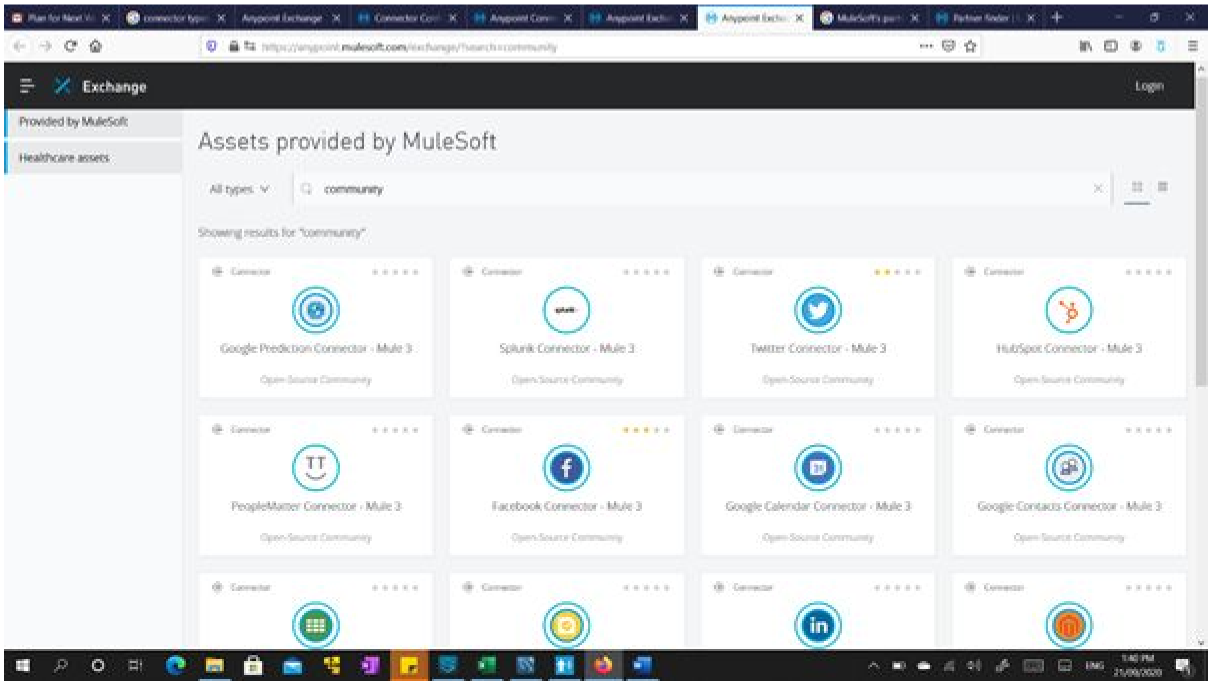
Module 3: Exchange

Anypoint Mulesoft offers a wide variety of Anypoint Connectors to provide a connection between Mule flows and external sources. Connectors facilitate easy integration with third-party applications.

3.1. Connector Types

3.1.1.Community

MuleSoft or members of the MuleSoft community write and maintain the Community connectors Connectors built by the community or MuleSoft are generally open-source, although each package may vary. Partner-built connectors may not be open-source. Contact the partner directly for more information. You do not need any special account or license to use a Community connector.
Developed by developer community


Exchange
Login
Provided by MuleSoft
Healthcare assets
Assets provided by MuleSoft
All types
community
Showing results for "community"
Google Prediction Connector - Mule 3
Splunk Connector - Mule 3
Twitter Connector - Mule 3
HubSpot Connector - Mule 3
PeopleMatter Connector - Mule 3
Facebook Connector - Mule 3
Google Calendar Connector - Mule 3
Google Contacts Connector - Mule 3
Open-Source Community

Examples of Community Connector:
Slack Connector - Mule 4
The Slack Connector enables organizations to connect directly with the Slack API, permitting users access to the Slack functionality with seamless integration.

Using this connector, businesses can create instant connectivity to popular collaboration, mobile, and social applications to streamline connectivity and integrate business processes.
Slack Connector is an easy and fast way to integrate with your organization's team chats, create notifications, automate responses, and much more

HubSpot Connector - Mule 3
The HubSpot connector enables instant API connectivity to numerous HubSpot APIs, allowing users to interface with HubSpot APIs to perform key functions without having to directly connect to the HubSpot platform.
Sync data and automate business processes between HubSpot and third-party CRMs, sales process management, mobile, and social applications. Increase ROI on marketing investments by integrating HubSpot with third party marketing automation SaaS applications and services via the MuleSoft Anypoint HubSpot connector. HubSpot offers SaaS inbound marketing solutions to aid companies in capturing leads and converting them to business opportunities.
With the Anypoint HubSpot connector, seamless integration between HubSpot with existing marketing automation, CRM, and social media applications such as Salesforce, Magento, NetSuite, and Twitter to further extend the reach of HubSpot is possible.
MuleSoft’s Anypoint Platform makes it simple for businesses to integrate sales, marketing, and social - giving organizations the information they need to drive better business.

Google Calendar Connector:The Anypoint Google Calendar connector provides instant API connectivity to the Google Calendar API, enabling third party applications to import, export, sync, search for, create, edit, and delete calendar events in real-time. Enhance the capabilities of Google Calendar by syncing third party applications and services through pre-built connectivity with the Anypoint Google Calendar connector from MuleSoft.

The Anypoint Google Calendar connector makes it simple to create seamless connectivity between Google Calendar and other third party applications and services such as Salesforce, Drupal, Google Spreadsheets, IBM, and Outlook. With the MuleSoft Anypoint™ Platform, businesses can extend the capabilities of their applications and create deeper integration with Google Calendar.

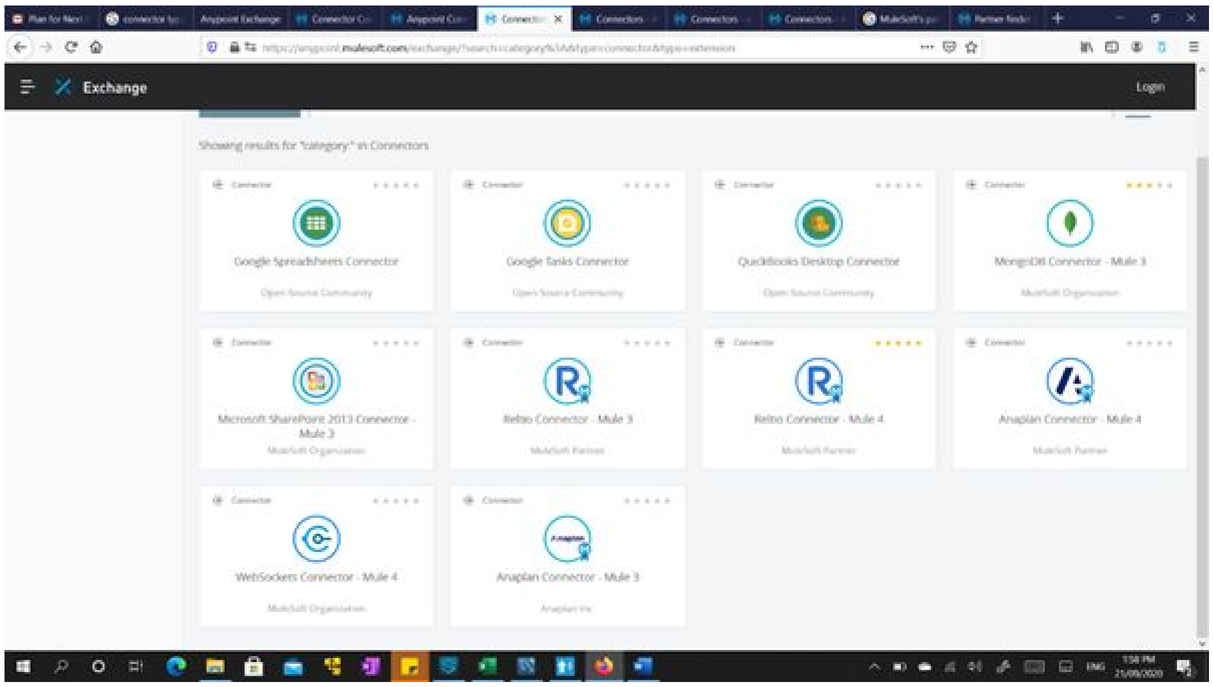
Splunk-Connector-Mule-3: The Splunk Enterprise Server Connector performs the most common tasks against the Splunk API in an easy and consistent way. This connector allows you to not only directly send data from Anypoint Platform into the Splunk Index, but also to perform a wide variety of searches and indexing operations.

Use the connector to work with saved searches, create and run real-time blocking and one-shot searches, retrieve data models, and create and modify data inputs on the fly.

About Community-Developed Connectors:This connector was developed by WhiteSky Labs, a MuleSoft partner, and reviewed by MuleSoft. MuleSoft disclaims any support obligation for Community Connectors.

3.1.2. Premium

MuleSoft maintains Premium connectors. We must purchase Premium connectors as add-ons to our subscription.
Developed by Mulesoft Organisation and certified by the Mulesoft partner.


Exchange
Login
Showing results for "category:" in Connectors
Google Spreadsheets Connector
Open Source Community
Google Tasks Connector
Open Source Community
QuickBooks Desktop Connector
Open Source Community
MongoDB Connector - Mule 3
MuleSoft Organization
Microsoft SharePoint 2013 Connector - Mule 3
MuleSoft Organization
Reltio Connector - Mule 3
MuleSoft Partner
Reltio Connector - Mule 4
MuleSoft Partner
Anaplan Connector - Mule 4
MuleSoft Partner
WebSockets Connector - Mule 4
MuleSoft Organization
Anaplan Connector - Mule 3
Anaplan Inc

Examples of Premium Connector:

AS2: MuleSoft's AS2 connector allows Anypoint customers to send and receive business transactions electronically with added security over HTTP or HTTPS protocols, from within their Mule Applications. The AS2 connector available within the Anypoint platform delivers below outcomes:

- Enables secured B2B communication with the trading partner community from the unified Anypoint platform; Organizations do not need to have personnel with different skill sets.
- Supports large file processing through the usage of streaming.
- Enables AS2 communications from the Cloud.

The AS2 Connector relies on S/MIME to offer encryption and signing capabilities protecting the document throughout its entire journey across the network through the use of digital signatures, providing confidence to a recipient that the sender is really who it claims to be. Enabling the AS2 Connector's full set of security features, helps provide non-repudiation of an origin and receipt that reduces or eliminates disagreements between business partners. The AS2 Connector provides asynchronous and compression features that reduce the risks of HTTP response timeouts and allow receivers to process documents at their own pace.

FTPS: This connector provides the same functionality as the standard FTP connector, but it adds secure connections over SSL.

To secure the connection, we set a TLS context in the connection:

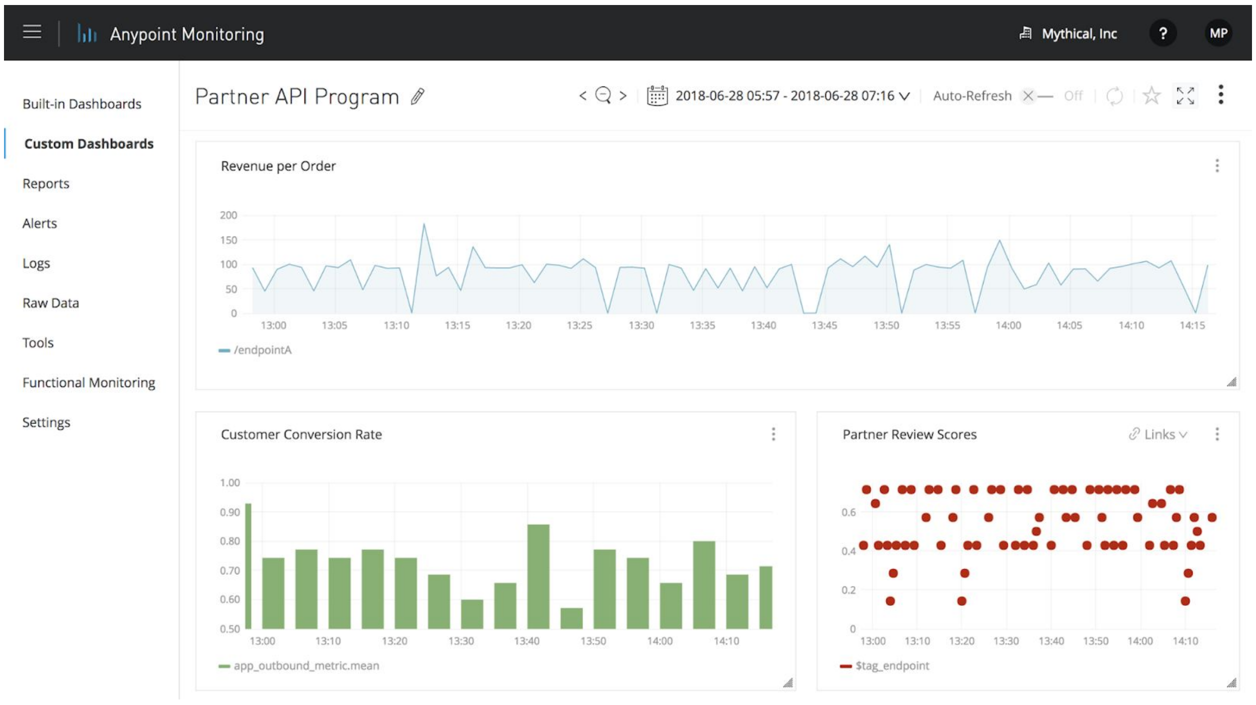
```
<ftp:config name="ftp">
    <ftps:connection username="anonymous" password="password"
     host="localhost" port="${ftpPort}"
     workingDir="${workingDir}">
        <tls:context>
        <tls:trust-store path="path/to/keystore" password="mulepassword"/>
        </tls:context>
    </ftps:connection>
</ftp:config>
```

To use the FTPS module, we simply add it to your Mule app through Studio or Design Center, or we can add the following dependency in our `pom.xml` file:

```
<dependency>
    <groupId>com.mulesoft.connectors</groupId>
    <artifactId>mule-ftps-connector</artifactId>
    <version>1.3.0</version> <!-- or newer -->
    <classifier>mule-plugin</classifier>
</dependency>
```

Anypoint Custom Metrics:The Anypoint Custom Metrics connector to send operational and business metric values and the dimensions you customize, to Anypoint Monitoring. Add a Custom Dashboard in Anypoint Monitoring to view the custom metrics in real time. We can include multiple custom and built-in metrics in a single dashboard, where we can filter and group the metrics along the dimensions you specify to further refine the custom metrics visualization for our needs.

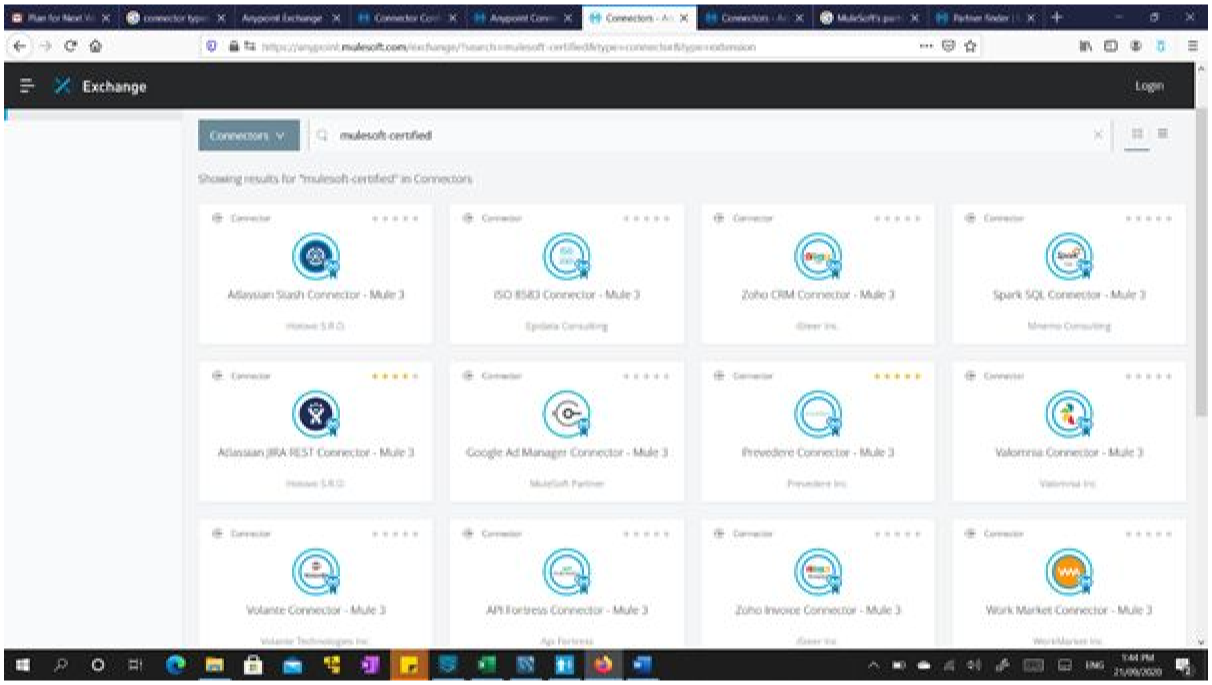
See the Custom Metrics documentation for more information.



3.1.3. Certified

MuleSoft Certified connectors are developed by MuleSoft's partners and developer community and are reviewed and certified by MuleSoft. For support, customers should contact the MuleSoft partner that created the MuleSoft Certified connector.
This type of connector does not require an additional fee to use with Mulesoft Enterprise Edition.

MuleSoft Connector Certification Program
MuleSoft's connector certification program guides you through the process of creating an enterprise-grade connector that is ready for certification by MuleSoft. Once certified, the connector will be made available through the Anypoint Exchange for discovery and use for our customers and developers.


Exchange
Login
Connectors
mulesoft-certified
Showing results for "mulesoft-certified" in Connectors
Atlassian Stash Connector - Mule 3
ISO 8583 Connector - Mule 3
Zoho CRM Connector - Mule 3
Spark SQL Connector - Mule 3
Atlassian JIRA REST Connector - Mule 3
Google Ad Manager Connector - Mule 3
Prevedere Connector - Mule 3
Valomnia Connector - Mule 3
Volante Connector - Mule 3
API Fortress Connector - Mule 3
Zoho Invoice Connector - Mule 3
Work Market Connector - Mule 3

https://docs.mulesoft.com/connector-devkit/3.9/connector-certification-program-guidelines

Examples of Certified Connector:

Google Analytics Connector: The Google Analytics Connector from Ksquare is an out-of-the-box solution to integrate your Google Analytics data with other business applications.

This connector enables your organization to have seamless integration of site analytic data with your enterprise applications to help with data decision making, customer targeting, etc. The MuleSoft Google Analytics Connector comes out of the box with custom metrics layered on top of analytic data provided by Google.

Easily design, build and integrate with Mule flows to control and access Google analytics data. The parameter attribution is highly customizable, allowing us to select what data/metrics to import and build reports.

Atlassian Jira Rest Connector: Integrate Jira Cloud with existing SaaS and on-premises applications quickly and easily using the Atlassian Jira Cloud REST Connector from Hotovo.org.

The Atlassian Jira Cloud Connector allows businesses to synchronize data and automate processes between Jira and third party collaboration, mobile, and social applications such as GitHub, Clarizen, Salesforce or Desk.com

Connectivity with the latest, up-to-date Jira Cloud REST API, gives users the ability to perform various operations, such as tracking of issues and issue statistics, working with issues, comments, work logs, attachments, projects, user groups and other information, across third party applications.

The Atlassian Jira Cloud REST Connector allows companies to extend the capabilities of Jira and benefit from its integration with other internal and external systems. The connector support is guaranteed by Hotovo.org, an Atlassian expert, who also provides further services related to Atlassian platform and tools.

Microsoft Office Excel 365 Connector: The Microsoft Office 365 Excel Connector enables you to seamlessly integrate with Microsoft Office 365 instance by leveraging the Office 365 Excel API. This connector allows you to read and modify Excel workbooks stored in OneDrive, SharePoint, or other supported storage platforms. The workbook (or Excel file) resource contains all the other Excel resources through relationships. You can access a workbook through the Office 365 Excel connector by identifying the location of the file in the URL.

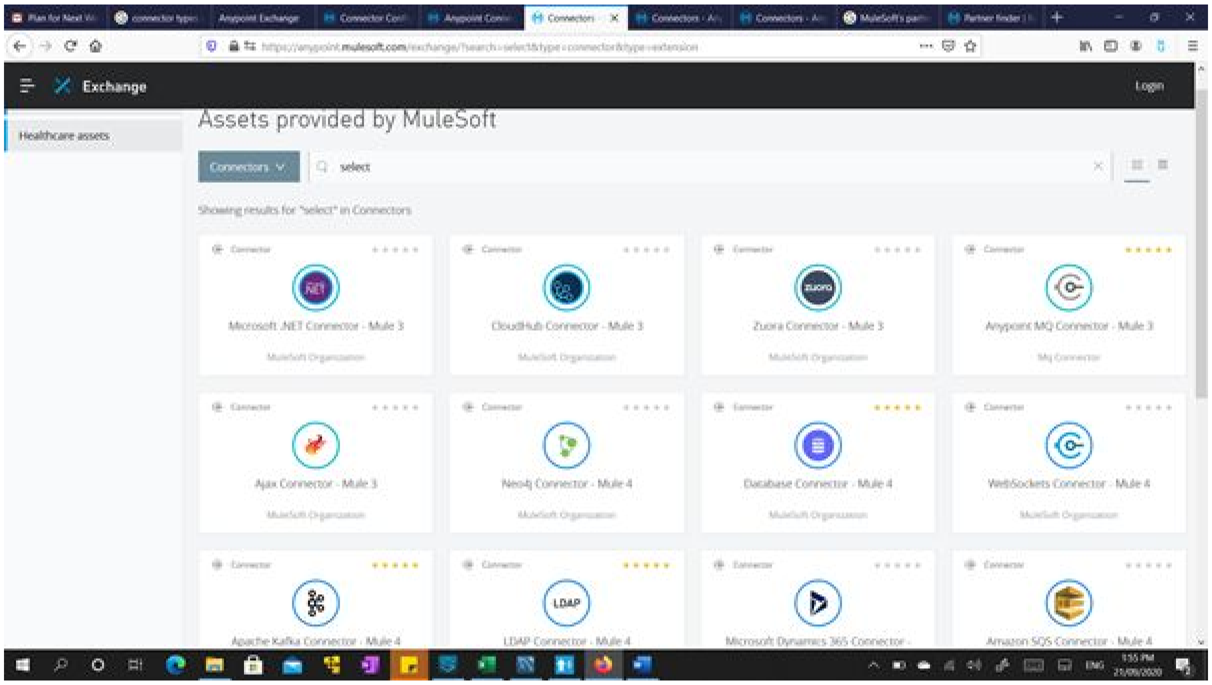
Features included within this connector:

- Excel worksheet modification: Edit workbooks, add worksheets and charts, and do a lot more with this connector.
- Authentication: Authkey based Office 365 Excel credentials and authentication mechanism.

3.1.4. Select

MuleSoft maintains Select connectors. Connectors included in the open source Mule distribution can be used by everyone, however support is only included in an Anypoint Platform subscription. To use all other Select Connectors and access support, you must have an active Anypoint Platform subscription.

Select connectors are developed by Mulesoft Organisation.


Exchange
Login
Healthcare assets
Assets provided by MuleSoft
Connectors
select
Showing results for "select" in Connectors
Microsoft .NET Connector - Mule 3
MuleSoft Organization
CloudHub Connector - Mule 3
MuleSoft Organization
Zuora Connector - Mule 3
MuleSoft Organization
Anypoint MQ Connector - Mule 3
Mq Connector
Ajax Connector - Mule 3
MuleSoft Organization
Neo4j Connector - Mule 4
MuleSoft Organization
Database Connector - Mule 4
MuleSoft Organization
WebSockets Connector - Mule 4
MuleSoft Organization
Apache Kafka Connector - Mule 4
LDAP Connector - Mule 4
Microsoft Dynamics 365 Connector -
Amazon SQS Connector - Mule 4

Examples of Select Connector:

MongoDB Connector - Mule 4:

MuleSoft's MongoDB Connector enables us to leverage MongoDB's open source document database, and the leading NoSQL database. Using this connector, businesses can access databases linked to their existing account and perform many of the functions available in the MongoDB driver, such as document and collections management.

Businesses can also integrate MongoDB with e-commerce platforms such as Magento, and CRM applications such as Zoho CRM to extend connectivity throughout the enterprise.

Associated Use Cases

Use MuleSoft's MongoDB connector for the following data integration scenarios:

- Create a single view of your most important data - Leverage MuleSoft's Anypoint platform to aggregate your siloed data from multiple enterprise systems, transform the data, and write the data to MongoDB.
- Mainframe offloading - Leverage the MongoDB connector to mirror or synchronize your operational data from mainframes into MongoDB to offload workloads from your mainframes while enabling the capability to meet your customer where they are.
- Real-time analytics - Use the MongoDB Connector to leverage MongoDB's rich query language to make near real-time decisions on shipping, inventory and pricing. Analyze any data in place and in real-time using documents in MongoDB.
- Personalization - Create personalized omni-channel retail experiences by leveraging MuleSoft's Anypoint platform to collect and synchronize data across your retail channels and your backend systems into MongoDB.

Web Service Consumer Connector - Mule 4:

When developing applications with Mule, users often need to consume a SOAP web service to acquire data from an external source.

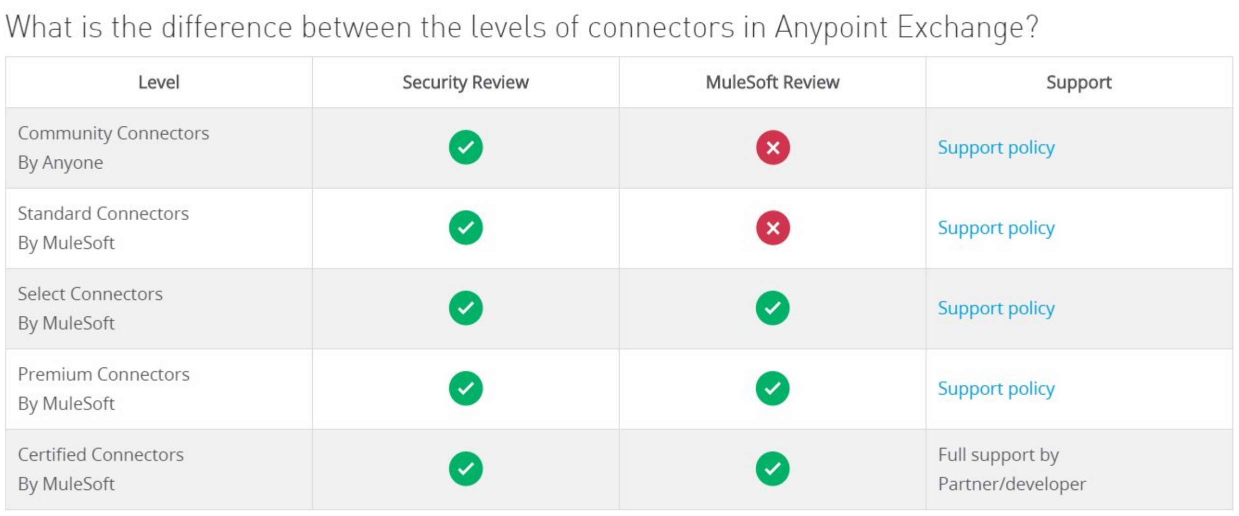
In the best case scenario, we can use an existing Anypoint Connector in our Mule application to connect to a specific service provider, such as Zuora or Avalara. However, where no service-specific connector exists to facilitate the connection, the quickest, easiest way to consume a Web service from within a Mule application is to use the Web Service Consumer.

Using the information contained in a service’s WSDL, this connector enables us to configure a few details in order to establish the connection you need to consume a service from within your Mule application.

Simply identify the location of the Web service’s WSDL file, then ask the Web Service Consumer to configure itself from the WSDL – host, port, address. This connector module only works with Mule 4 and Studio 7.

SharePoint Connector-Mule-04: The Microsoft SharePoint connector provides support for both SharePoint 2013 and SharePoint Online for use in the cloud and on-premises. This connector enables content and document management, access to intranet and extranet portals and websites, collaboration, and enterprise search. This connector also enables integration with SharePoint through its REST API.

Difference between levels of connectors in Anypoint Exchange:



What is the difference between the levels of connectors in Anypoint Exchange?

Level	Security Review	MuleSoft Review	Support
Community Connectors By Anyone	✓	✗	Support policy
Standard Connectors By MuleSoft	✓	✗	Support policy
Select Connectors By MuleSoft	✓	✓	Support policy
Premium Connectors By MuleSoft	✓	✓	Support policy
Certified Connectors By MuleSoft	✓	✓	Full support by Partner/developer

3.2. Custom Connector

Custom connectors are the connectors created by the mulesoft developers community.
We can develop our own connector using the new Mule SDK platform for Mule Runtime 4.
Mule SDK replaces Devkit for developing Custom Connectors.

MuleSoft's Anypoint Connectors helps to connect third-party APIs and services through the Mule flow. We can use the connector to send or receive a message from Mule flow to one or more external services over protocol or API. Mule flow designing is graphical, so we don't need to write a single line of code to connect external services when we are using Anypoint connectors. Today, MuleSoft has a large collection of different connectors to connect and integrate diverse systems including Salesforce, Database, different Google and AWS services, and many more.

You can develop your own connector using the new Mule SDK platform, which is the replacement of earlier Mule devkit tool.

Prerequisites for Custom Connector

1. Java JDK Version 8.x
2. Anypoint Studio 7.x
3. Apache Maven 3.3.9 or higher

Steps to Create a Connector

Go to the directory where you want to create connector (studio-workspace) and execute a command to create project structure with sample test cases:

```
 $ mvn
org.mule.extensions:mule-extensions-archetype-maven-plugin:generate
```

Complete the configuration through the console by answering a couple of questions:

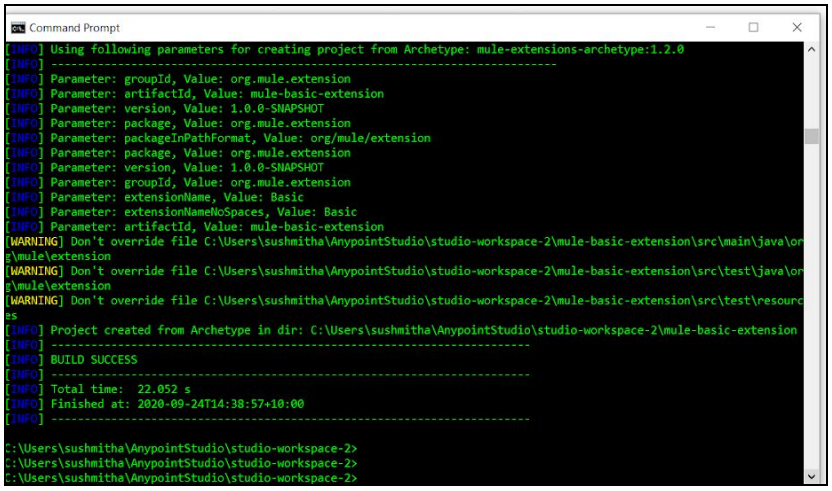
Enter the name of the extension (empty for default):

Enter the extension's groupId (empty for default):

Enter the extension's artifactId (empty for default):

Enter the extension's version (empty for default):

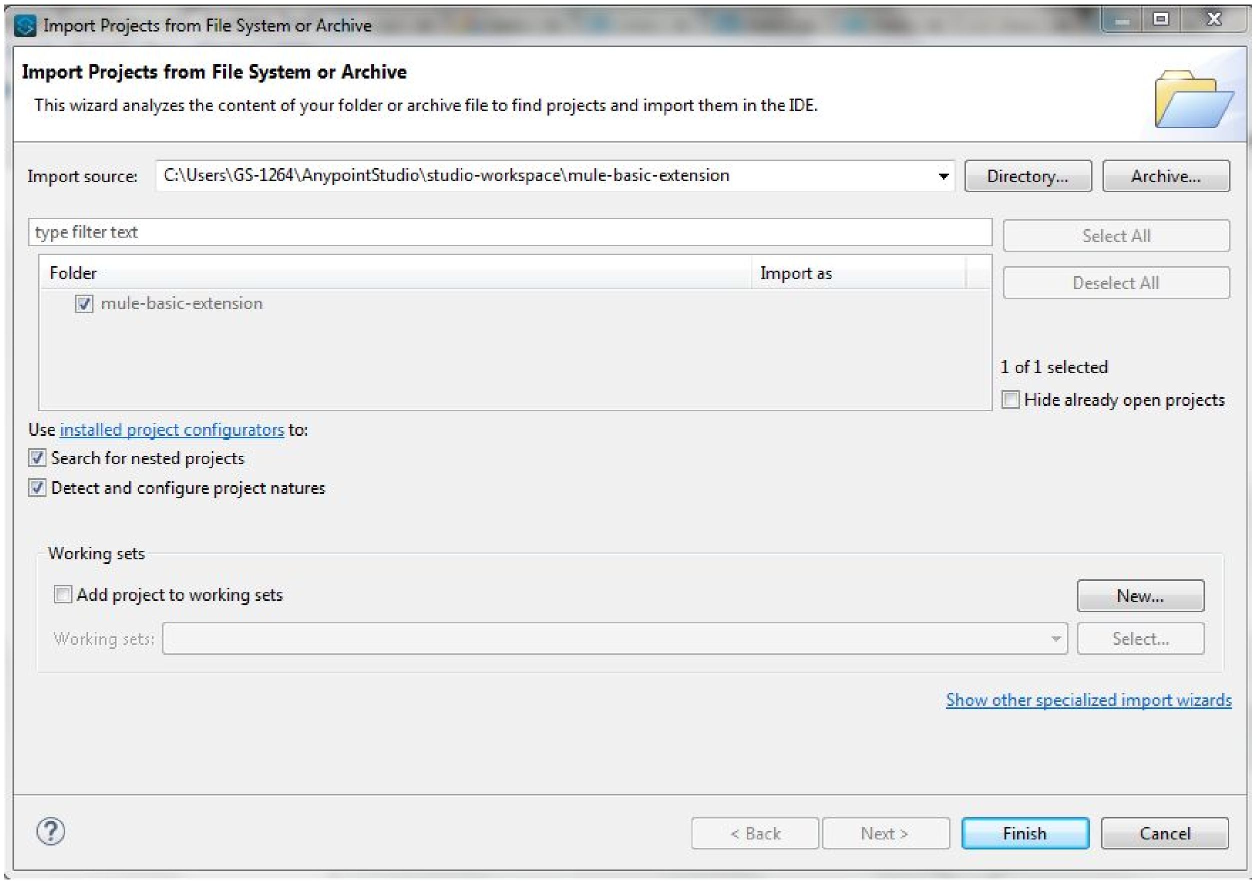
Enter the extension's main package (empty for default):



```
[INFO] Using following parameters for creating project from Archetype: mule-extensions-archetype:1.2.0
[INFO] ----------------------------------------------------------------------------
[INFO] Parameter: groupId, Value: org.mule.extension
[INFO] Parameter: artifactId, Value: mule-basic-extension
[INFO] Parameter: version, Value: 1.0.0-SNAPSHOT
[INFO] Parameter: package, Value: org.mule.extension
[INFO] Parameter: packageInPathFormat, Value: org/mule/extension
[INFO] Parameter: package, Value: org.mule.extension
[INFO] Parameter: version, Value: 1.0.0-SNAPSHOT
[INFO] Parameter: groupId, Value: org.mule.extension
[INFO] Parameter: extensionName, Value: Basic
[INFO] Parameter: extensionNameNoSpaces, Value: Basic
[INFO] Parameter: artifactId, Value: mule-basic-extension
[WARNING] Don't override file C:\Users\sushmitha\AnypointStudio\studio-workspace-2\mule-basic-extension\src\main\java\or
g\mule\extension
[WARNING] Don't override file C:\Users\sushmitha\AnypointStudio\studio-workspace-2\mule-basic-extension\src\test\java\or
g\mule\extension
[WARNING] Don't override file C:\Users\sushmitha\AnypointStudio\studio-workspace-2\mule-basic-extension\src\test\resourc
es
[INFO] Project created from Archetype in dir: C:\Users\sushmitha\AnypointStudio\studio-workspace-2\mule-basic-extension
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  22.052 s
[INFO] Finished at: 2020-09-24T14:38:57+10:00
[INFO] ------------------------------------------------------------------------

C:\Users\sushmitha\AnypointStudio\studio-workspace-2>
C:\Users\sushmitha\AnypointStudio\studio-workspace-2>
C:\Users\sushmitha\AnypointStudio\studio-workspace-2>
```

Fig 2.1: Default Extension Project created successfully

Go to Anypoint studio, **File > Open Project from File System,** and select the project directory you have created in the last step.


Import Projects from File System or Archive

Import Projects from File System or Archive

This wizard analyzes the content of your folder or archive file to find projects and import them in the IDE.

Import source: C:\Users\GS-1264\AnypointStudio\studio-workspace\mule-basic-extension

type filter text

Folder	Import as
☑ mule-basic-extension	

1 of 1 selected

☐ Hide already open projects

Use installed project configurators to:

☑ Search for nested projects

☑ Detect and configure project natures

Working sets

☐ Add project to working sets

Working sets:

After you open this project in Anypoint studio, you will get the number of classes, annotated with mule SDK annotations as below:

The connector is called as an **extension** in Mule 4, and it's class defined by the annotation **@Extension**. A **@Configurations** annotation is used to point the configuration class. }

We can install this connector into local maven repository using the command:

```
$mvn clean install
```

You can use this connector in your **Mule 4** application by adding the following dependency in pom.xml:

```
<dependency>
<groupId>org.mule.extension</groupId>
  <artifactId>mule-basic-extension</artifactId>
  <version>1.0.0-SNAPSHOT</version>
 <classifier>mule-plugin</classifier
</dependency>
```

3.3. Common Connector

The most widely used Anypoint Connectors are generally referred to as Common Connectors. Common connectors can be different in the context of the project one is referring to. In general, the common connector of an API-led Connectivity approach would be HTTP connector. To make it easier for the developers to add connectors to the projects, Anypoint Studio features a few ‘Select’ connectors as modules in Mule Palette which are the most used or downloaded connectors from Anypoint Exchange. Fig 3.1 shows the connectors and modules that are featured in the Mule Palette in Anypoint Studio.

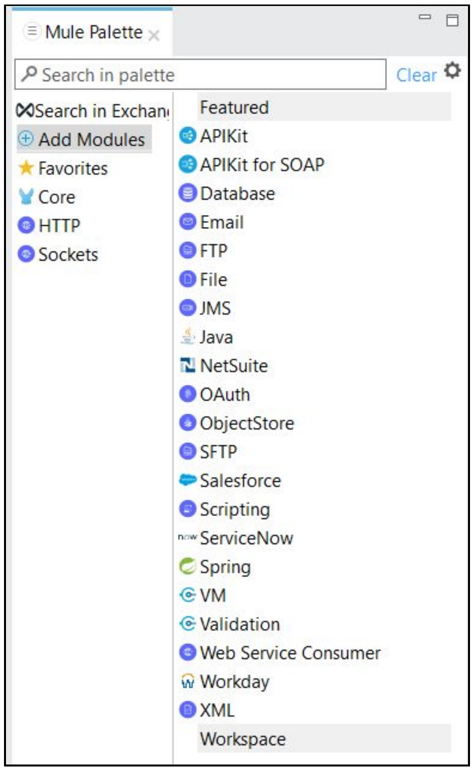


Fig 3.1: Connectors and Modules in Anypoint Studio 7.6-Mule Palette

The common connectors in according to wide usage are as follows:

- HTTP
- Database
- Email
- File
- FTP
- SFTP
- ObjectStore
- JMS
- Salesforce

Few of the common connectors were discussed as examples of Select Connector Types in *Section 1.4* .The remaining section would describe some common connectors.

3.3.1 HTTP Connector - Mule 4

Anypoint HTTP Connector provides HTTP client and server functionality. It enables us to declare HTTP servers that listen to requests and trigger flows, as well as HTTP clients that can communicate with any HTTP service. In simple words, using HTTP and HTTPS protocols the connector performs one of the following operations:

- Listener - Starts executing the Mule flow upon receiving an HTTP request.
- Request - Consumes an HTTP service.
- Basic Security Filter - Authenticate user using basic authentication
- Load Static resource - Load a static resource, such as your home page or a script

HTTP Connector effectively allows us to both expose and consume HTTP based APIs. For example, APIkit and RESTConnect both use the HTTP connector internally to work with APIs.

By default, the Listener configuration connects to CloudHub using HTTP and the Request configuration connects to port 80 using HTTP and GET. You can quickly configure an app to consume an HTTP service using the two HTTP connector instances as follows:

- Listener configuration: Path = /somepath, for example
- Request configuration: Path Or URL = the URL of the HTTP service you want to consume

The HTTP connector can be added to a project by adding the dependency snippet to the project's pom.xml file as follows:

```
<dependency>
  <groupId>org.mule.connectors</groupId>
  <artifactId>mule-http-connector</artifactId>
  <version>RELEASE</version>
  <classifier>mule-plugin</classifier>
</dependency>
```

Mule converts RELEASE to the latest version. To specify a version, view Anypoint Exchange and click **Dependency Snippets**.
Release Notes: HTTP Connector Release Notes
Exchange: HTTP Connector

3.3.1.1 HTTP Listener

The HTTP listener is an event source that enables us to set up an HTTP server and trigger flows when HTTP requests are received.

We can choose what methods the source accepts, such as GET, POST or a list of methods, and on which path to accept requests, thereby allowing the routing of requests through different flows.

Once a request is accepted by the listener, the corresponding flow is triggered with the HTTP body as payload and the HTTP data as attributes (headers, query parameters and so on).

When the flow finishes its execution, the HTTP listener enables us to customize the HTTP response based on whether the execution was successful or not, so that different status codes can be returned.

More information at HTTP Listener Configuration Reference

3.3.1.2 HTTP Request

A typical use of the HTTP request operation is to consume an external HTTP service using the default GET method. By default, the GET and OPTIONS methods do not send the payload in the request; the body of the HTTP request is empty. The other methods send the message payload as the body of your request.

After sending a request, the connector receives the response and passes it to the next element in your application's flow.

The required minimum setting for the request operation is a host URI, which can include a path. You can configure the following types of authentication in the request operation:

- Basic
- OAuth
- NT LAN Manager (NTLM)
- Digest

More information at HTTP Request Configuration Reference

3.3.1.3 Basic Security Filter

HTTP Basic Security Filter operation is generally added only after the HTTP Listener operation to authenticate the user using basic authentication. To authorize the user against the authority ROLE_ADMIN(like admin role), we must use Basic Security Filter along with authorization-filter of Spring Module.

More information on configuring and using authorization-filter with Basic Security Filter at Spring Module

3.3.1.4 Load Static Resource

We can configure the HTTP connector to load a static resource, such as our home page or a script. We set the path and file name of a resource to load. We optionally set the name of another file in the same directory as the resource to use in case of failure.

More information at To Load a Static Resource

3.3.2 Database Connector

Anypoint Connector for Database (Database Connector) establishes communication between our Mule app and a relational database.

Database Connector can connect to almost any Java Database Connectivity (JDBC) relational database and run SQL operations. We can specify Dataweave expressions in connector fields and configure attributes dynamically, depending on the database configuration we use. An application can support multi-tenant scenarios using the same configuration element, changing the connection attributes based on, for example, information coming from each request.

The Database connector lets us perform predefined queries as well as queries that take the connector’s input to specify variable parameters or even to construct sections of the query dynamically. It also allows the use of template queries that are both self-sufficient and customizable. We can also perform multiple SQL requests in a single bulk update. The connector also allows us to perform Data Definition Language (DDL) requests that alter the data structure rather than the data itself. The Database connector is available with both Mule Community and Mule Enterprise runtimes.

Database Connector Supports the following Basic operations:

- Select
- Insert
- Update
- Delete
- Stored Procedure
- Bulk Execute
- DDL operations such as CREATE, ALTER, etc.

3.3.2.1. Supported Database Types

The Database Connector has connection providers that automatically set the driver class name and create JDBC URLs with the given parameters for the following databases:

- MySQL
- Oracle
- Microsoft SQL Server
- Derby
- JDBC

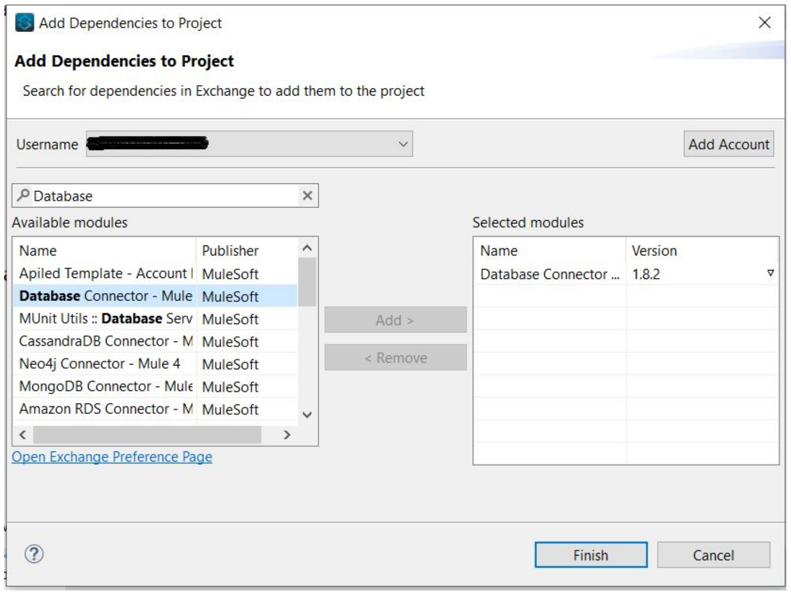
We can set up other JDBC databases using a generic JDBC configuration. We can also reference a JDBC DataSource object or a XADataSource object, which is required for XA transactions. We typically create the object in Studio using Spring.

3.3.2.2. Install Connector

Anypoint Database Connector is not readily available in Mule Palette, instead we need to import it from Anypoint Exchange. Following are the steps to install the connector:

1. In Anypoint Studio, click the Anypoint Exchange icon in the Studio task bar.

2. In Anypoint Exchange, click Login and supply your Anypoint Platform username and password.
3. Search for the Database connector, click the card for the connector, and click Add to project. (*Fig 3.2*)



Add Dependencies to Project

Add Dependencies to Project

Search for dependencies in Exchange to add them to the project

Username [redacted] Add Account

Database

Available modules

Name	Publisher
Apiled Template - Account I	MuleSoft
Database Connector - Mule	MuleSoft
MUnit Utils :: **Database** Serv	MuleSoft
CassandraDB Connector - M	MuleSoft
Neo4j Connector - Mule 4	MuleSoft
MongoDB Connector - Mule	MuleSoft
Amazon RDS Connector - M	MuleSoft

Selected modules

Name	Version
Database Connector ...	1.8.2

Fig 3.2: Add Dependencies to Project

4. Follow the prompts.

3.3.2.3. Configure an Input Source

Configure an input source for the connector such as a connector operation, using HTTP Listener, or Scheduler. The Database operations are used as event processors in the Mule flow which are triggered by the event sources such as HTTP Listener or Scheduler.

We can put the On table row operation for the connector in the Source area of the Studio canvas to trigger a mule flow.

3.3.2.4. Create a Database Configuration

A database configuration which indicates the type of database server and credentials to connect to the database needs to be created as global elements to allow the project to successfully establish a communication to the database server. This can be done in two ways:

1. Create connector configuration from Mule properties.
2. Create Database Connector Configuration in Global elements section. (*Fig 3.3*)

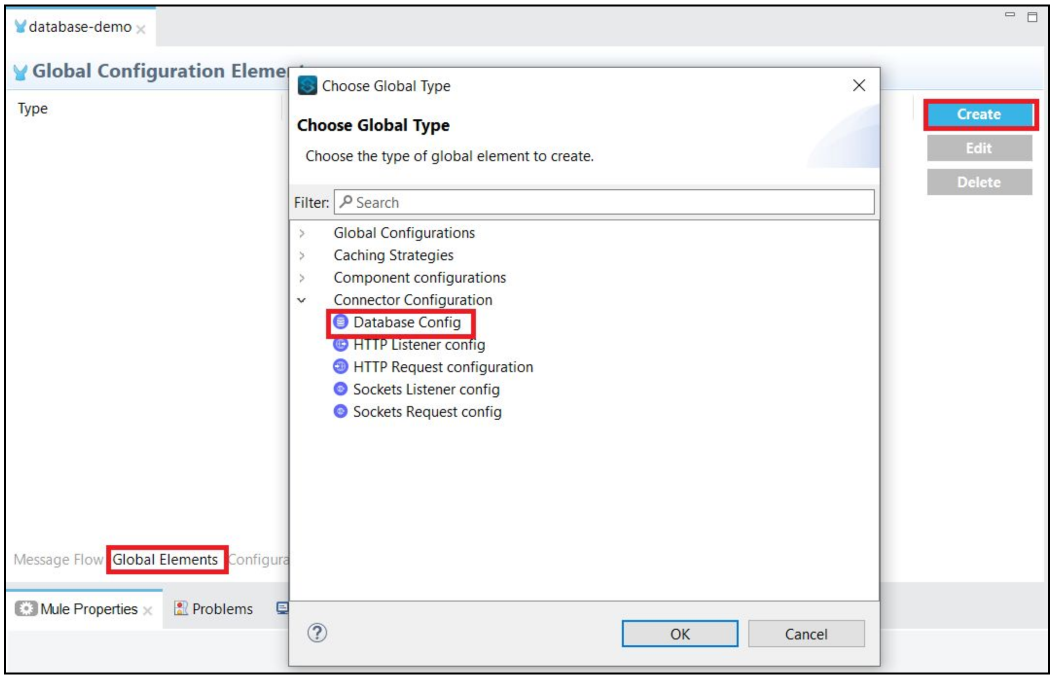


Fig 3.3: Create Global Database Connector Configuration

More information at Database Connector

3.4. REST Connector

REST Connectors are connector assets with REST endpoints. These connectors are accessed by sending an HTTP request to the host defined in the connector configuration.
Anypoint's REST Connect and Mule 4 facilitate the conversion of an API-specification to a connector when published to Anypoint Exchange. This accelerates the building of APIs in an organization.

Most of the connectors in Anypoint Exchange are REST Connectors. Following are few examples of REST Connectors:

- JIRA REST Connector
- Anypoint MQ Connector
- Salesforce Connector
- NetSuite Connector

3.4.1 API to REST Connector

We can create a simple REST connector by publishing an API in Anypoint Exchange. Following steps would enable us to create a REST Connector and use it in our mule applications:

1. Create an API specification (default is RAML 1.0) in Anypoint Design Center.
2. Define the resources, HTTP methods, reponses and other assets in the specification.
3. Publish the API specification to Exchange.

After the API is published to Exchange, an Anypoint Console and Anypoint Notebook is created for the API in Exchange. Also this API is discoverable to all the users in our particular organization. Every user in our organization can now import this API as a REST connector to their Anypoint Studio. See *Exercise 5.2* for steps to consume a REST Connector.

3.4.2 Differences in features when you compare with HTTP Connector

An HTTP Connector is a component of Anypoint that facilitates establishing a communication channel with external sources using HTTP. A REST Connector enables Mule applications to interact with external applications, mostly using HTTP. Under the hood, REST connector also uses HTTP connector for interacting with other applications.

An HTTP Connector is configured to create an HTTP server or client, whereas a REST Connector is triggered by sending an HTTP request under the hood.

The Listener operation of HTTP Connector is used as the event source of a Mule flow, whereas most of the REST connectors are rather event processors of the Mule flow.

3.5. Exercise

3.5.1. Custom Connector

Steps to Create a Connector

Go to the directory where you want to create connector (studio-workspace) and execute a command to create project structure with sample test cases:

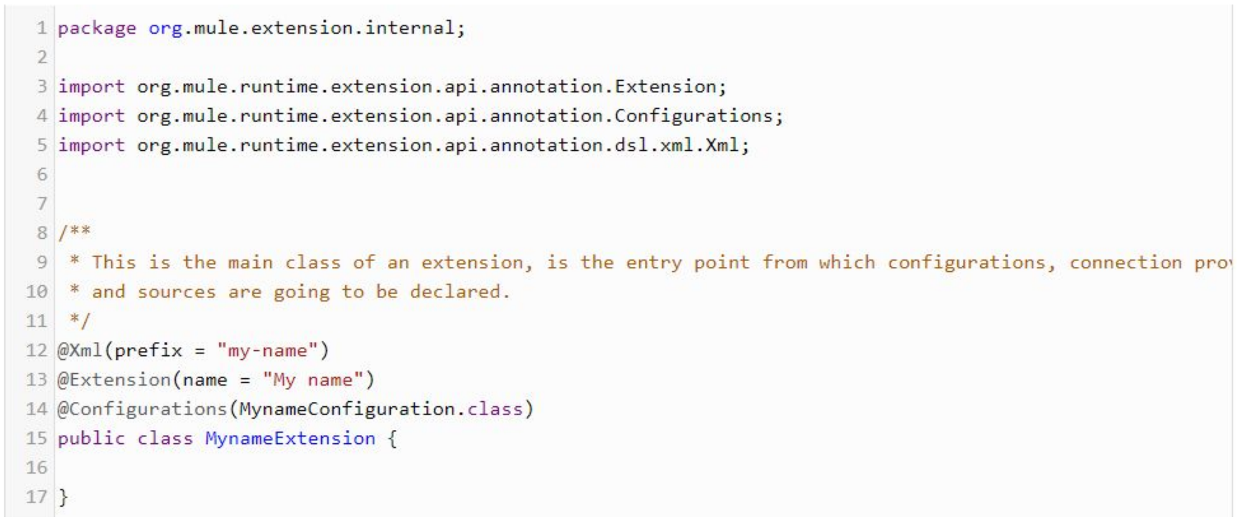
```
$ mvn org.mule.extensions:mule-extensions-archetype-maven-plugin:generate
```

Complete the configuration through the console by answering a couple of questions:

```
Enter the name of the extension (empty for default):
Enter the extension's groupId (empty for default):
Enter the extension's artifactId (empty for default):
Enter the extension's version (empty for default):
Enter the extension's main package (empty for default):
```

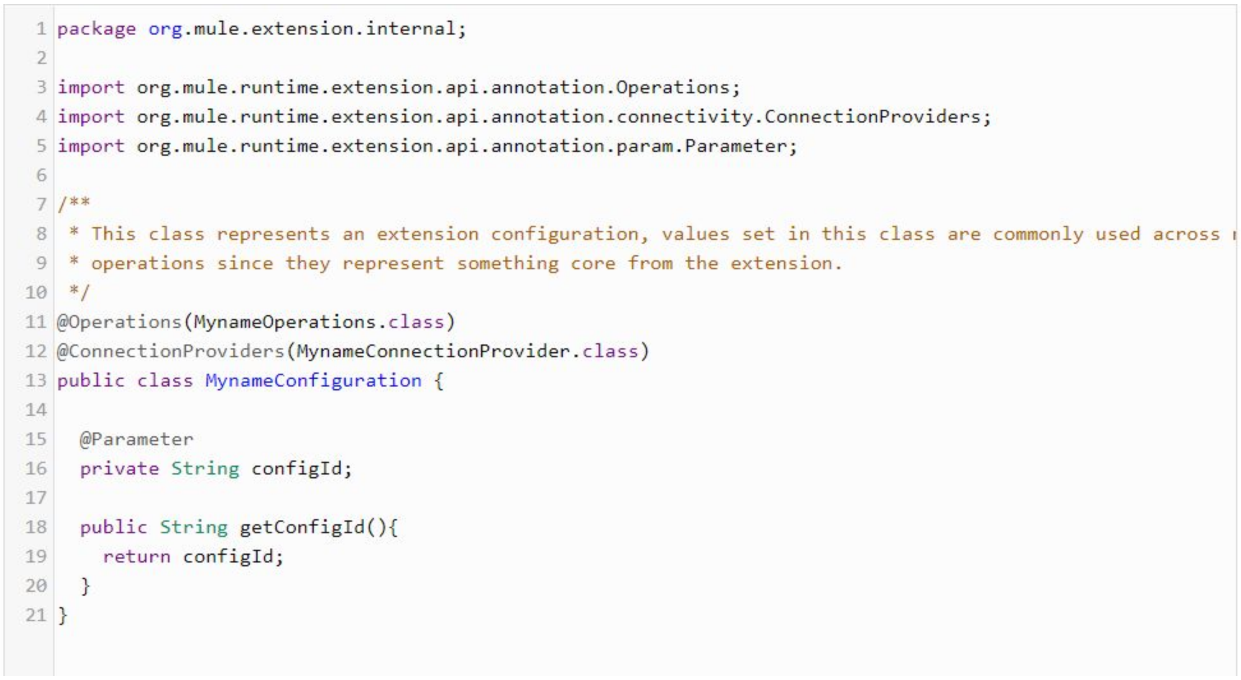
After we open this project in Anypoint studio, we will get the number of classes, annotated with mule SDK annotations as below:

The connector is called as an extension in Mule 4, and it's class defined by the annotation @Extension. A @Configurations annotation is used to point the configuration class.



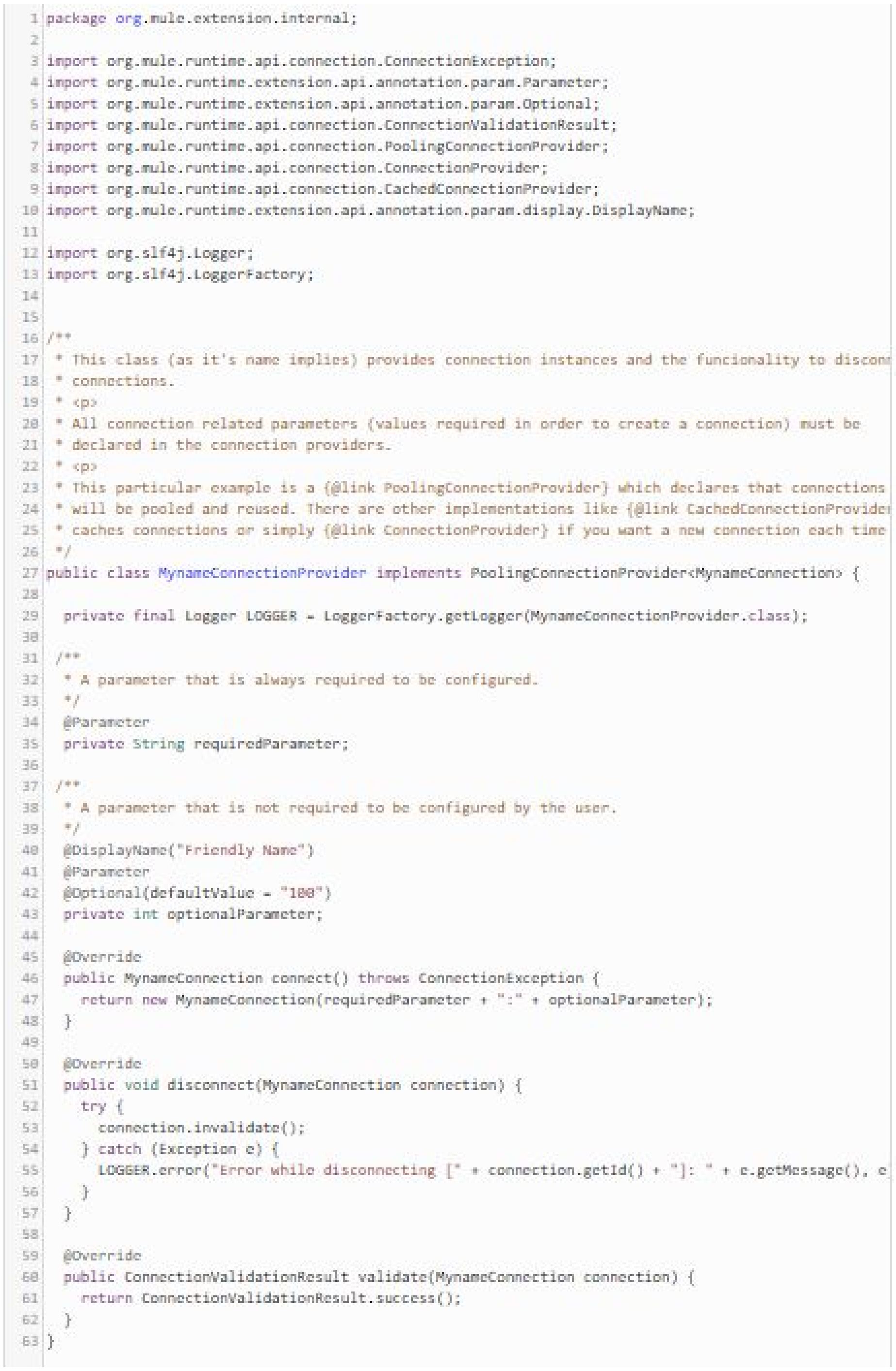
```java
package org.mule.extension.internal;

import org.mule.runtime.extension.api.annotation.Extension;
import org.mule.runtime.extension.api.annotation.Configurations;
import org.mule.runtime.extension.api.annotation.dsl.xml.Xml;


/**
 * This is the main class of an extension, is the entry point from which configurations, connection pro
 * and sources are going to be declared.
 */
@Xml(prefix = "my-name")
@Extension(name = "My name")
@Configurations(MynameConfiguration.class)
public class MynameExtension {

}
```

Configuration class defines the parameters that we expect appears in configuration part of the connector. We can use **@ConnectionProvider** and **@Operation** annotation with this class to point out the connection providers and the operations connector has implemented.



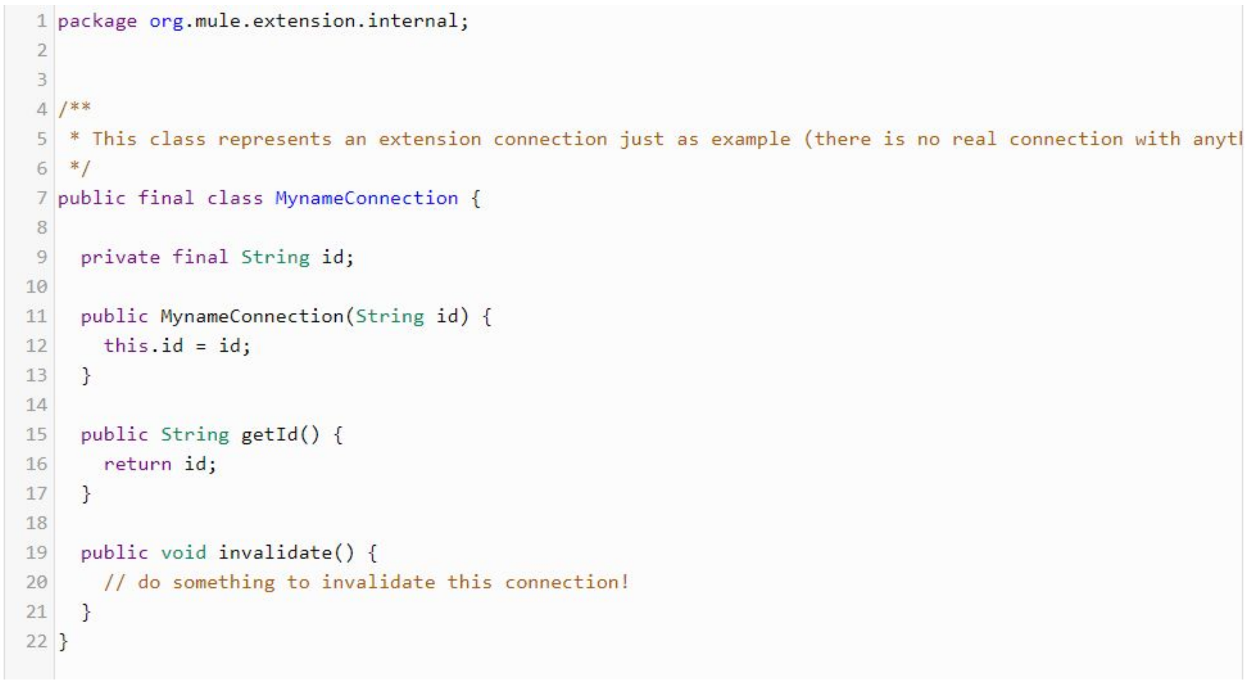
```java
package org.mule.extension.internal;

import org.mule.runtime.extension.api.annotation.Operations;
import org.mule.runtime.extension.api.annotation.connectivity.ConnectionProviders;
import org.mule.runtime.extension.api.annotation.param.Parameter;

/**
 * This class represents an extension configuration, values set in this class are commonly used across
 * operations since they represent something core from the extension.
 */
@Operations(MynameOperations.class)
@ConnectionProviders(MynameConnectionProvider.class)
public class MynameConfiguration {

  @Parameter
  private String configId;

  public String getConfigId(){
    return configId;
  }
}
```

The next important class is *MynameConnectionProvider*, which is annotated as **@ConnectionProviders in** configuration class.



```java
package org.mule.extension.internal;

import org.mule.runtime.api.connection.ConnectionException;
import org.mule.runtime.extension.api.annotation.param.Parameter;
import org.mule.runtime.extension.api.annotation.param.Optional;
import org.mule.runtime.api.connection.ConnectionValidationResult;
import org.mule.runtime.api.connection.PoolingConnectionProvider;
import org.mule.runtime.api.connection.ConnectionProvider;
import org.mule.runtime.api.connection.CachedConnectionProvider;
import org.mule.runtime.extension.api.annotation.param.display.DisplayName;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;


/**
 * This class (as it's name implies) provides connection instances and the funcionality to discon
 * connections.
 * <p>
 * All connection related parameters (values required in order to create a connection) must be
 * declared in the connection providers.
 * <p>
 * This particular example is a {@link PoolingConnectionProvider} which declares that connections
 * will be pooled and reused. There are other implementations like {@link CachedConnectionProvide
 * caches connections or simply {@link ConnectionProvider} if you want a new connection each time
 */
public class MynameConnectionProvider implements PoolingConnectionProvider<MynameConnection> {

  private final Logger LOGGER = LoggerFactory.getLogger(MynameConnectionProvider.class);

 /**
  * A parameter that is always required to be configured.
  */
  @Parameter
  private String requiredParameter;

 /**
  * A parameter that is not required to be configured by the user.
  */
  @DisplayName("Friendly Name")
  @Parameter
  @Optional(defaultValue = "100")
  private int optionalParameter;

  @Override
  public MynameConnection connect() throws ConnectionException {
    return new MynameConnection(requiredParameter + ":" + optionalParameter);
  }

  @Override
  public void disconnect(MynameConnection connection) {
    try {
      connection.invalidate();
    } catch (Exception e) {
      LOGGER.error("Error while disconnecting [" + connection.getId() + "]: " + e.getMessage(), e
    }
  }

  @Override
  public ConnectionValidationResult validate(MynameConnection connection) {
    return ConnectionValidationResult.success();
  }
}
```

The Connection class is used by connection providers to handle the connection. By having the single connection class and multiple connection providers, we can create multiple connection configurations for our Connector.

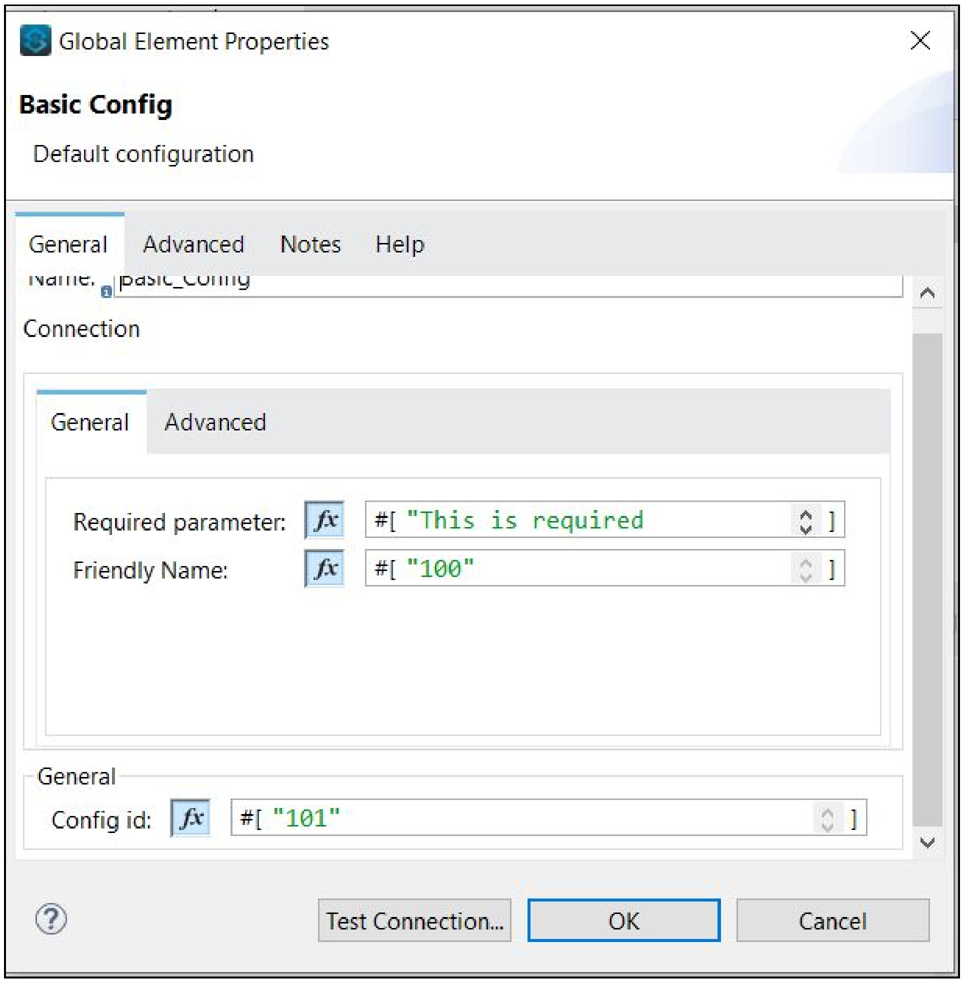


```java
package org.mule.extension.internal;


/**
 * This class represents an extension connection just as example (there is no real connection with anytl
 */
public final class MynameConnection {

  private final String id;

  public MynameConnection(String id) {
    this.id = id;
  }

  public String getId() {
    return id;
  }

  public void invalidate() {
    // do something to invalidate this connection!
  }
}
```

And finally, we have operation class. We can define any number of operation classes and all public methods from this class will be treated as a connector operation. We can inject configurations and connections to particular operation using **@Config** and **@Connection** annotation in method arguments.

```java
package org.mule.extension.internal;

import static org.mule.runtime.extension.api.annotation.param.MediaType.ANY;

import org.mule.runtime.extension.api.annotation.param.MediaType;
import org.mule.runtime.extension.api.annotation.param.Config;
import org.mule.runtime.extension.api.annotation.param.Connection;


/**
 * This class is a container for operations, every public method in this class will be taken as an exte
 */
public class MynameOperations {

  /**
   * Example of an operation that uses the configuration and a connection instance to perform some acti
   */
  @MediaType(value = ANY, strict = false)
  public String retrieveInfo(@Config MynameConfiguration configuration, @Connection MynameConnection co
    return "Using Configuration [" + configuration.getConfigId() + "] with Connection id [" + connectio
  }

  /**
   * Example of a simple operation that receives a string parameter and returns a new string message th
   */
  @MediaType(value = ANY, strict = false)
  public String sayHi(String person) {
    return buildHelloMessage(person);
  }

  /**
   * Private Methods are not exposed as operations
   */
  private String buildHelloMessage(String person) {
    return "Hello " + person + "!!!";
  }
}
```

Publishing to exchange
https://javastreets.com/blog/publish-connectors-to-anypoint-exchange.html


Global Element Properties

Basic Config

Default configuration

Name: basic_config

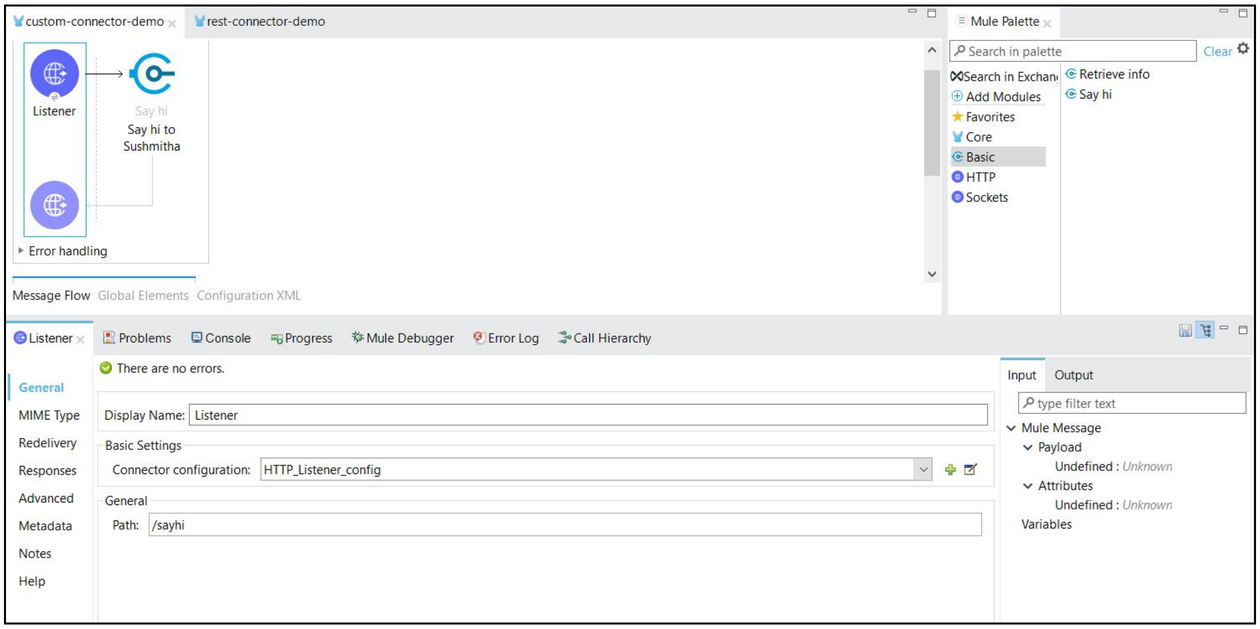
Connection

Required parameter: #["This is required]

Friendly Name: #["100"]

General

Config id: #["101"]



There are no errors.

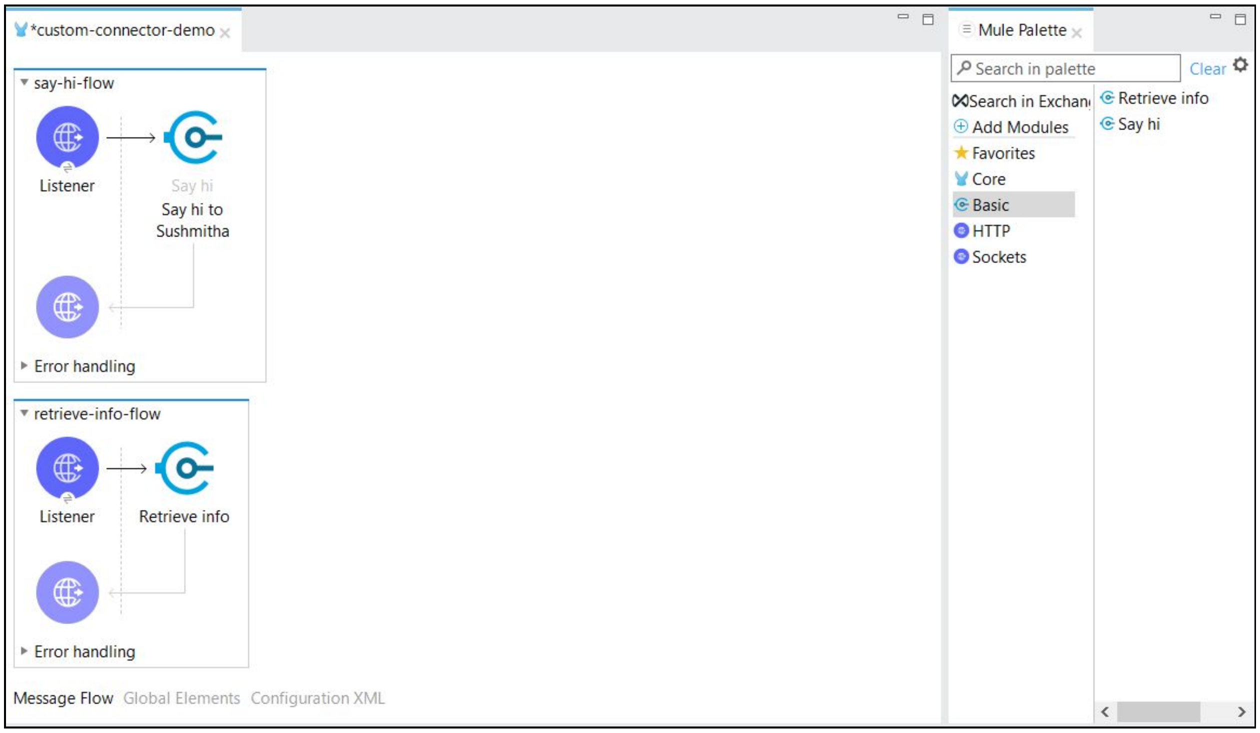
Display Name: Listener

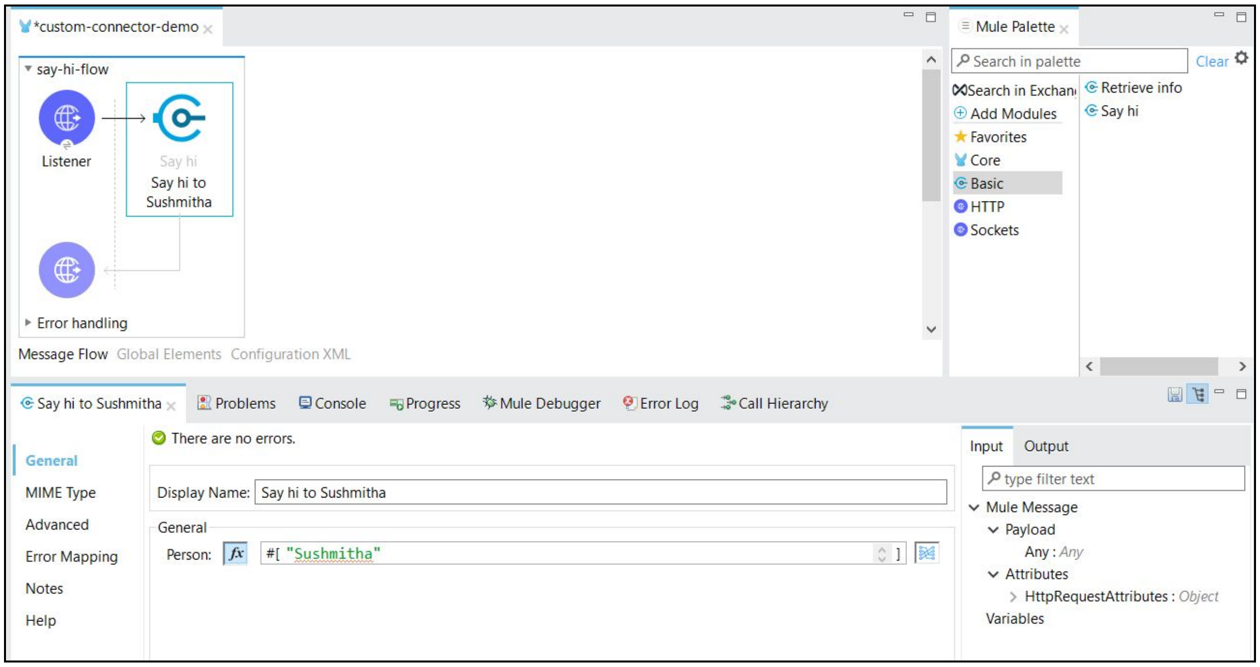
Basic Settings

Connector configuration: HTTP_Listener_config

General

Path: /sayhi





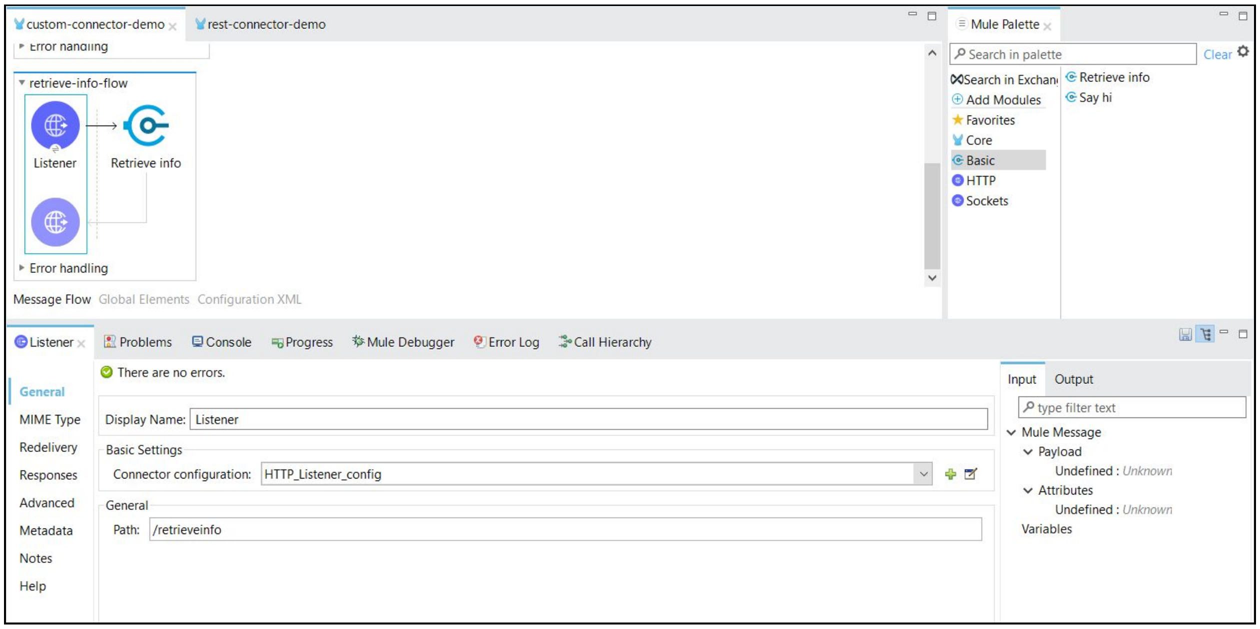
There are no errors.

Display Name: Say hi to Sushmitha

General

Person: #["Sushmitha"]

Mule Message
Payload
Any : *Any*
Attributes
HttpRequestAttributes : *Object*
Variables



There are no errors.

Display Name: Listener

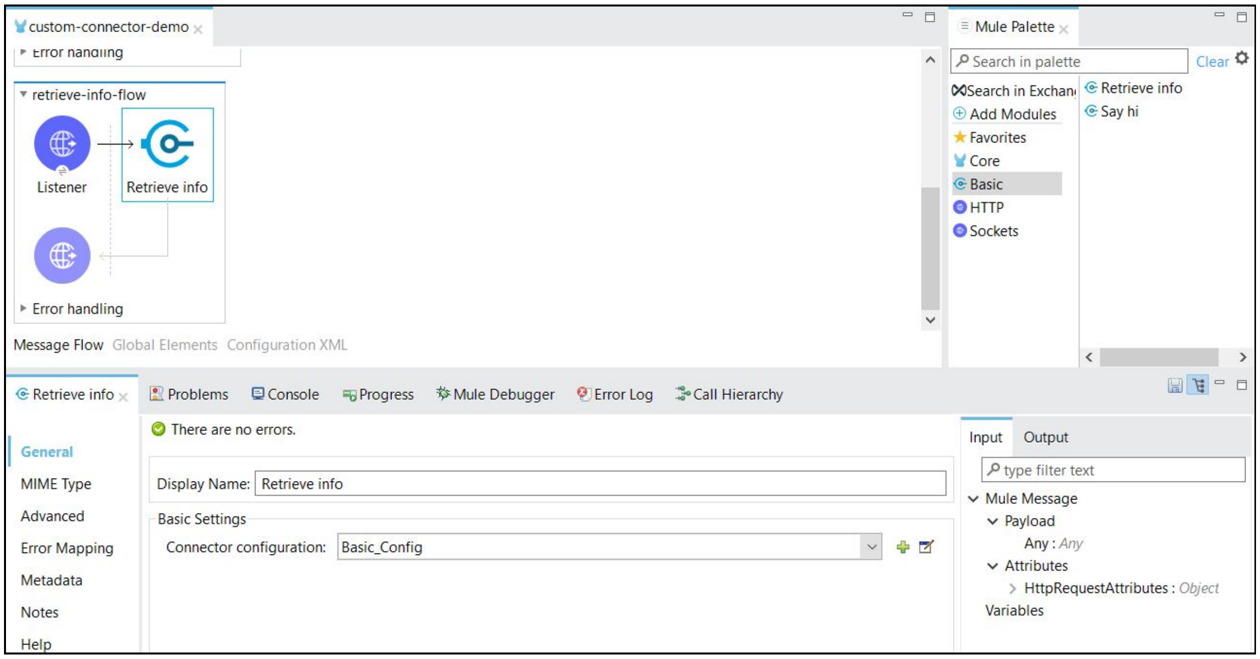
Basic Settings

Connector configuration: HTTP_Listener_config

General

Path: /retrieveinfo

Mule Message
Payload
Undefined : *Unknown*
Attributes
Undefined : *Unknown*
Variables

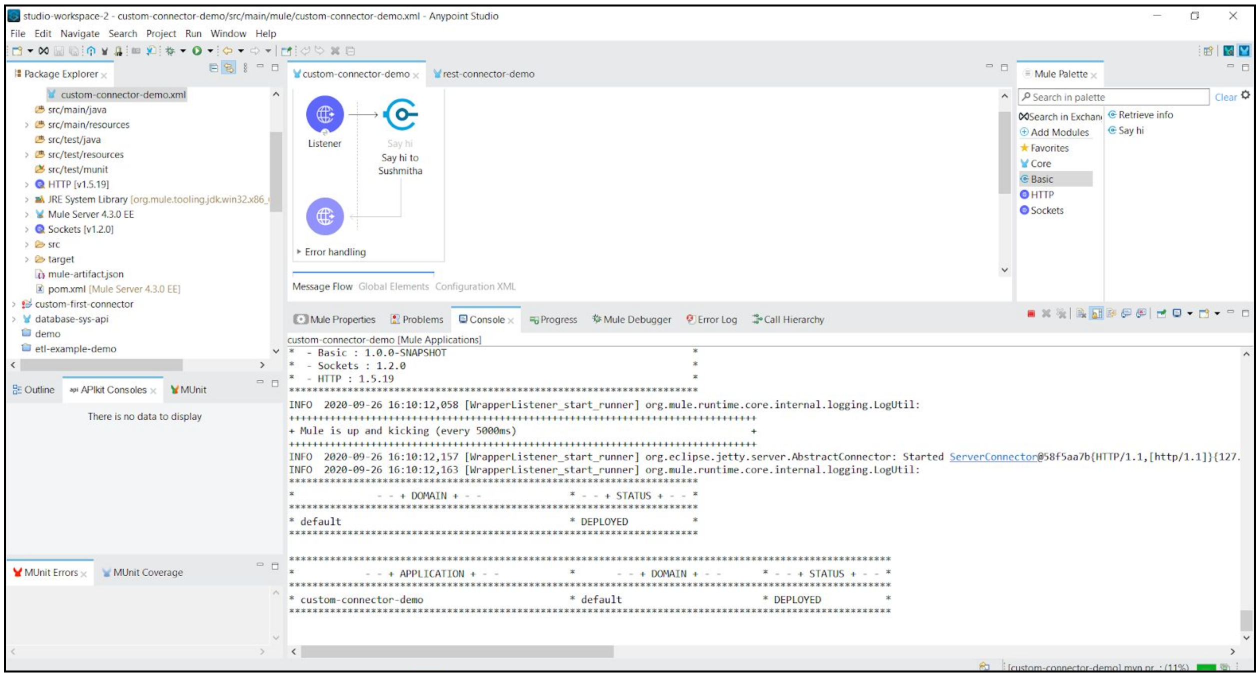


There are no errors.

Display Name: Retrieve info

Basic Settings

Connector configuration: Basic_Config

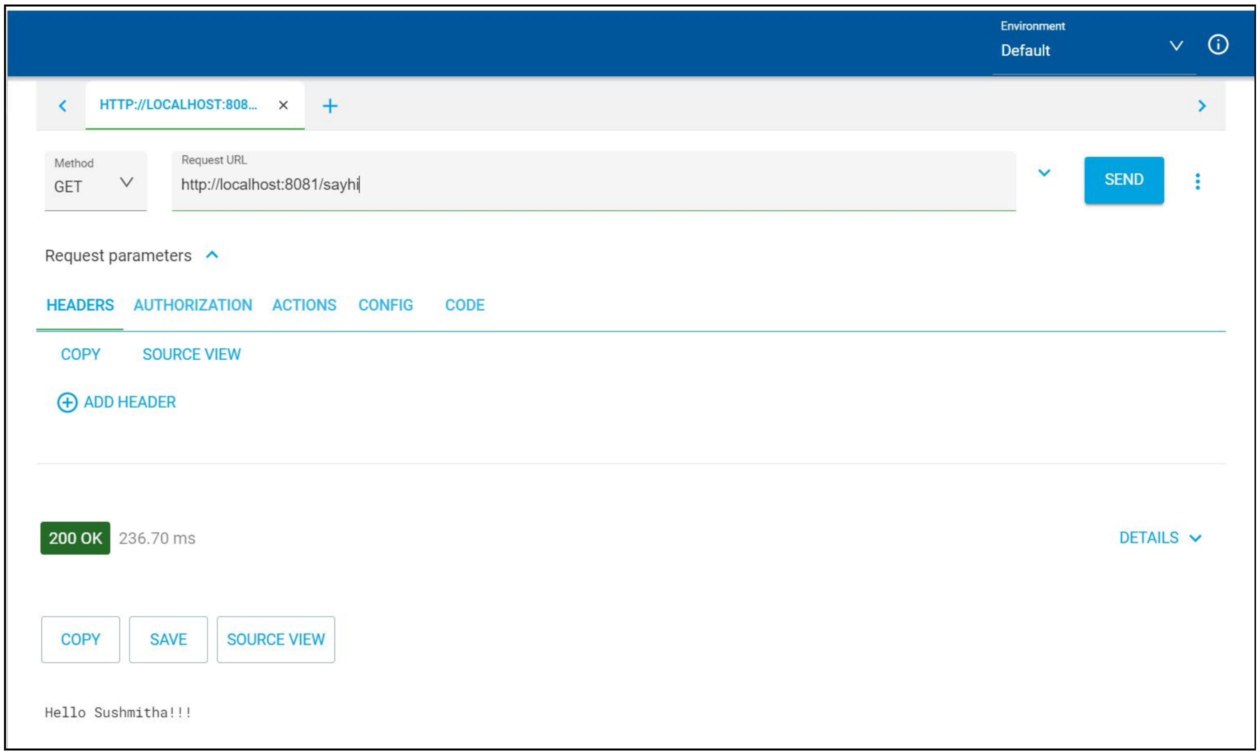


custom-connector-demo [Mule Applications]

```
*  - Basic : 1.0.0-SNAPSHOT                                          *
*  - Sockets : 1.2.0                                                 *
*  - HTTP : 1.5.19                                                   *
**********************************************************************
INFO  2020-09-26 16:10:12,058 [WrapperListener_start_runner] org.mule.runtime.core.internal.logging.LogUtil:
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+ Mule is up and kicking (every 5000ms)                                        +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
INFO  2020-09-26 16:10:12,157 [WrapperListener_start_runner] org.eclipse.jetty.server.AbstractConnector: Started ServerConnector@58f5aa7b{HTTP/1.1,[http/1.1]}{127.
INFO  2020-09-26 16:10:12,163 [WrapperListener_start_runner] org.mule.runtime.core.internal.logging.LogUtil:
**********************************************************************
*              - - + DOMAIN + - -               * - - + STATUS + - - *
**********************************************************************
* default                                       * DEPLOYED           *
**********************************************************************

****************************************************************************************************************
*            - - + APPLICATION + - -            *        - - + DOMAIN + - -         * - - + STATUS + - - *
****************************************************************************************************************
* custom-connector-demo                         * default                           * DEPLOYED           *
****************************************************************************************************************
```

Test the application from one of the HTTP/REST client tools like Advanced REST Client (ARC) or Postman.

Send a HTTP GET request to http://localhost:8081/sayhi to test say-hi-flow

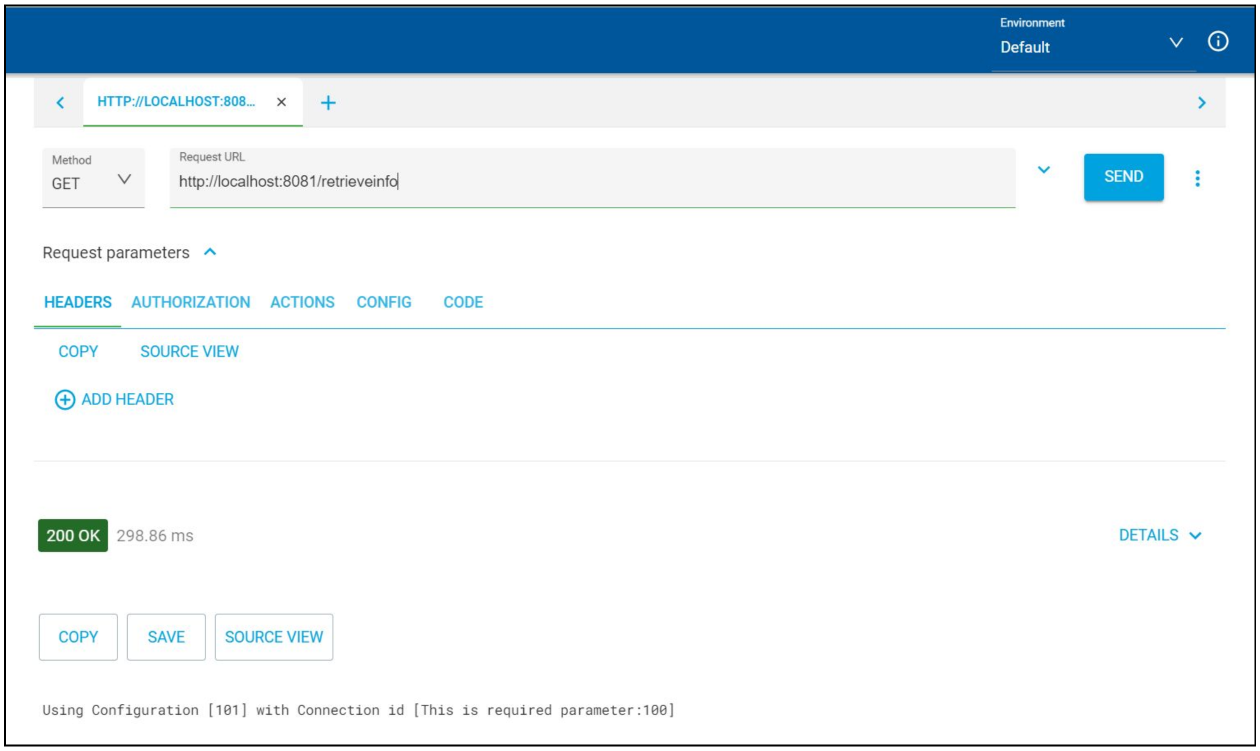


http://localhost:8081/sayhi

200 OK 236.70 ms

```
Hello Sushmitha!!!
```

Send a HTTP GET request to http://localhost:8081/retrieveinfo to test retrieve-info-flow



http://localhost:8081/retrieveinfo

200 OK 298.86 ms

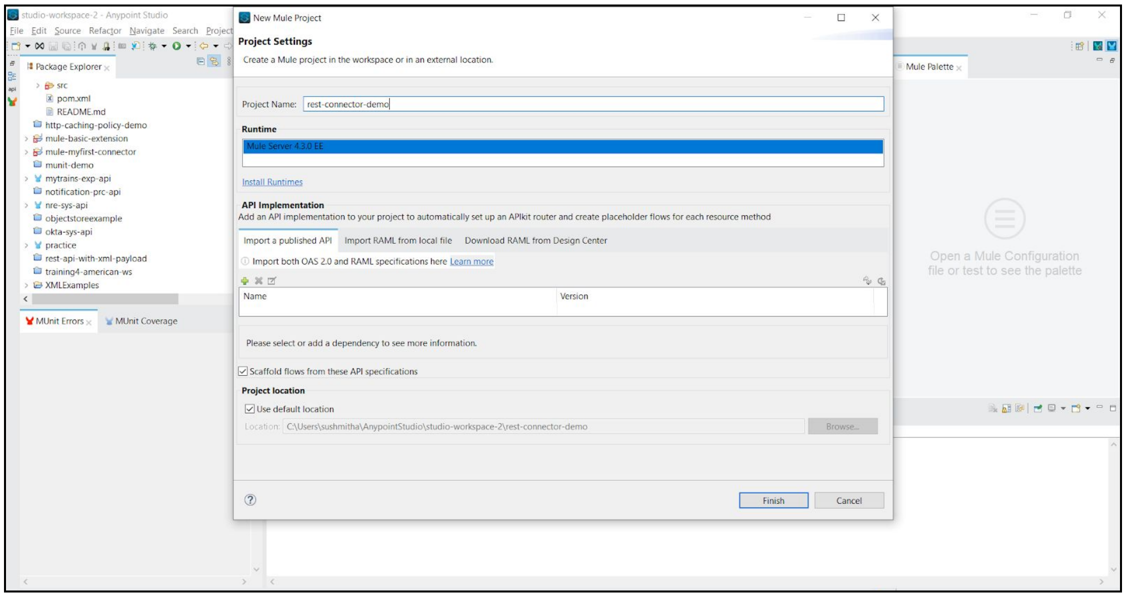
```
Using Configuration [101] with Connection id [This is required parameter:100]
```

3.5.2. REST Connector

Consume a rest connector in anypoint studio for the already published API in exchange

Following are the steps to consume a Salesforce REST connector:

1. Create a Mule Project.



New Mule Project

Project Settings

Create a Mule project in the workspace or in an external location.

Project Name: rest-connector-demo

Runtime: Mule Server 4.3.0 EE

API Implementation

Add an API implementation to your project to automatically set up an APIkit router and create placeholder flows for each resource method

Import both OAS 2.0 and RAML specifications here Learn more

Please select or add a dependency to see more information.

☑ Scaffold flows from these API specifications

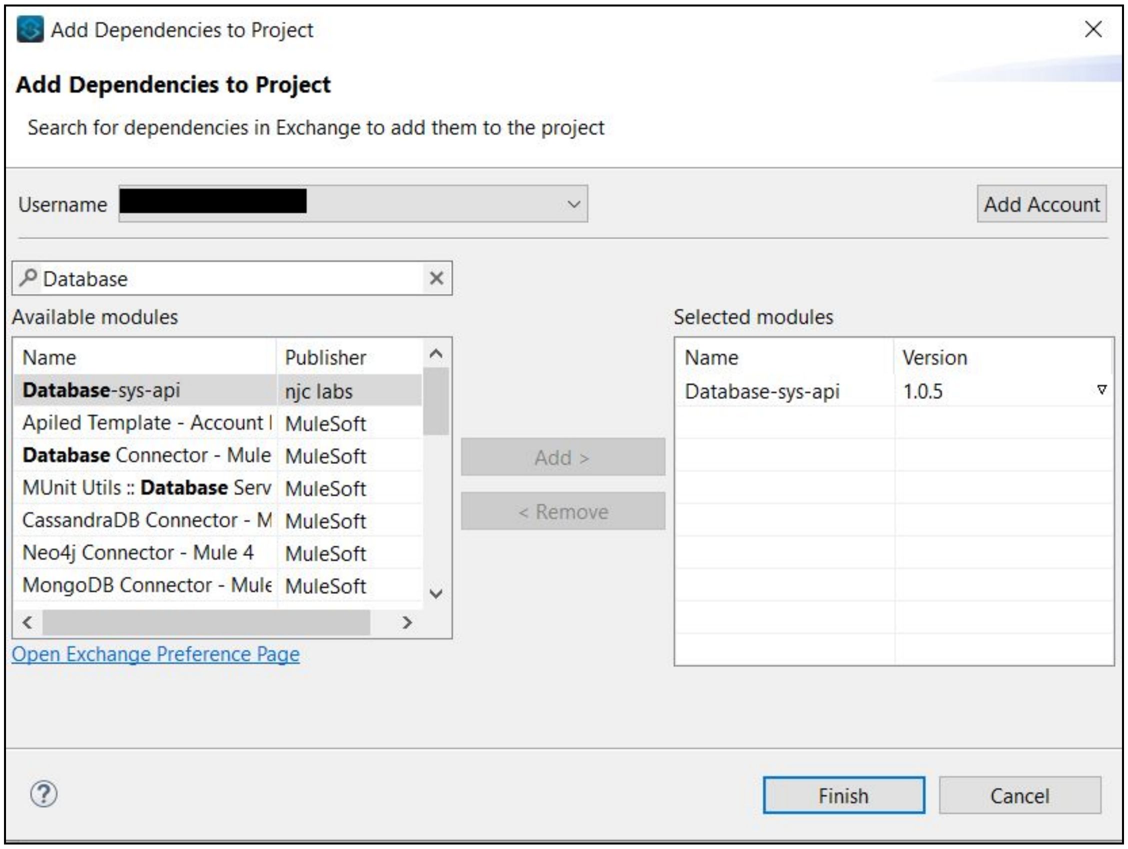
Project location

☑ Use default location

Location: C:\Users\sushmitha\AnypointStudio\studio-workspace-2\rest-connector-demo

Fig 5.1: Creating Mule Project

2. Search Connector from Exchange(Here, I have added *Database-sys-api* that I published in my Anypoint Exchange).
3. Add Connector to Project (here, *Database-sys-api)*. The connector and its operations are added to Mule Palette.


Add Dependencies to Project

Add Dependencies to Project

Search for dependencies in Exchange to add them to the project

Username [redacted]

Database

Available modules

Name	Publisher
Database-sys-api	njc labs
Apiled Template - Account I	MuleSoft
Database Connector - Mule	MuleSoft
MUnit Utils :: **Database** Serv	MuleSoft
CassandraDB Connector - M	MuleSoft
Neo4j Connector - Mule 4	MuleSoft
MongoDB Connector - Mule	MuleSoft

Selected modules

Name	Version
Database-sys-api	1.0.5

Fig 5.2: Adding Connector to Project

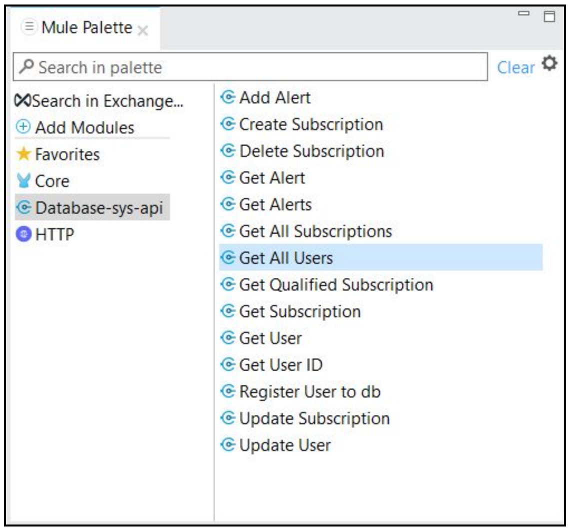
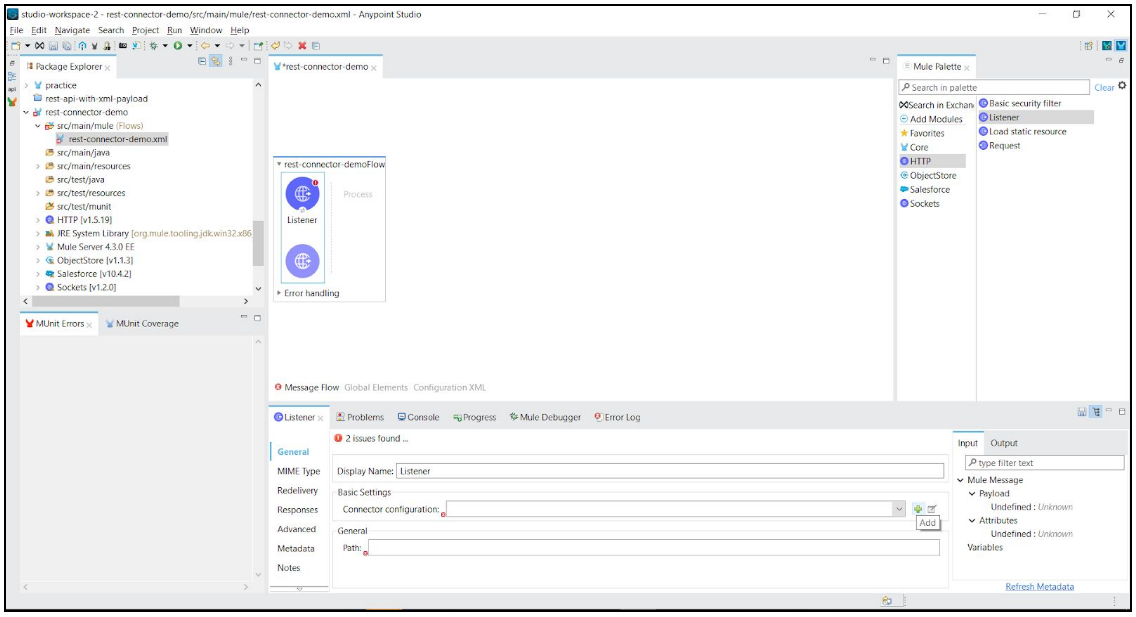


Fig 5.3: Connector in Mule Palette

4. Add an input source, HTTP listener to the project to create a mule flow.



2 issues found ...

Display Name: Listener

Basic Settings

Connector configuration:

General

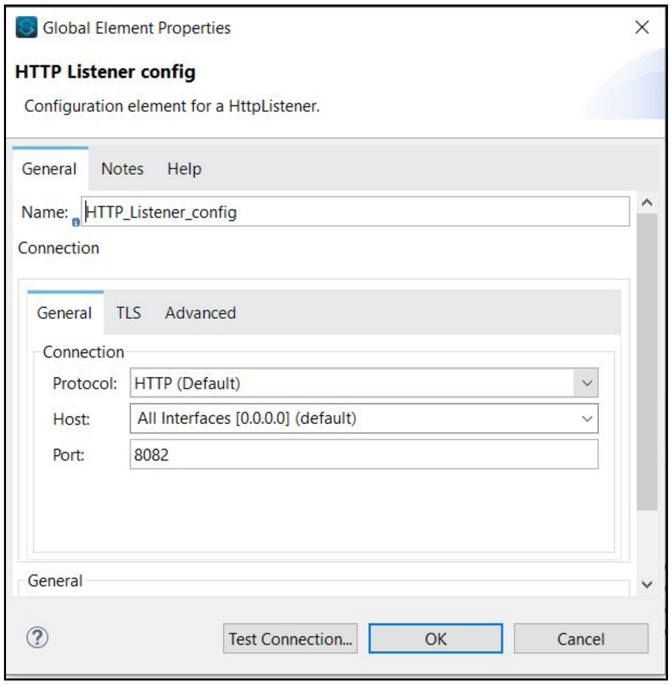
Path:

Payload Undefined : Unknown

Attributes Undefined : Unknown

Fig 5.4: Add input source (HTTP listener)

5. Create HTTP Connector Configuration with default values. (Here, I have changed the port to 8081 since my API is deployed at 8081)



HTTP Listener config

Configuration element for a HttpListener.

Name: HTTP_Listener_config

Connection

Connection

Protocol: HTTP (Default)

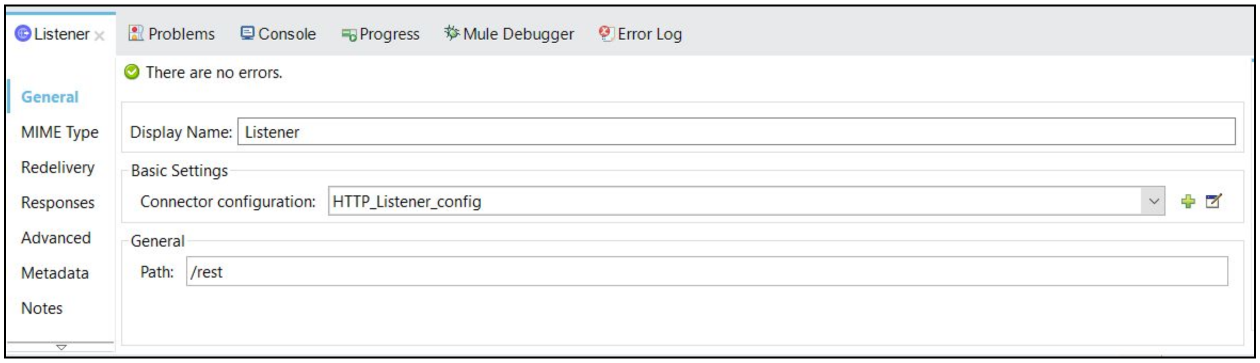
Host: All Interfaces [0.0.0.0] (default)

Port: 8082

General

Fig 5.5: Create HTTP Connector configuration

6. Provide the path to listen, like '/rest'.



There are no errors.

Display Name: Listener

Basic Settings

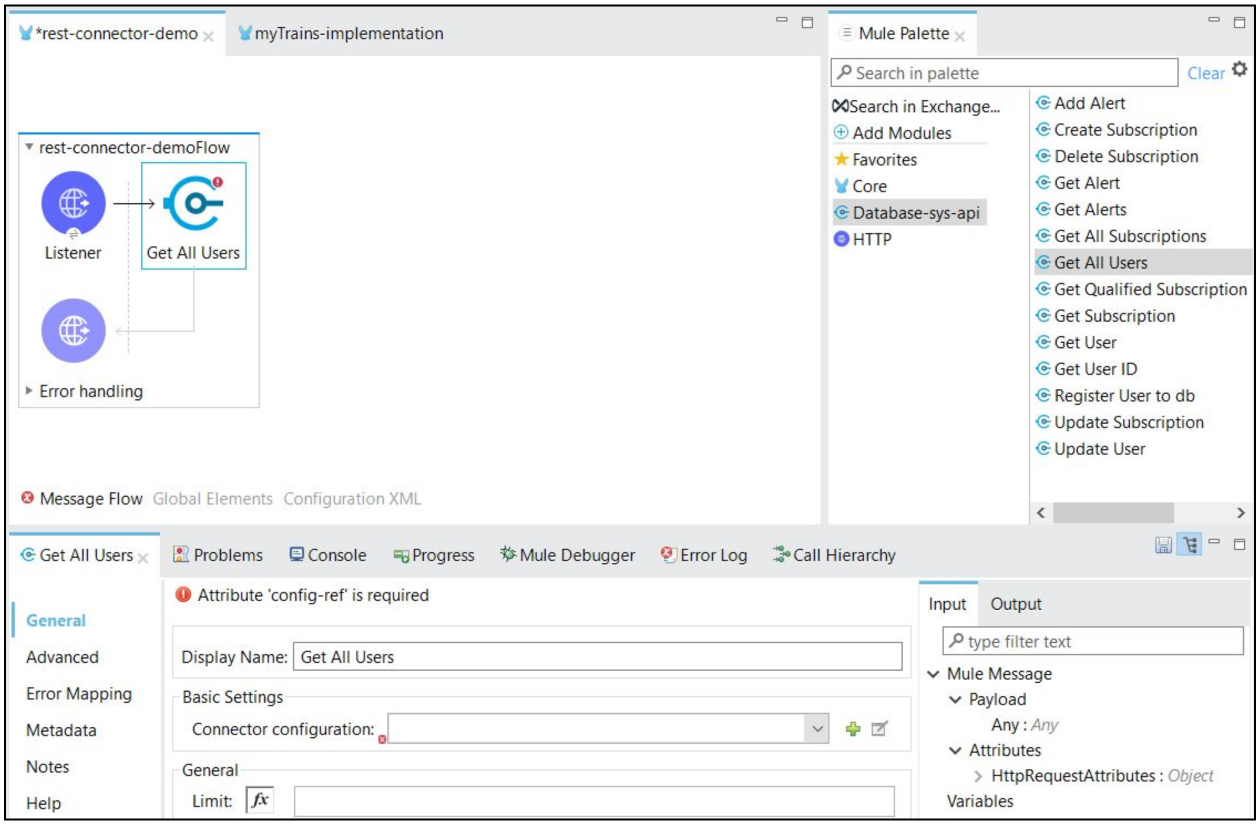
Connector configuration: HTTP_Listener_config

General

Path: /rest

Fig 5.6: Provide HTTP Listener Path

7. Drag one of the Connector's operations to mule flow (here, Get All).



Attribute 'config-ref' is required

Display Name: Get All Users

Basic Settings

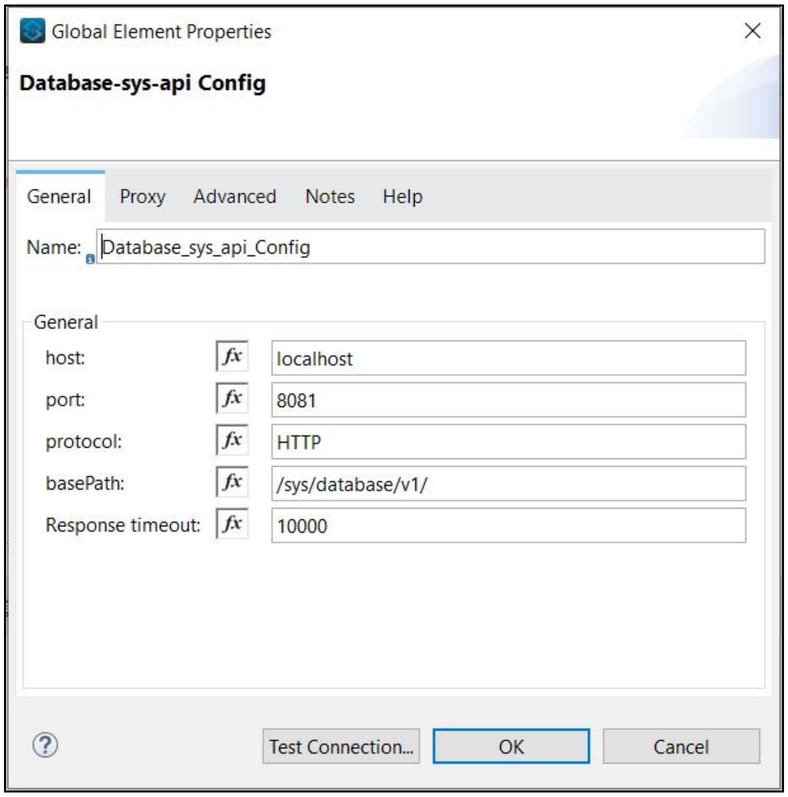
Connector configuration:

General

Limit:

Fig 5.7: Add Connector to mule flow

8. Create Connector configuration by providing necessary parameter values.


Global Element Properties

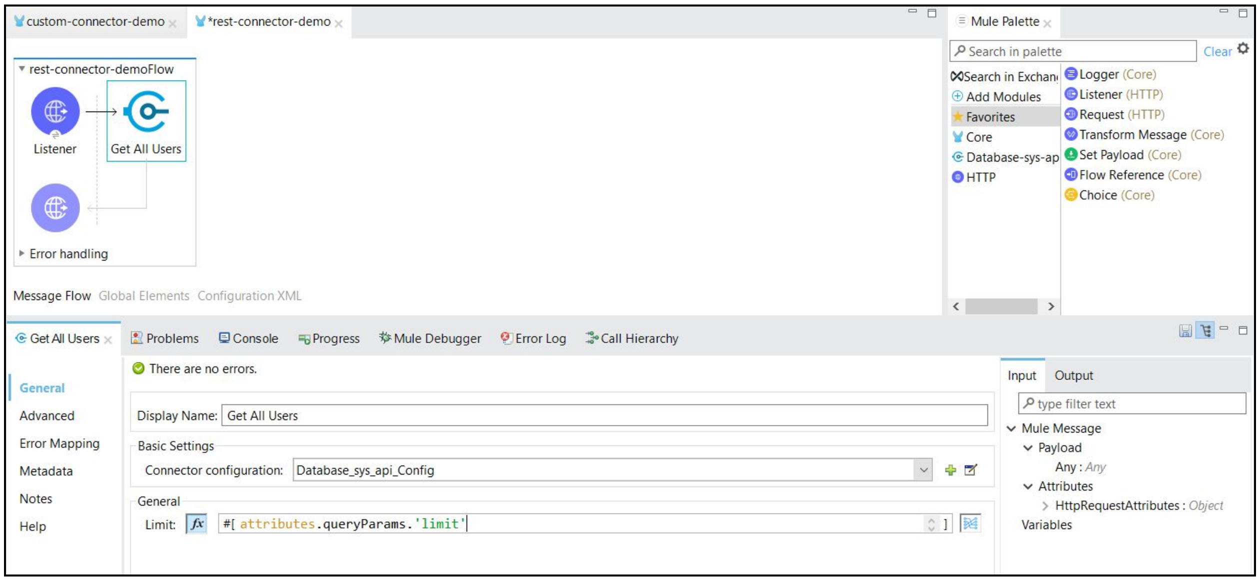
Database-sys-api Config

Name: Database_sys_api_Config

General
- host: localhost
- port: 8081
- protocol: HTTP
- basePath: /sys/database/v1/
- Response timeout: 10000

Fig 5.8: Create Database-sys-api Connector Configuration

9. Add any required query parameters or payload to be passed to the connector. Here, the connector requests a parameter, ‘limit’.



There are no errors.

Display Name: Get All Users

Basic Settings

Connector configuration: Database_sys_api_Config

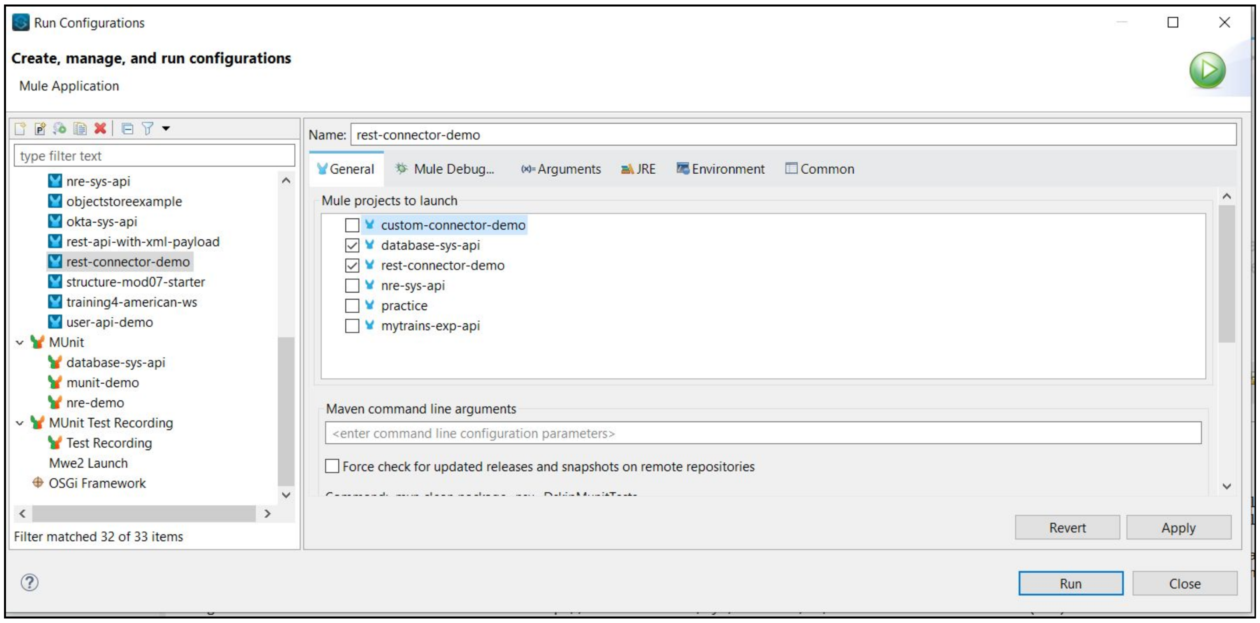
General

Limit: `#[ attributes.queryParams.'limit' ]`

Fig 5.9: Add parameters to connector

10. Save and run the application.
    Here, since my API is not deployed to cloudhub and is located in the located system. I deployed both the applications: Database-sys-api and rest-connector-demo. A maximum

of three applications can be run without increasing the meta size of the runtime by selecting the applications in the Run-> Run Configurations.



Run Configurations

Create, manage, and run configurations

Mule Application

Name: rest-connector-demo

General | Mule Debug... | Arguments | JRE | Environment | Common

Mule projects to launch

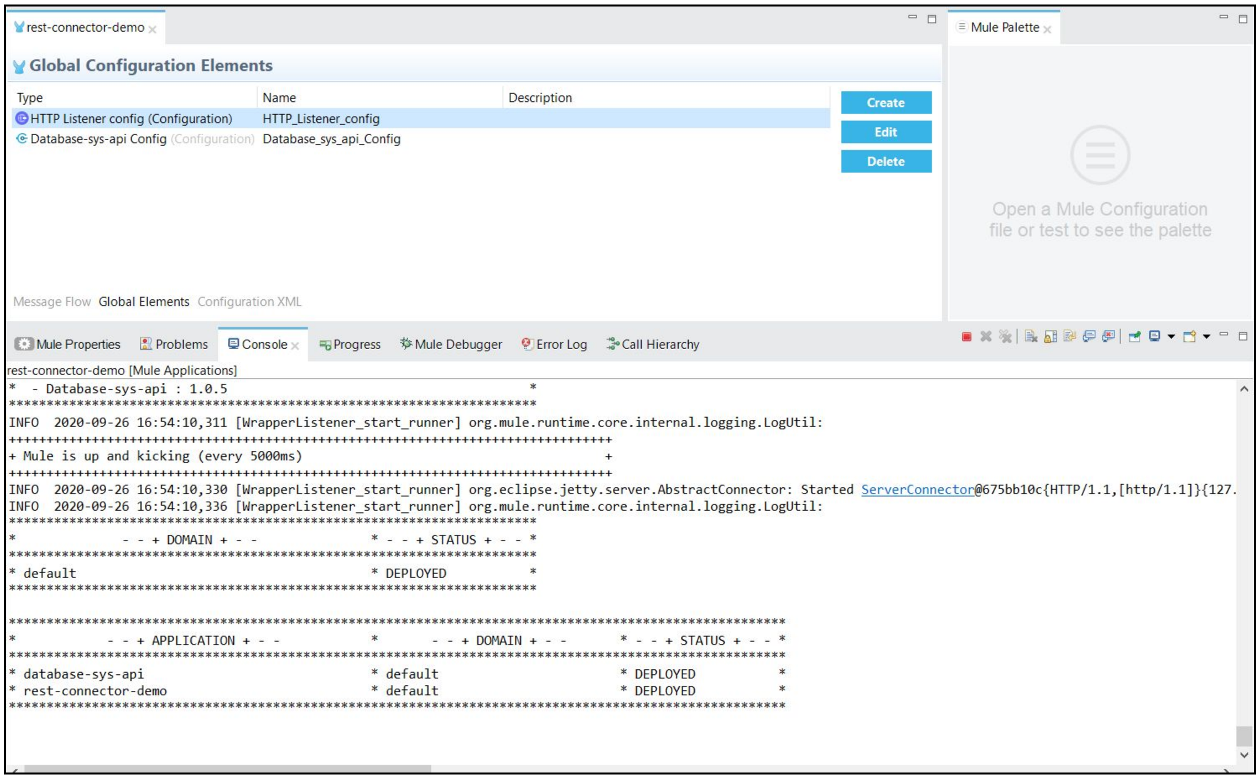
- [ ] custom-connector-demo
- [x] database-sys-api
- [x] rest-connector-demo
- [ ] nre-sys-api
- [ ] practice
- [ ] mytrains-exp-api

Maven command line arguments

<enter command line configuration parameters>

- [ ] Force check for updated releases and snapshots on remote repositories

Fig 5.10: Selecting multiple applications to run



rest-connector-demo

Global Configuration Elements

Type	Name	Description
HTTP Listener config (Configuration)	HTTP_Listener_config	
Database-sys-api Config (Configuration)	Database_sys_api_Config	

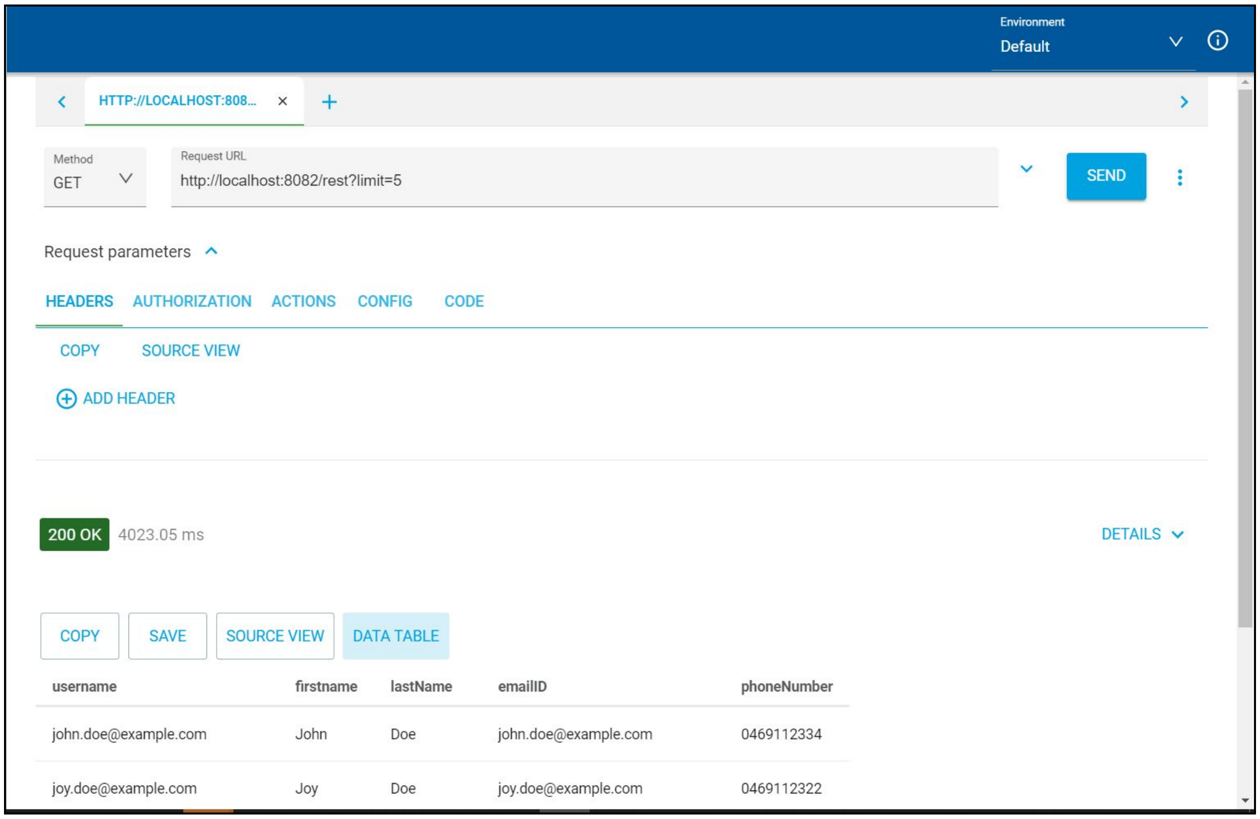
rest-connector-demo [Mule Applications]

```
*  - Database-sys-api : 1.0.5                                         *
**********************************************************************
INFO  2020-09-26 16:54:10,311 [WrapperListener_start_runner] org.mule.runtime.core.internal.logging.LogUtil:
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+ Mule is up and kicking (every 5000ms)                                         +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
INFO  2020-09-26 16:54:10,330 [WrapperListener_start_runner] org.eclipse.jetty.server.AbstractConnector: Started ServerConnector@675bb10c{HTTP/1.1,[http/1.1]}{127.
INFO  2020-09-26 16:54:10,336 [WrapperListener_start_runner] org.mule.runtime.core.internal.logging.LogUtil:
**********************************************************************
*              - - + DOMAIN + - -               * - - + STATUS + - - *
**********************************************************************
* default                                       * DEPLOYED           *
**********************************************************************

*******************************************************************************************************
*            - - + APPLICATION + - -            *       - - + DOMAIN + - -       * - - + STATUS + - - *
*******************************************************************************************************
* database-sys-api                              * default                        * DEPLOYED           *
* rest-connector-demo                           * default                        * DEPLOYED           *
*******************************************************************************************************
```

Fig 5.10: Console log when application successfully deployed

11. Test the application from one of the HTTP/REST client tools like Advanced REST Client (ARC) or Postman. Here, the REST client is ARC.

Send a HTTP GET request with query parameter 'limit' to http://localhost:8081/rest to test the connector.



http://localhost:8082/rest?limit=5

200 OK 4023.05 ms

username	firstname	lastName	emailID	phoneNumber
john.doe@example.com	John	Doe	john.doe@example.com	0469112334
joy.doe@example.com	Joy	Doe	joy.doe@example.com	0469112322

Fig 5.11: Test application in ARC

References

- https://help.mulesoft.com/s/question/0D52T0000597ei3/need-helpwhat-is-common-connector-in-mule-4
- https://blogs.mulesoft.com/dev/connectivity-dev/top-download-connectors-anypoint-platform/
- https://docs.mulesoft.com/exchange/to-deploy-using-rest-connect
- https://docs.mulesoft.com/connector-devkit/3.6/implementing-a-rest-connector
- https://www.mulesoft.com/resources/api/restful-api
- https://www.mulesoft.com/webinars/api/soap-connect
- https://docs.mulesoft.com/exchange/to-deploy-using-rest-connect
- https://docs.mulesoft.com/connector-devkit/3.9/creating-an-anypoint-connector-project
- https://dzone.com/articles/mulesoft-custom-connector-using-mule-sdk-for-mule
- https://dzone.com/articles/how-to-create-a-custom-connector-in-new-mule-sdk-4
- https://dzone.com/articles/mulesoft-custom-connector-using-mule-sdk-for-mule
- https://docs.mulesoft.com/mule-sdk/1.1/getting-started
- https://docs.mulesoft.com/mule-sdk/1.1/xml-sdk#create-and-test-an-xml-sdk-project
- http://www.slf4j.org/codes.html#StaticLoggerBinder
- https://dzone.com/articles/adding-slf4j-your-maven