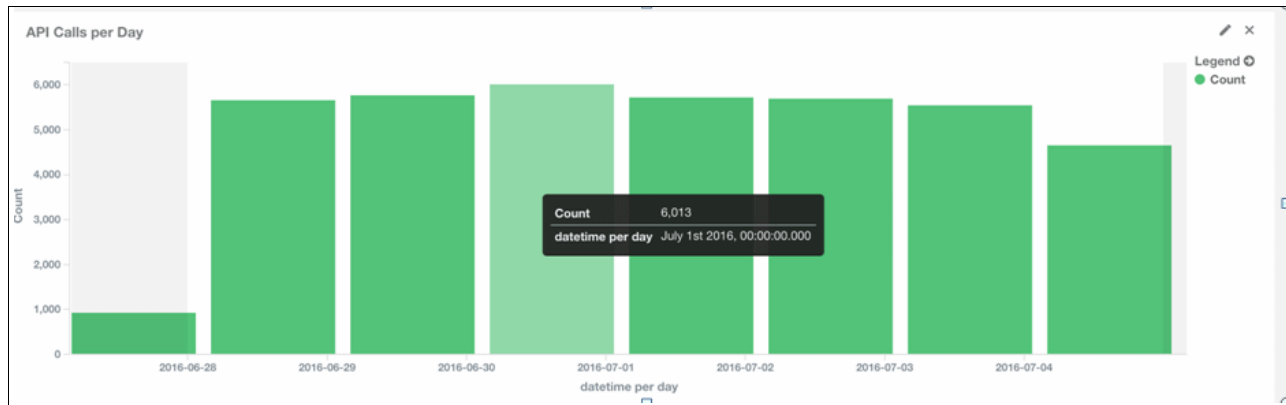


Figure 6-9 Details view of API calls per day

This dashboard gives you high-level visibility into the use of your Product. It can answer the following questions:

- 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

Complete the following steps to review the default API dashboard:

1. Return to the Product View in the catalog. Select the **graph icon** that is next to the Inventory API under the Inventory Product, as shown in Figure 6-10.



Figure 6-10 Select the graph icon

The Default API dashboard opens, which features the following visualizations:

- A pie chart that shows the HTTP Status codes that were 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 reflect error codes only.
- A visualization that shows the minimum response time for all calls to the Inventory API in the specified time.
- 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. An example of visualizations 3, 4, and 5 is shown in Figure 6-11 on page 108. 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. These figures might reflect differences in specific endpoints or a problem that is causing occasional slow response times.

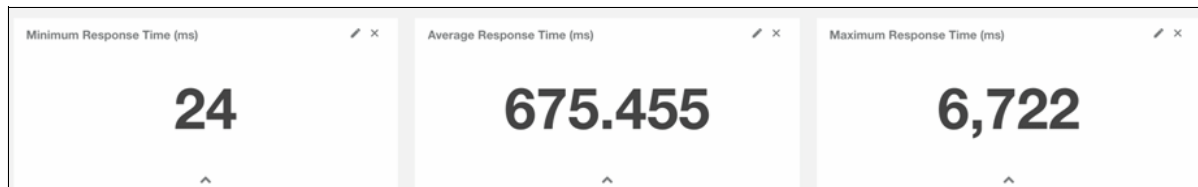


Figure 6-11 Maximum response time for all calls

- 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 the following questions:

- What are the successes 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

Complete the following steps to review the default plan dashboard:

1. Select the **analytics** icon that is next to the Silver plan under Inventory Product. The default plan dashboard features the following visualizations:
 - The number of API calls that were made by applications that are subscribed to the Silver plan.
 - The number of applications that are subscribed to the Silver plan.
 - A stacked graph that shows the maximum rate limit and rate limit count for each application over time. This visualization can be used to show if applications are approaching rate limits. If your plan is an unlimited plan, this visualization is empty.

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

An example of the Success rate graph is shown in Figure 6-12 on page 109. You can see that there was a spike in usage and that the failure rate remains constant, although there were a higher percentage of failures at the start.

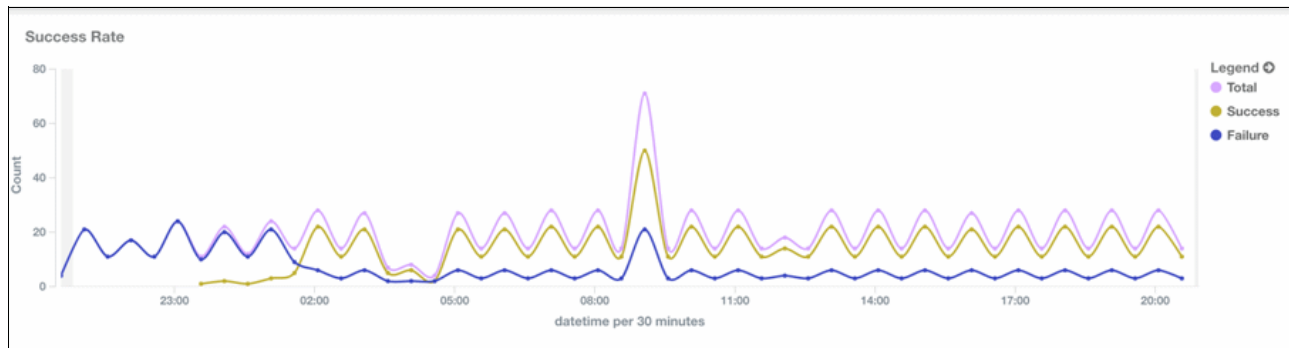


Figure 6-12 Success rate graph

For more information about Portal Analytics, see this IBM Knowledge Center website:

<https://ibm.biz/BdrAjx>

6.3 Customizing visualizations and dashboards

Now that you explored the dashboards and visualizations, maybe you noticed some data that you want to review more deeply to get further insights; for example, the response times on the API dashboard featured a large range, from 24 ms to almost 7 seconds. Maybe you spotted many error response codes. In both cases, you might want to obtain more information about which operations were affected.

Complete the following steps:

1. Return to the API dashboard for the Inventory API. Browse to the Errors visualization in the upper right and select the **pencil icon** to edit this visualization, as shown in Figure 6-13.

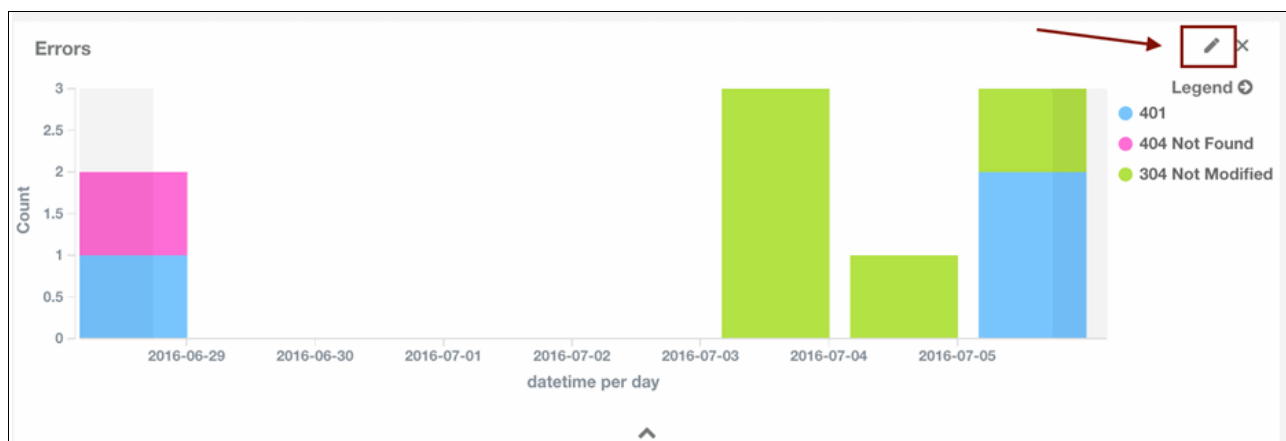


Figure 6-13 Errors visualization

2. The Kibana visualization editor opens. You can make changes here and then click **Play** to check the results. You can then save this visualization as new, as shown in Figure 6-14 on page 110.

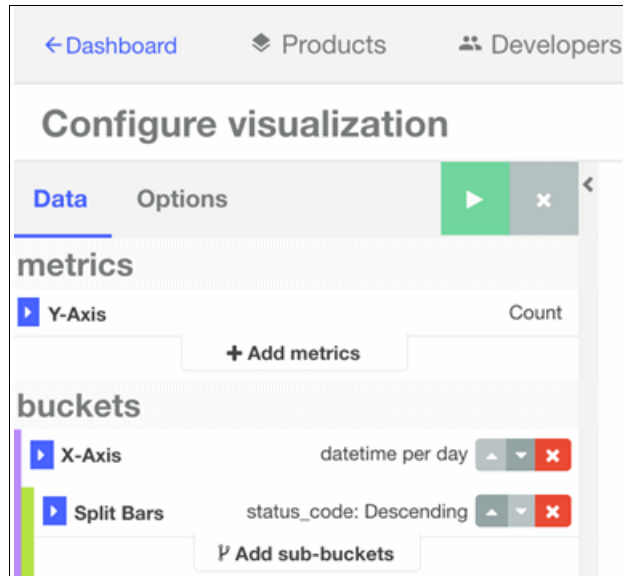


Figure 6-14 New visualization

3. Modify this visualization to show error codes by individual resource. Select **Add sub-buckets**. Then, choose the default **Split Chart**, as shown in Figure 6-15.

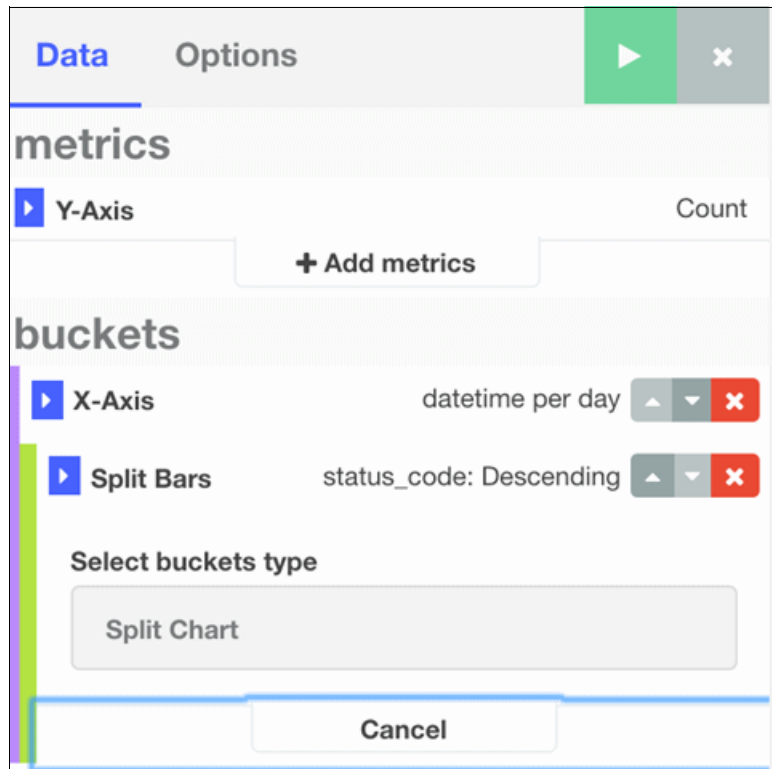


Figure 6-15 Modifying the visualization to show error codes

4. Make the selections that are shown in Figure 6-16 on page 111. You want to choose a Sub Aggregation that uses “Terms” where the field is `resource_id`. This field is the identifier for each resource or operation. Within the aggregation, you order by `metric:Count`.

Split Chart

Rows
Columns

Sub Aggregation

Terms

Field

resource_id

Order By

metric: Count

Order
Descending

Size
5

Figure 6-16 Sub Aggregation choice

Tip: For more information about all of the IBM API Connect terms, see this IBM Knowledge Center website:

<https://ibm.biz/BdrAjH>

- If you click the green **Run** button at the top of the window, you see an example of the resulting visualization, as shown in Figure 6-17. What you see depends on how many calls you have made and what the resulting responses are. You should see individual operations that are listed along the x-axis and then further divided by time. The bars are stacked, which shows 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 started the editor.

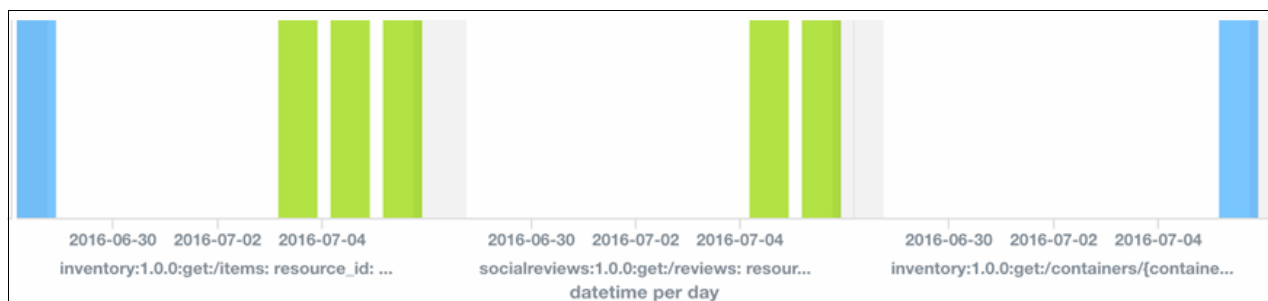


Figure 6-17 Resulting visualization

6. You might decide that you also want to see non-error status codes in this view. If so, return to the editor on the left. Under X-Axis, click **Split Bars** → **Advanced**. Exclude and Include Patterns can be specified here. Under JSON Input, you see `{"exclude" : "2."}`. This instruction is telling the visualization to exclude all response codes that start with 2; that is, it is excluding successful responses. Delete this text, as shown in Figure 6-18.

X-Axis datetime per day

Split Bars

Sub Aggregation

Terms

Field

status_code

Order By

metric: Count

Order

Descending

Size

5

Advanced

Exclude Pattern

Exclude Pattern Flags (reference)

CANON_EQ
CASE_INSENSITIVE
COMMENTS
DOTALL

Include Pattern

Include Pattern Flags (reference)

CANON_EQ
CASE_INSENSITIVE
COMMENTS
DOTALL

JSON Input

`{"exclude" : "2."}` Delete

Add sub-buckets

Figure 6-18 Deleting the text

7. After running the visualization again, you might see more operations across the bottom of the window. These operations are listed because those operations that were empty were not displayed previously.

Now, you might see that some operations are running at 100% success rate, while others failed 100% of the time. You can investigate to determine whether this issue is a usage error, a defect in the implementation of the operation, a downstream outage, or some other issue, as shown in Figure 6-19 on page 113.

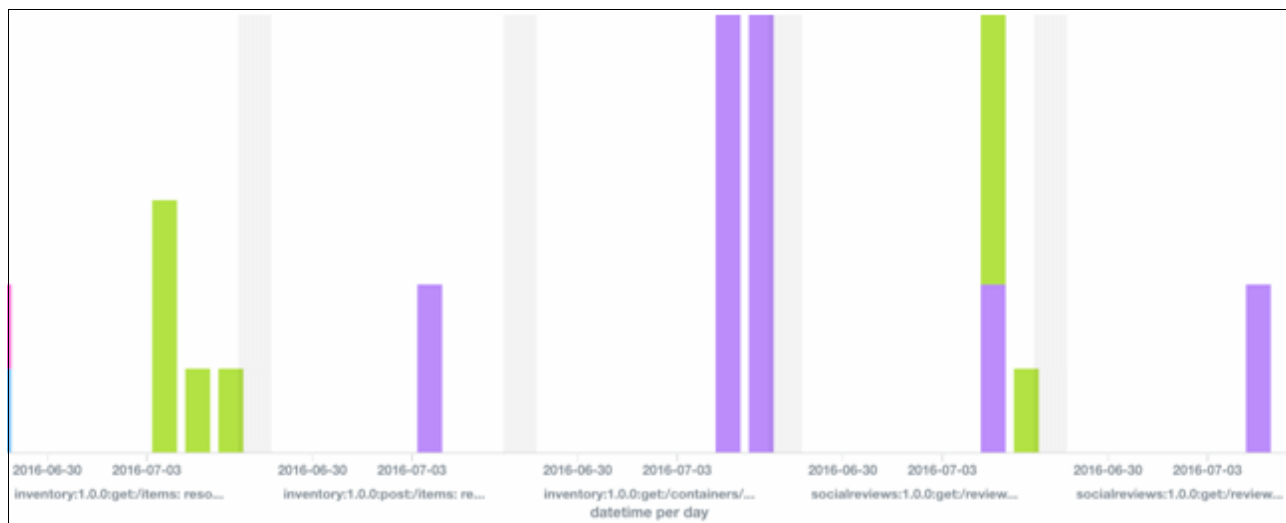


Figure 6-19 Same view with non-error status codes

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

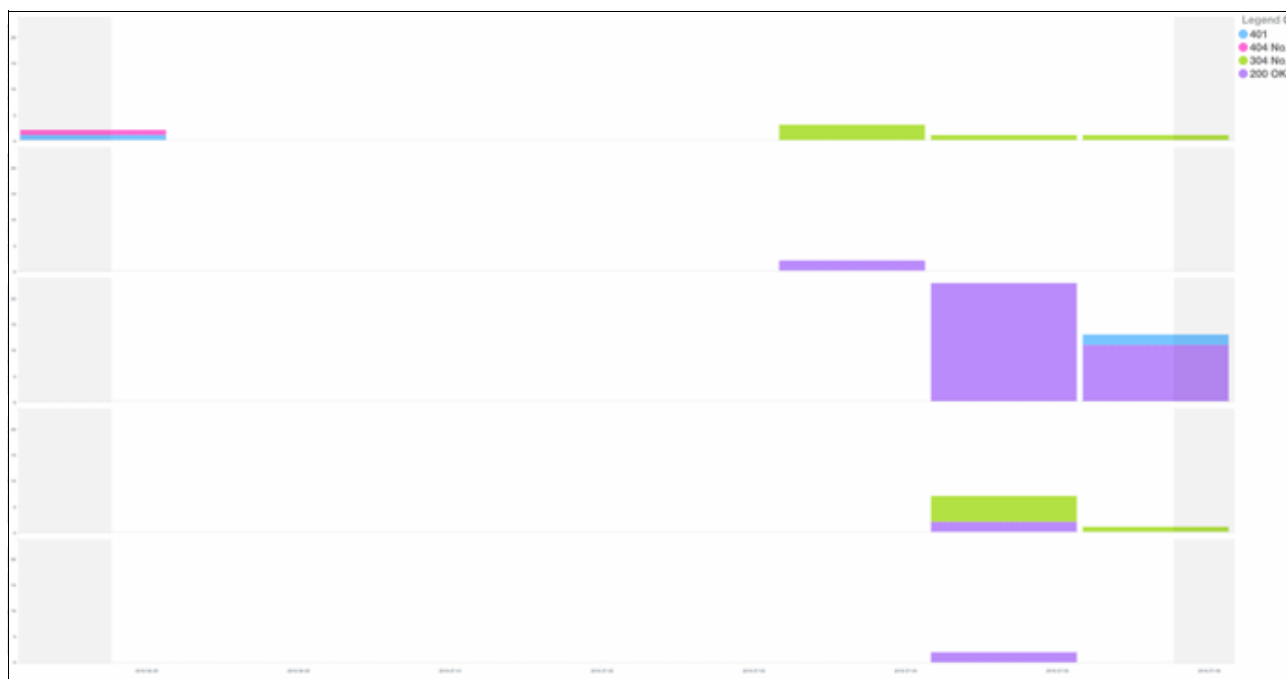


Figure 6-20 Subaggregation on the y-axis

9. If this visualization looks the way you want and is useful, it must be saved. Click the **Save** icon at the upper right of the visualization, as shown in Figure 6-21 on page 114.

Important: Enter a new name for the visualization so that you do not replace the default Errors visualization.

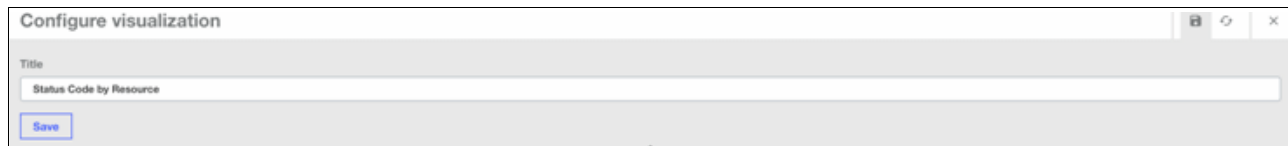


Figure 6-21 Configure visualization

10. Click **Save** to save this visualization. Then, click **X** at the upper right to close the editor. You are returned to in the Product Dashboard. Add our new Visualization to this Dashboard by clicking 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 do not see your new visualization in the list, start entering the name in the Visualization Filter field, as shown in Figure 6-23.

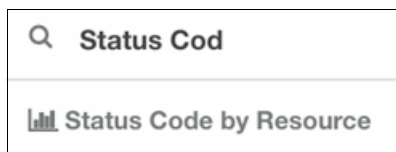


Figure 6-23 Entering the name in the Visualization Filter field

12. Select your visualization. Scroll down and you see your visualization on the dashboard, as shown in Figure 6-24.



Figure 6-24 Your visualization on the dashboard

13.If the visualization is not sized or in the area that you want, you can resize and move it by clicking the corners and dropping it, as shown in Figure 6-25.

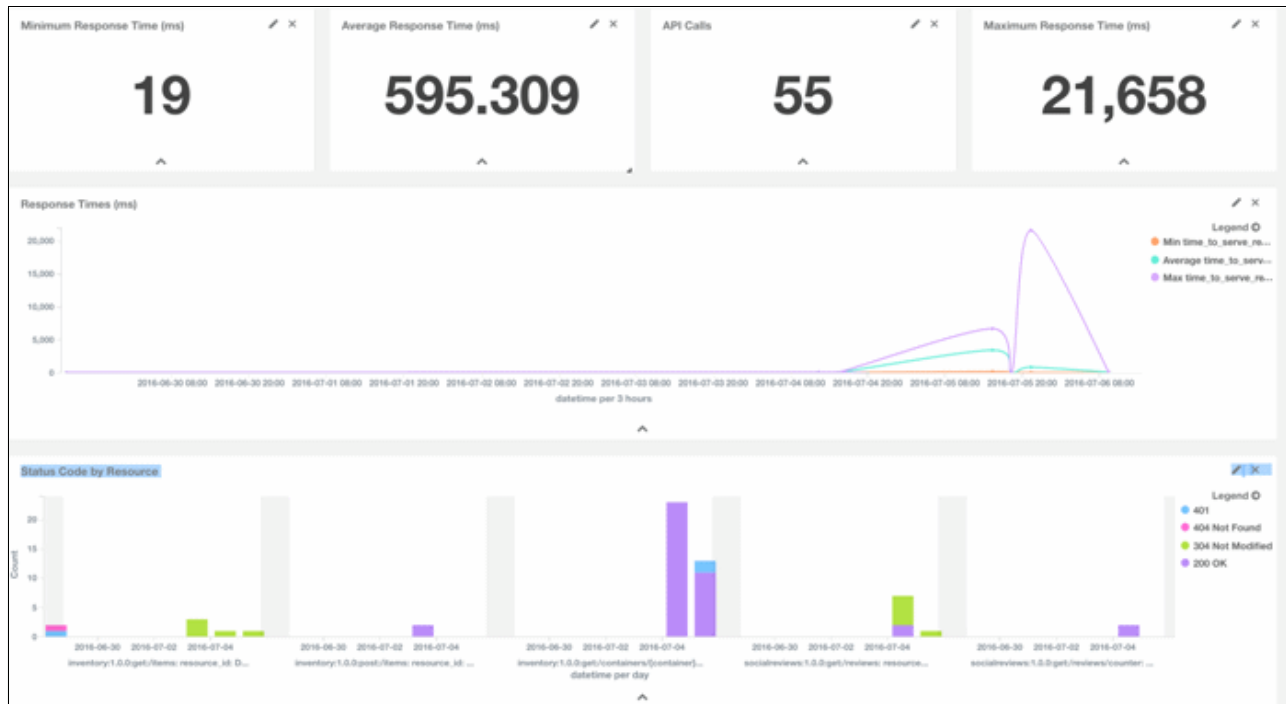


Figure 6-25 Resizing the visualization

You can now save the Dashboard so that it always includes this visualization. Select the **Save** icon on the upper right. You can enter a name for this Dashboard to create a new dashboard. Alternatively, you can save this default dashboard.

Tip: If you save the default dashboard with changes, you cannot reset to the original default. For this reason, it is recommended that you enter a new name for the customized dashboard.

6.4 Sharing dashboards

For a user to access visualizations, they must be authorized in two places. Complete the following steps to grant this authorization:

1. In API Manager, browse to Admin, and then, to the Roles view. A user must have a Role that is permitted to “View catalog analytics”. You can create a custom role that is specialized for Analytics access, as shown in Figure 6-26 on page 116.

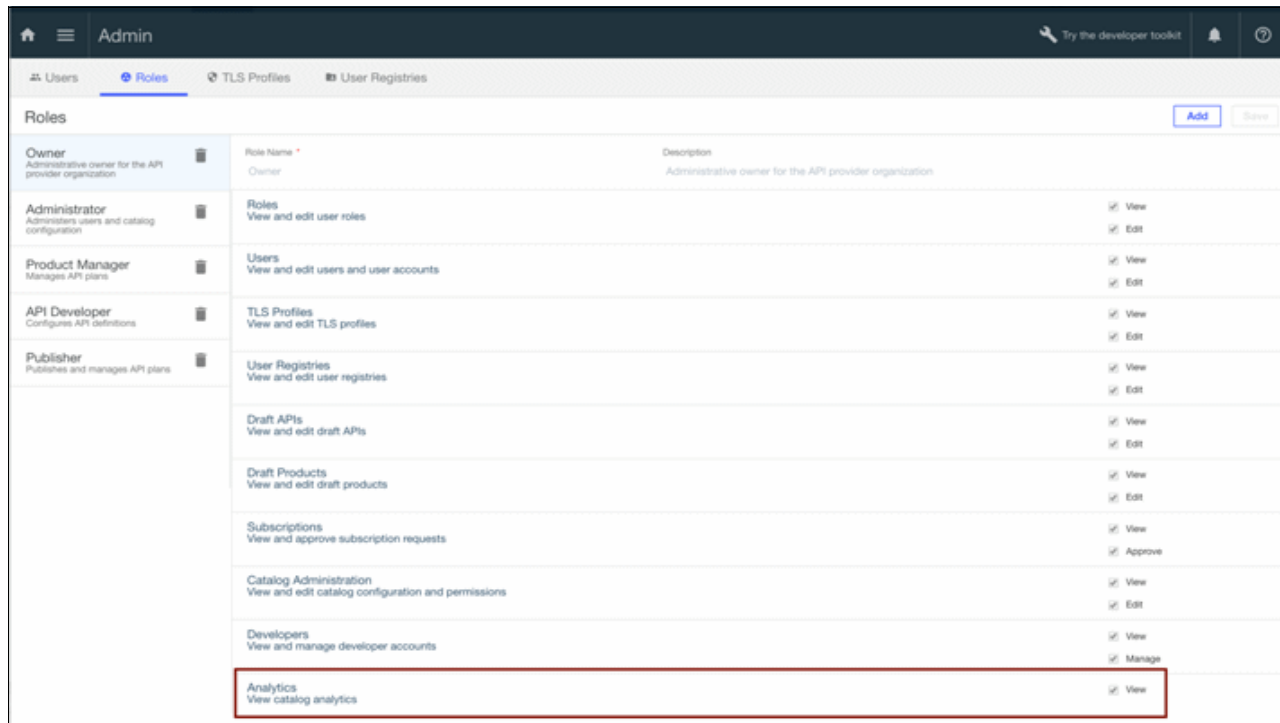


Figure 6-26 Roles view

2. Click **ApicStore Catalog** → **Settings** → **Permissions** (see Figure 6-27).

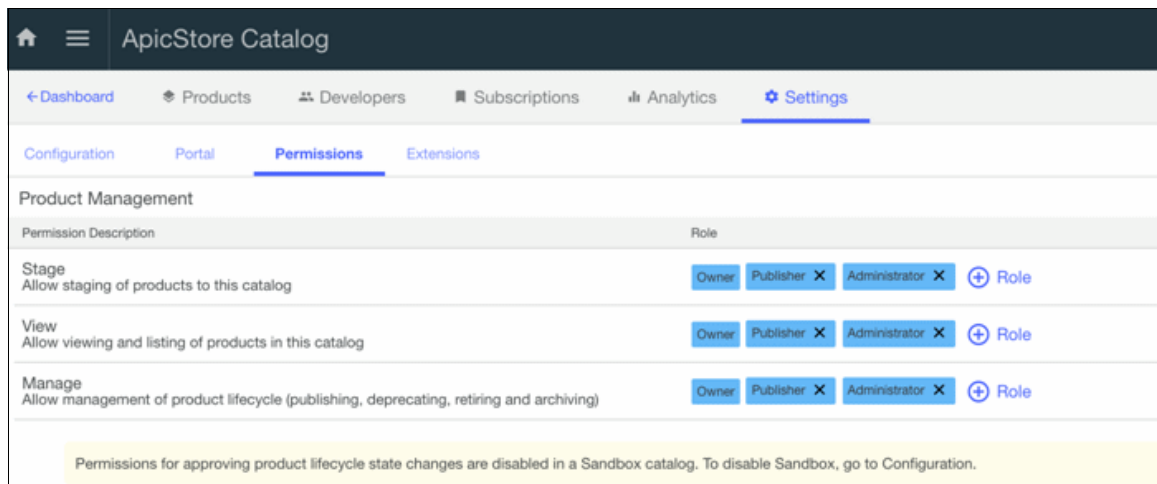


Figure 6-27 Permissions view

3. Confirm that the role of the user is listed in “View”. Now that the permissions were confirmed, we need to browse to the dashboard that we want to share.
4. Click the **link** on the dashboard configuration, as shown in Figure 6-28.

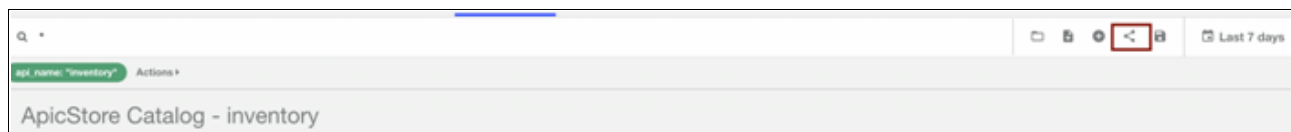


Figure 6-28 Clicking the link icon

5. You see two large strings. One string is an iframe that includes the dashboard, which can be merged into a larger dashboard.

Alternatively, you can copy the second string (which is a link) and share it. When you paste the link in a browser, you are prompted for your IBM API Connect credentials. If you are not authorized to view IBM API Connect analytics, you are not granted access.

6.5 Summary

In this chapter, we explored the visualizations and dashboards that are included with IBM API Connect. We described how these visualizations and dashboards can help your business understand the usage and health of your APIs.

We also showed how to customize visualizations and dashboards to study specific aspects of API usage. Finally, we showed 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 that is described in Chapter 4, “Securing the APIs” on page 65. To deploy this application, you use the cloud foundry command line.

In this appendix, we described how to log in and use the authentication utility application.

Deploying the authentication utility application

Complete the following steps to deploy the authentication utility application. For more information about downloading the Cloud Foundry CLI package and other plug-ins, see this website:

<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`. For more information, see “Cloning the GitHub repository” on page 11. In the following instructions, replace `D:\APIM50\Redbooks\redp5350` with the directory that you use in your setup.

2. Browse to the following directory:

```
cd D:\APIM50\Redbooks\redp5350\0Auth\authentication-app
```

3. Enter `ls`. You see the following files:

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

4. Enter `bluemix api https://api.ng.bluemix.net`. You receive the following output (in this example, we use the user and organization `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 the `cf push authentication-app` command. You see the following output:

```
Using manifest file
D:\APIM50\Redbooks\redp5350\0Auth\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 also are shown in Figure A-1 on page 121 and Figure A-2 on page 122.

```

D:\APIM50\Redbooks\redp5350\0Auth\authentication-app>ls
app.js manifest.yml package.json public

D:\APIM50\Redbooks\redp5350\0Auth\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:           buperepa@us.ibm.com
Org:            buperepa@us.ibm.com
Space:          dev

D:\APIM50\Redbooks\redp5350\0Auth\authentication-app>cf push authentication-app
Using manifest file D:\APIM50\Redbooks\redp5350\0Auth\authentication-app\manifes
t.yml

Updating app authentication-app in org buperepa@us.ibm.com / space dev as bupere
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\0Auth\authentication-app
Uploading 8.1K, 7 files
Done uploading
OK

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

Starting app authentication-app in org buperepa@us.ibm.com / space dev as bupere
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: Part 1

```

D:\API50\Redbooks\redp5350\0Auth\authentication-app>ls
app.js  manifest.yml  package.json  public

D:\API50\Redbooks\redp5350\0Auth\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:            buperepa@us.ibm.com
Org:             buperepa@us.ibm.com
Space:          dev

D:\API50\Redbooks\redp5350\0Auth\authentication-app>cf push authentication-app
Using manifest file D:\API50\Redbooks\redp5350\0Auth\authentication-app\manifes
t.yml

Updating app authentication-app in org buperepa@us.ibm.com / space dev as bupere
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:\API50\Redbooks\redp5350\0Auth\authentication-app
Uploading 8.1K, 7 files
Done uploading
OK

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

Starting app authentication-app in org buperepa@us.ibm.com / space dev as bupere
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: Part 2

The URL to test the application is shown in Figure A-3. We use this link to test the application. Enter the following URL in a browser:

<http://authentication-app.mybluemix.net/login.html>

```

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
    ├── vary@1.0.1
    ├── content-disposition@0.5.1
    ├── fresh@0.3.0
    ├── etag@1.7.0
    ├── range-parser@1.0.3
    ├── path-to-regexp@0.1.7
    ├── depd@1.1.0
    ├── qs@4.0.0
    ├── finalhandler@0.4.1 (unpipe@1.0.0)
    ├── on-finished@2.3.0 (ee-first@1.1.1)
    ├── debug@2.2.0 (ms@0.7.1)
    ├── proxy-addr@1.0.10 (forwarded@0.1.0, ipaddr.js@1.0.5)
    ├── send@0.13.1 (destroy@1.0.4, statuses@1.2.1, ms@0.7.1, mime@1.3.4, htt
p-errors@1.3.1)
    ├── type-is@1.6.13 (media-typer@0.3.0, mime-types@2.1.11)
    ├── accepts@1.2.13 (negotiator@0.5.3, mime-types@2.1.11)
    └── serve-static@1.10.3 (send@0.13.2)
cfenv@1.0.3 node_modules/cfenv
├── ports@1.1.0
└── underscore@1.8.3
-----> Installing App Management
-----> Caching build
Clearing previous node cache
Saving 2 cacheDirectories (default):
- node_modules
- bower_components (nothing to cache)
-----> Build succeeded!
├── basic-auth@1.0.4
├── body-parser@1.15.2
├── cfenv@1.0.3
└── express@4.13.4
-----> Uploading droplet (16M)
0 of 1 instances running, 1 starting
1 of 1 instances running

App started

OK

App authentication-app was started using this command './vendor/initial_startup.
rb'

Showing health and status for app authentication-app in org buperepa@us.ibm.com
/ space dev as buperepa@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 16:37:01 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      det
#0 running  2016-07-23 02:39:49 PM  0.0%    67.3M of 512M  58.4M of 1G

D:\APIM50\Redbooks\redp5350\0Auth\authentication-app>

```

Figure A-3 Deploying the authentication utility application: Part 3

The login window that is shown in Figure A-4 opens.

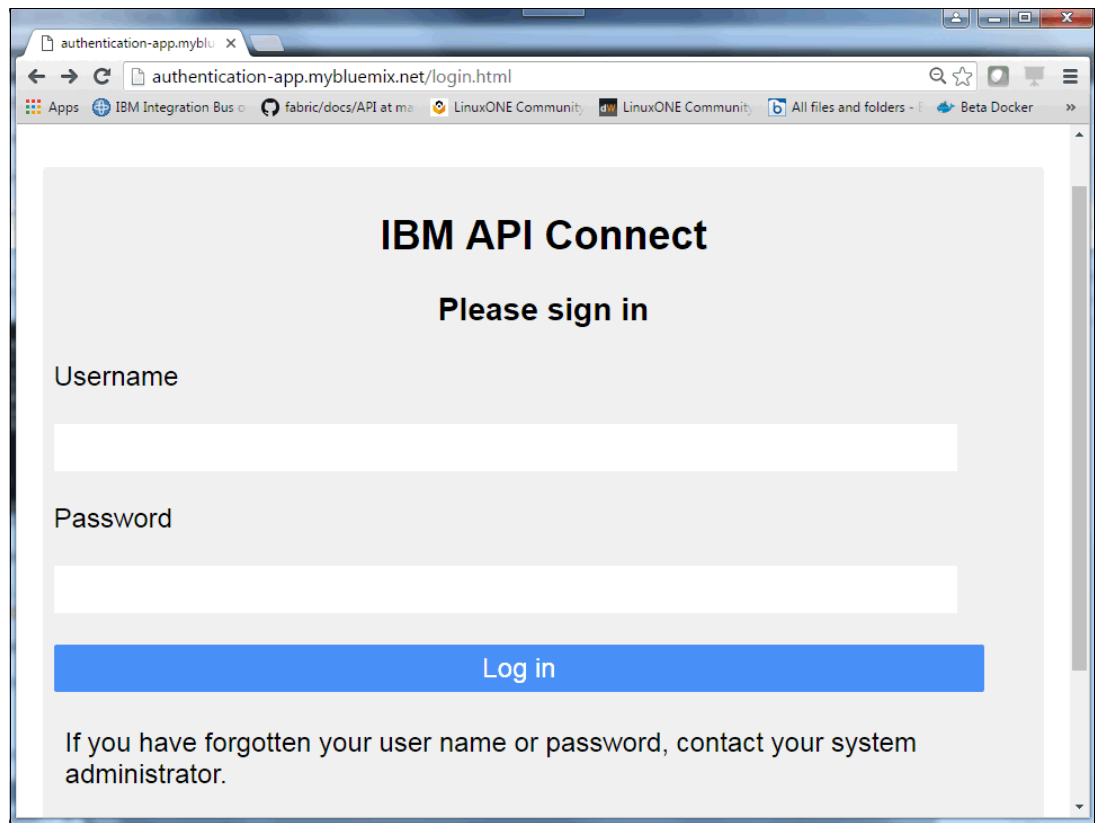


Figure A-4 Log in window

6. Test the deployed application by entering the following URL in a browser:

<http://authentication-app.mybluemix.net/grant.html>

The window that is shown in Figure A-5 opens.

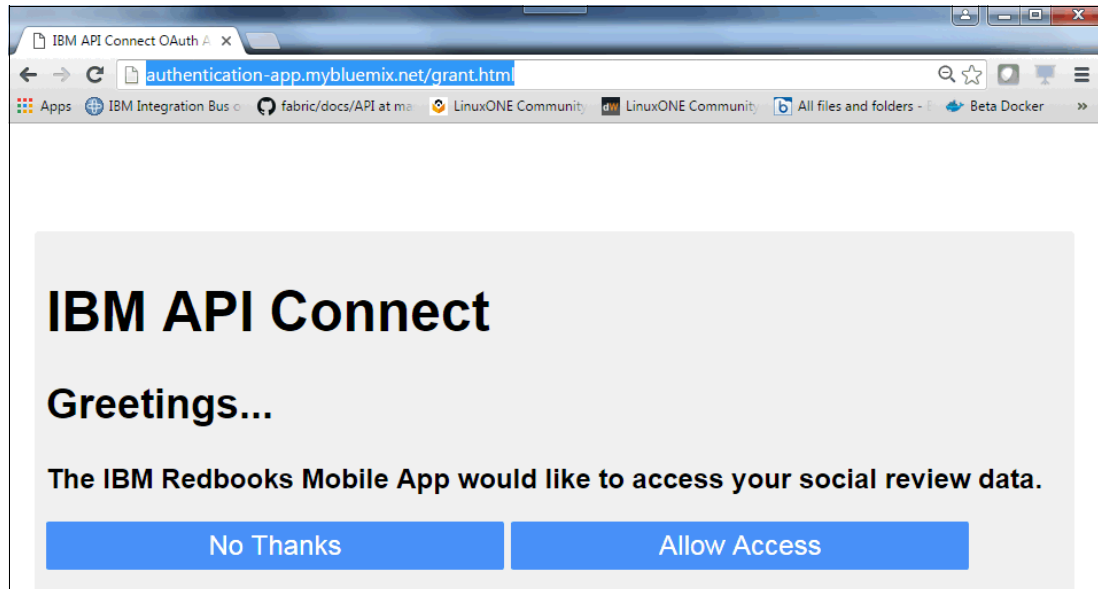


Figure A-5 Testing the deployed application

This window indicates that you successfully deployed the authentication utility application.



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 that is associated with this paper is available in softcopy on the Internet at the following IBM Redbooks Web server:

<ftp://www.redbooks.ibm.com/redbooks/REDP5350>

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

ibm.com/redbooks

At the Redbooks website, select **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 file:

| <i>File name</i> | <i>Description</i> |
|---------------------|-------------------------------------|
| 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 that are listed in this section are considered particularly suitable for a more detailed discussion of the topics that are covered in this paper.

IBM Redbooks

The following IBM Redbooks publications provide more information about the topic in this document. Some publications that are referenced in this list might be available in softcopy only:

- ▶ *Getting Started with IBM API Connect: Concepts and Architecture 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 other materials, at the following website:

ibm.com/redbooks

Online resource

The following IBM API Connect V5.0 IBM Knowledge Center website also is relevant as another information source:

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

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services



REDP-5350-00

ISBN 0738455547

Printed in U.S.A.

Get connected

