Data-Services

Astera Software

Sep 24, 2023

GETTING STARTED

CHAPTER

ONE

ASTERA API MANAGEMENT – SYSTEM REQUIREMENTS

Note: The overall speed and performance of the application depend on the configuration of your machine. More memory and higher processing speed on the system will result in faster performance, especially when transferring large amounts of data as the application takes advantage of the multicore hardware to parallelize operations.

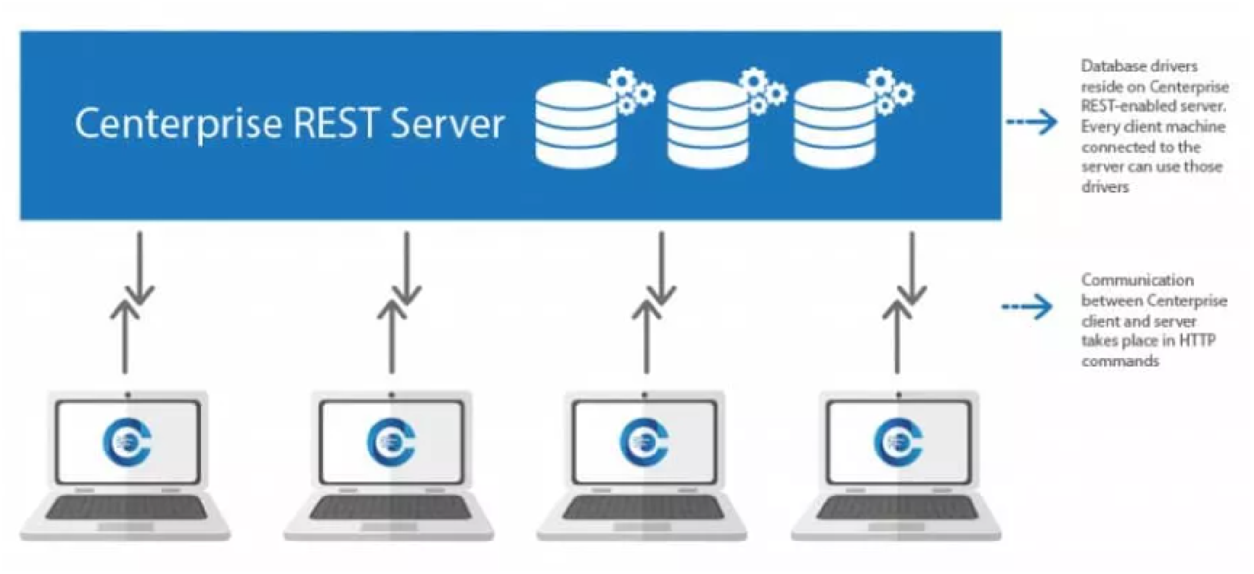
CHAPTER

TWO

ASTERA API MANAGEMENT – PRODUCT ARCHITECTURE

Astera API Management is built on a client-server architecture. The client is the part of the application which a user can run locally on their machine, whereas the server performs processing and querying requested by the client. In simple words, the client sends a request to the server, and the server, in turn, responds to the request. Therefore, database drivers are installed only on the Centerprise server. This enables horizontal scaling by adding multiple clients to an existing cluster of servers and eliminating the need to install drivers on every machine.

The Astera API Management client and server applications communicate on REST architecture. REST-compliant systems, often called RESTful systems, are characterized by statelessness and separate concerns of the client and server, which means that the implementation of both can be done independently if each side knows what format of messages to send to the other. The server communicates with the client using HTTPS commands, which are encrypted using a certified key/certificate signed by an authority. This saves the data from being intercepted by an attacker as the plaintext is encrypted as a random string of characters.


Centerprise REST Server
Database drivers reside on Centerprise REST-enabled server. Every client machine connected to the server can use those drivers
Communication between Centerprise client and server takes place in HTTP commands

CHAPTER
THREE

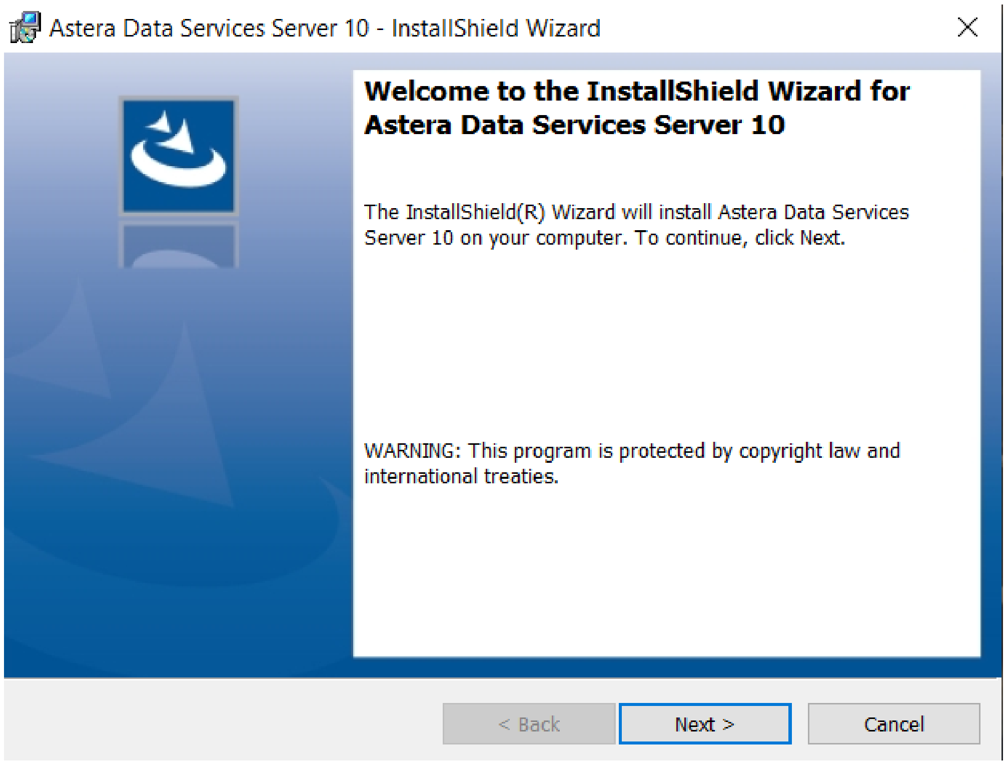
INSTALLING CLIENT AND SERVER APPLICATIONS

In this section, we will discuss how to install and configure Astera API Management Server and Centerprise Lean Client applications.

3.1 How to Install Data Services Server

1. Run ‘DataServicesServer.exe’ from the installation package to start the server installation setup.

2. You’ll be directed to the welcome screen. Click *Next* to continue.



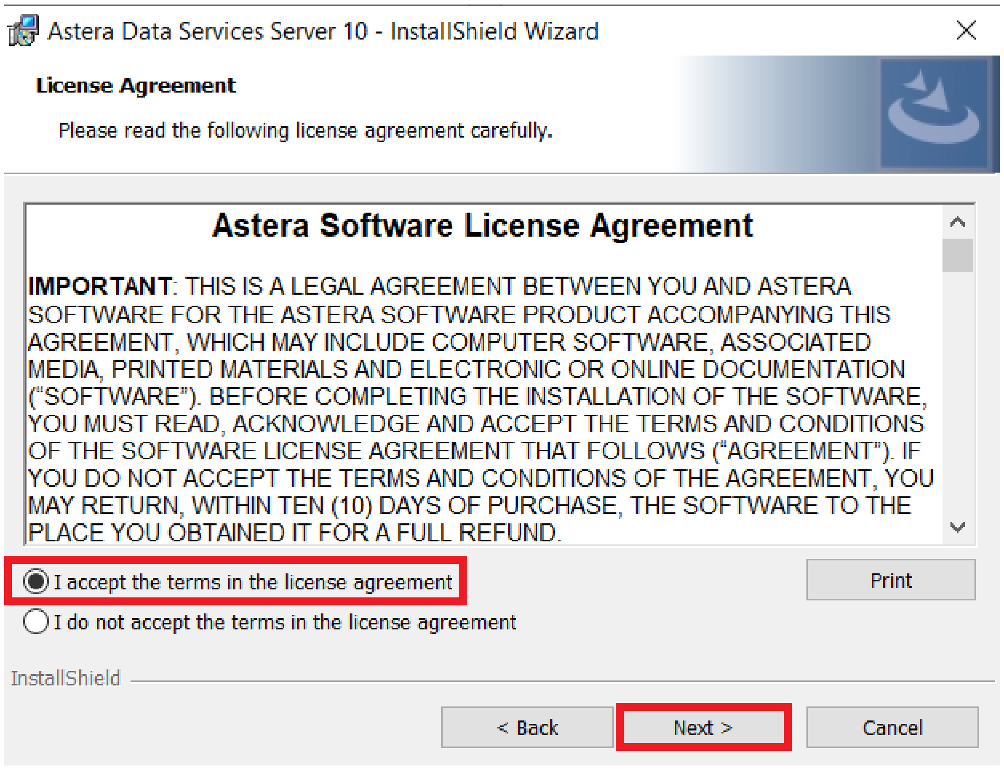
Astera Data Services Server 10 - InstallShield Wizard

Welcome to the InstallShield Wizard for Astera Data Services Server 10

The InstallShield(R) Wizard will install Astera Data Services Server 10 on your computer. To continue, click Next.

WARNING: This program is protected by copyright law and international treaties.

3. On the next screen you will see the license agreement. You can only continue if you choose to accept the terms of the license agreement. Click *Next* to continue.



Astera Data Services Server 10 - InstallShield Wizard

License Agreement

Please read the following license agreement carefully.

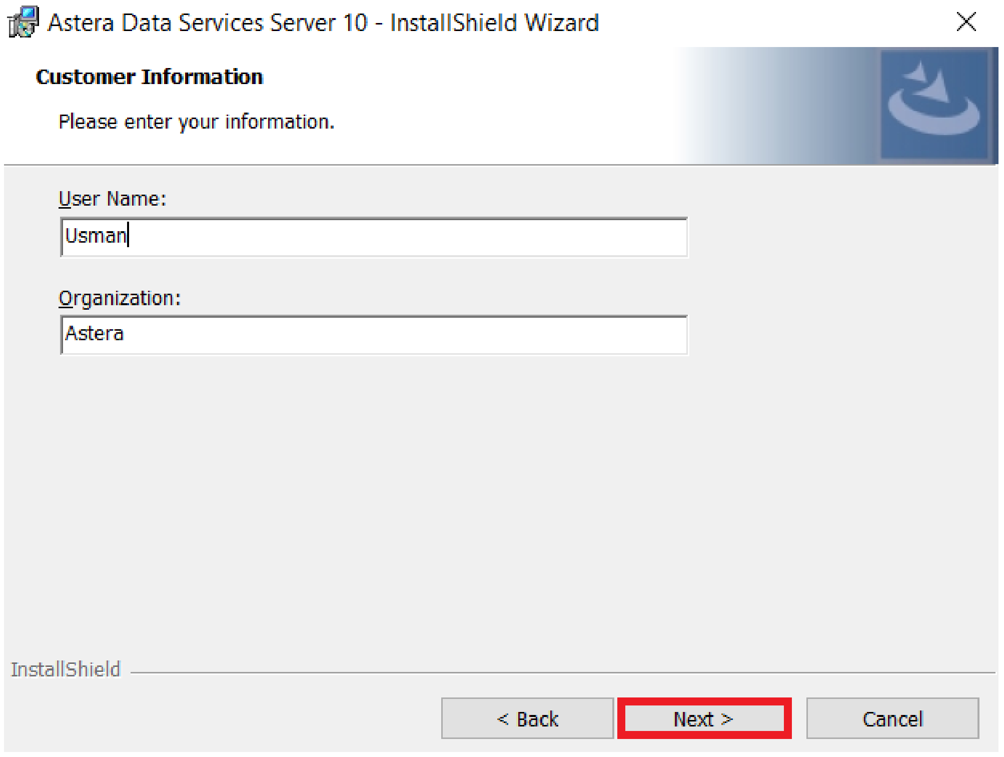
Astera Software License Agreement

IMPORTANT: THIS IS A LEGAL AGREEMENT BETWEEN YOU AND ASTERA SOFTWARE FOR THE ASTERA SOFTWARE PRODUCT ACCOMPANYING THIS AGREEMENT, WHICH MAY INCLUDE COMPUTER SOFTWARE, ASSOCIATED MEDIA, PRINTED MATERIALS AND ELECTRONIC OR ONLINE DOCUMENTATION ("SOFTWARE"). BEFORE COMPLETING THE INSTALLATION OF THE SOFTWARE, YOU MUST READ, ACKNOWLEDGE AND ACCEPT THE TERMS AND CONDITIONS OF THE SOFTWARE LICENSE AGREEMENT THAT FOLLOWS ("AGREEMENT"). IF YOU DO NOT ACCEPT THE TERMS AND CONDITIONS OF THE AGREEMENT, YOU MAY RETURN, WITHIN TEN (10) DAYS OF PURCHASE, THE SOFTWARE TO THE PLACE YOU OBTAINED IT FOR A FULL REFUND.

[✓ I accept the terms in the license agreement]

I do not accept the terms in the license agreement

4. On the next screen, enter the user details and click *Next* to continue.



Astera Data Services Server 10 - InstallShield Wizard

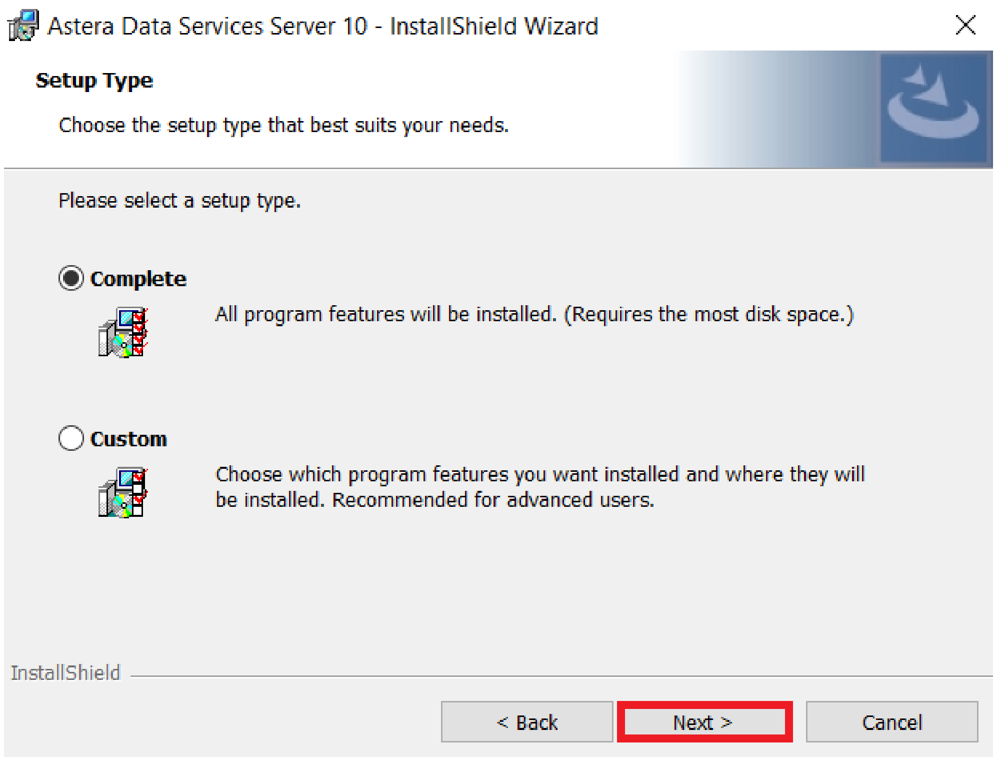
Customer Information

Please enter your information.

User Name: Usman

Organization: Astera

5. Select the type of installation (Complete or Custom) you want to proceed with and click *Next*.



Astera Data Services Server 10 - InstallShield Wizard

Setup Type

Choose the setup type that best suits your needs.

Please select a setup type.

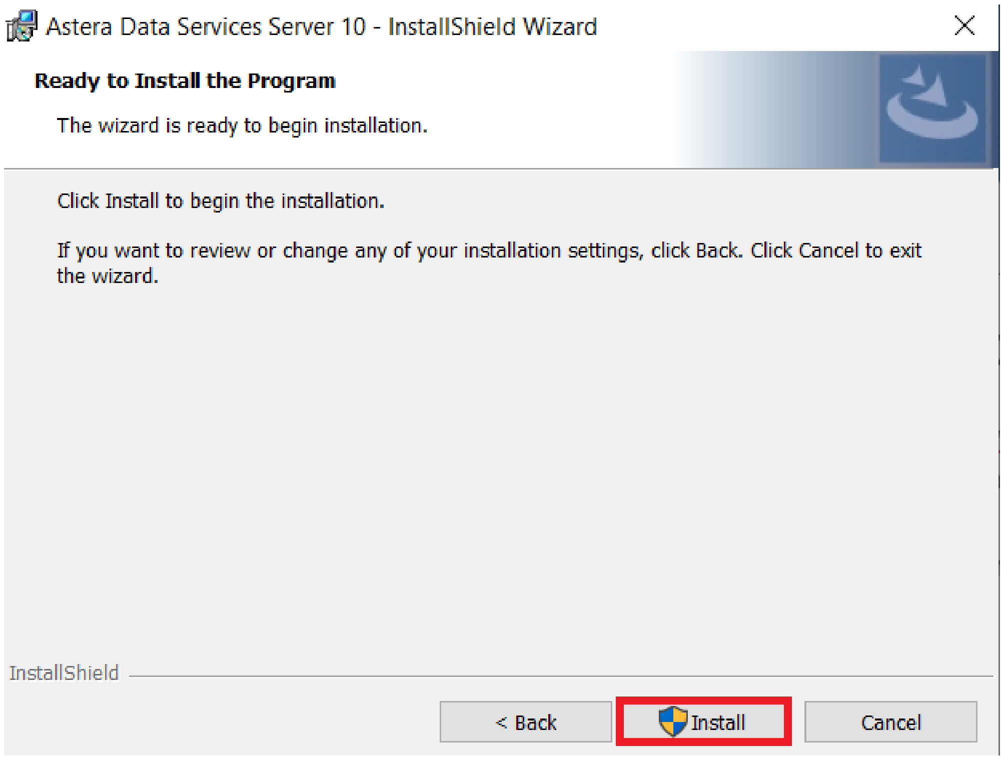
[✓ Complete]

All program features will be installed. (Requires the most disk space.)

Custom

Choose which program features you want installed and where they will be installed. Recommended for advanced users.

6. Select *Install* to complete the installation.



Astera Data Services Server 10 - InstallShield Wizard

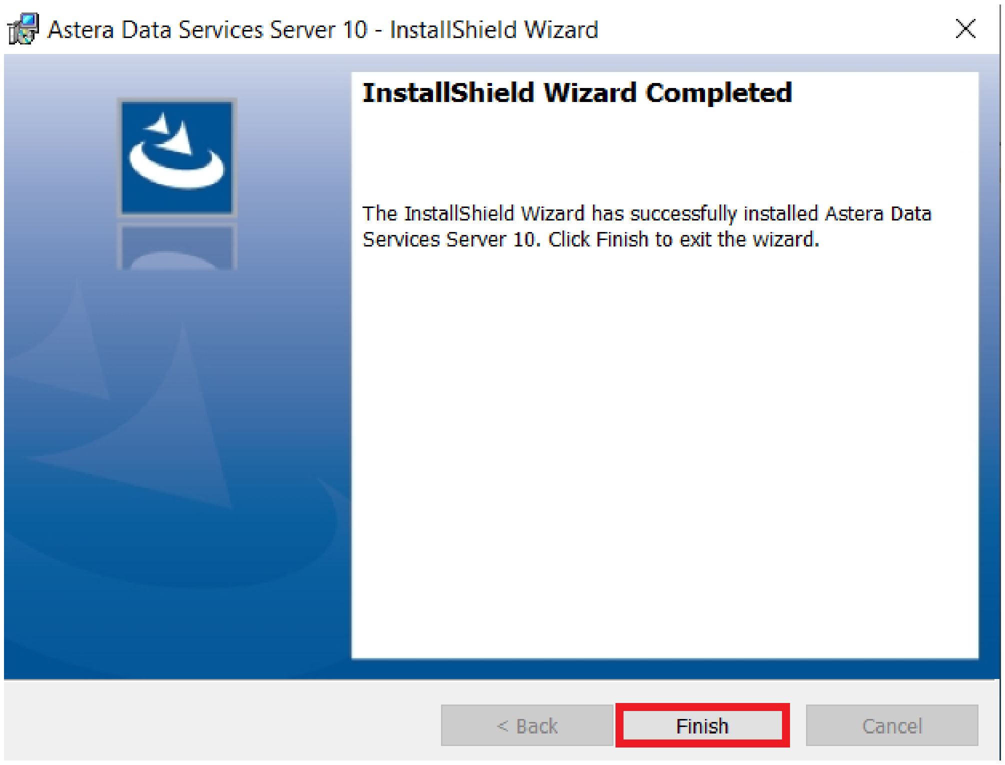
Ready to Install the Program

The wizard is ready to begin installation.

Click Install to begin the installation.

If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.

7. Select *Finish* to finish the installation process.

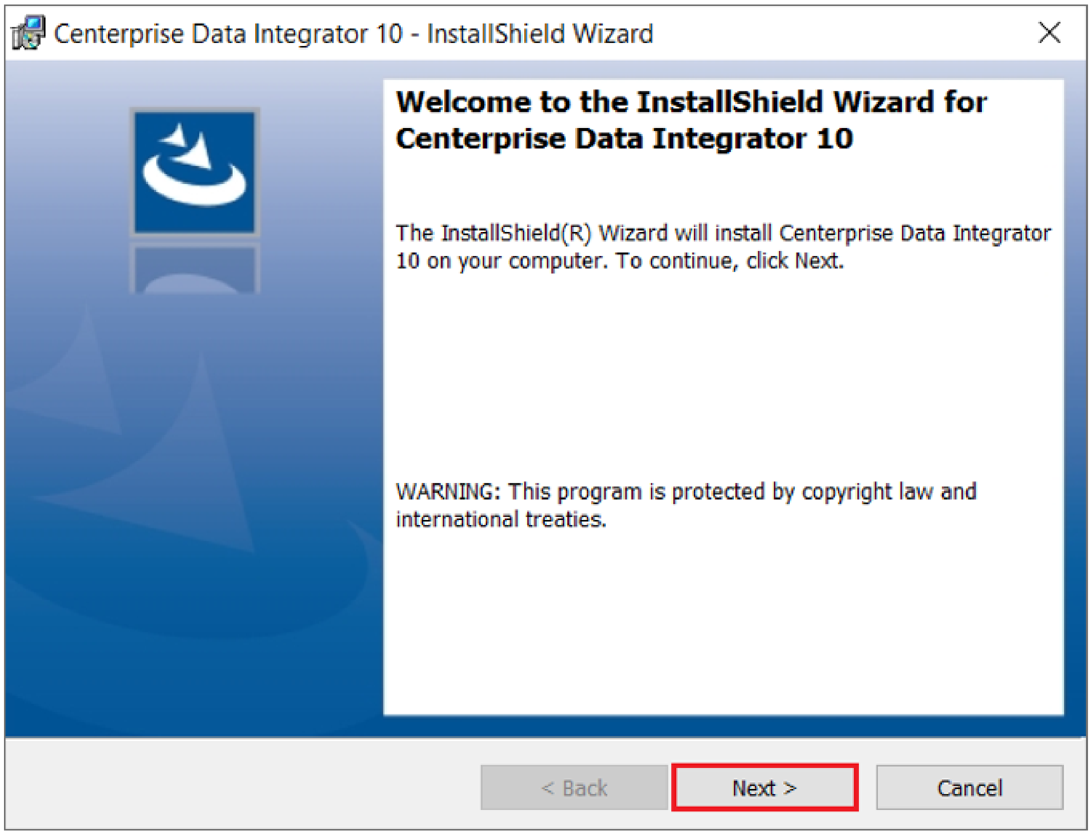

Astera Data Services Server 10 - InstallShield Wizard

InstallShield Wizard Completed

The InstallShield Wizard has successfully installed Astera Data Services Server 10. Click Finish to exit the wizard.

3.2 How to Install Centerprise Lean Client

1. Run the *'CenterpriseDataIntegrator'* application from the installation package to start the client installation setup.
2. You'll be directed to the welcome screen. Click *Next* to continue.

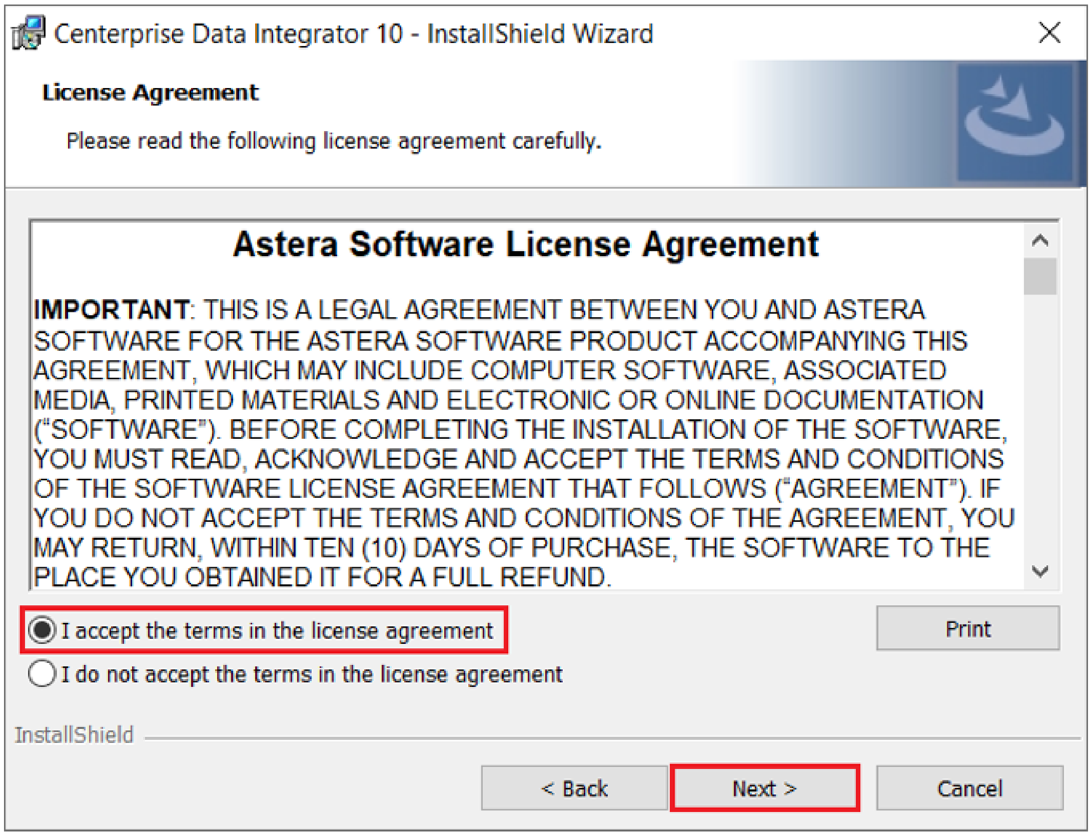

Centerprise Data Integrator 10 - InstallShield Wizard

Welcome to the InstallShield Wizard for Centerprise Data Integrator 10

The InstallShield(R) Wizard will install Centerprise Data Integrator 10 on your computer. To continue, click Next.

WARNING: This program is protected by copyright law and international treaties.

3. On the next screen you will see the license agreement. You can only continue if you choose to accept the terms of the license agreement. Click *Next* to continue.


Centerprise Data Integrator 10 - InstallShield Wizard

License Agreement

Please read the following license agreement carefully.

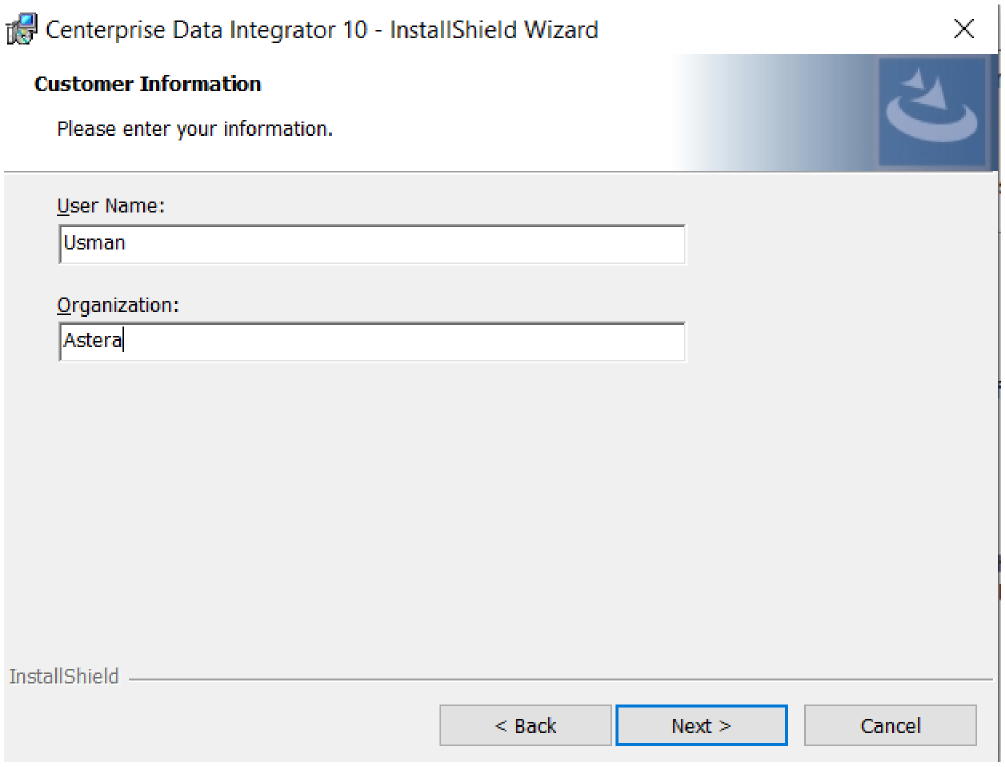
Astera Software License Agreement

IMPORTANT: THIS IS A LEGAL AGREEMENT BETWEEN YOU AND ASTERA SOFTWARE FOR THE ASTERA SOFTWARE PRODUCT ACCOMPANYING THIS AGREEMENT, WHICH MAY INCLUDE COMPUTER SOFTWARE, ASSOCIATED MEDIA, PRINTED MATERIALS AND ELECTRONIC OR ONLINE DOCUMENTATION ("SOFTWARE"). BEFORE COMPLETING THE INSTALLATION OF THE SOFTWARE, YOU MUST READ, ACKNOWLEDGE AND ACCEPT THE TERMS AND CONDITIONS OF THE SOFTWARE LICENSE AGREEMENT THAT FOLLOWS ("AGREEMENT"). IF YOU DO NOT ACCEPT THE TERMS AND CONDITIONS OF THE AGREEMENT, YOU MAY RETURN, WITHIN TEN (10) DAYS OF PURCHASE, THE SOFTWARE TO THE PLACE YOU OBTAINED IT FOR A FULL REFUND.

[✓ I accept the terms in the license agreement]

I do not accept the terms in the license agreement

4. On the next screen, enter the user details and click *Next* to continue.


Centerprise Data Integrator 10 - InstallShield Wizard

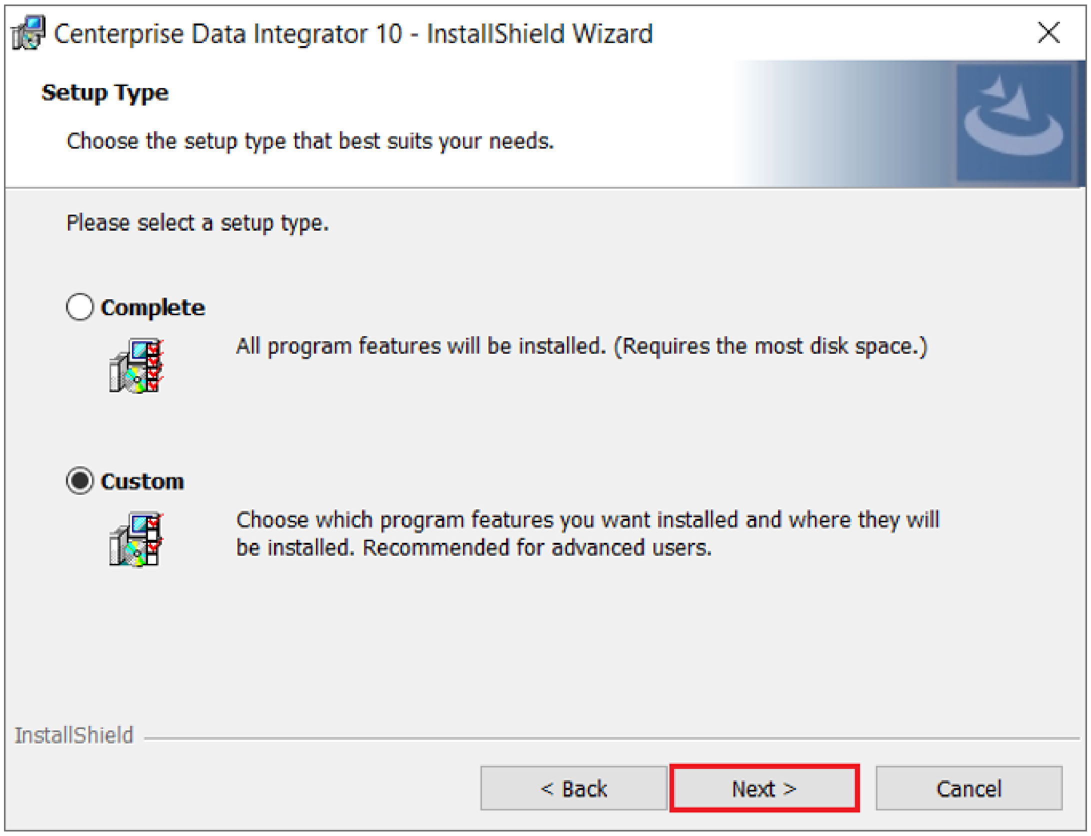
Customer Information

Please enter your information.

User Name: Usman

Organization: Astera

5. Select the type of installation (Complete or Custom) you want to proceed with and click *Next*.



Centerprise Data Integrator 10 - InstallShield Wizard

Setup Type

Choose the setup type that best suits your needs.

Please select a setup type.

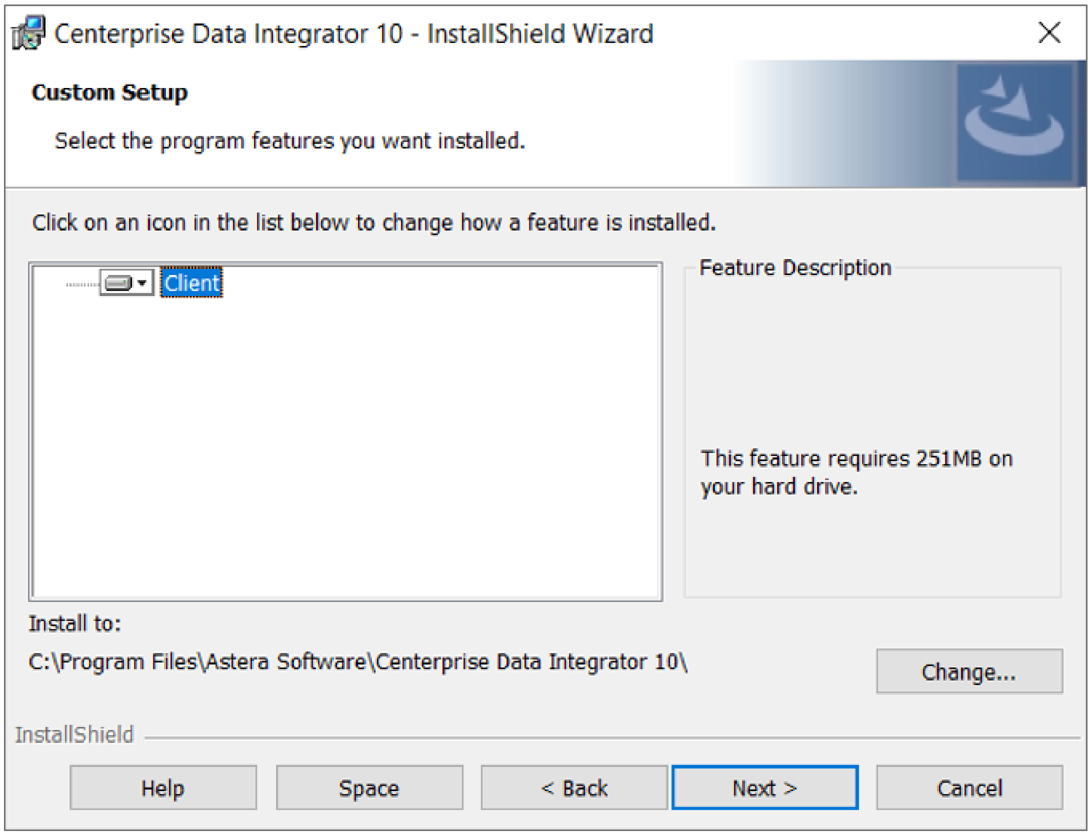
Complete

All program features will be installed. (Requires the most disk space.)

[✓ Custom]

Choose which program features you want installed and where they will be installed. Recommended for advanced users.

If you select custom installation, you can choose specific component(s) that you want to download.


Centerprise Data Integrator 10 - InstallShield Wizard

Custom Setup

Select the program features you want installed.

Click on an icon in the list below to change how a feature is installed.

Client

Feature Description

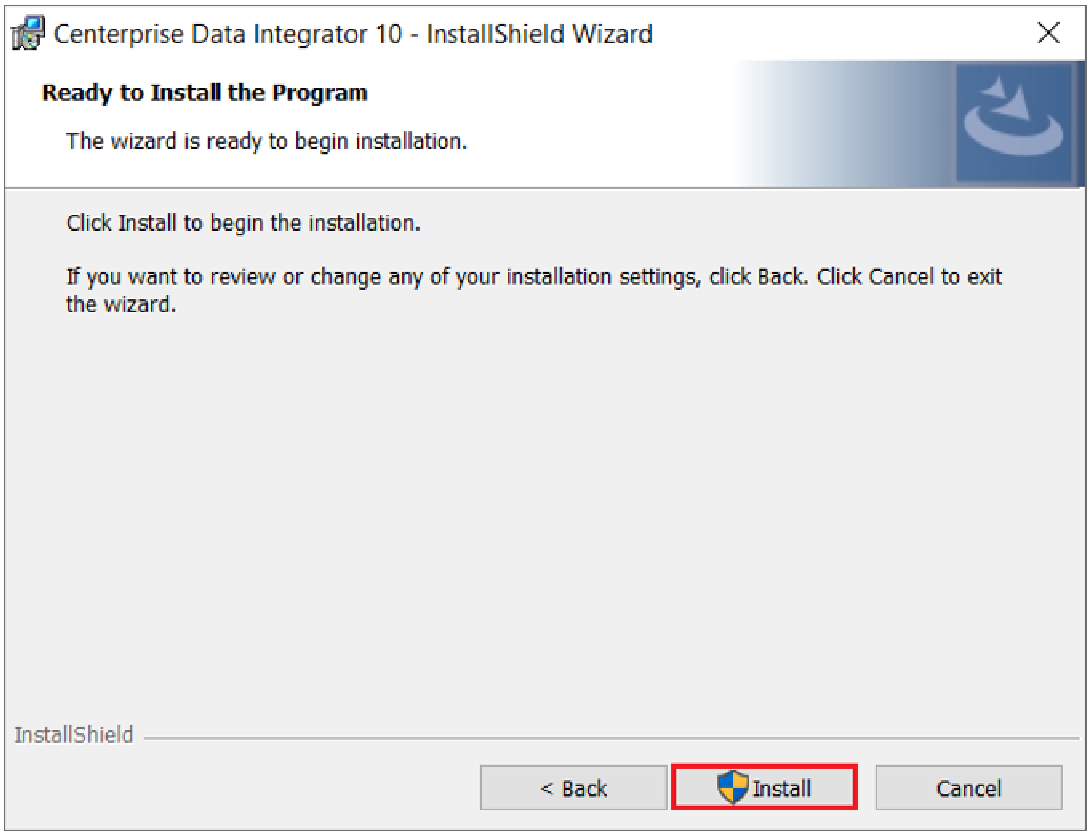
This feature requires 251MB on your hard drive.

Install to:

C:\Program Files\Astera Software\Centerprise Data Integrator 10\

We want to install the complete package therefore, We'll select *Complete* on the *Setup Type* screen and click *Next*.

6. Select *Install* to complete the installation.


Centerprise Data Integrator 10 - InstallShield Wizard

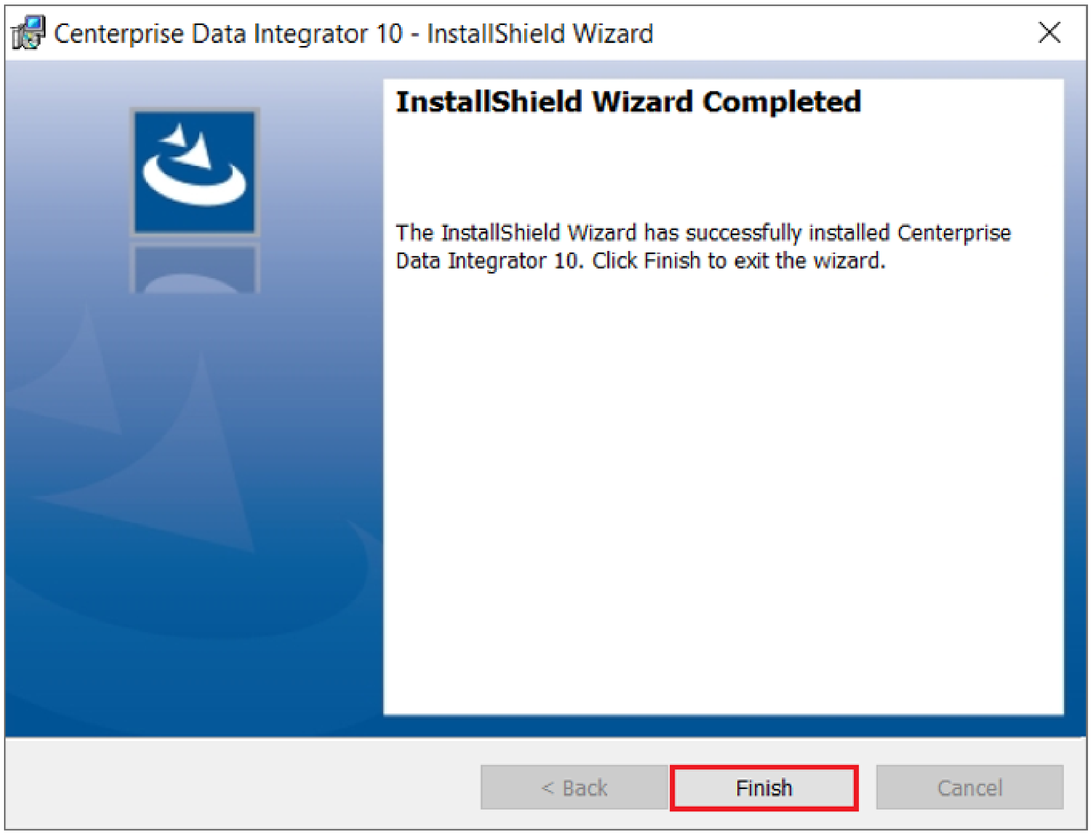
Ready to Install the Program

The wizard is ready to begin installation.

Click Install to begin the installation.

If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.

7. Select *Finish* to finish the installation process.



InstallShield Wizard Completed

The InstallShield Wizard has successfully installed Centerprise Data Integrator 10. Click Finish to exit the wizard.

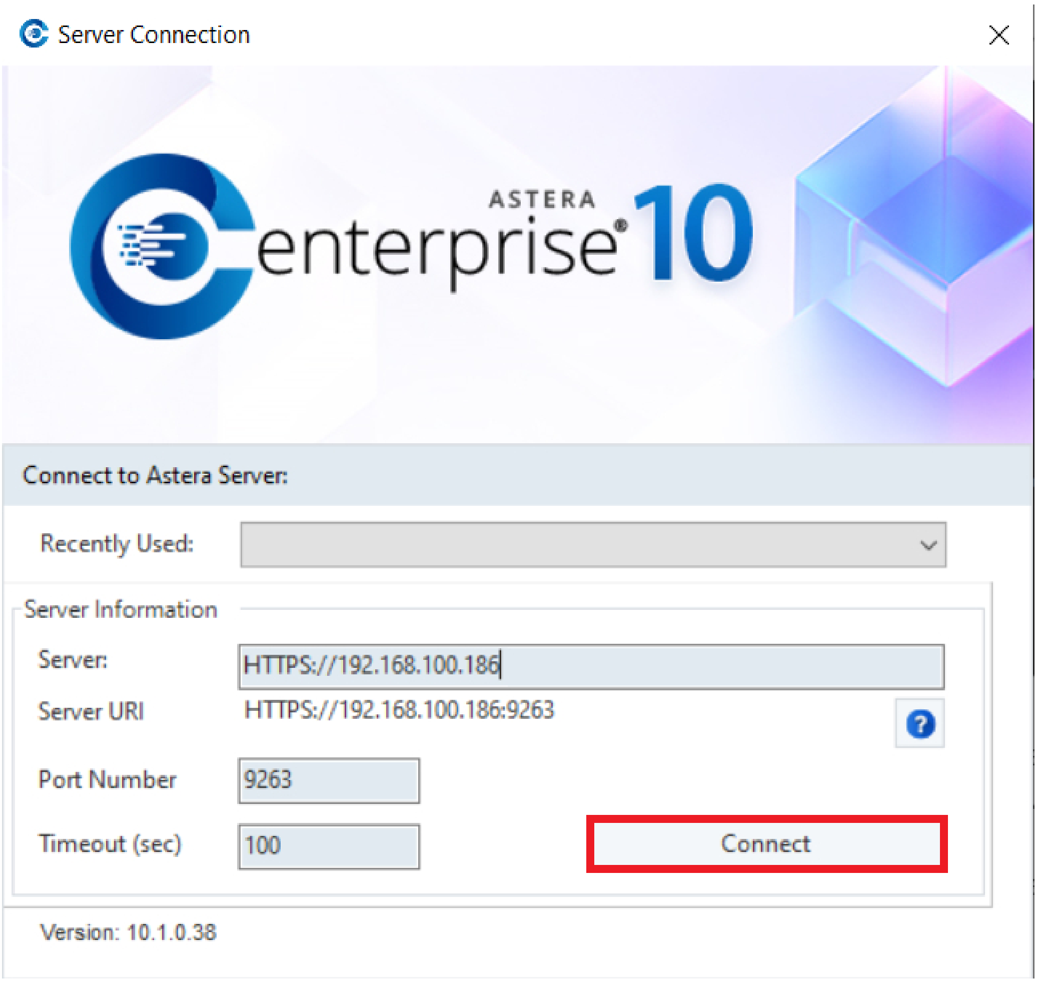
This is how you install Astera API Management Server and Centerprise client applications. The next step is to establish a connection between the client and server.

CHAPTER

FOUR

CONNECTING TO AN ASTERA API MANAGEMENT SERVER USING LEAN CLIENT

After you have successfully installed Centerprise client and Astera API Management server applications, open the client application and you will see the *Server Connection* screen as pictured below.



Server: HTTPS://192.168.100.186

Server URI HTTPS://192.168.100.186:9263

Port Number 9263

Timeout (sec) 100

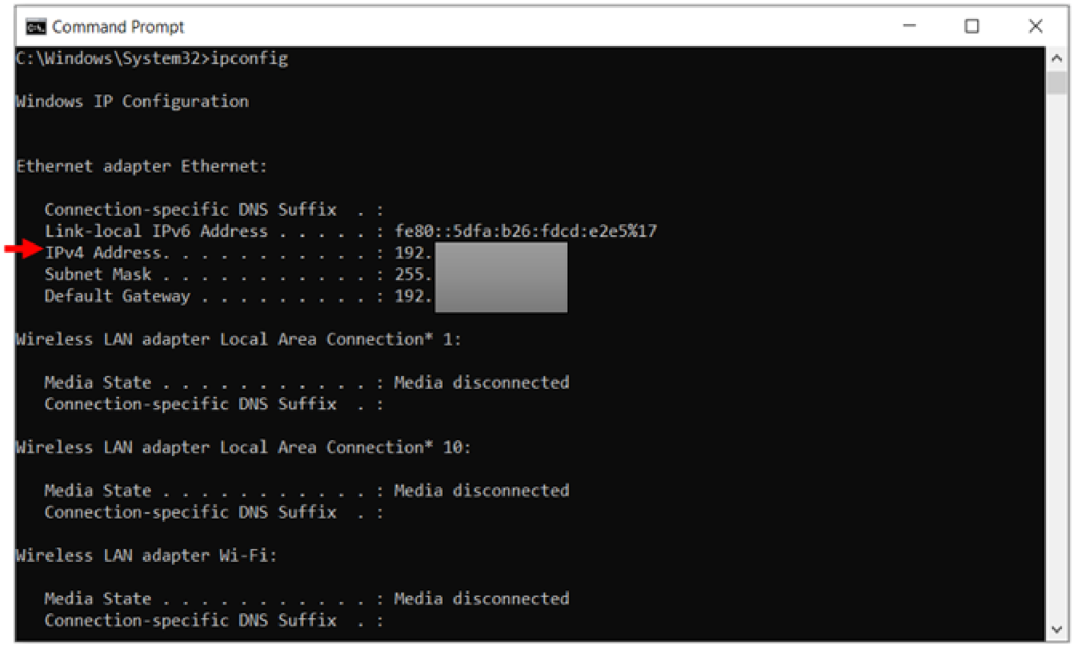
Version: 10.1.0.38

Enter the *Server URI* and *Port Number* to establish the connection.

The server URI will be the IP address of the machine where Astera Data Services Server is installed.

Server URI: (HTTPS://IP_address)

Note: You can get help of your network administrator to get the IP address of the machine where Astera API Management Server is installed. Or you can launch the command prompt and type the command *ipconfig* to get the IP configuration details for the machine and use that information to provide Server URI.



```
C:\Windows\System32>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::5dfa:b26:fdcd:e2e5%17
   IPv4 Address. . . . . . . . . . . : 192.[redacted]
   Subnet Mask . . . . . . . . . . . : 255.
   Default Gateway . . . . . . . . . : 192.

Wireless LAN adapter Local Area Connection* 1:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 10:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Wi-Fi:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
```

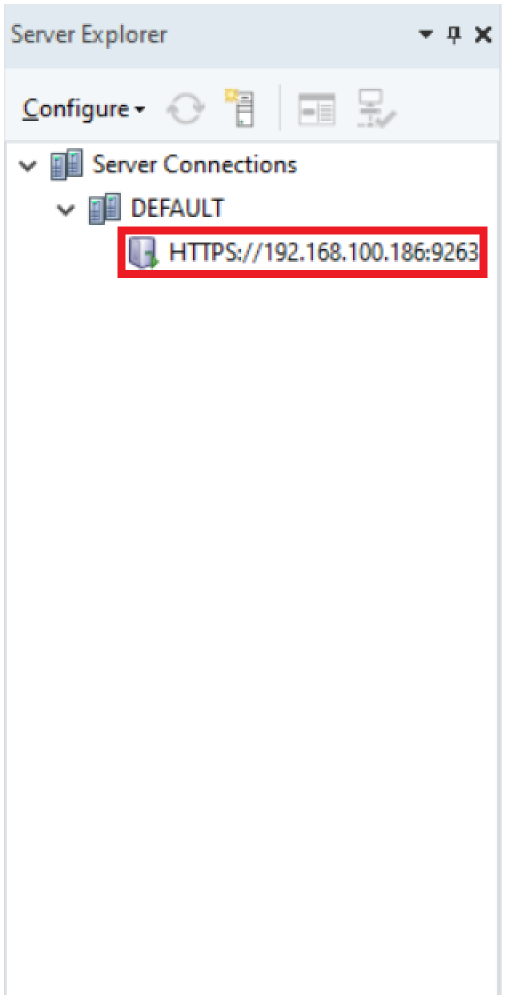
The default port for the secure connection between the Lean client and Astera API Management Server is 9263.

If you have connected to any server recently, you can automatically connect to that server by selecting that server from the *Recently Used* drop-down list.

Click *Connect* after you have filled out the information required.

The client will now connect to the selected server. You should be able to see the server listed in the *Server Explorer* tree when the client application opens.

To open *Server Explorer* go to *Server* > *Server Explorer* or use the keyboard shortcut (Ctrl + Alt + E).



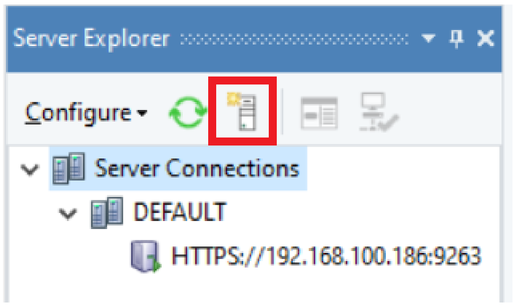
The yellow icon with an exclamation mark means that the server is not configured. Before you can start working with the Centerprise Lean client, you will have to create a repository and configure the server.

CHAPTER

FIVE

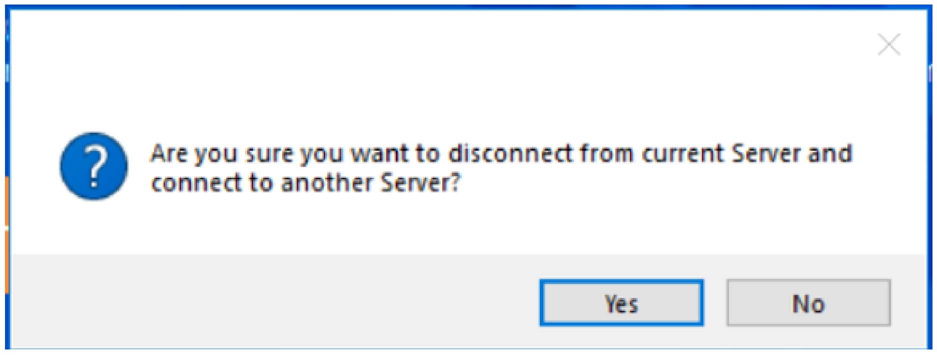
HOW TO CONNECT TO A DIFFERENT ASTERA API MANAGEMENT SERVER FROM THE LEAN CLIENT

You can connect to different servers right from the Server Explorer window in Lean Client. Go to the *Server Explorer* window and click on the Connect to Server icon.



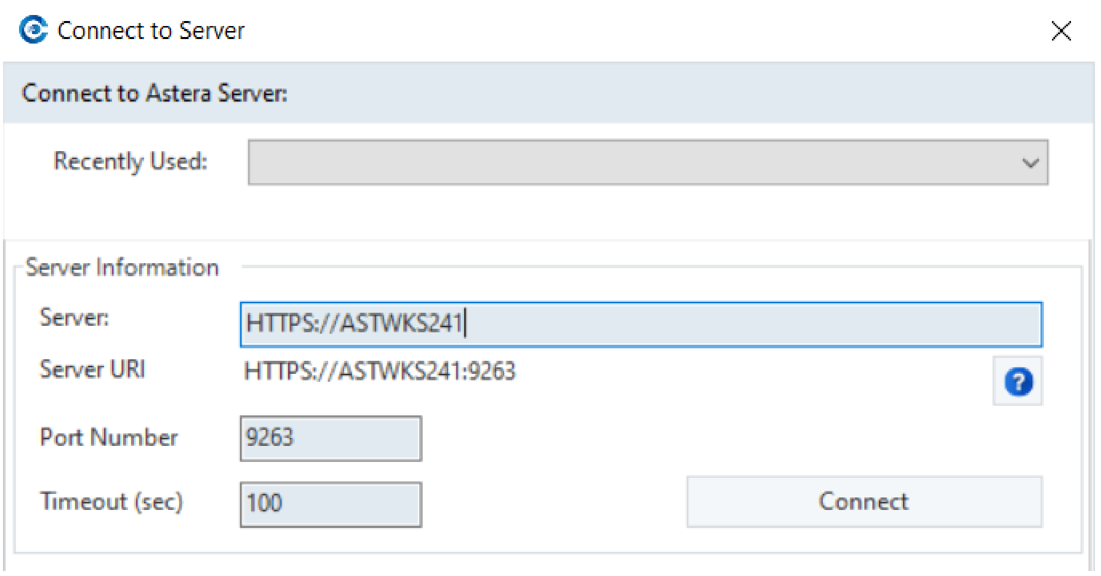
A prompt will appear that will confirm if you want to disconnect from the current Server and establish a connection to a different server. Click *Yes* to proceed.

Note: A client cannot be connected to multiple servers at once.



Are you sure you want to disconnect from current Server and connect to another Server?

You will be directed to the *Server Connection* screen. Enter the required server information (Server URI and Port Number) to connect to the server and click *Connect*.

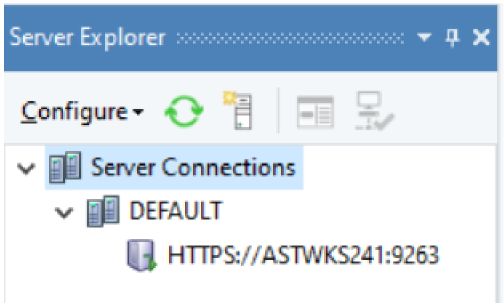


Recently Used:

Server Information

Server:	HTTPS://ASTWKS241
Server URI	HTTPS://ASTWKS241:9263
Port Number	9263
Timeout (sec)	100

If the connection is successfully established, you should be able to see the connected server in the Server Explorer window.



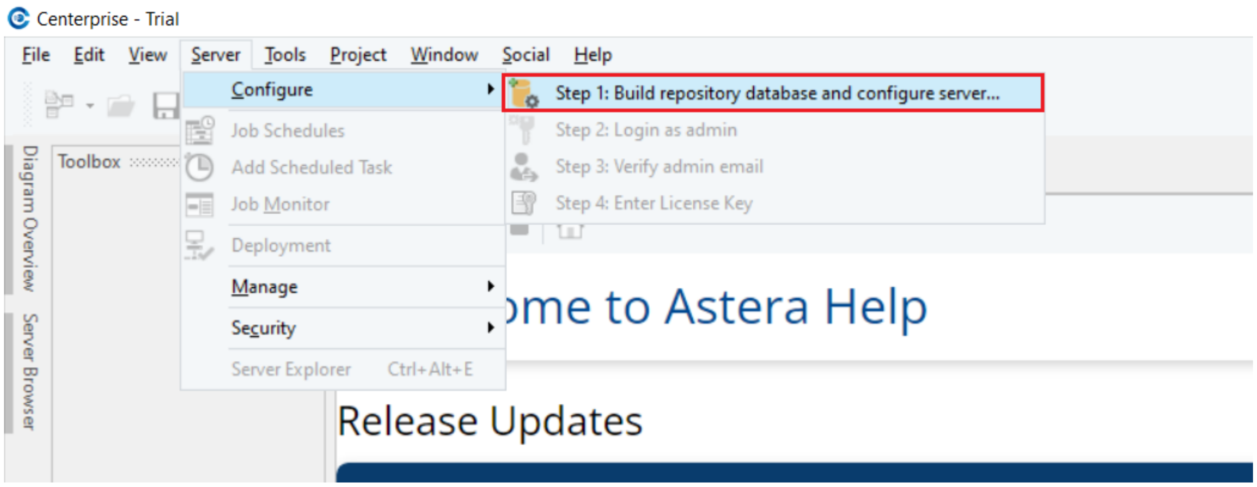
CHAPTER

SIX

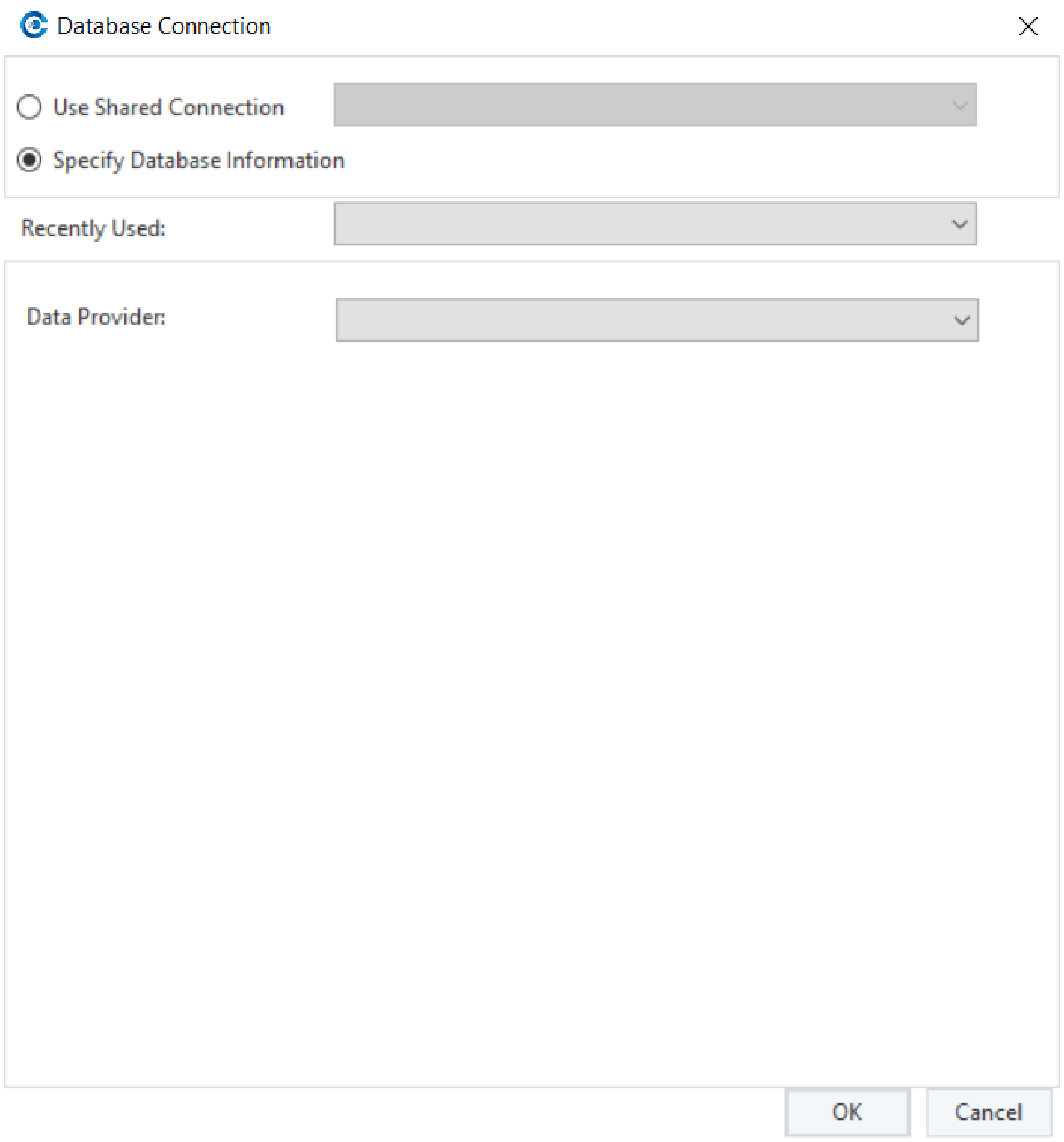
HOW TO BUILD A CLUSTER DATABASE AND CREATE REPOSITORY

Before you start using the Astera API Management server, a repository must be set up. Astera Server supports SQL Server and PostgreSQL for building cluster databases, which can then be used for maintaining the repository. The repository is where request logs, request queues, and deployment information is stored.

To see these options, go to *Server > Configure > Step 1: Build repository database and configure server*.


Centerprise - Trial
File Edit View Server Tools Project Window Social Help
Configure
Job Schedules
Add Scheduled Task
Job Monitor
Deployment
Manage
Security
Server Explorer Ctrl+Alt+E
Step 1: Build repository database and configure server...
Step 2: Login as admin
Step 3: Verify admin email
Step 4: Enter License Key
Toolbox
Diagram Overview
Server Browser
ome to Astera Help
Release Updates

The first step is to point to the SQL Server or PostgreSQL instance where you want to build the repository and provide the credentials to establish the connection.



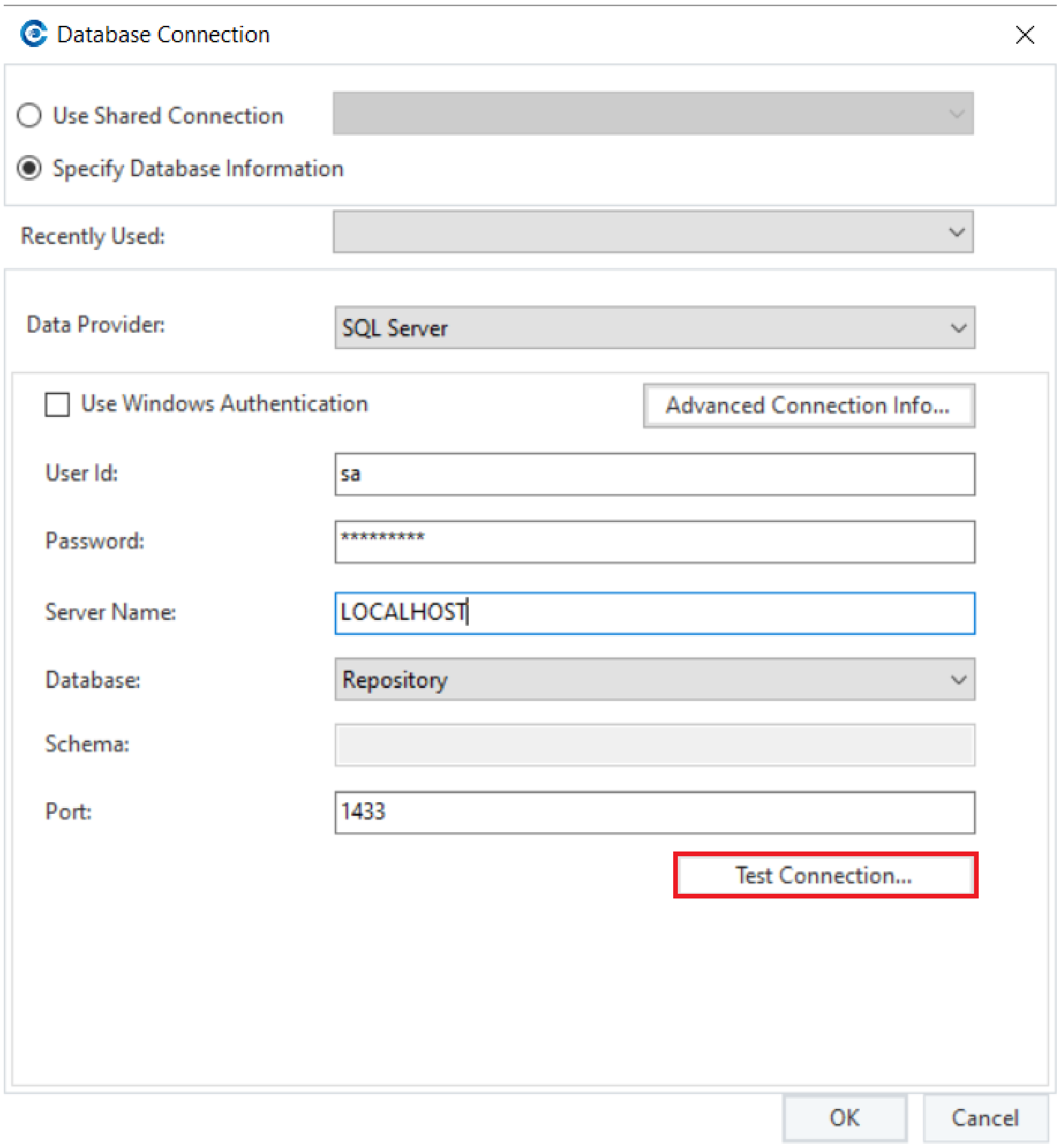
Note: The Astera API Management Server will not create the database itself, just the tables. A database will have to be created beforehand or an existing database can be used. We recommend the Astera API Management Server to have its own database for this purpose.

6.1 Building a Repository on SQL Server

1. Go to *Server > Configure > Step 1: Build repository database and configure server*.

2. Select SQL Server from the Data Provider drop-down list and provide the credentials for establishing the connection.

3. From the drop-down list next to the Database option, select the database on the SQL instance where you want to host the repository.



Database Connection

- ( ) Use Shared Connection
- (●) Specify Database Information

Recently Used:

Data Provider: SQL Server

☐ Use Windows Authentication

User Id: sa

Password: *********

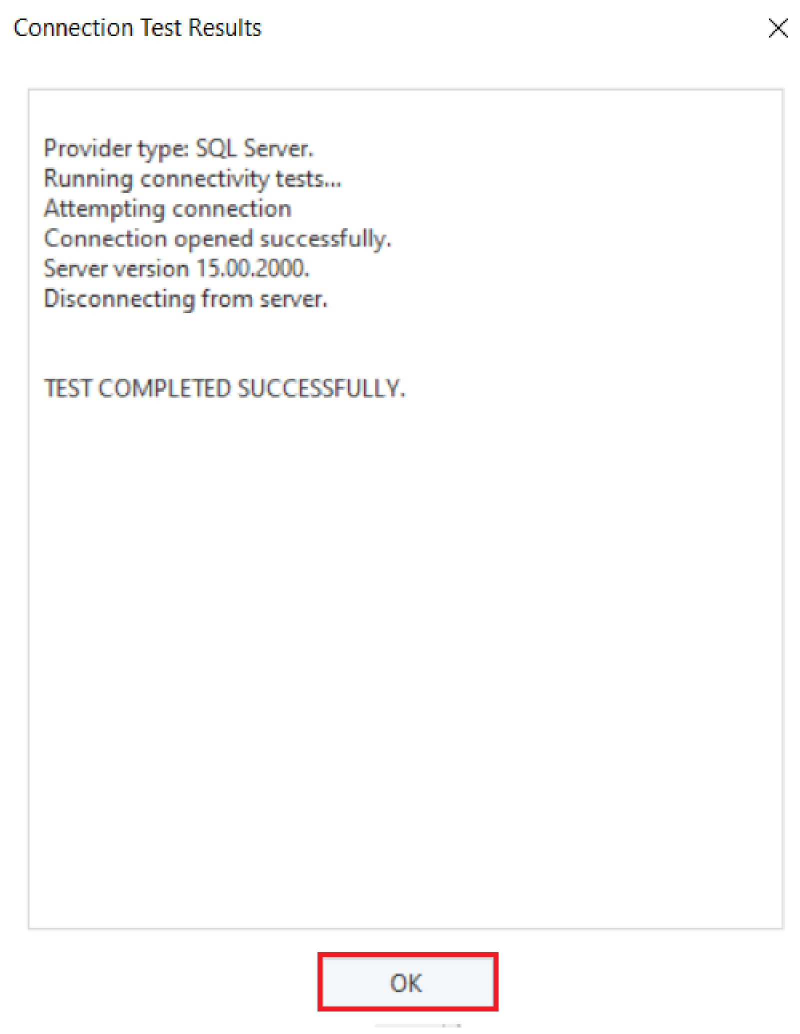
Server Name: LOCALHOST

Database: Repository

Schema:

Port: 1433

4. Click *Test Connection* to test whether the connection is successfully established or not. You should be able to see the following message if the connection is successfully established.

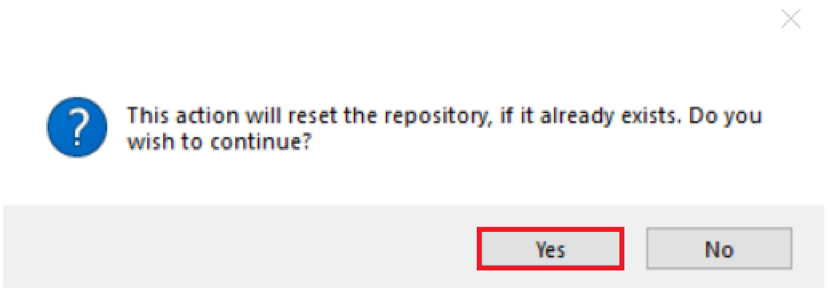


Connection Test Results

```
Provider type: SQL Server.
Running connectivity tests...
Attempting connection
Connection opened successfully.
Server version 15.00.2000.
Disconnecting from server.


TEST COMPLETED SUCCESSFULLY.
```

5. Click *OK* to exit out of the test connection window and again click *OK*, the following message will appear. Select *Yes* to proceed.



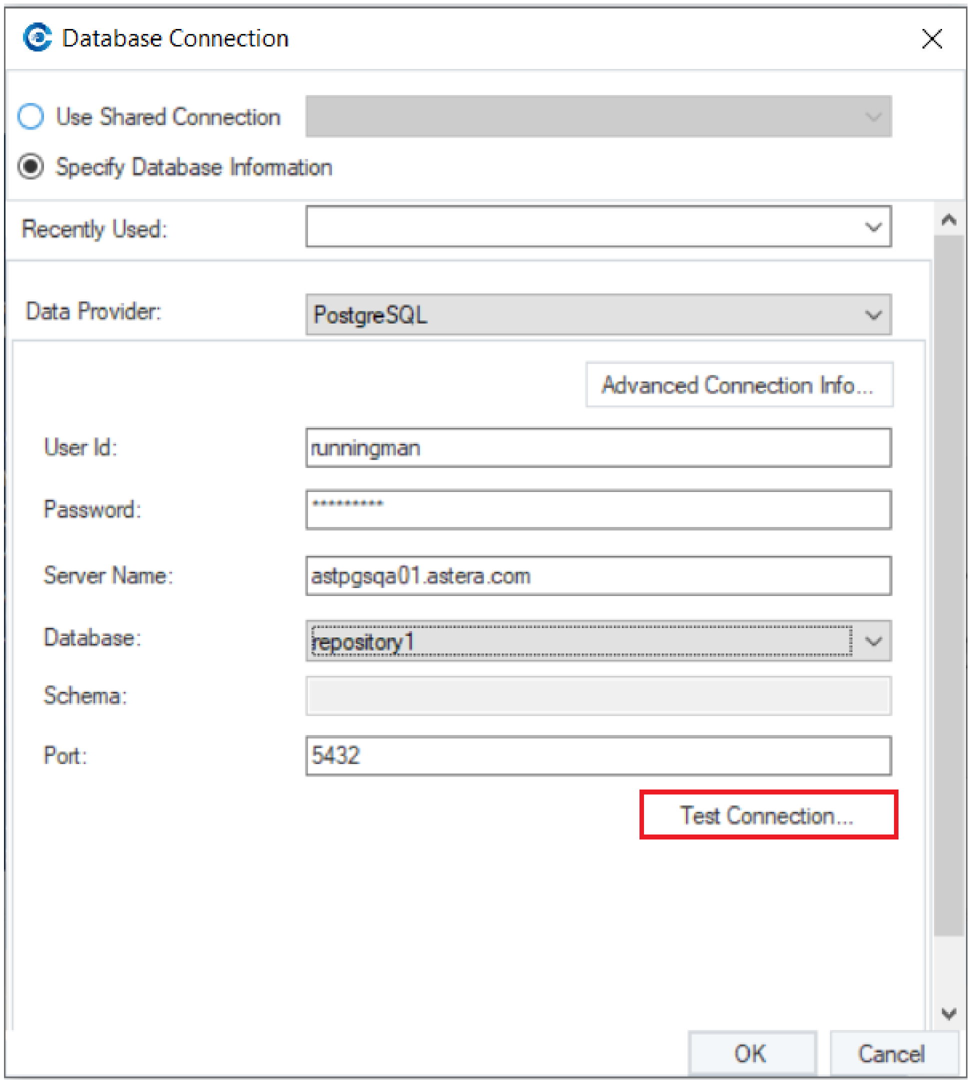
This action will reset the repository, if it already exists. Do you wish to continue?

The repository is now set up and configured with the server to be used.

The next step is to log in using your credentials.

6.2 Building a Repository on PostgreSQL

1. Go to *Server > Configure > Step 1: Build repository database and configure server.*
2. Select PostgreSQL from the Data Provider drop-down list and provide the credentials for establishing the connection.
3. From the drop-down list next to the *Database* option, select the database on the PostgreSQL instance where you want to host the repository.



Recently Used:

Data Provider: PostgreSQL

User Id: runningman

Password: *********

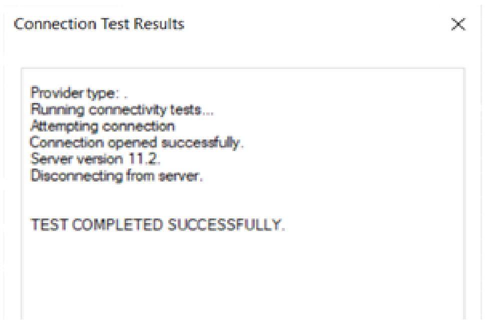
Server Name: astpgsqa01.astera.com

Database: repository1

Schema:

Port: 5432

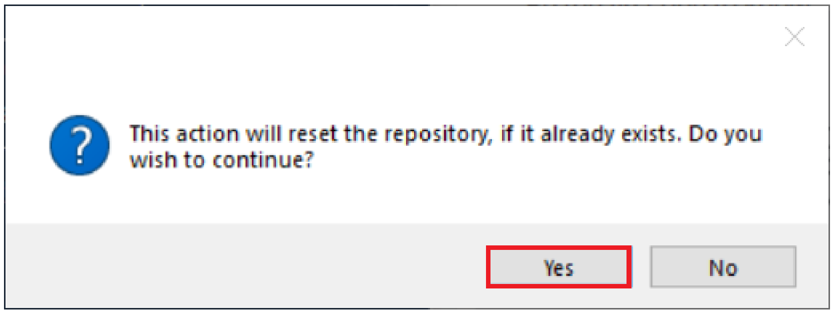
4. Click *Test Connection* to test whether the connection is successfully established or not. You should be able to see the following message if the connection is successfully established.


Connection Test Results

```
Provider type: .
Running connectivity tests...
Attempting connection
Connection opened successfully.
Server version 11.2.
Disconnecting from server.

TEST COMPLETED SUCCESSFULLY.
```

5. Click *OK* and the following message will appear. Select *Yes* to proceed.


This action will reset the repository, if it already exists. Do you wish to continue?

The repository is now set up and configured with the server to be used.

The next step is to log in using your credentials.

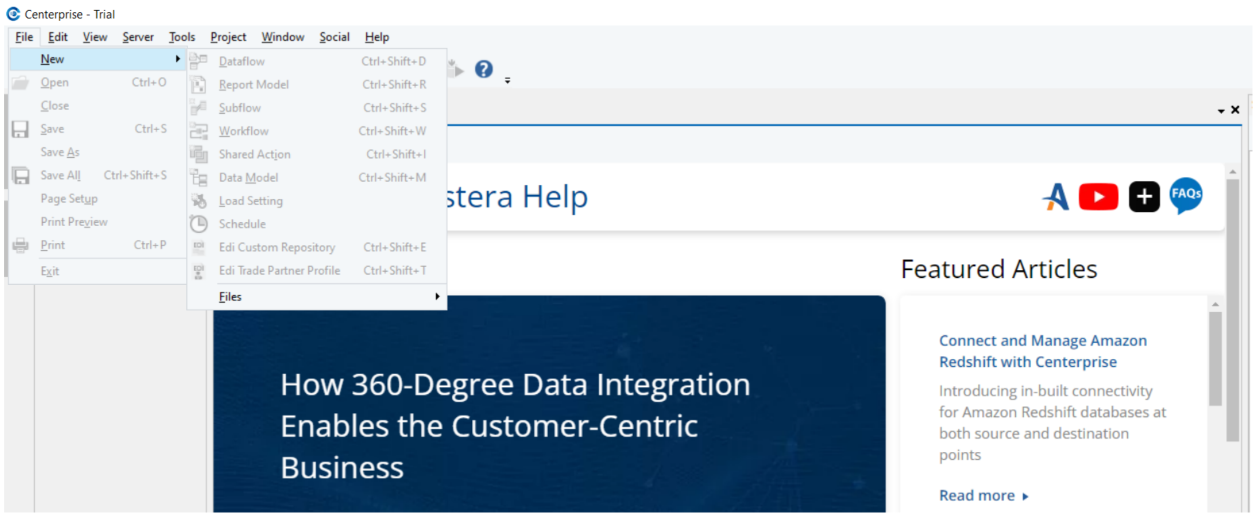
CHAPTER

SEVEN

HOW TO LOGIN FROM LEAN CLIENT

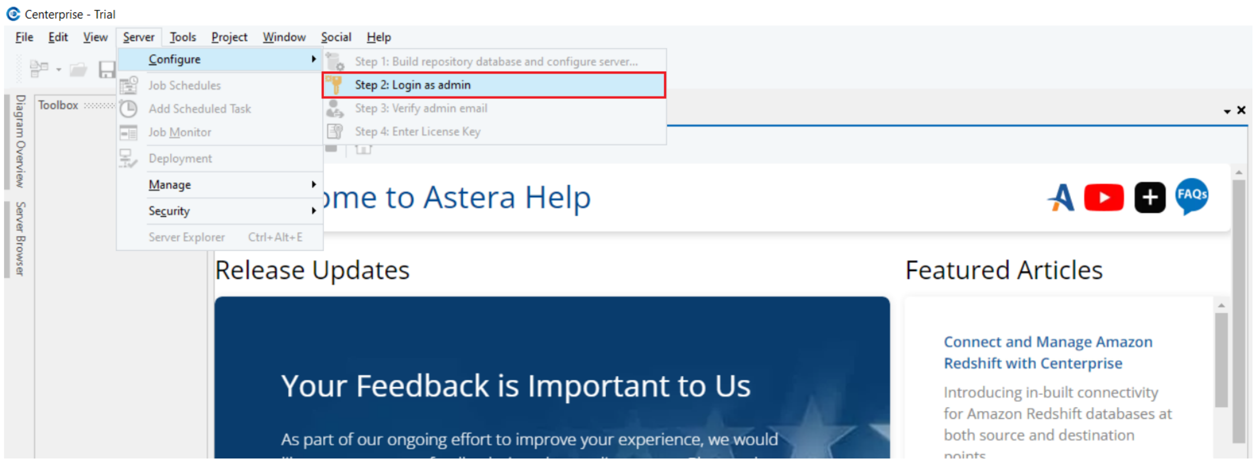
Once you have created the repository and configured the server, the next step is to login using your Astera Centerprise Client account credentials.

You will not be able to design any API flows on the Lean client if you haven't logged in. The options will be disabled.

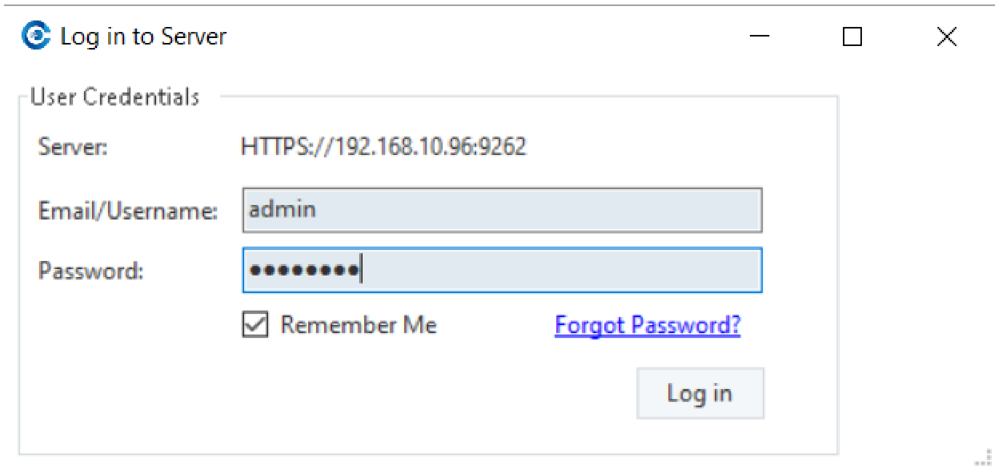

Centerprise - Trial
File Edit View Server Tools Project Window Social Help
New
Open Ctrl+O
Close
Save Ctrl+S
Save As
Save All Ctrl+Shift+S
Page Setup
Print Preview
Print Ctrl+P
Exit
Dataflow Ctrl+Shift+D
Report Model Ctrl+Shift+R
Subflow Ctrl+Shift+S
Workflow Ctrl+Shift+W
Shared Action Ctrl+Shift+I
Data Model Ctrl+Shift+M
Load Setting
Schedule
Edi Custom Repository Ctrl+Shift+E
Edi Trade Partner Profile Ctrl+Shift+T
Files
stera Help
Featured Articles
How 360-Degree Data Integration Enables the Customer-Centric Business
Connect and Manage Amazon Redshift with Centerprise
Introducing in-built connectivity for Amazon Redshift databases at both source and destination points
Read more

7.1 Log in to your user account

1. Go to *Server > Configure > Step 2: Login as admin*.


Centerprise - Trial
File Edit View Server Tools Project Window Social Help
Configure
Job Schedules
Add Scheduled Task
Job Monitor
Deployment
Manage
Security
Server Explorer Ctrl+Alt+E
Step 1: Build repository database and configure server...
Step 2: Login as admin
Step 3: Verify admin email
Step 4: Enter License Key
Toolbox
Diagram Overview
Server Browser
ome to Astera Help
Release Updates
Featured Articles
Your Feedback is Important to Us
As part of our ongoing effort to improve your experience, we would
Connect and Manage Amazon Redshift with Centerprise
Introducing in-built connectivity for Amazon Redshift databases at both source and destination

2. This will direct you to a login screen where you can provide your user credentials.



Log in to Server

User Credentials

Server: HTTPS://192.168.10.96:9262

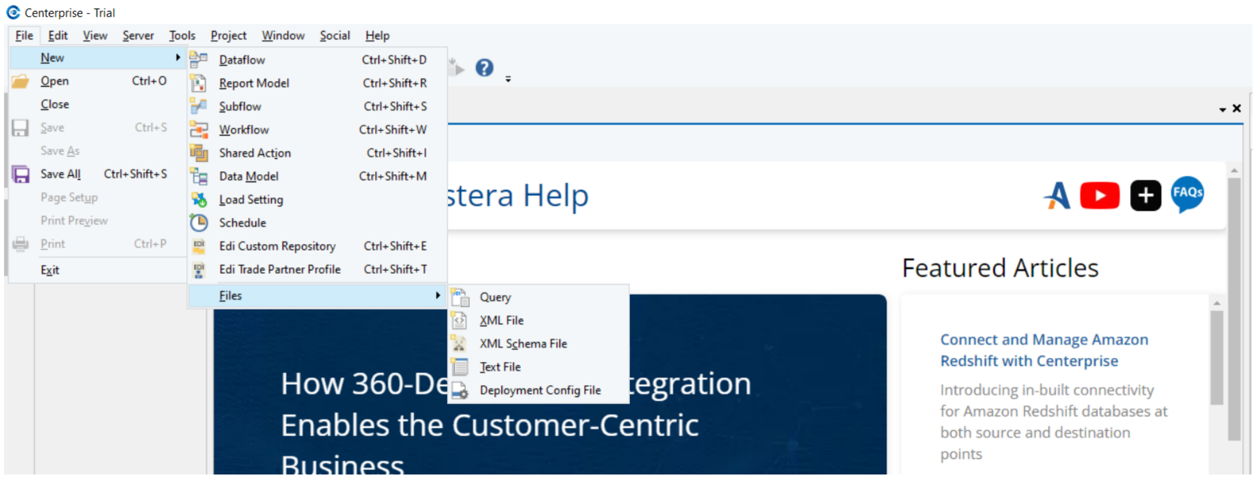
Email/Username: admin

Password: ••••••••

☑ Remember Me

If you are logging in for the first time, you can login using the default credentials as follows: *Username: admin Password: Admin123*

After you log in, you will see that the options in the Centerprise Lean Client are enabled.



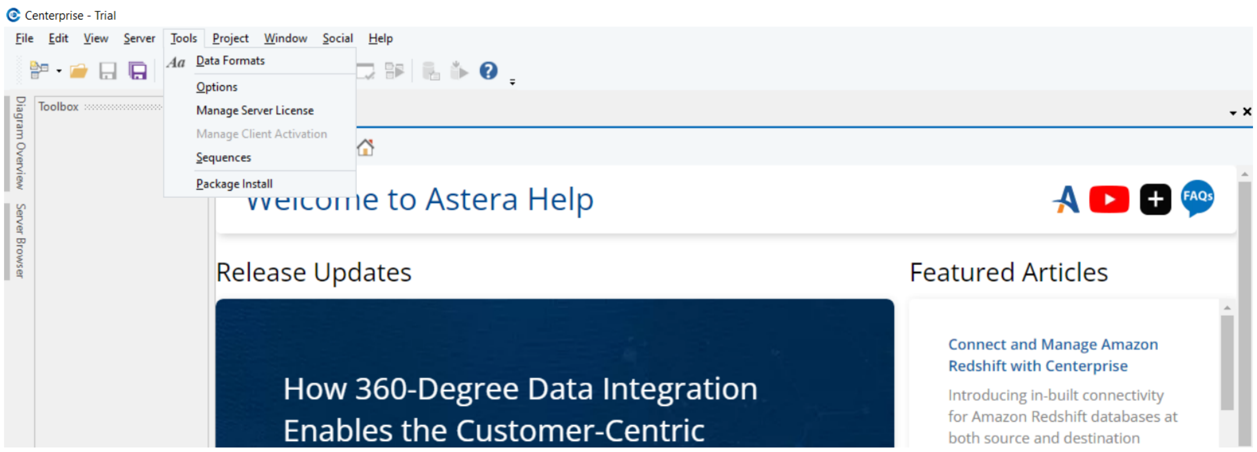
Centerprise - Trial
File Edit View Server Tools Project Window Social Help
New
Open Ctrl+O
Close
Save Ctrl+S
Save As
Save All Ctrl+Shift+S
Page Setup
Print Preview
Print Ctrl+P
Exit
Dataflow Ctrl+Shift+D
Report Model Ctrl+Shift+R
Subflow Ctrl+Shift+S
Workflow Ctrl+Shift+W
Shared Action Ctrl+Shift+I
Data Model Ctrl+Shift+M
Load Setting
Schedule
Edi Custom Repository Ctrl+Shift+E
Edi Trade Partner Profile Ctrl+Shift+T
Files
Query
XML File
XML Schema File
Text File
Deployment Config File
stera Help
Featured Articles
How 360-De tegration Enables the Customer-Centric Business
Connect and Manage Amazon Redshift with Centerprise
Introducing in-built connectivity for Amazon Redshift databases at both source and destination points

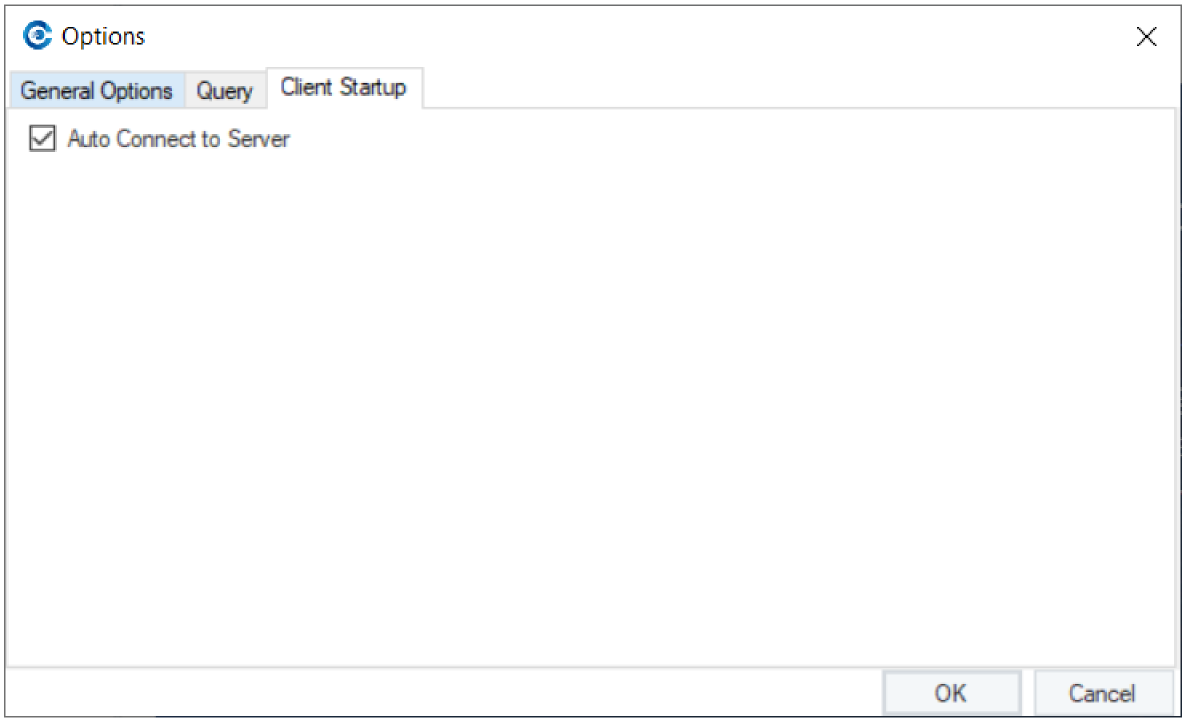
You can use these options until your trial period is active. For fully activating the options and the product, you'll have to enter your license.

7.2 How to automatically reconnect on client startup

If you don't want Centerprise to show you the server connection screen every time you run the client application, you can skip that by modifying the settings.

To do that go to **Tools > Options > Client Startup** and select the *Auto Connect to Server* option. On enabling the option, Centerprise will store the server details you entered previously and will use those details to automatically reconnect to the server every time you run the application.


Centerprise - Trial
File Edit View Server Tools Project Window Social Help
Data Formats
Options
Manage Server License
Manage Client Activation
Sequences
Package Install
Toolbox
Diagram Overview
Server Browser
Welcome to Astera Help
Release Updates
Featured Articles
How 360-Degree Data Integration Enables the Customer-Centric
Connect and Manage Amazon Redshift with Centerprise
Introducing in-built connectivity for Amazon Redshift databases at both source and destination



Auto Connect to Server [✓]

The next step after logging in is to unlock Centerprise using the License key.

CHAPTER

EIGHT

HOW TO VERIFY ADMIN EMAIL

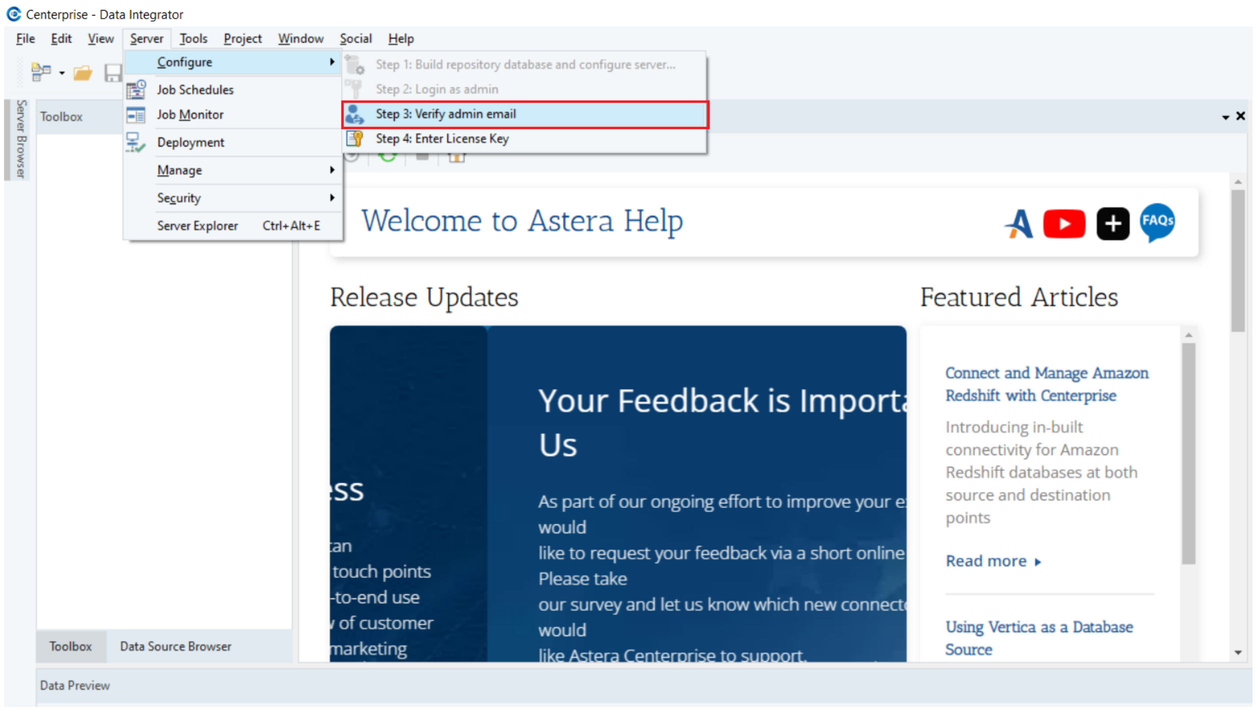
Once you have logged into the Astera Centerprise client, you can set up an admin email to access the Centerprise server. This will also allow you to be able to use the "*Forgot Password*" option at the time of log in.

In this document, we will discuss how to verify admin email in Astera Centerprise.

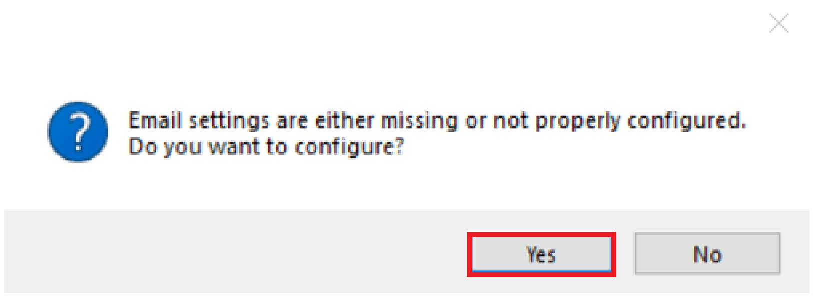
8.1 Verifying Admin Email

1. Once logged in, we will now proceed to enter an email address to associate with the admin user by verifying the email address.

Go to *Server* > *Configure* > *Step 3: Verify Admin Email*

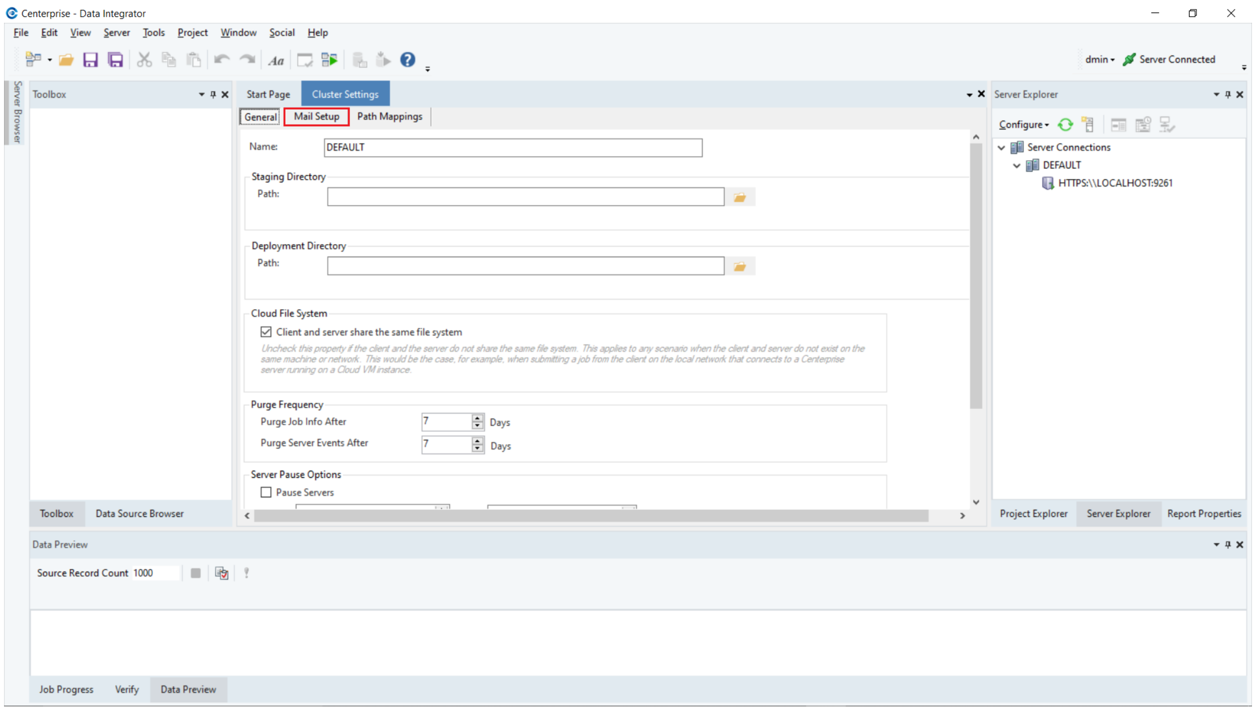

Centerprise - Data Integrator
File Edit View Server Tools Project Window Social Help
Configure
Job Schedules
Job Monitor
Deployment
Manage
Security
Server Explorer Ctrl+Alt+E
Step 1: Build repository database and configure server...
Step 2: Login as admin
Step 3: Verify admin email
Step 4: Enter License Key
Toolbox
Server Browser
Welcome to Astera Help
Release Updates
Featured Articles
Data Source Browser
Data Preview

2. Unless you have already set up an email address in the *Mail Setup* section of *Cluster settings,* the following dialogue box will pop up asking you to configure your email settings.



Email settings are either missing or not properly configured.
Do you want to configure?

Click on *Yes* to open your cluster settings.



Name: DEFAULT

Staging Directory
Path:

Deployment Directory
Path:

Cloud File System
☑ Client and server share the same file system
Uncheck this property if the client and the server do not share the same file system. This applies to any scenario when the client and server do not exist on the same machine or network. This would be the case, for example, when submitting a job from the client on the local network that connects to a Centerprise server running on a Cloud VM instance.

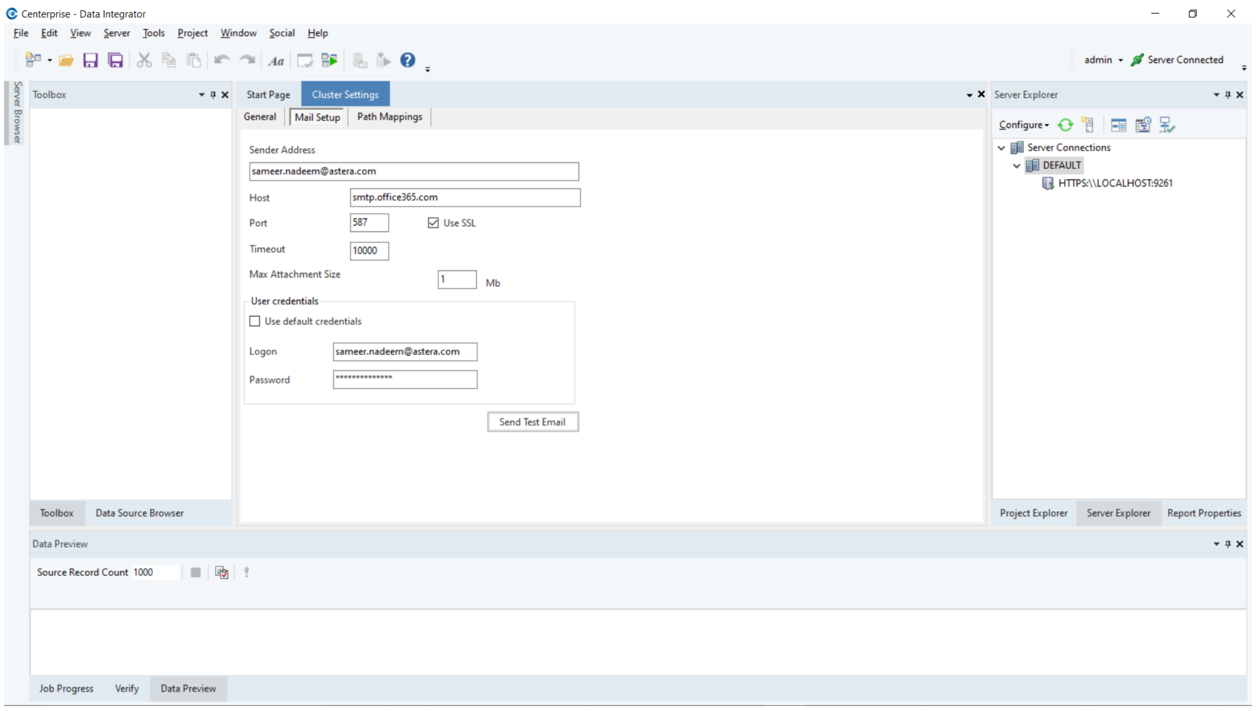
Purge Frequency
Purge Job Info After 7 Days
Purge Server Events After 7 Days

Server Pause Options
☐ Pause Servers

Source Record Count 1000

Click on the *Mail Setup* tab.

3. Enter your email server settings.



Sender Address: sameer.nadeem@astera.com

Host: smtp.office365.com

Port: 587 [✓ Use SSL]

Timeout: 10000

Max Attachment Size: 1 Mb

User credentials

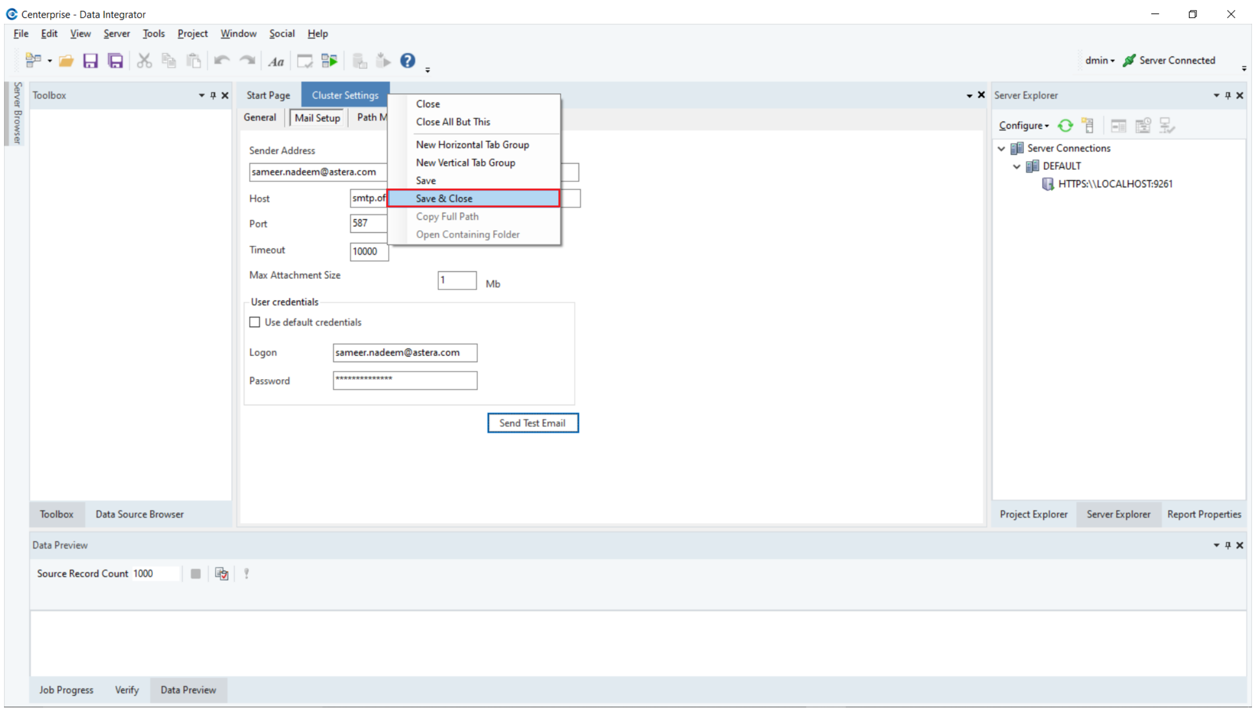
☐ Use default credentials

Logon: sameer.nadeem@astera.com

Password: **************

Send Test Email

4. Now, right-click on the Cluster Settings active tab and click on *Save & Close* in order to save the mail setup.



Sender Address: sameer.nadeem@astera.com

Host: smtp.of

Port: 587

Timeout: 10000

Max Attachment Size: 1 Mb

User credentials

☐ Use default credentials

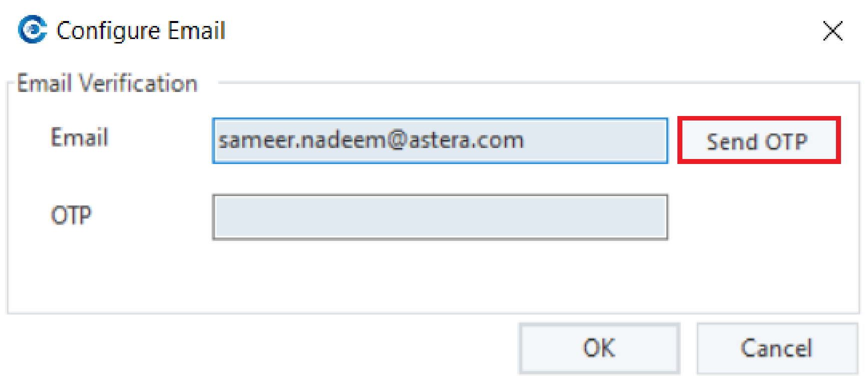
Logon: sameer.nadeem@astera.com

Password: **************

Send Test Email

5. Re-visit the *Verify Admin Email* step by going to *Server* > *Configure* > *Step 3: Verify Admin Email*.

This time, the *Configure Email* dialogue box will open.


Configure Email

Email Verification

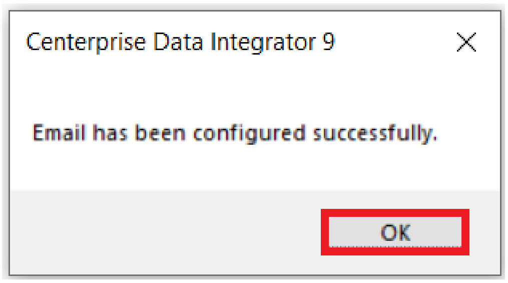
Email: sameer.nadeem@astera.com

OTP:

6. Enter the email address you previously set up and click on *Send OTP*.

7. Use the OTP from the email you received and enter it in the *Configure Email* dialogue and proceed.

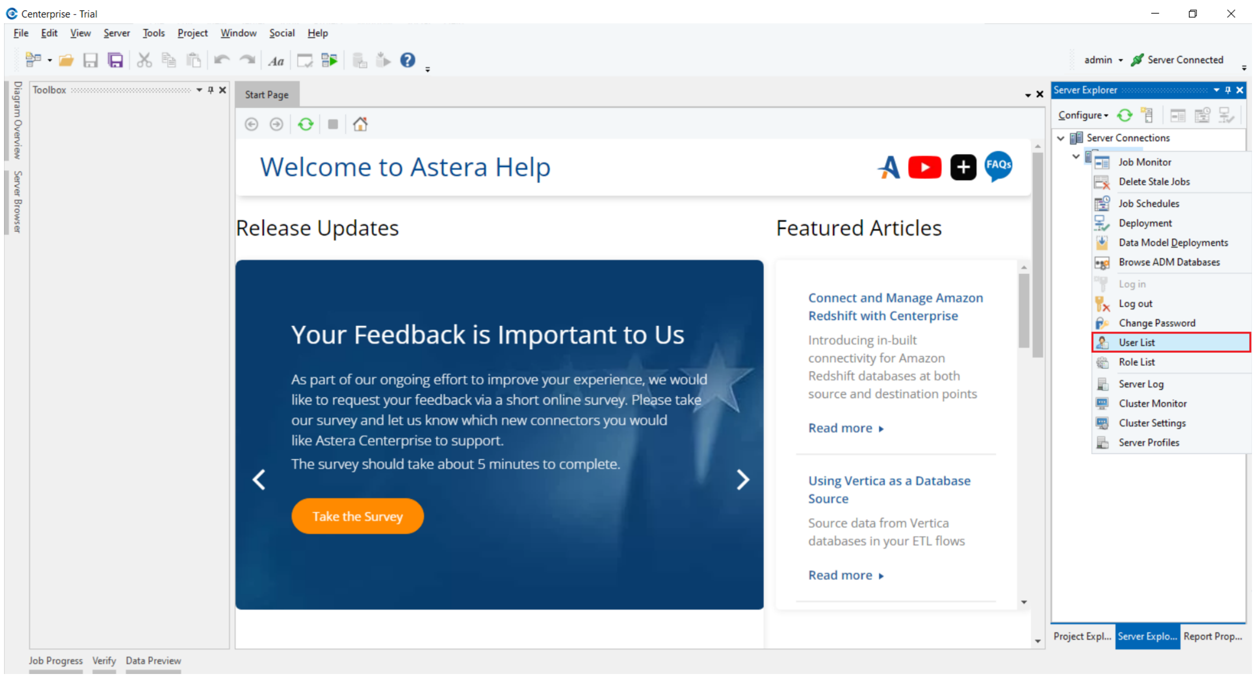
On the correct entry of the OTP, an email successfully configured dialogue will appear.


Centerprise Data Integrator 9

Email has been configured successfully.

8. Click *OK* to exit it. We can confirm our email configuration by going to the *User List*.

Right-click on *DEFAULT* under *Server Connections* in the *Server Explorer* and go to *User List*.



Welcome to Astera Help

Release Updates

Your Feedback is Important to Us

As part of our ongoing effort to improve your experience, we would like to request your feedback via a short online survey. Please take our survey and let us know which new connectors you would like Astera Centerprise to support.

The survey should take about 5 minutes to complete.

Featured Articles

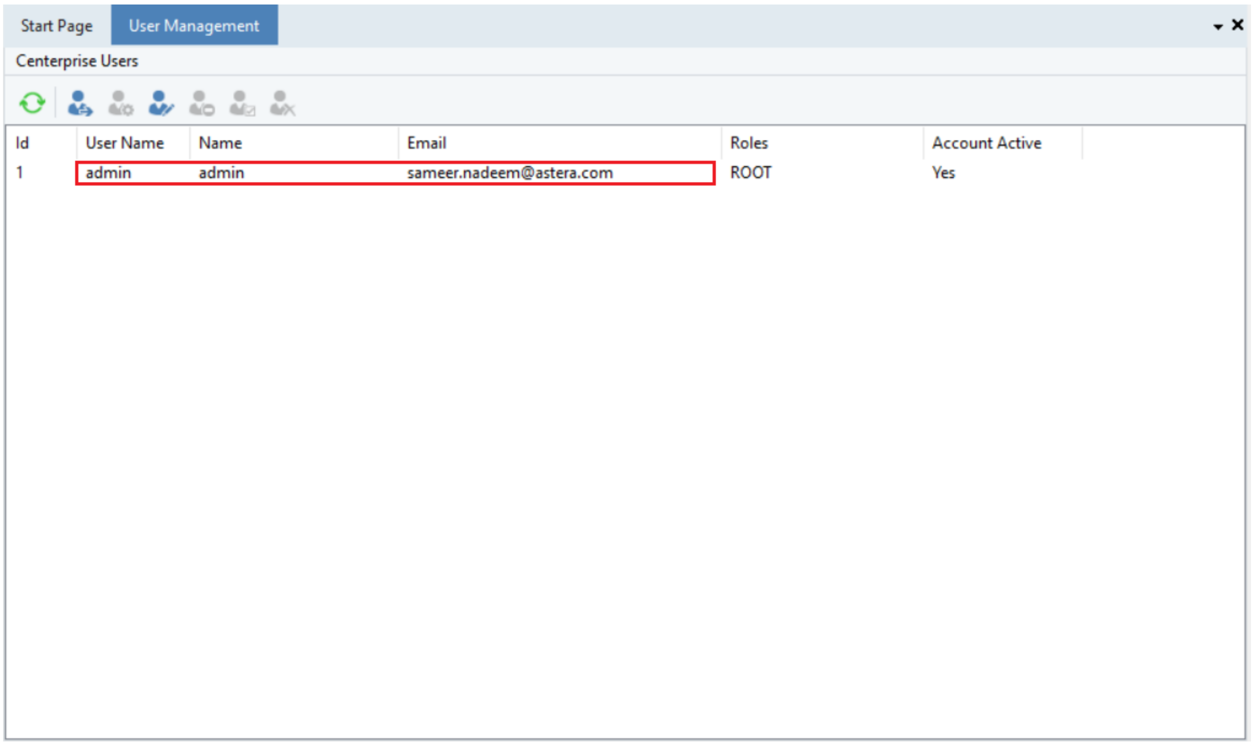
Connect and Manage Amazon Redshift with Centerprise

Introducing in-built connectivity for Amazon Redshift databases at both source and destination points

Using Vertica as a Database Source

Source data from Vertica databases in your ETL flows

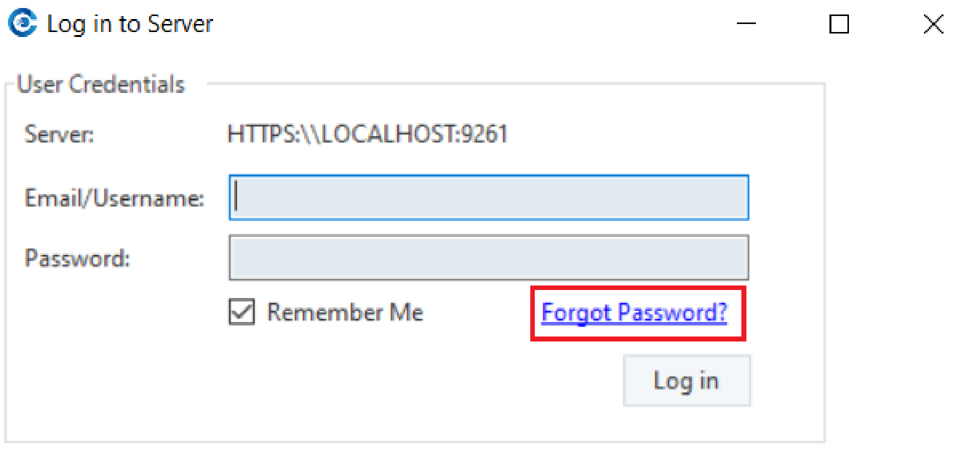
9. This opens the *User List* where you can confirm that the email address has been configured with the admin user.



Id	User Name	Name	Email	Roles	Account Active
1	admin	admin	sameer.nadeem@astera.com	ROOT	Yes

8.2 Using Forgot Password feature

The feature is now configured and can be utilized when needed by clicking on *Forgot Password* in the log in window.

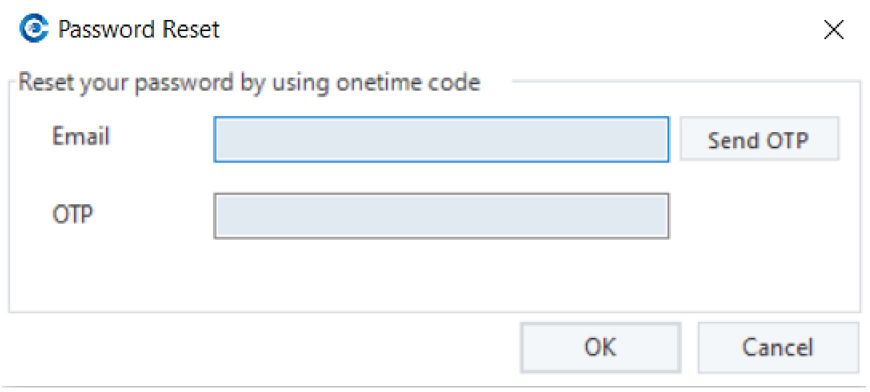


Server: HTTPS:\\LOCALHOST:9261

Email/Username:

Password:

This opens the *Password Reset* window, where you can enter the OTP sent to the specified e-mail for the user and proceed to reset your password.



This concludes our discussion on verifying admin email in Astera Centerprise.

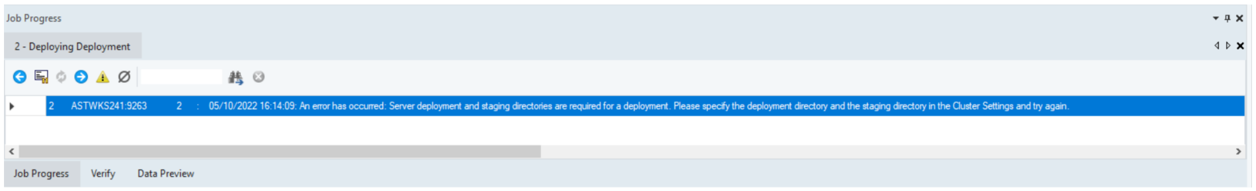
CHAPTER

NINE

CONFIGURING THE DEPLOYMENT DIRECTORY IN ASTERA API MANAGEMENT

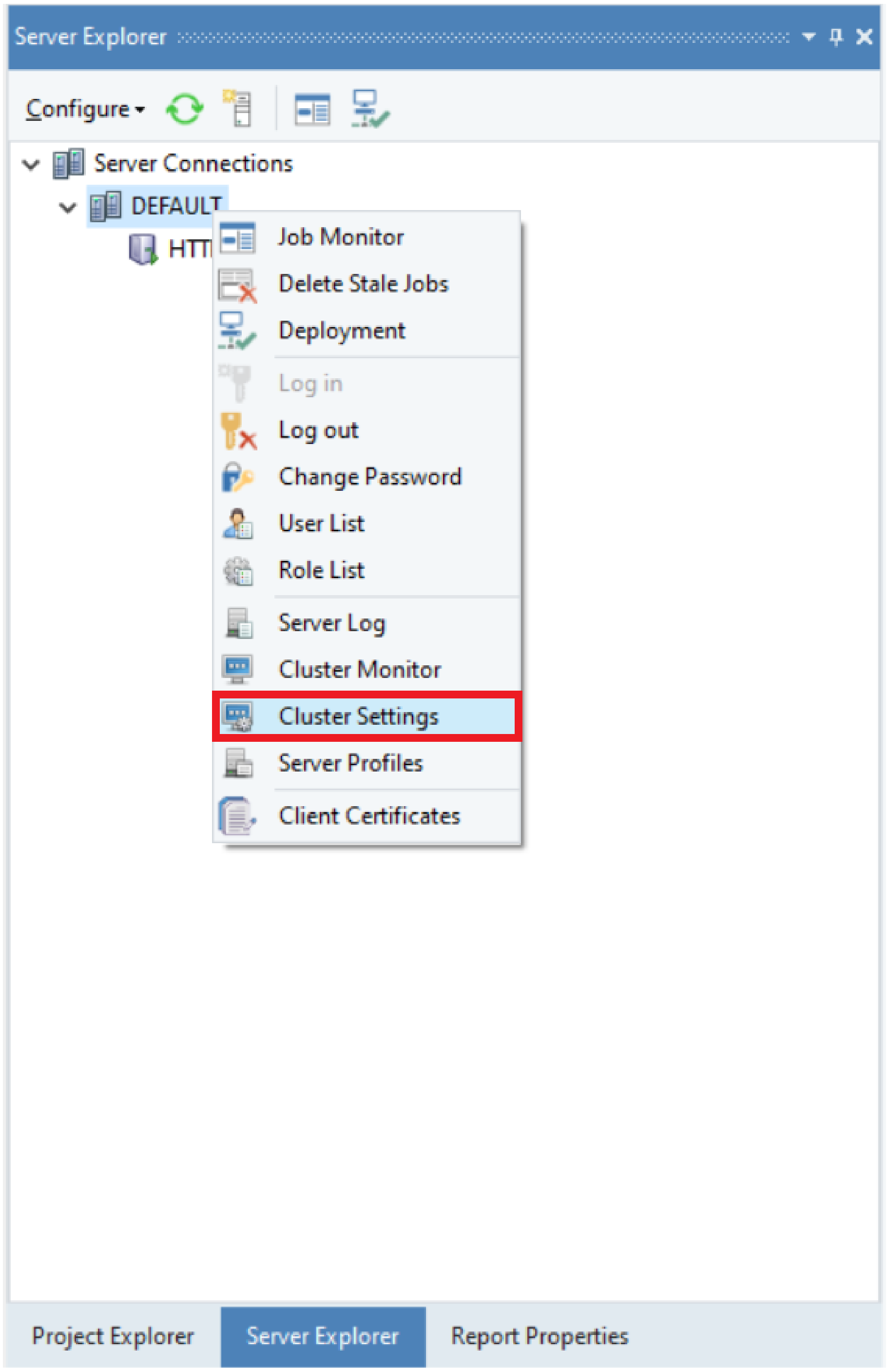
Before API deployments can be created in Astera API Management, the deployment directory must be defined. It is used to maintain runtime executable archives of all the deployments made on the server.

Without defining the deployment directory, creating a deployment will produce an error, asking the user to set the directory, in the Job Progress window.

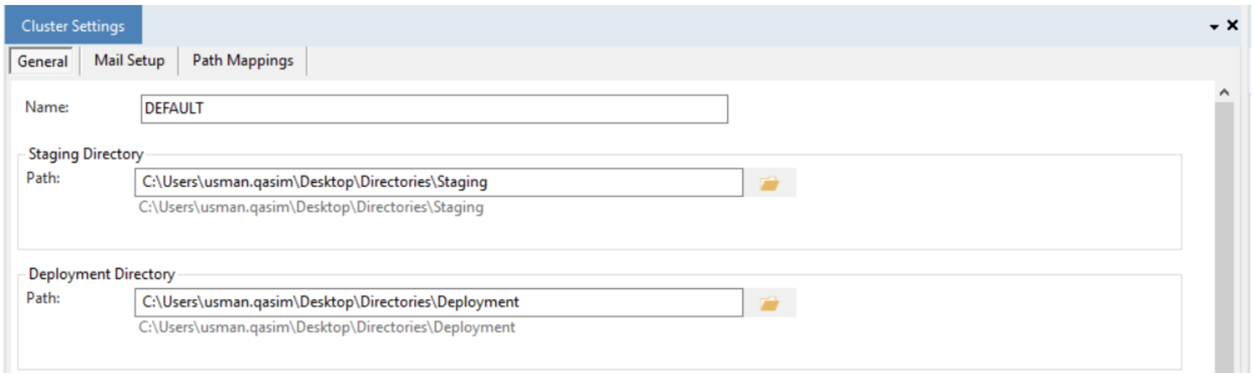


```
2    ASTWKS241:9263    2  :  05/10/2022 16:14:09: An error has occurred: Server deployment and staging directories are required for a deployment. Please specify the deployment directory and the staging directory in the Cluster Settings and try again.
```

1. Right-click on the cluster's node in the Server Explorer window and select *Cluster Settings* from the context menu.



This will open a new tab.



Name: DEFAULT

Staging Directory
Path: C:\Users\usman.qasim\Desktop\Directories\Staging
C:\Users\usman.qasim\Desktop\Directories\Staging

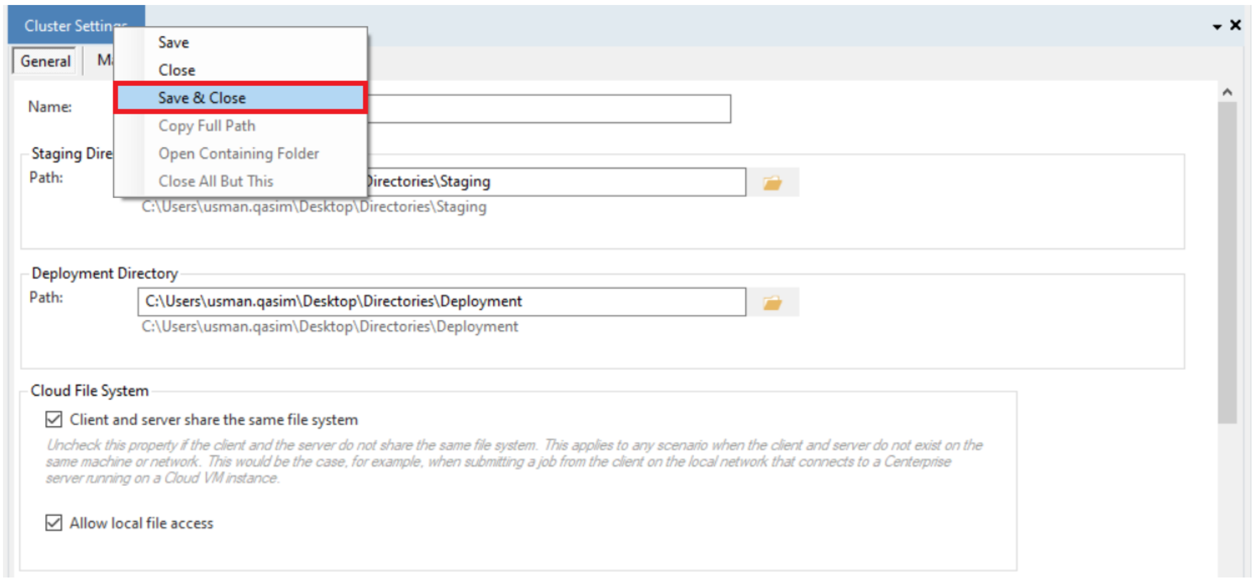
Deployment Directory
Path: C:\Users\usman.qasim\Desktop\Directories\Deployment
C:\Users\usman.qasim\Desktop\Directories\Deployment

Deployment Directory – Path: This is where we browse and add a location for our deployment directory.

As you can see above, the directory has been set.

Note: The directory can be set from both local and remote locations.

2. Now, save the *Cluster Settings* to enable deployment generation on the server.



Name:

Staging Directory
Path: C:\Users\usman.qasim\Desktop\Directories\Staging
C:\Users\usman.qasim\Desktop\Directories\Staging

Deployment Directory
Path: C:\Users\usman.qasim\Desktop\Directories\Deployment
C:\Users\usman.qasim\Desktop\Directories\Deployment

Cloud File System
☑ Client and server share the same file system
Uncheck this property if the client and the server do not share the same file system. This applies to any scenario when the client and server do not exist on the same machine or network. This would be the case, for example, when submitting a job from the client on the local network that connects to a Centerprise server running on a Cloud VM instance.

☑ Allow local file access

This concludes our discussion on the configuration of the deployment directory in Astera API Management.

CHAPTER

TEN

API PUBLISHING

10.1 Designing an API Flow

10.1.1 What is an API Flow?

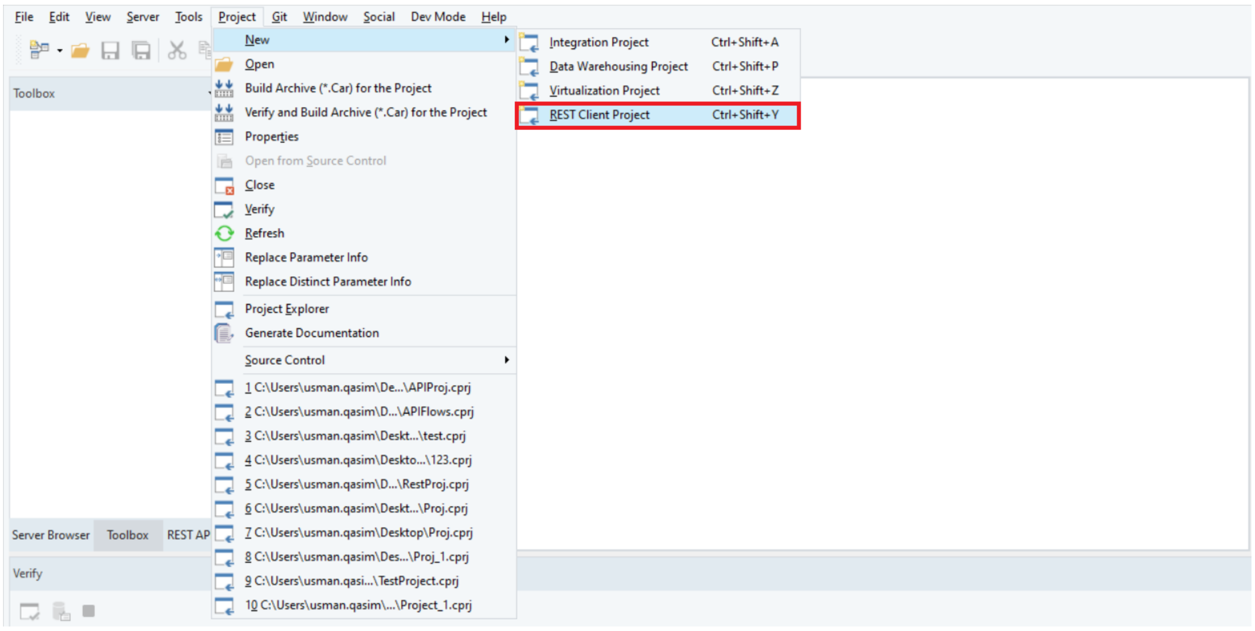
An API flow is an artifact to design a data or a function service endpoint incorporating various data connectors, transformations, quality checks, task-based operations, integrating services, and much more. It defines an end-to-end flow for processing an input, applying transformations and integrations, and routing to response definitions.

Astera API Management holds the ability to create an API Flow as a REST endpoint by defining its request and response objects with in-built abilities to apply sort, filter, pagination, and error handling on responses.

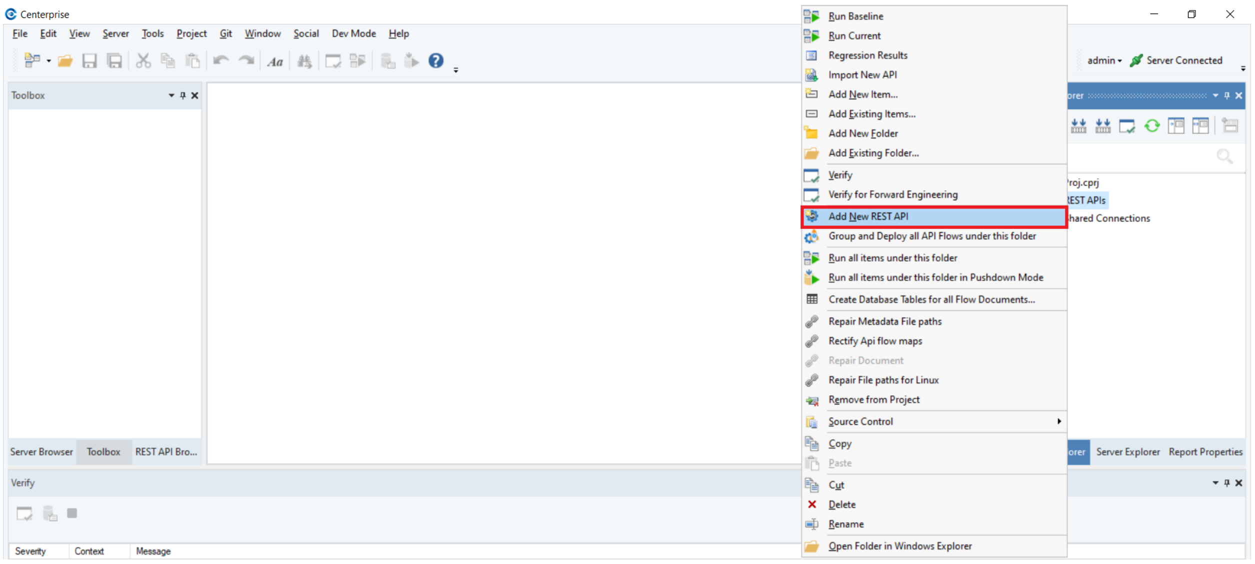
10.1.2 Creating an API Flow

Let's see how we can create API flows contained in a project:

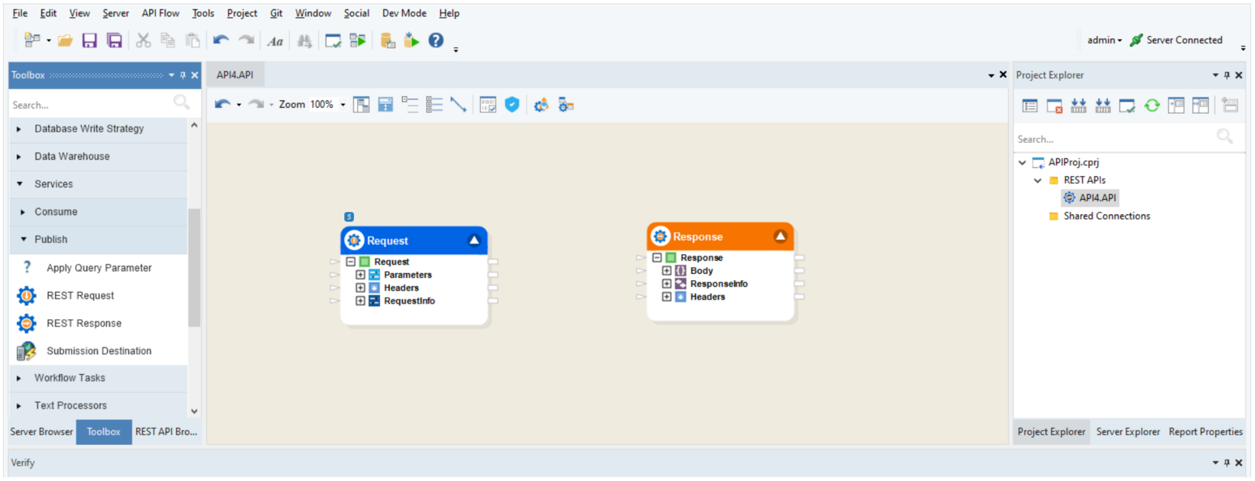
1. To create an API Flow, navigate to the main toolbar, select *Project,* and click on it. Then, hover over *New* and select a project type. API Flows can only be deployed from a project, but they can be added to any project type.


File Edit View Server Tools Project Git Window Social Dev Mode Help
Toolbox
New
Open
Build Archive (*.Car) for the Project
Verify and Build Archive (*.Car) for the Project
Properties
Open from Source Control
Close
Verify
Refresh
Replace Parameter Info
Replace Distinct Parameter Info
Project Explorer
Generate Documentation
Source Control
1 C:\Users\usman.qasim\De...\APIProj.cprj
2 C:\Users\usman.qasim\D...\APIFlows.cprj
3 C:\Users\usman.qasim\Deskt...\test.cprj
4 C:\Users\usman.qasim\Deskto...\123.cprj
5 C:\Users\usman.qasim\D...\RestProj.cprj
6 C:\Users\usman.qasim\Deskt...\Proj.cprj
7 C:\Users\usman.qasim\Desktop\Proj.cprj
8 C:\Users\usman.qasim\Des...\Proj_1.cprj
9 C:\Users\usman.qasi...\TestProject.cprj
10 C:\Users\usman.qasim\...\Project_1.cprj
Integration Project Ctrl+Shift+A
Data Warehousing Project Ctrl+Shift+P
Virtualization Project Ctrl+Shift+Z
REST Client Project Ctrl+Shift+Y
Server Browser Toolbox REST AP
Verify

2. Once the project is created, head to the *Project Explorer.* Right-click on any of the folders in the project and select *Add New REST API.*


Run Baseline
Run Current
Regression Results
Import New API
Add New Item...
Add Existing Items...
Add New Folder
Add Existing Folder...
Verify
Verify for Forward Engineering
Add New REST API
Group and Deploy all API Flows under this folder
Run all items under this folder
Run all items under this folder in Pushdown Mode
Create Database Tables for all Flow Documents...
Repair Metadata File paths
Rectify Api flow maps
Repair Document
Repair File paths for Linux
Remove from Project
Source Control
Copy
Paste
Cut
Delete
Rename
Open Folder in Windows Explorer

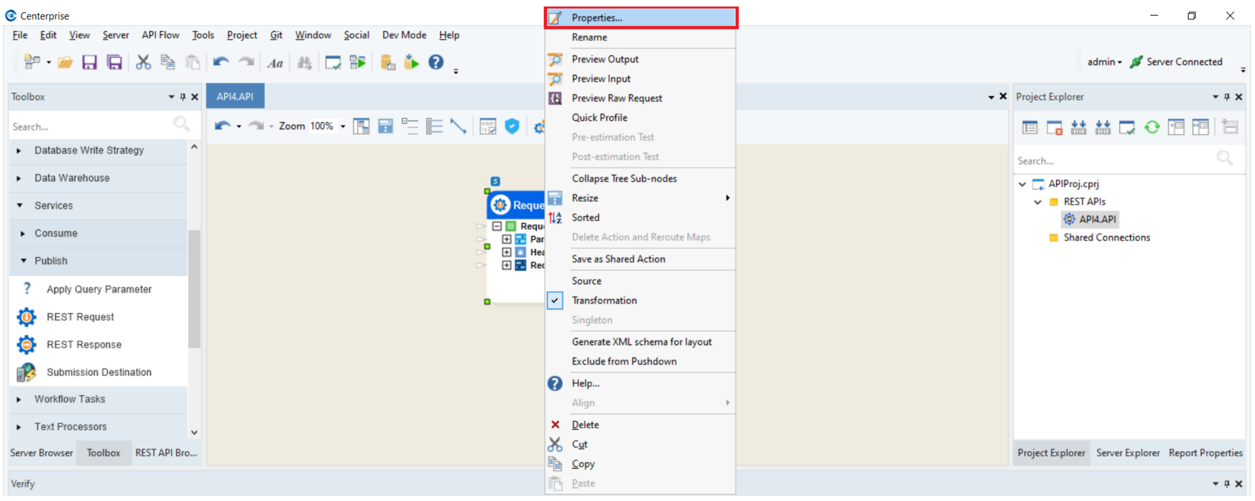
By default, your API Flow file will contain the two required objects, *REST Request*~~,~~ and *REST response*, which act as start and stop objects for an API flow. This flow will be saved with a .API extension.



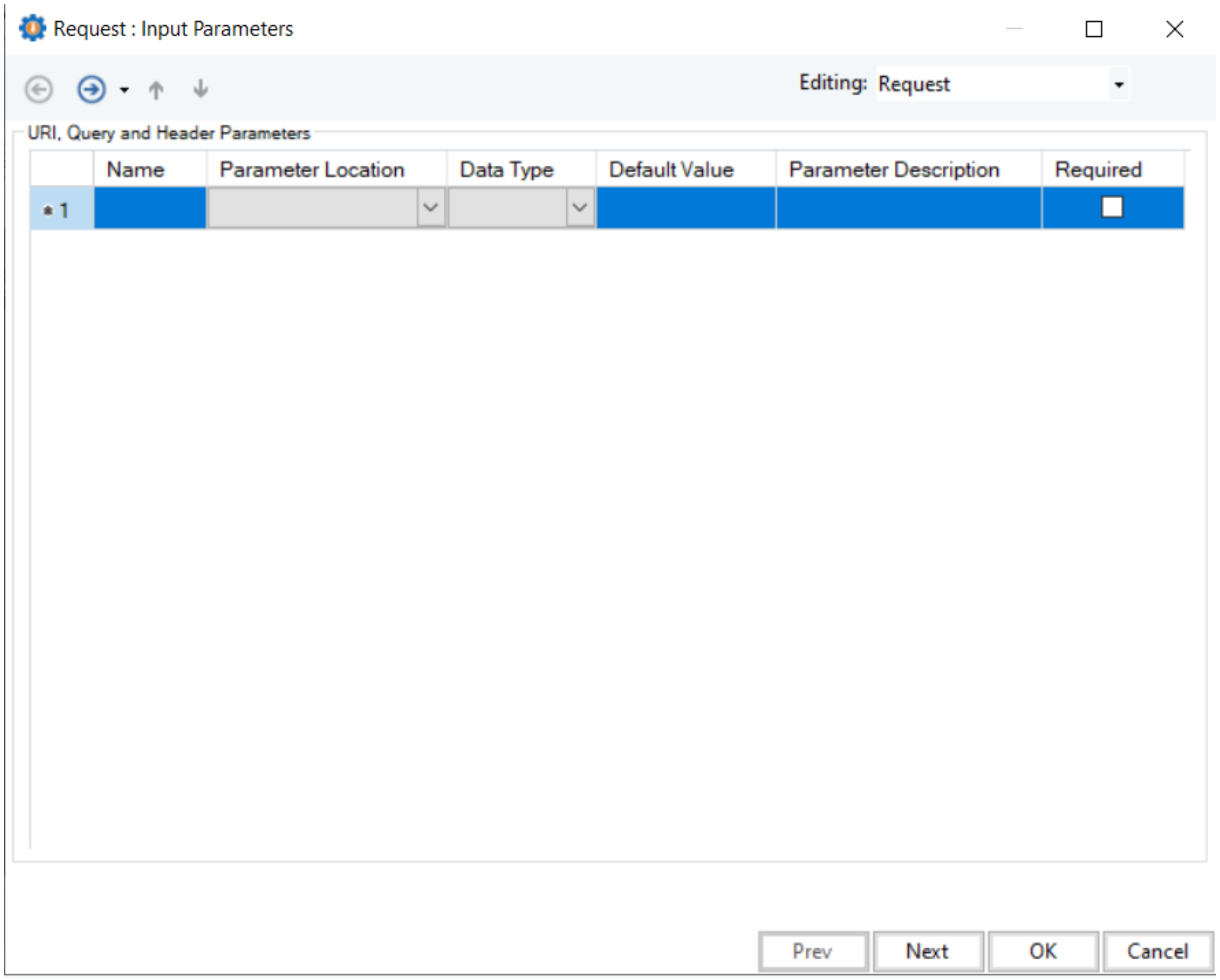
Configuring the REST Request Object

The *REST Request* object is used to define the request endpoint resource, input parameters, and message payloads expected from the API user which would then be used in the flow processing.

1. To start, right-click on the object and select *Properties* from the context menu.



Once done, the parameter configuration window will open.



Here, you can define all the expected parameters from the request. To define a parameter, specify a name, location, and the data type.

Name: To define the name of the parameter.

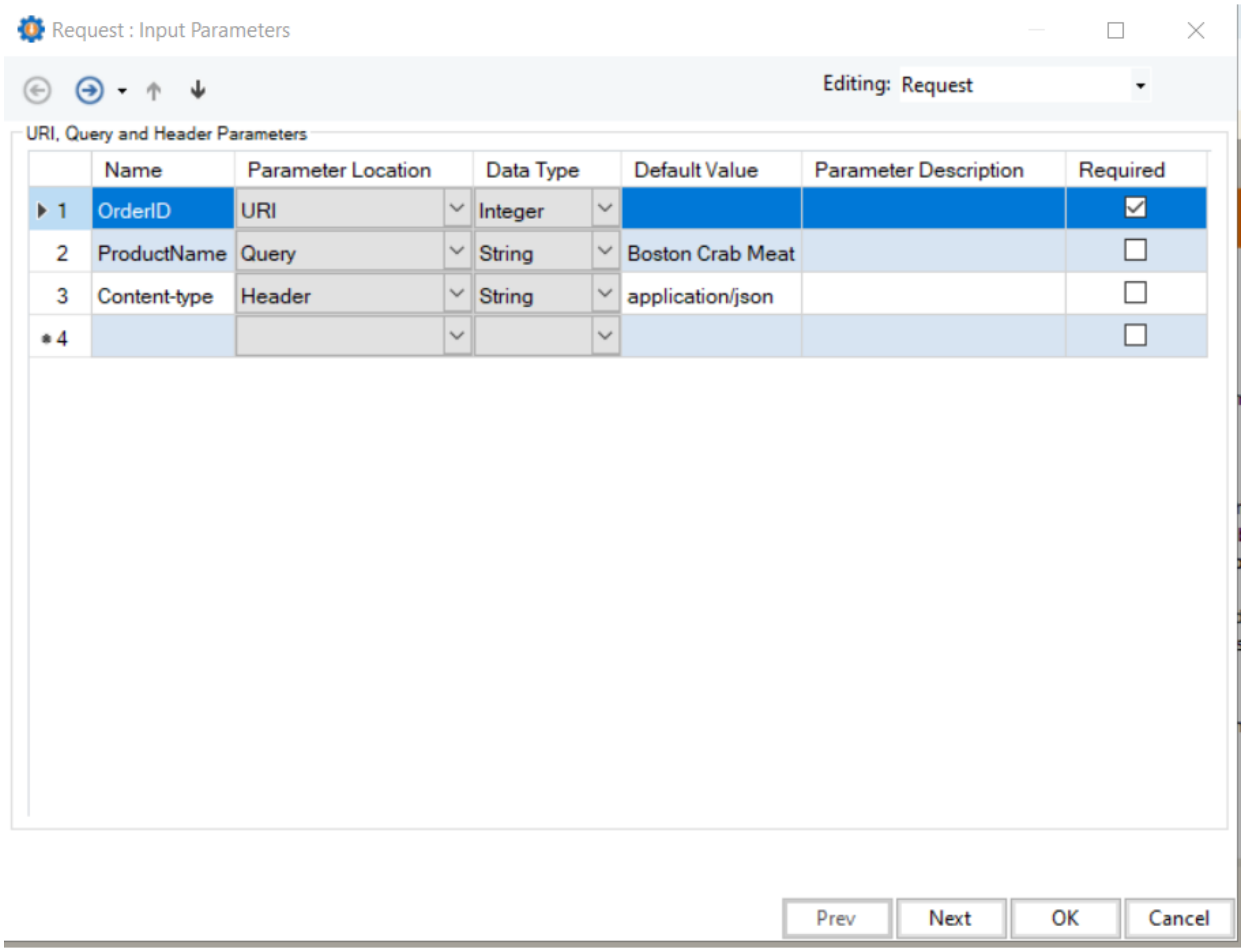
Parameter Location: Here, the location of the parameter is selected between either *URI*, *Query*, or *Header*.

Data Type: To define the parameter's data type expected from the request. Sending an incorrect data type would result in a 400 BadRequest error response.

Default Value: Add a default value, if any, to be used for optional query or header parameters when the incoming request is missing these parameters.

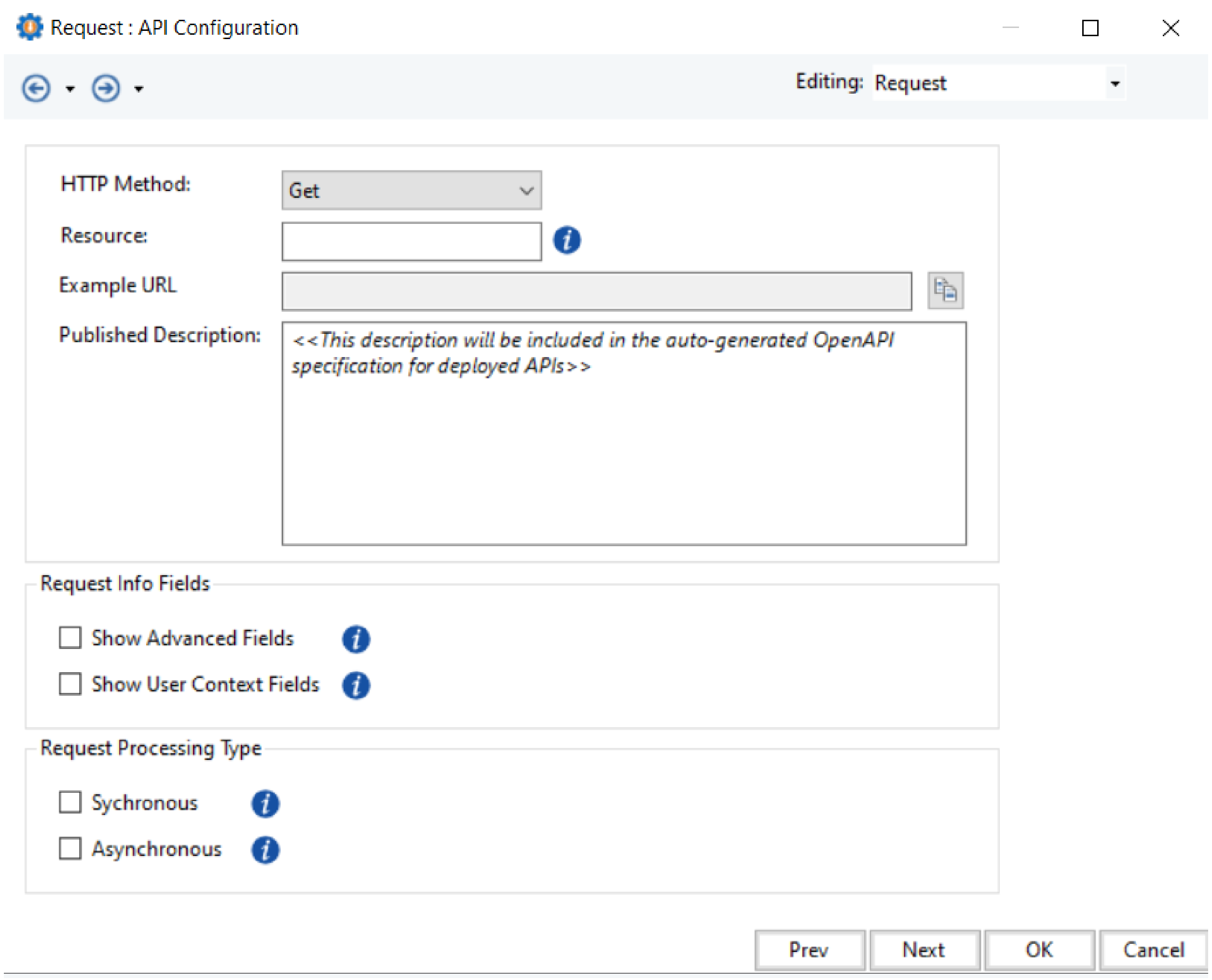
Parameter Description: These will be *u*sed in the auto-generated Open API Swagger specification.

Required: This checkbox is selected if the parameter must be included for a request to be valid. The API returns a *400-Bad* Request error response if required parameters are not provided.



	Name	Parameter Location	Data Type	Default Value	Parameter Description	Required
1	OrderID	URI	Integer			✓
2	ProductName	Query	String	Boston Crab Meat		☐
3	Content-type	Header	String	application/json		☐
4						☐

2. Once done, select *Next,* and the *API Configuration* screen will appear.


Request : API Configuration

Editing: Request

HTTP Method: Get

Resource:

Example URL

Published Description: <<This description will be included in the auto-generated OpenAPI specification for deployed APIs>>

Request Info Fields

☐ Show Advanced Fields

☐ Show User Context Fields

Request Processing Type

☐ Sychronous

☐ Asynchronous

HTTP Configuration for REST Request Object

HTTP Method: There are five methods based on which the request object can be configured. The method depends on the resource operation happening in the flow. These options are,

1. *Get*: Used when the flow is fetching data from a resource based on the given request parameters.

2. *Delete*: Used when the flow deletes an object based on the given request parameters.

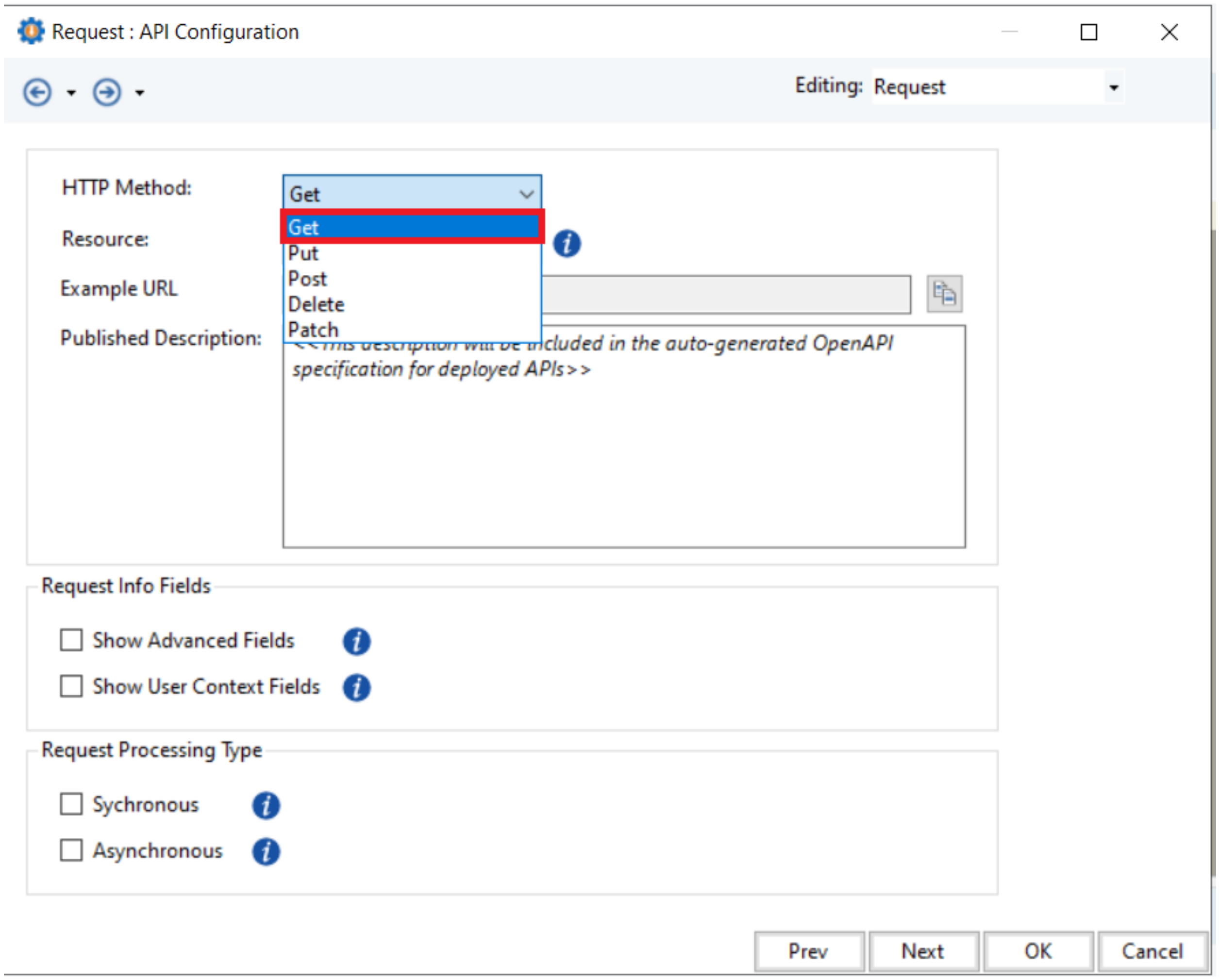
3. Post: Used to create a new object resource. In addition to request parameters, the POST method also allows a request payload which can be defined as the input layout.

4. *Put: U*sed to update an existing resource. It also allows input parameters and a payload in the request definition.

5. *Patch*: This method qualifies to partially update an existing resource. It also allows input parameters and a payload in the request definition.

Note: For this demonstration, we will be configuring a GET API flow.

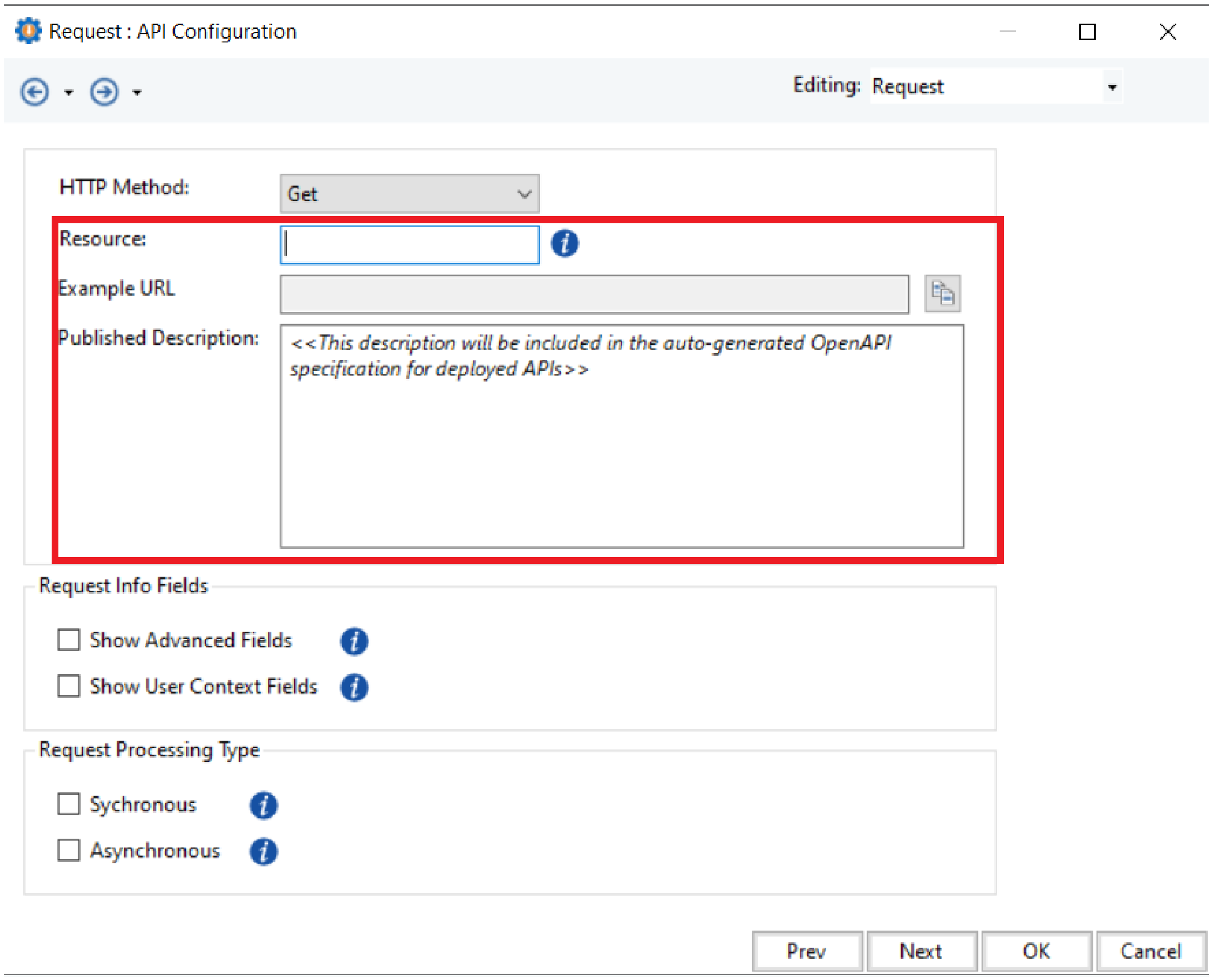
Resource: The resource entity for the REST API operation. It becomes part of the request URL. Here, since we are designing an API to read order items, we will call our resource “OrderItems”.



Note: Nested resources can also be defined using a /.

Example URL: It displays the complete request URL formed with the appended resource and parameters.

Published Description: A description for the API operation on the given resource. You could use the default generated description or modify it to your own description. This description for the API endpoint will be used in the auto-generated Open API Swagger specification.


Request : API Configuration

Editing: Request

HTTP Method: Get

Resource:

Example URL

Published Description: *<<This description will be included in the auto-generated OpenAPI specification for deployed APIs>>*

Request Info Fields

- [ ] Show Advanced Fields
- [ ] Show User Context Fields

Request Processing Type

- [ ] Sychronous
- [ ] Asynchronous

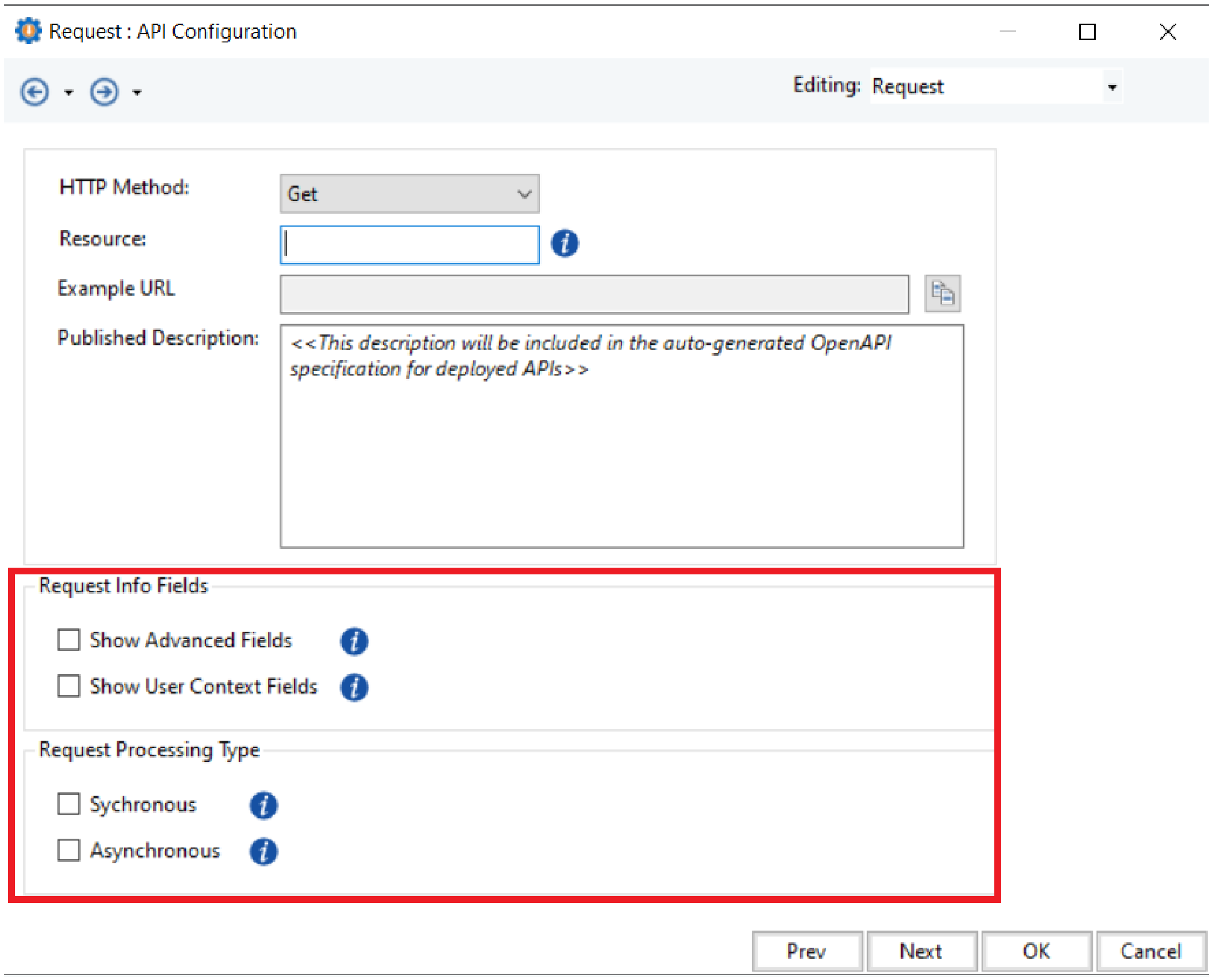
Show Advanced Fields: Enabling this will display additional information fields in the *requestinfo* node. These include information about the incoming request that can be further used in the API flow, such as connection, local address, local port, IP Address, etc.

Show User Context Fields: Enabling this will display user profile fields from the incoming request in the *requestinfo* node. These fields show information such as username, email address, whether the user is locked out or not, etc.

Synchronous: API request executes synchronously such that an API call blocks and returns to the client only when a response from the server is ready.

Asynchronous: To deploy the API as an asynchronous operation. This implies that the requestee does not need to wait for the response to be processed. On making a request, the server responds with a *202 Accepted* message and starts to process. The client can periodically check the status and read the response when available.

To learn more about Synchronous and Asynchronous, click here.


Request : API Configuration

Editing: Request

HTTP Method: Get

Resource:

Example URL

Published Description: <<This description will be included in the auto-generated OpenAPI specification for deployed APIs>>

Request Info Fields

☐ Show Advanced Fields

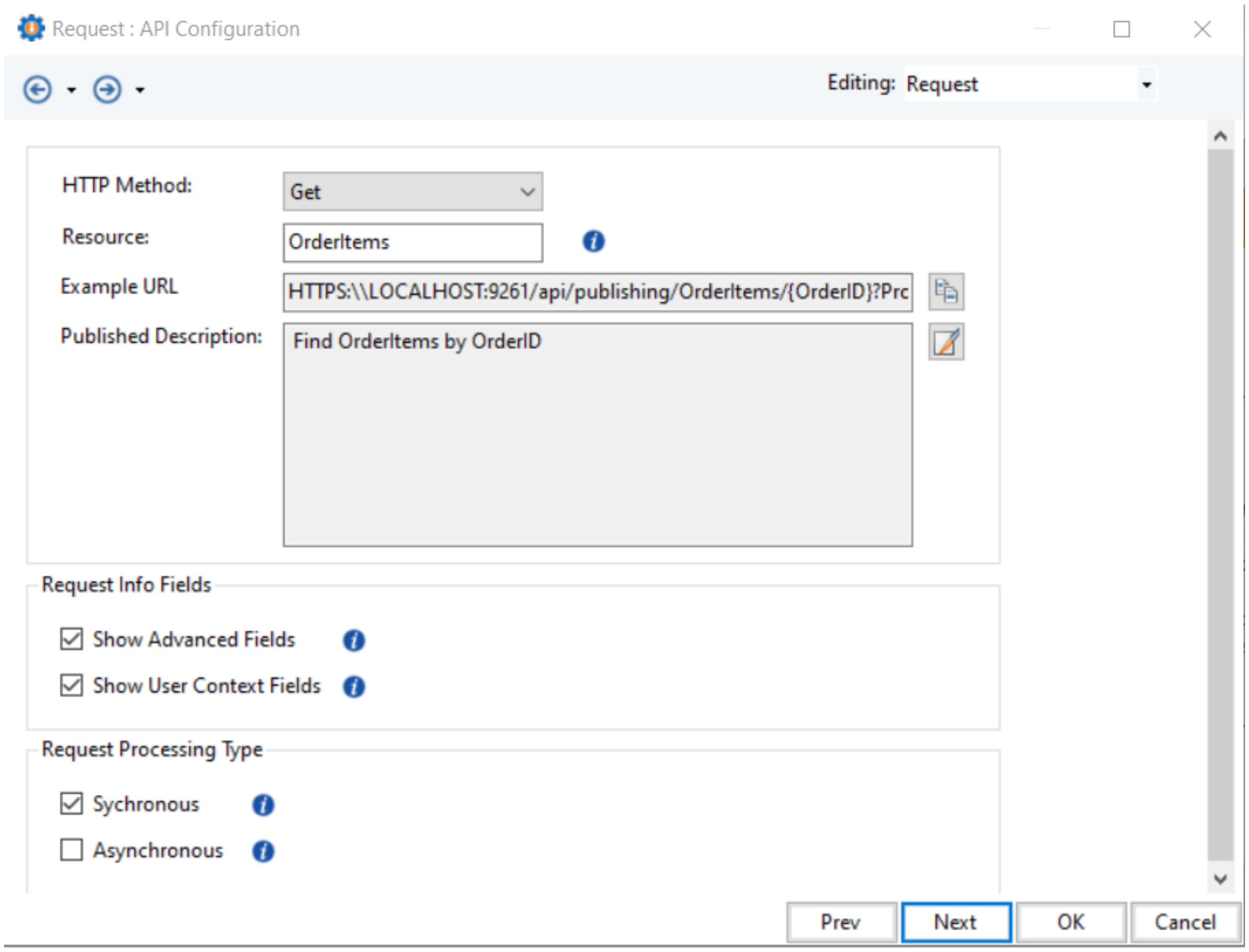
☐ Show User Context Fields

Request Processing Type

☐ Sychronous

☐ Asynchronous

1. For our use case, we are using the *Get* method to find order items by ‘OrderID’.



HTTP Method: Get

Resource: OrderItems

Example URL: HTTPS:\\LOCALHOST:9261/api/publishing/OrderItems/{OrderID}?Prc

Published Description: Find OrderItems by OrderID

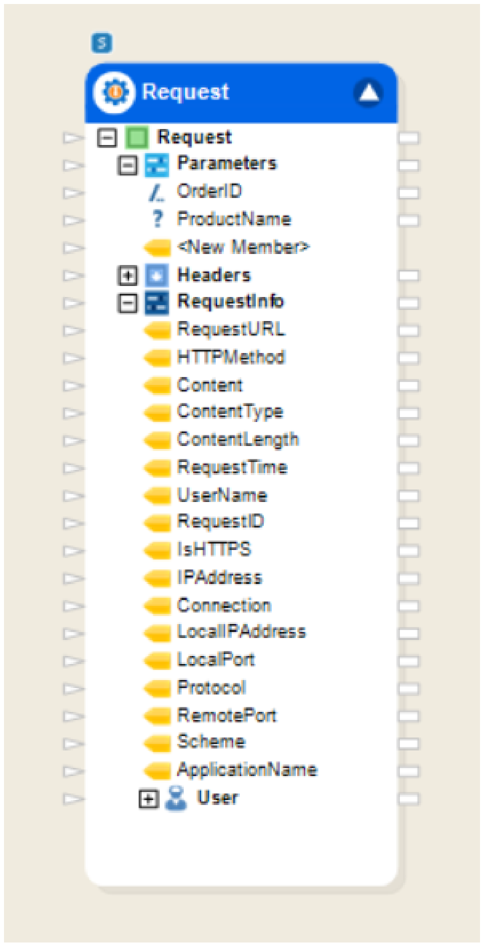
Request Info Fields

- [✓] Show Advanced Fields
- [✓] Show User Context Fields

Request Processing Type

- [✓] Sychronous
- [ ] Asynchronous

2. Click *OK*, and your request object will be configured using the *Get* method.



As you can see above, the object has been configured by our requirements.

Note: Under the requestinfo node, additional fields will appear depending on which checkbox is selected in the *API Configuration* window.

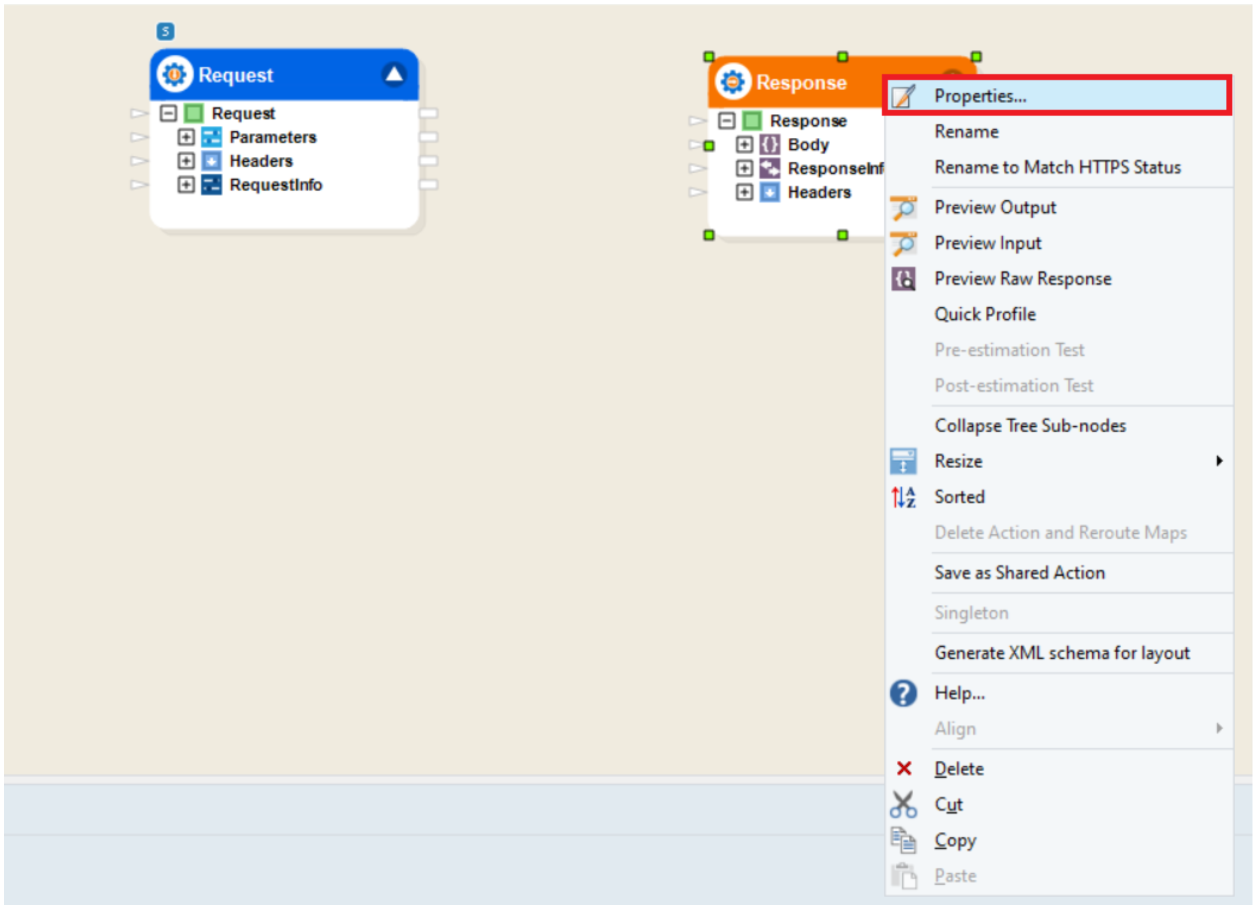
This concludes the configuration of the *REST Request* object in Astera Centerprise.

Configuring The REST Response Object

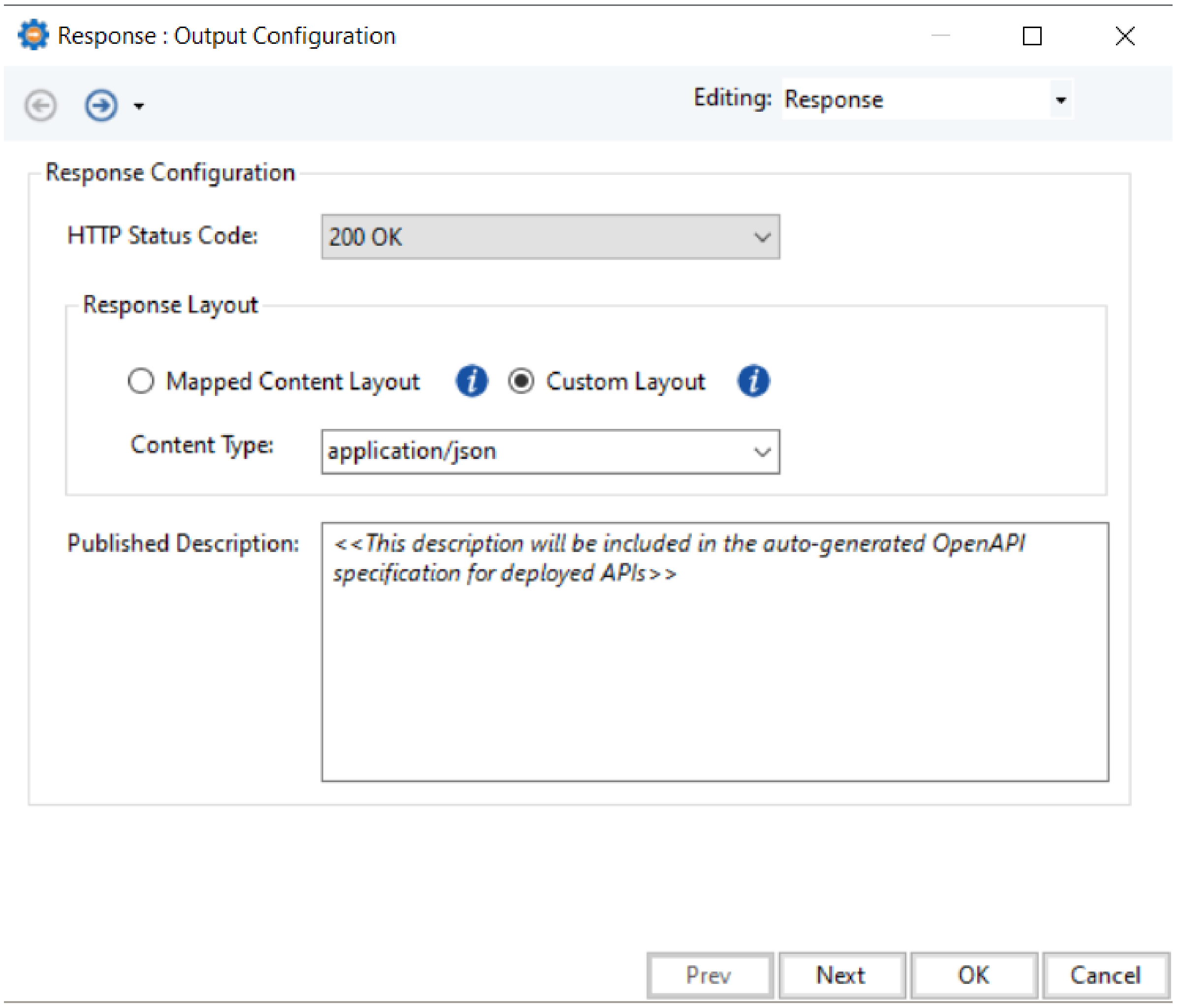
An API endpoint flow begins with a request object and ends at one of the many responses defined as per the flow execution route.

Note: At least one response object must be configured to complete the API endpoint flow.

1. To start, right-click on the object and select *Properties* from the context menu.



Then, the response configuration window will open, where you can specify the HTTP status code to be returned.



HTTP Status Code: 200 OK

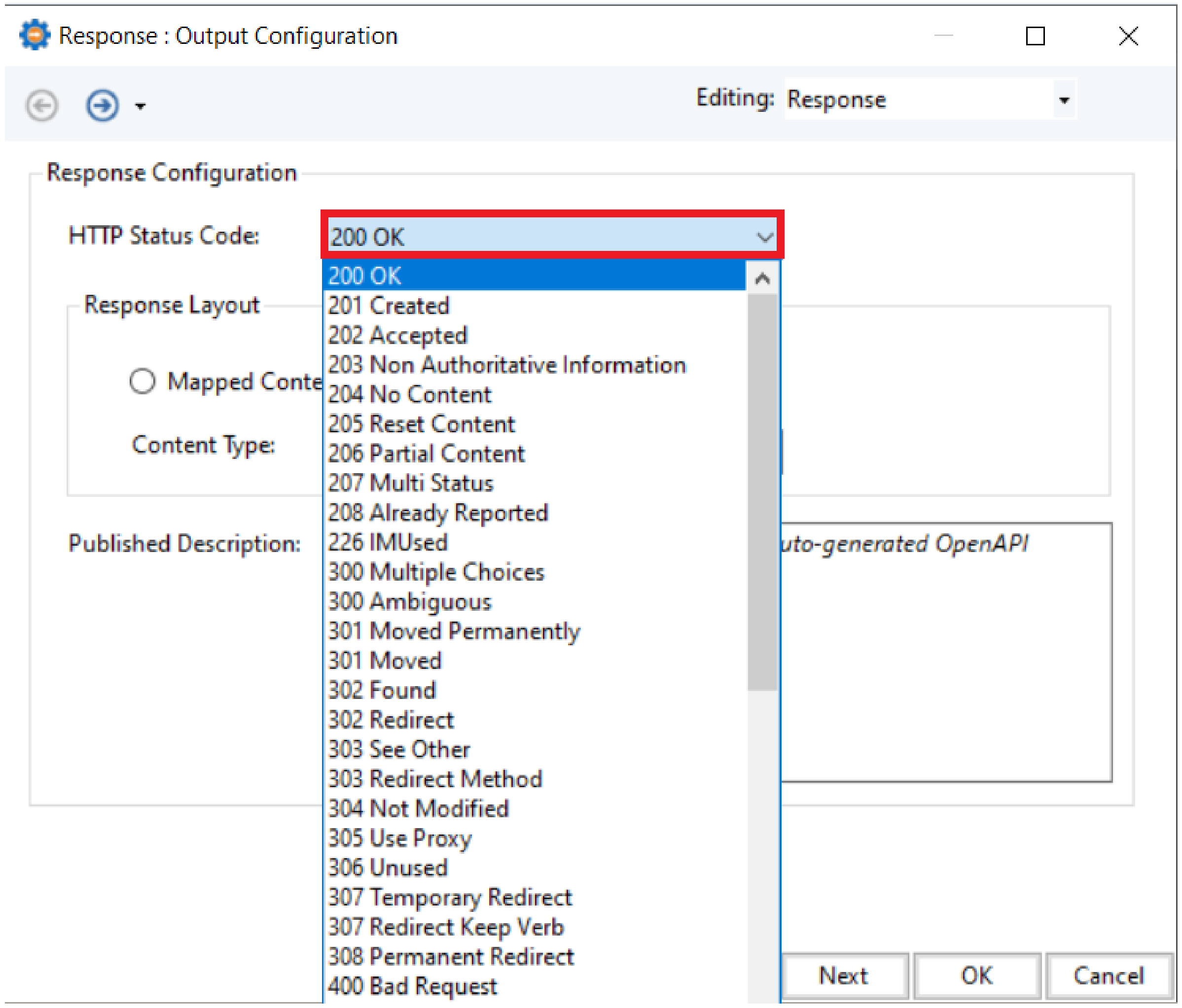
Mapped Content Layout [✓ Custom Layout]

Content Type: application/json

Published Description: <<This description will be included in the auto-generated OpenAPI specification for deployed APIs>>

HTTP Status Code: API response will be based on this selection of the HTTP status code and the API flow orchestration designed. These codes can be designed based on successful runs or errors etc.

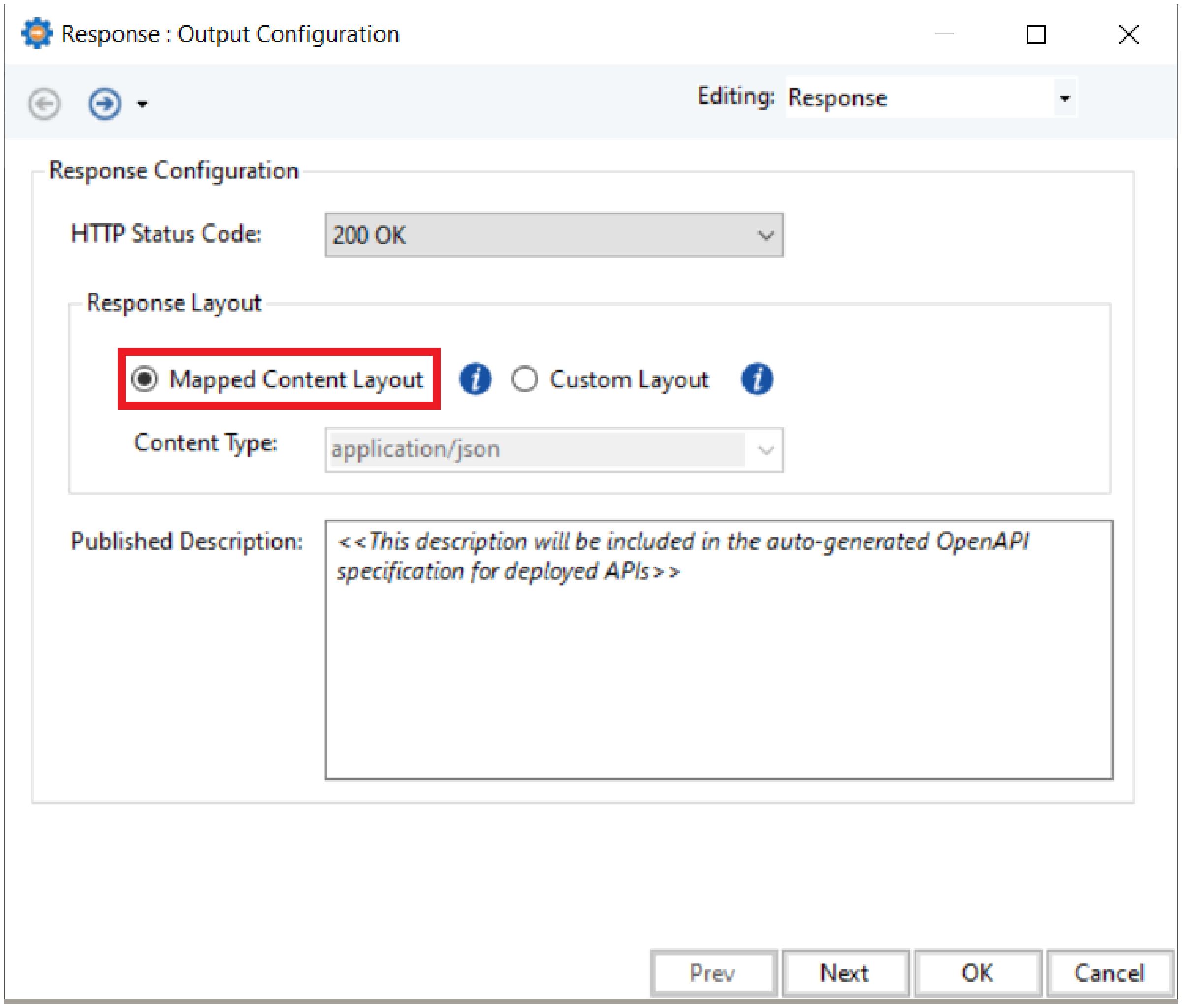
To learn more about multiple responses in an API flow, click here.



HTTP Status Code: 200 OK

200 OK
201 Created
202 Accepted
203 Non Authoritative Information
204 No Content
205 Reset Content
206 Partial Content
207 Multi Status
208 Already Reported
226 IMUsed
300 Multiple Choices
300 Ambiguous
301 Moved Permanently
301 Moved
302 Found
302 Redirect
303 See Other
303 Redirect Method
304 Not Modified
305 Use Proxy
306 Unused
307 Temporary Redirect
307 Redirect Keep Verb
308 Permanent Redirect
400 Bad Request

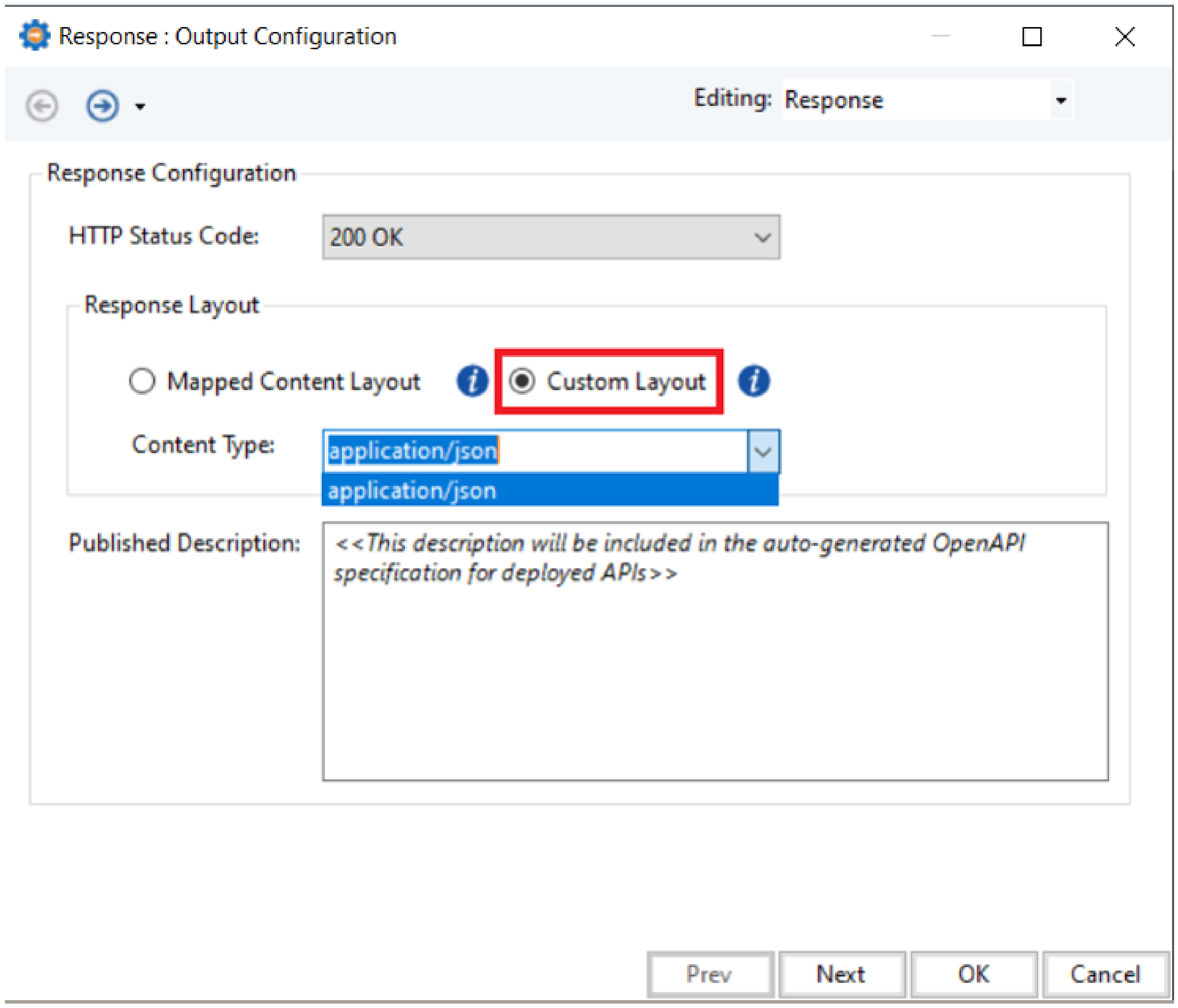
Mapped Content Layout: Selecting this option allows you to map a pre-serialized response string and its content type as input to the response object.



<<This description will be included in the auto-generated OpenAPI specification for deployed APIs>>

Custom Layout: Selecting this will allow you to build a layout for the response on the next screen based on the content type defined.

Content Type: You can select a standard media type for the response payload and define its custom layout on the next screen. Currently, you can only define custom layouts of JSON type.



Response : Output Configuration

Editing: Response

Response Configuration

HTTP Status Code: 200 OK

Response Layout

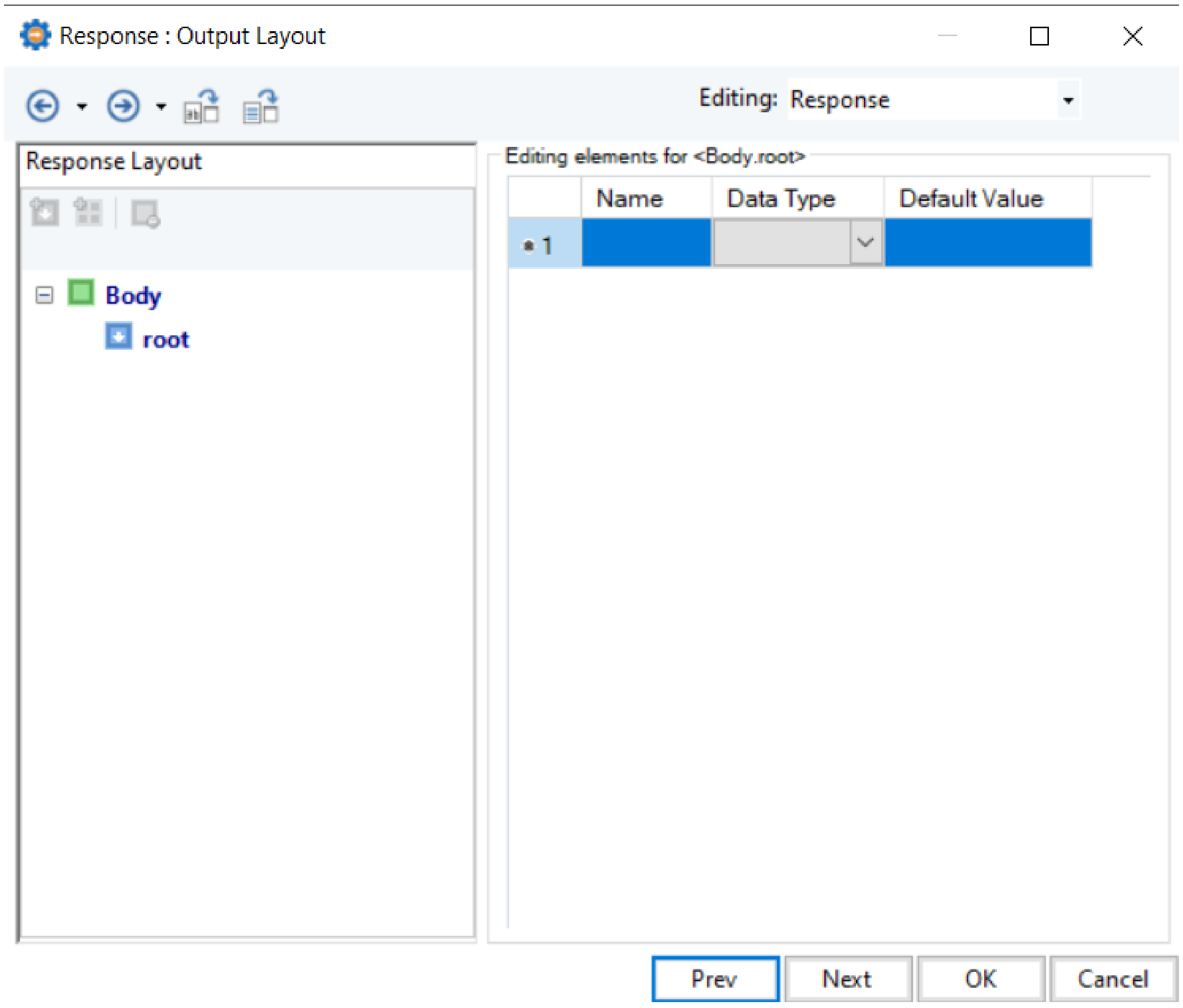
Mapped Content Layout [✓ Custom Layout]

Content Type: application/json

Published Description: <<This description will be included in the auto-generated OpenAPI specification for deployed APIs>>

Published Description: This description becomes a part of the auto-generated open API definition.

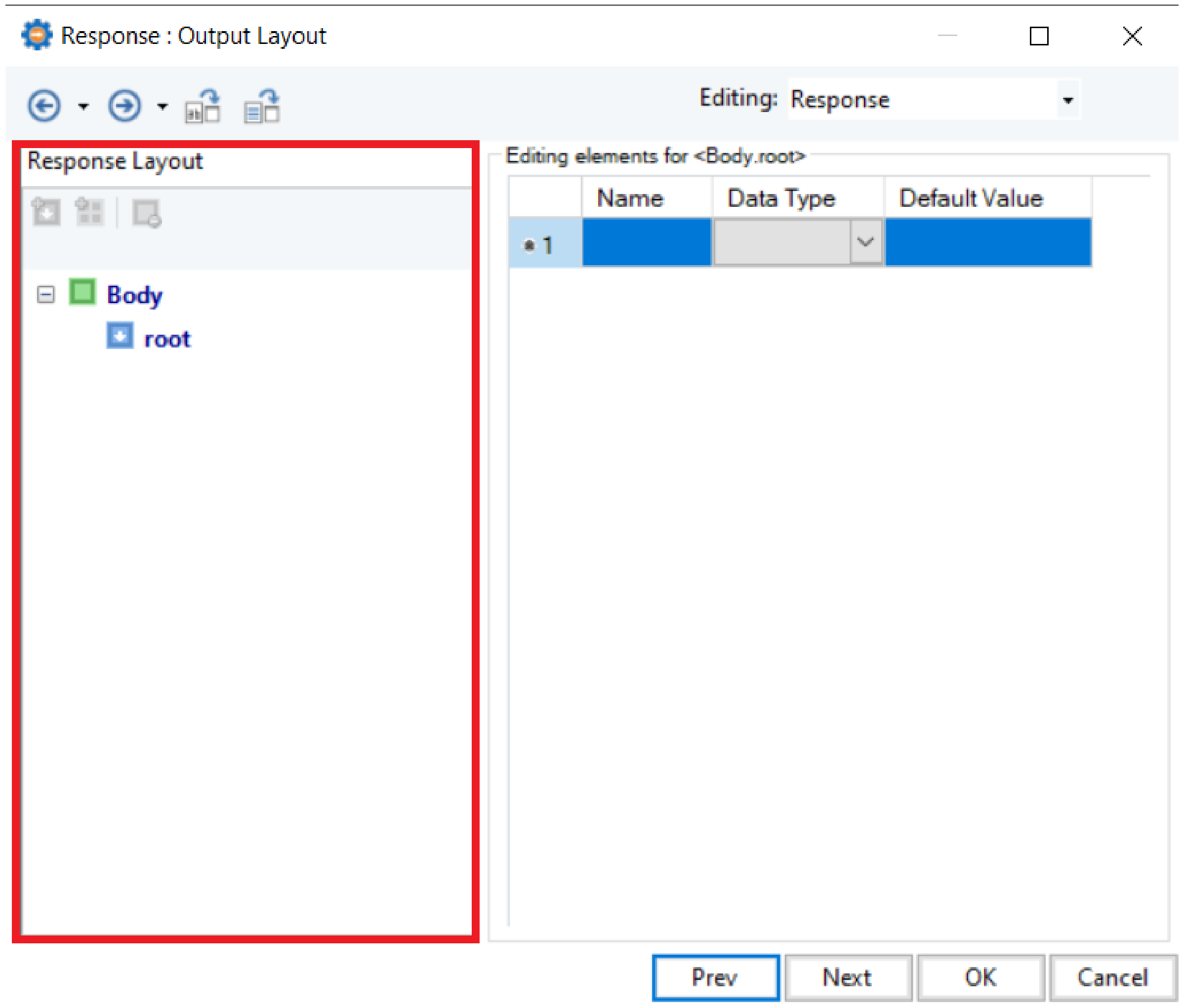
2. Once done, click *Next* and you will be taken to the next screen.



	Name	Data Type	Default Value
1			

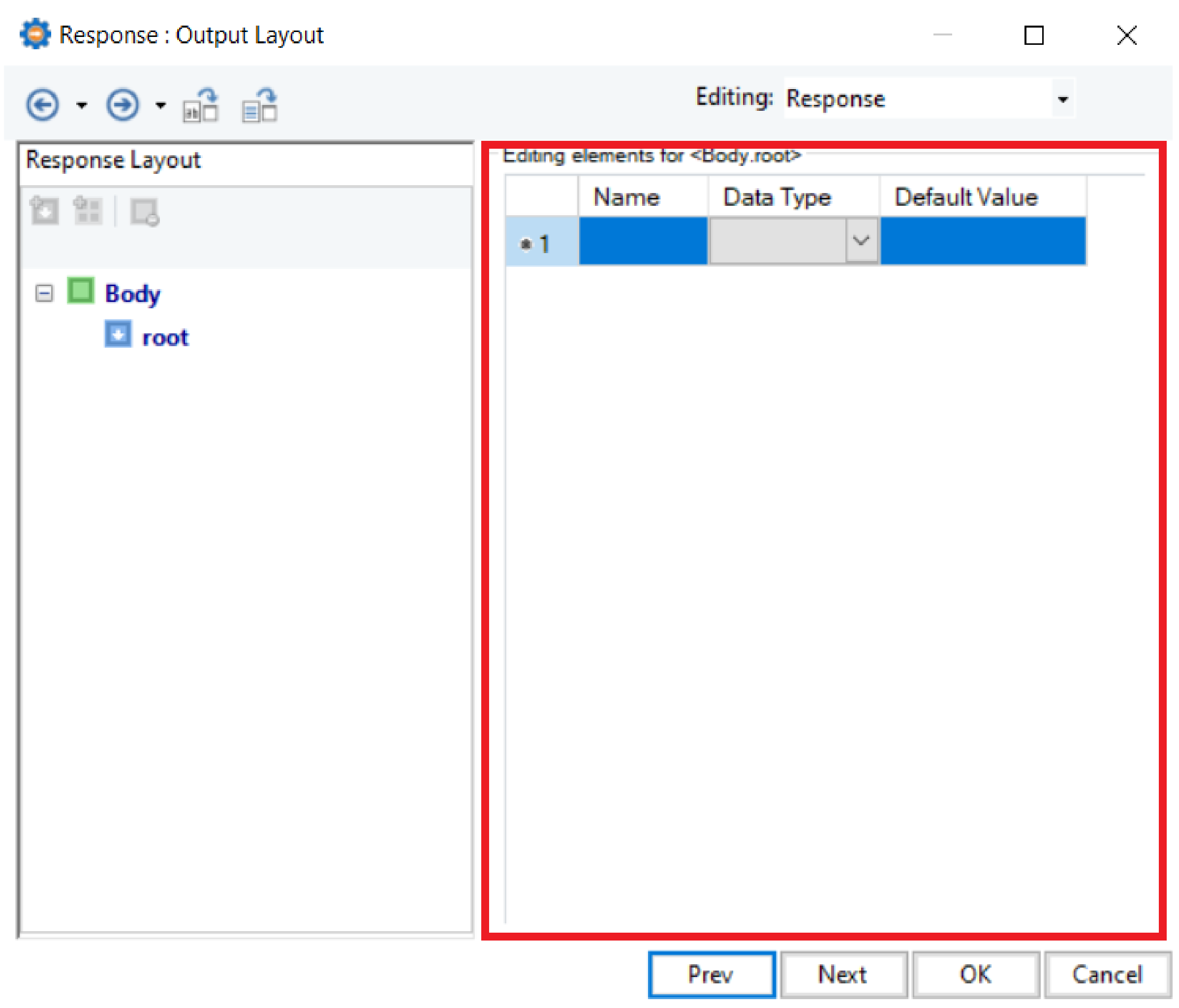
Here, the output layout of the response object can be defined.

On the left side of the screen is the hierarchy of the nodes in your layout. You can add or delete a single instance or collection members here to create the desired layout.



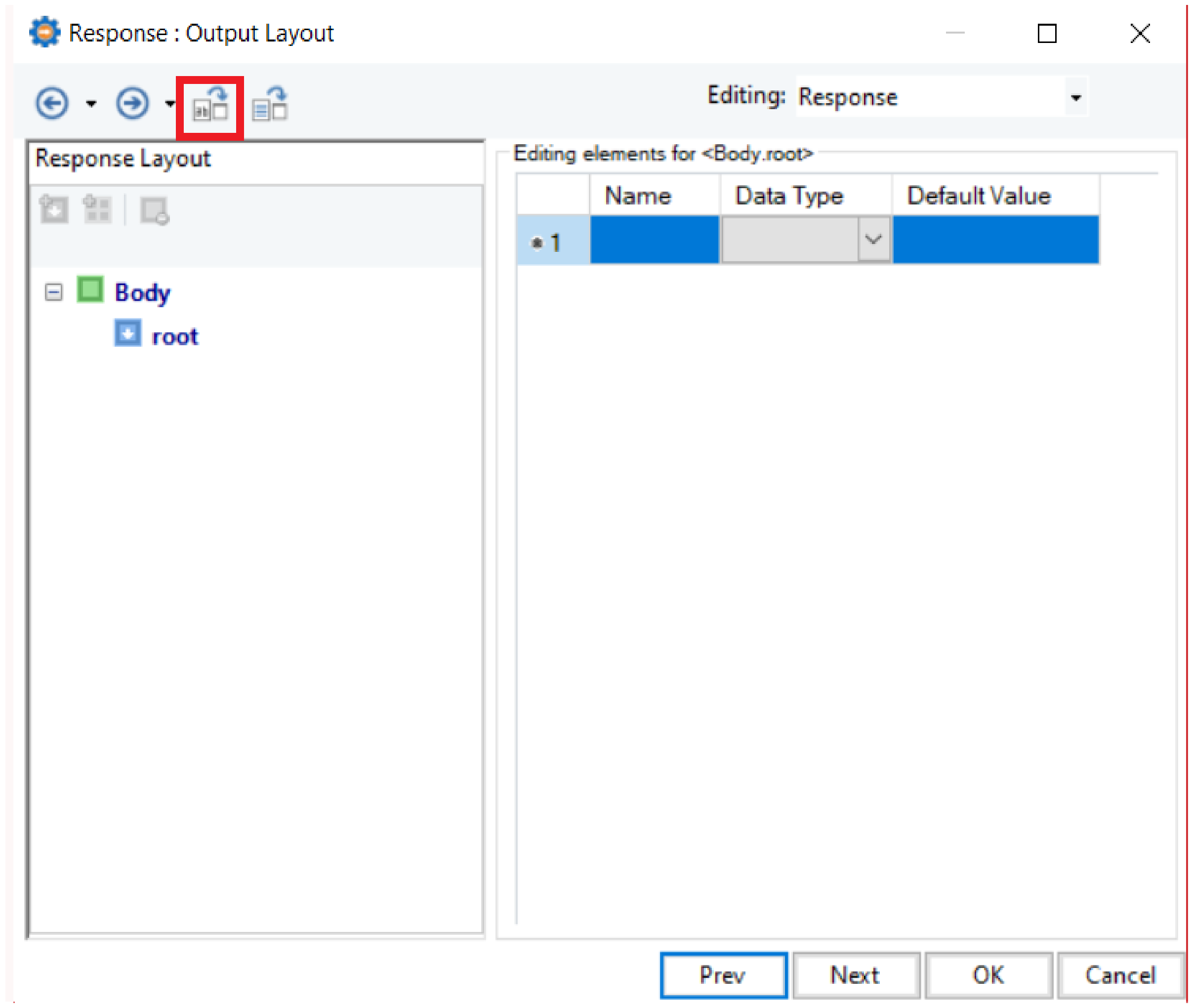
The right side of the screen is where the layout is to be added. There are three ways to map the output layout,

- Manually defining objects and fields,



- Using a sample text.

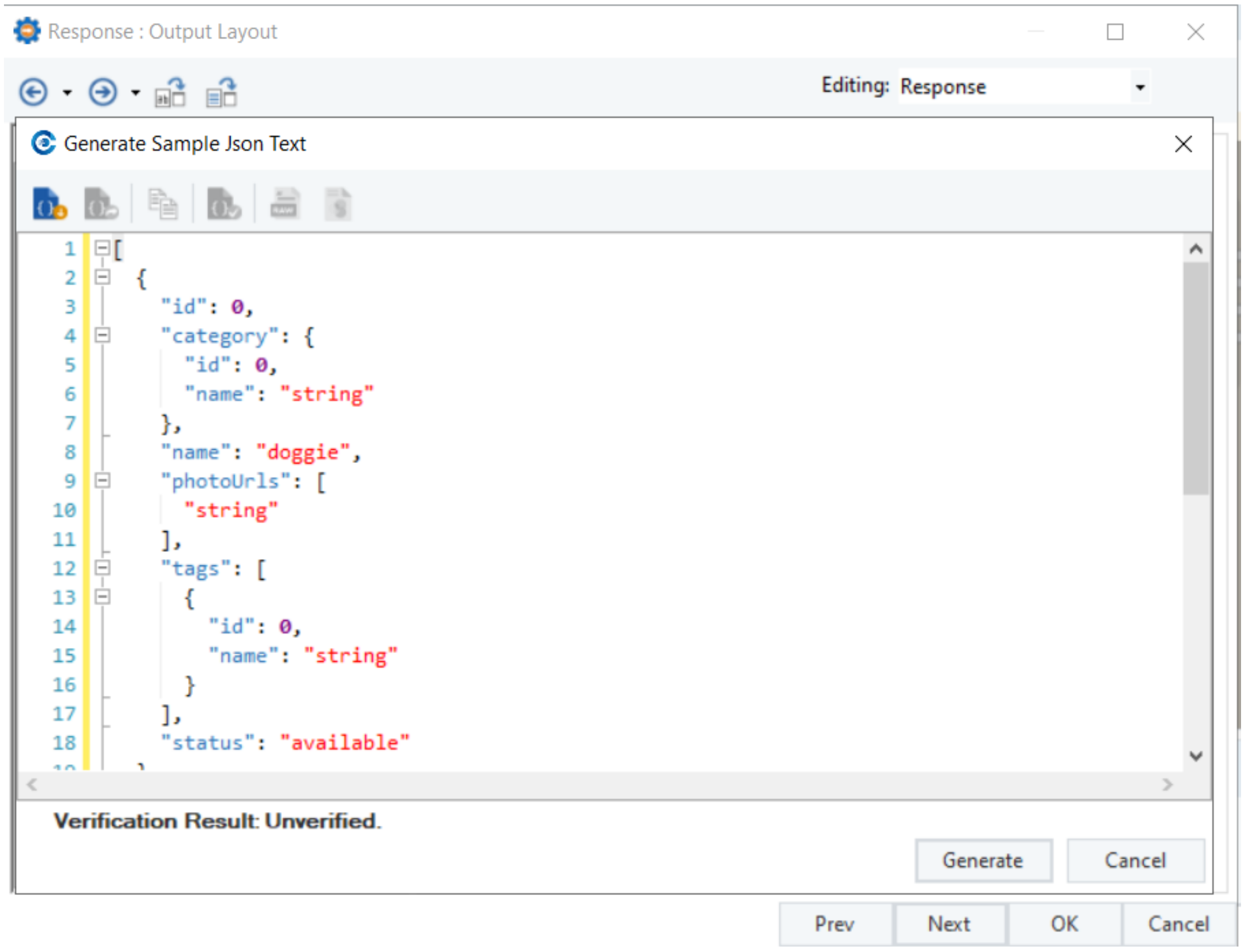
Selecting this option will open a new window where a sample text, based on the Content-type defined, is given to generate the layout accordingly.



Editing elements for <Body.root>

	Name	Data Type	Default Value
* 1			

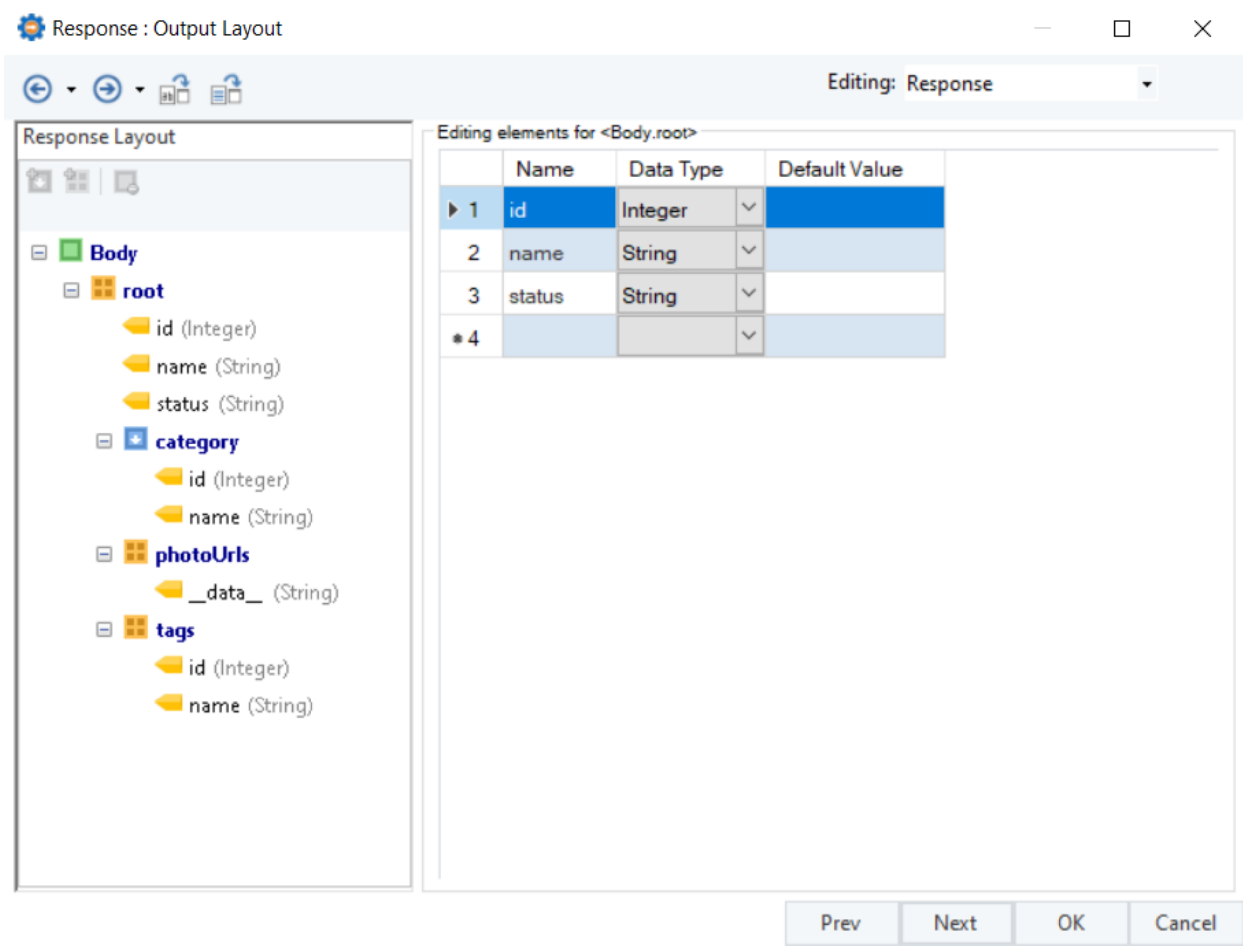
Clicking this will open a new sample text window where the text can be pasted.



```
[
  {
    "id": 0,
    "category": {
      "id": 0,
      "name": "string"
    },
    "name": "doggie",
    "photoUrls": [
      "string"
    ],
    "tags": [
      {
        "id": 0,
        "name": "string"
      }
    ],
    "status": "available"
```

Verification Result: Unverified.

Clicking *Generate* will produce a layout.


Response : Output Layout

Editing: Response

Response Layout

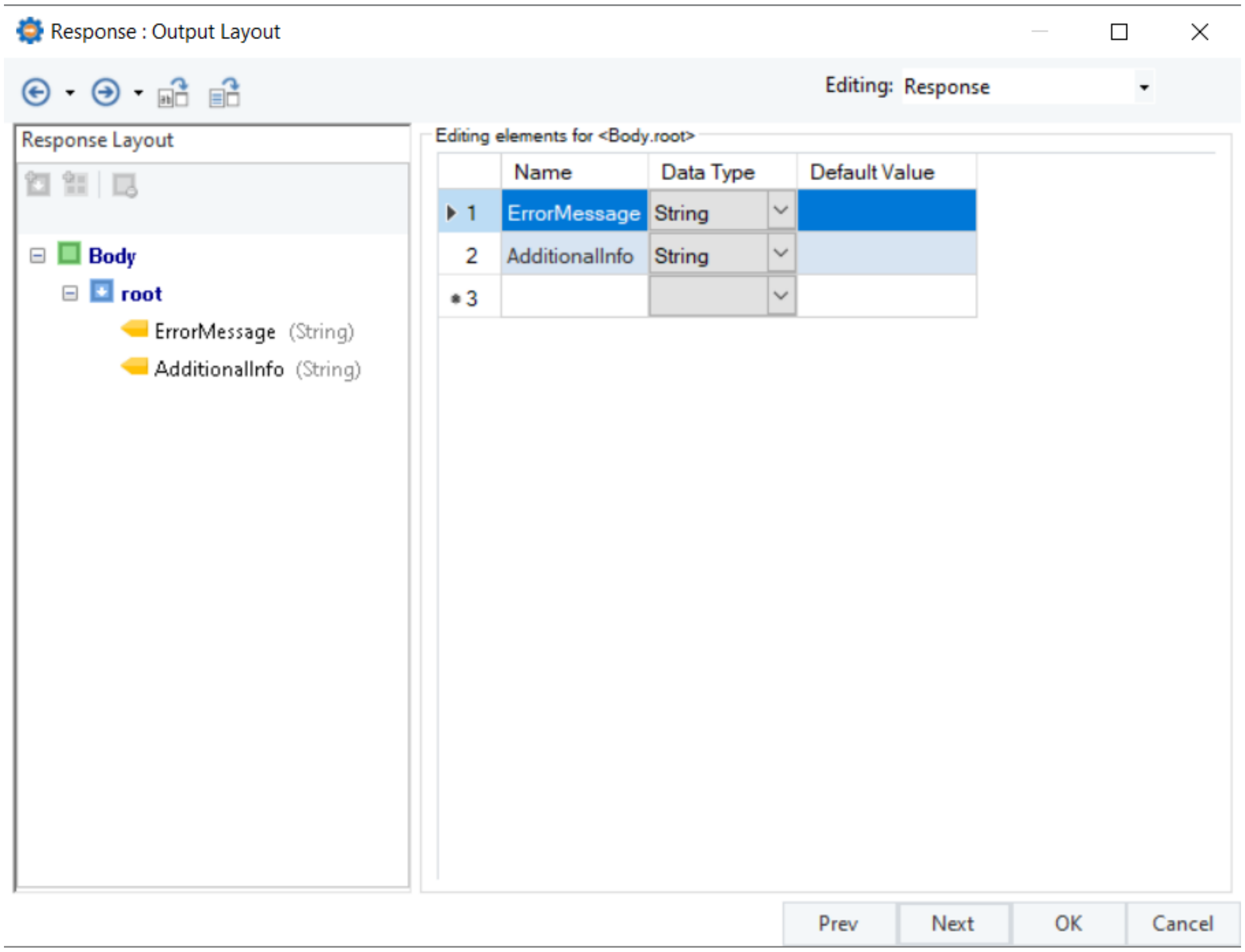
- Body
  - root
    - id (Integer)
    - name (String)
    - status (String)
    - category
      - id (Integer)
      - name (String)
    - photoUrls
      - __data__ (String)
    - tags
      - id (Integer)
      - name (String)

Editing elements for <Body.root>

	Name	Data Type	Default Value
1	id	Integer	
2	name	String	
3	status	String	
4			

- Generate default layout

Selecting this option will generate a default Centerprise layout which can be used to output an error and any additional messages.

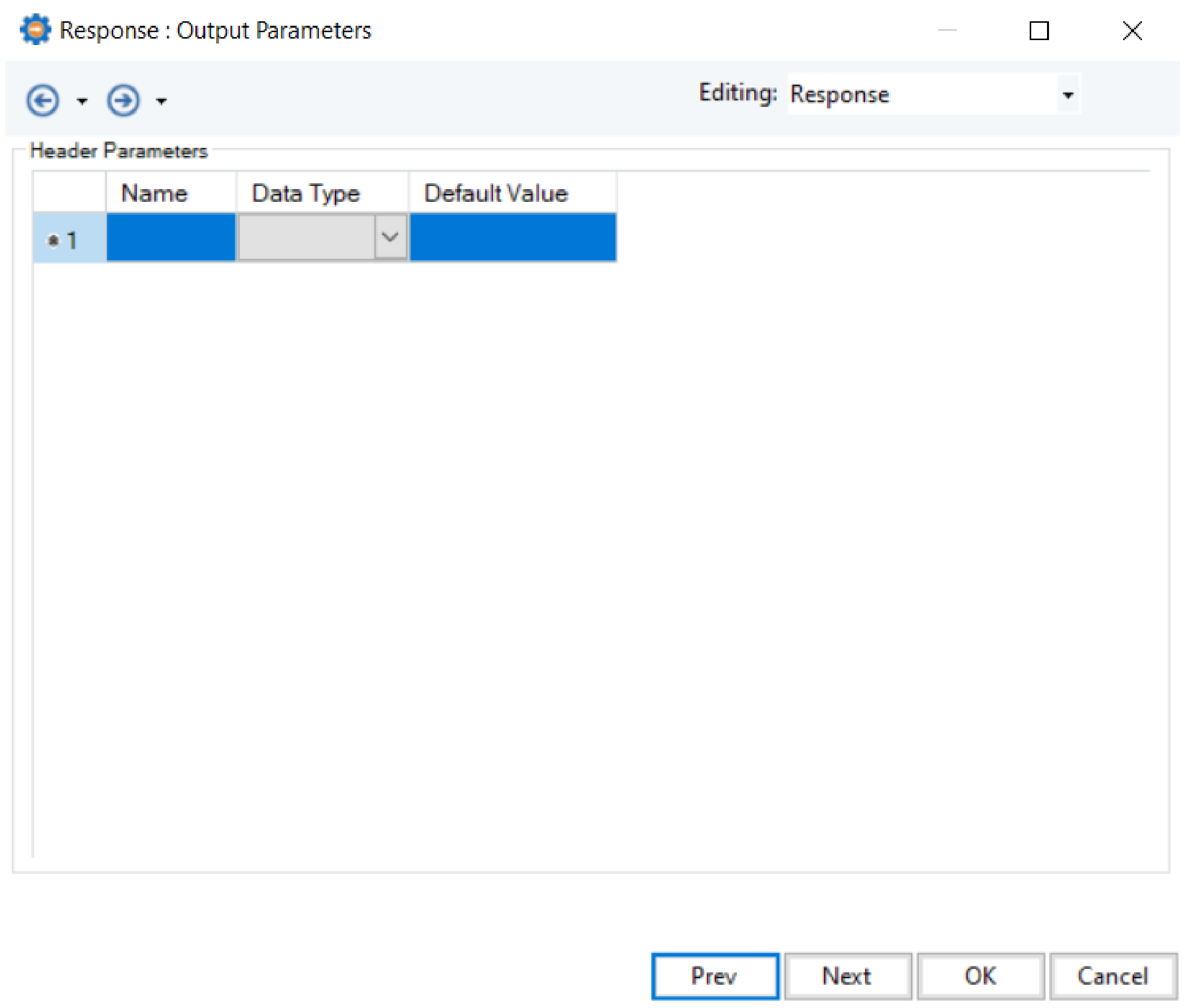


	Name	Data Type	Default Value
1	ErrorMessage	String	
2	AdditionalInfo	String	
3			

As we click on this option, the box is populated.

3. Once done, click *Next* and you will be led to the *Output Parameters* screen.

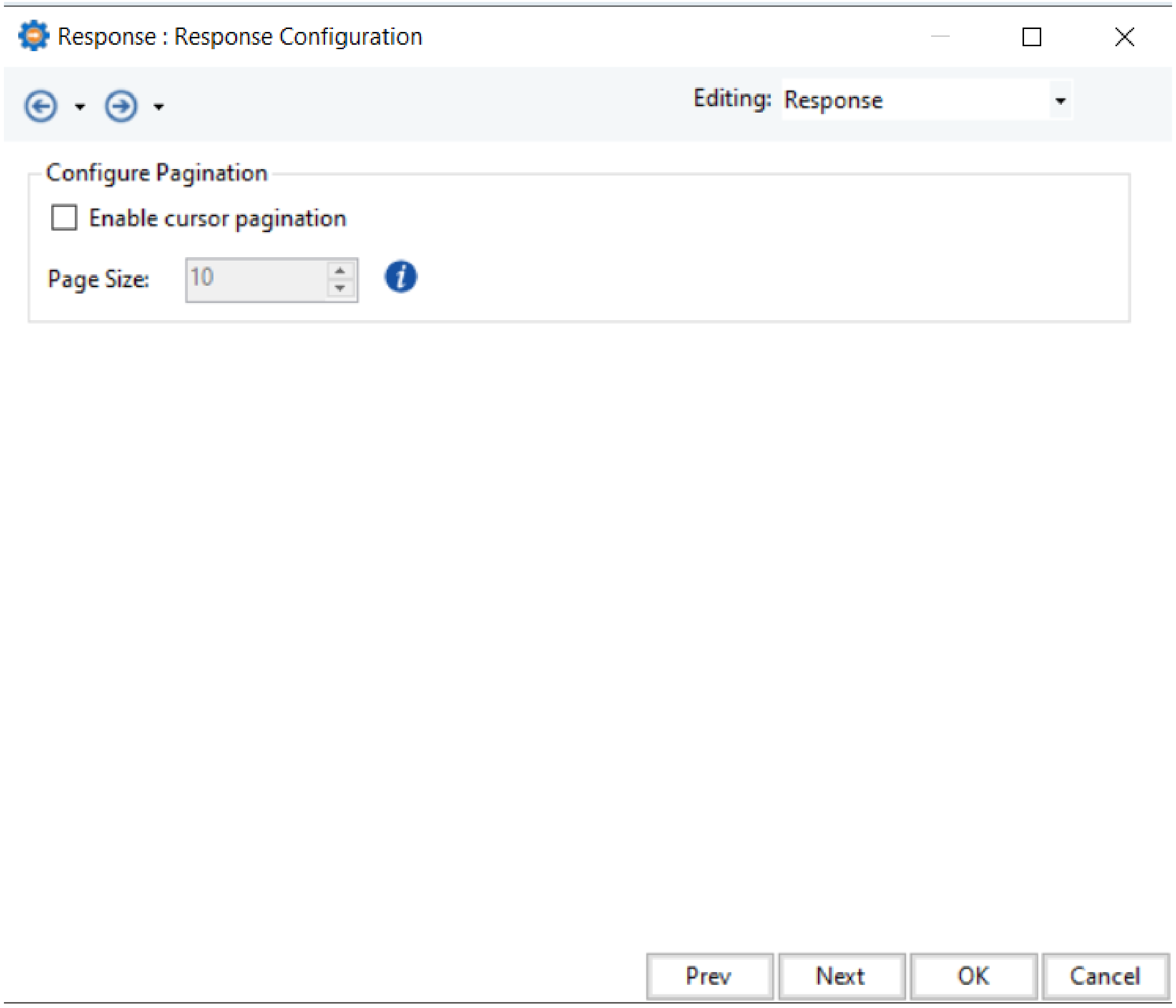
Here, you can define header parameters to be returned as part of the response.



4. Click *Next* when done and you will be led to the *Pagination* screen.

The pagination screen allows you to set a form of pagination on the *REST Response* data. You can configure Cursor Pagination for 200 OK responses to retrieve ordered data in small discrete sets, as requested. The first request returns the records as per *Page Size* and a cursor field which can then be iteratively used to read the next set of records.

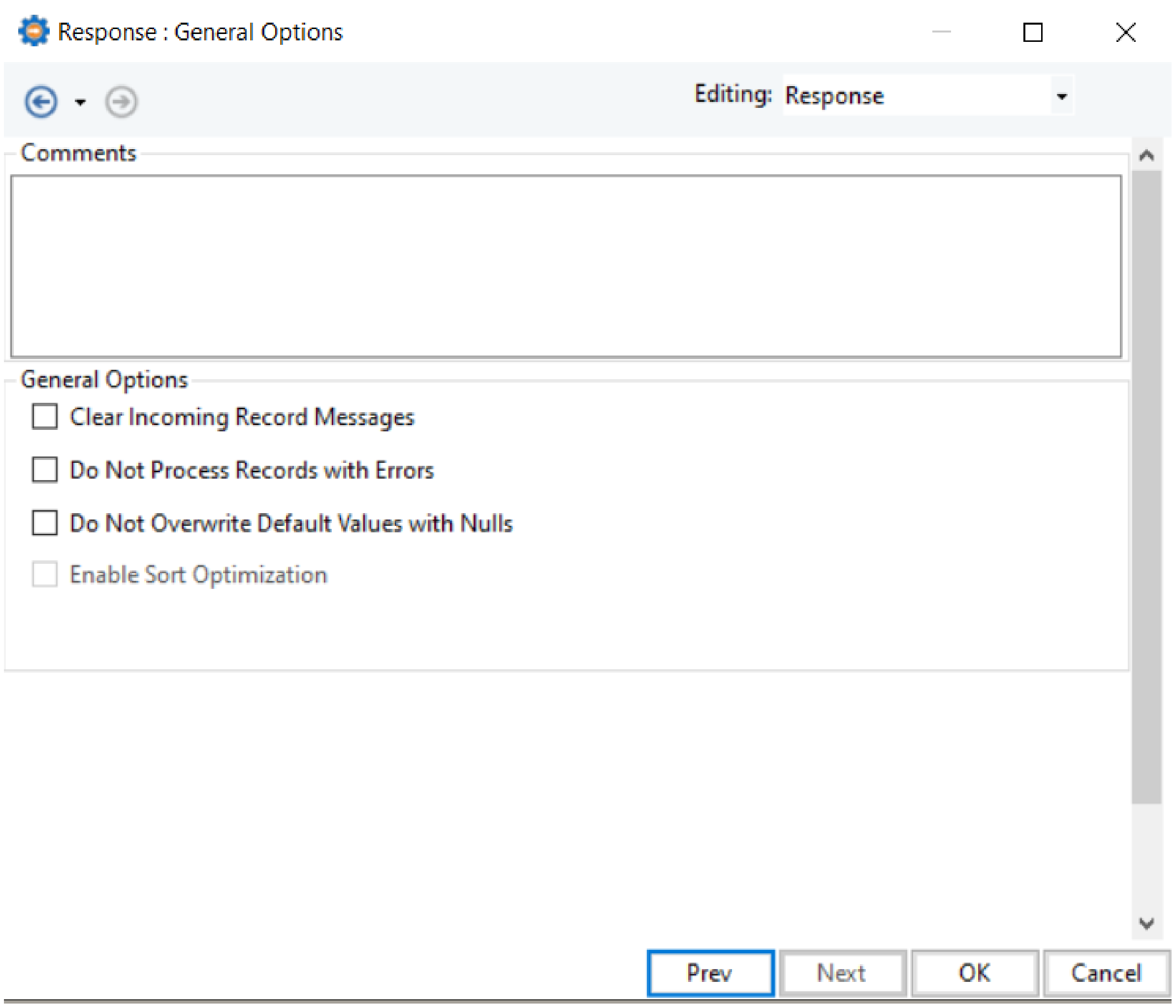
To learn more about pagination, click here.



Enable Cursor Pagination: Selecting this is going to enable *Cursor Pagination* on the data payload returned.

Page Size: This determines the size of a page in cursor pagination.

5. Once that has been done, select *Next* and you will be led to the *General Options* screen.


Response : General Options

Editing: Response

Comments

General Options

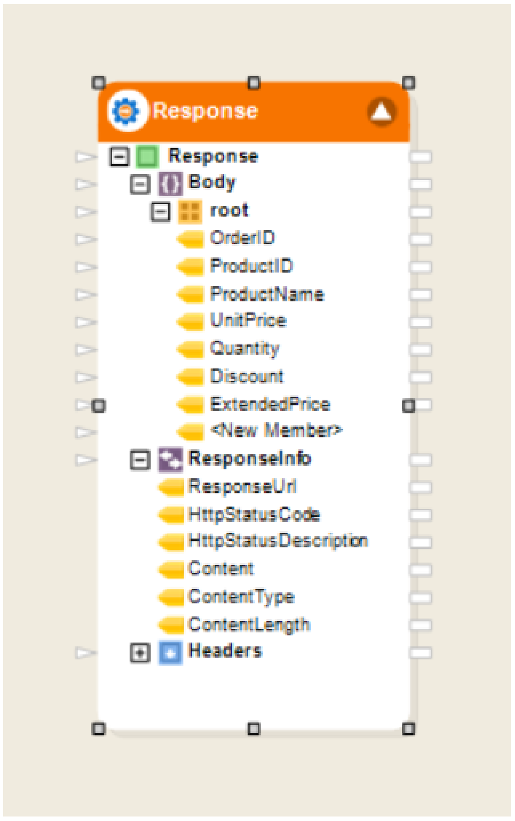
☐ Clear Incoming Record Messages

☐ Do Not Process Records with Errors

☐ Do Not Overwrite Default Values with Nulls

☐ Enable Sort Optimization

6. Click *OK* and the *REST Response* object will be configured. You can now map the fields and parameters from the flow to the response object.



Body: This node will show the output layout hierarchy that has been configured within the properties.

Responseinfo: Upon expansion, this node will display additional information that can be mapped to an output to be processed further in the test-flow, after the API response has been submitted.

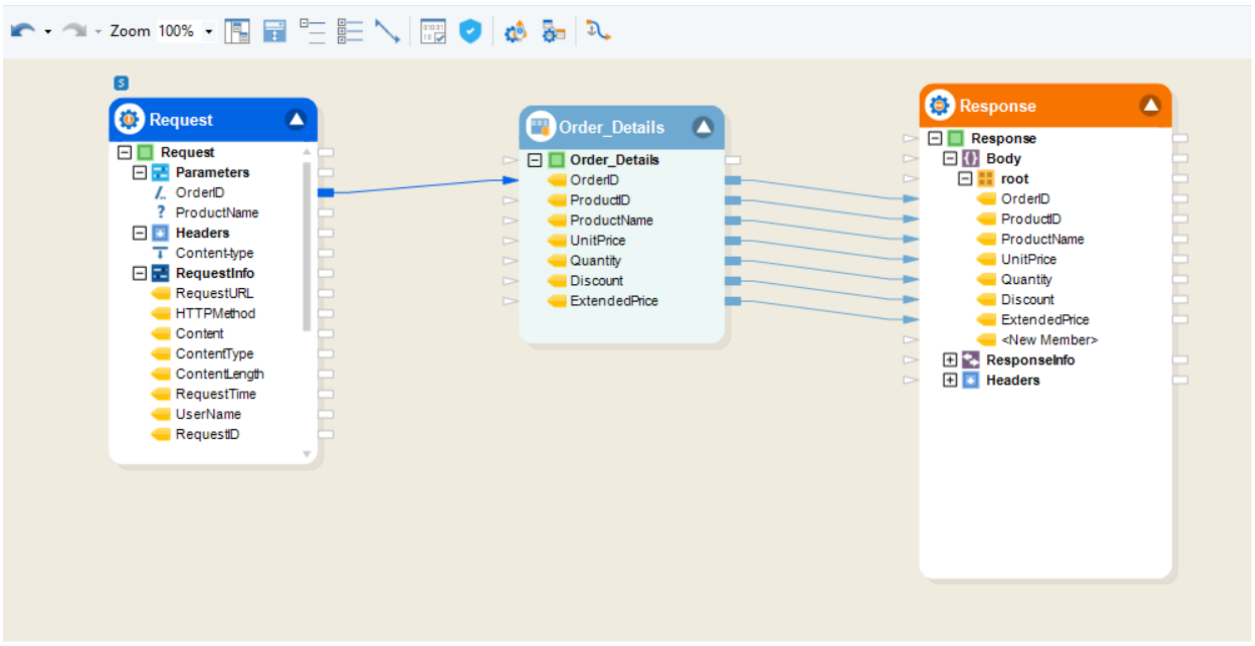
Headers: Expanding this node will show any headers that have been defined within the object.

This concludes the *REST Response* object configuration in Astera API Management.

Using the REST Request and REST Response objects in a flow

Since we have now configured the *REST Request* and *REST Response* objects, we can map them together and use them in a flow. The API flow feeds the input Order ID to a database lookup and returns the output in the response.

The following is a use case built on 'Order_Details' data.

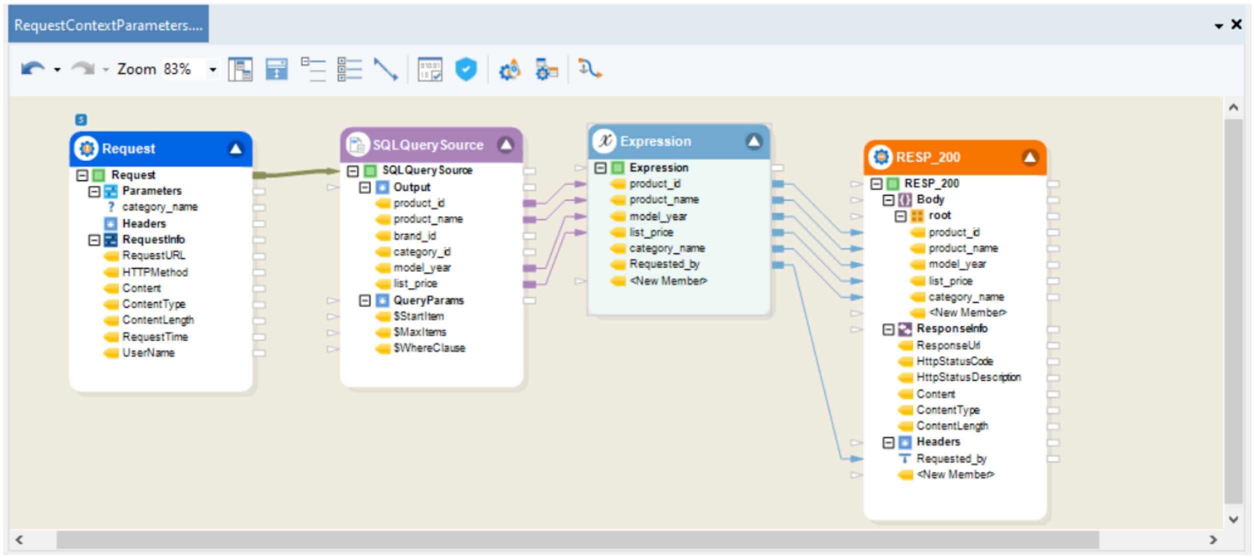


This concludes using the designing of an API flow.

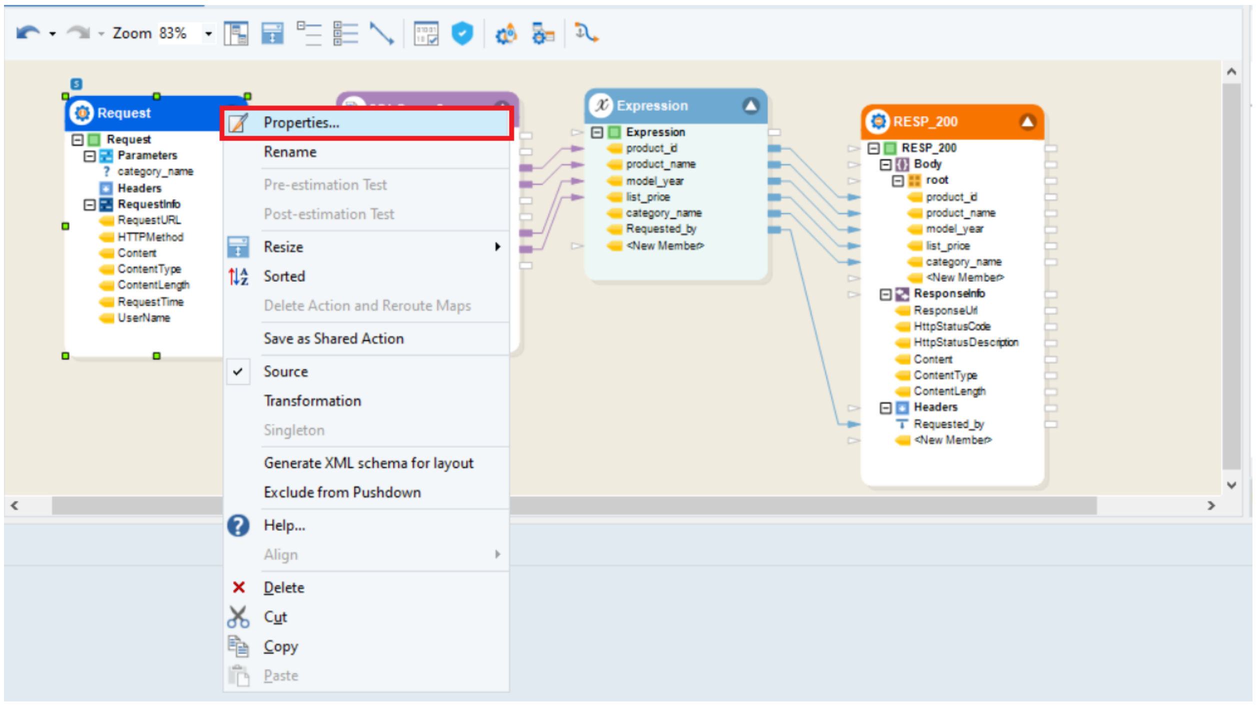
10.2 Request Context Parameters

Request Context Parameters allow us to use Request Parameters anywhere within the scope of the API flow. They can be used at any point within the flow following the Request object.

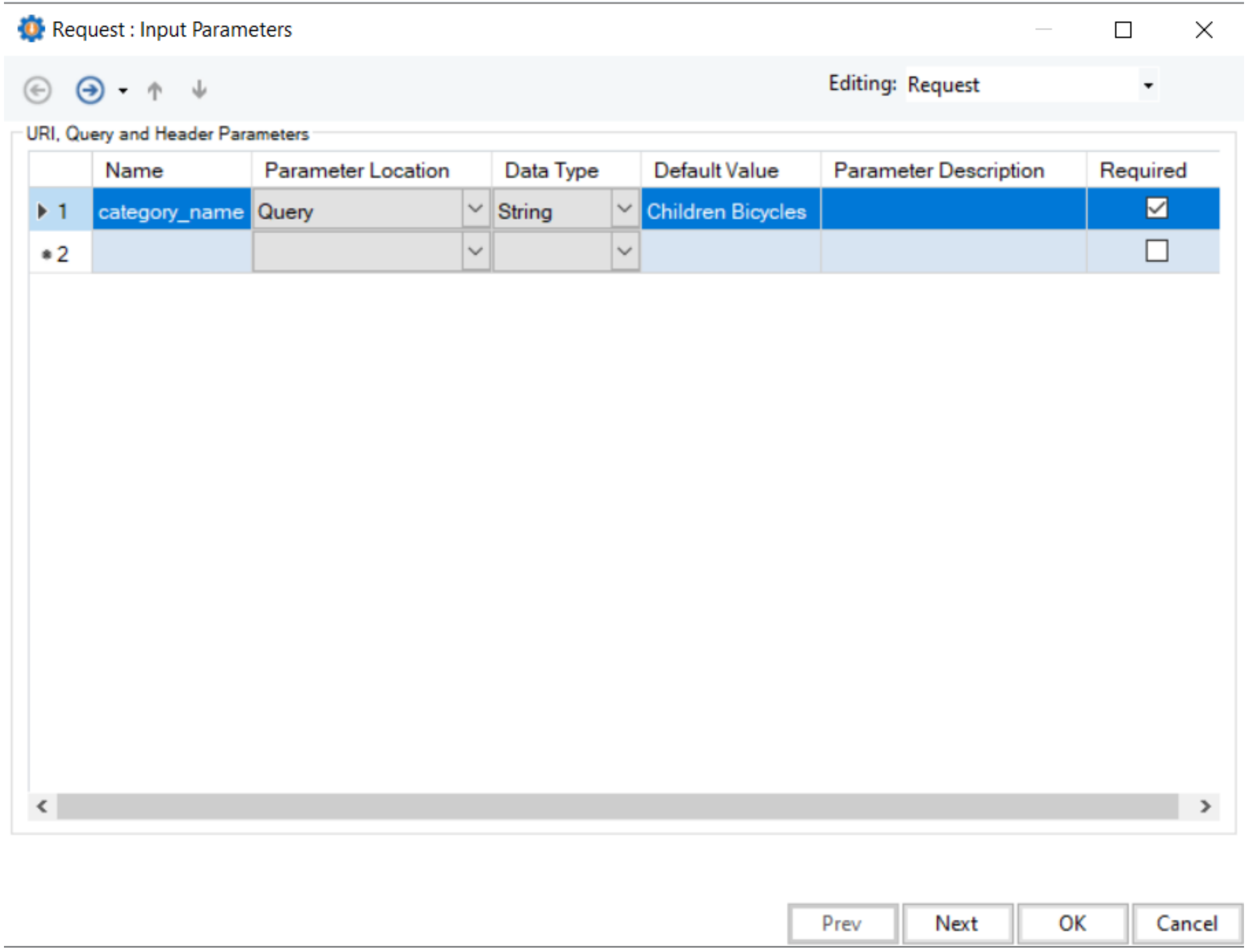
For our use case, we will look at the parameters defined in our flow.



1. Right-click on the *REST Request* object and select *Properties* from the context menu.



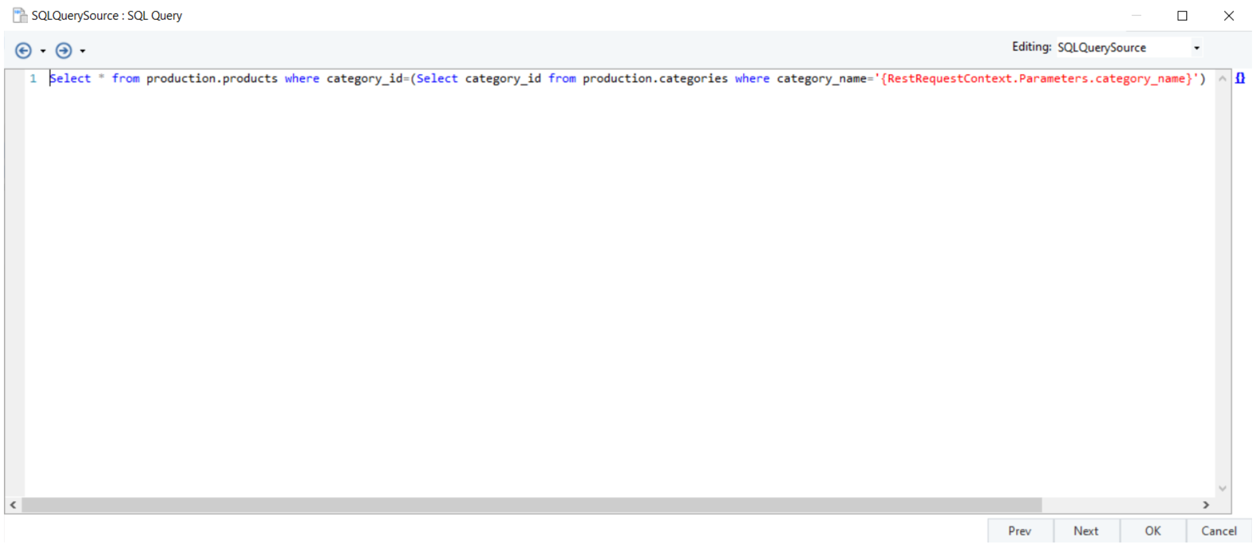
This will open a new window.



	Name	Parameter Location	Data Type	Default Value	Parameter Description	Required
1	category_name	Query	String	Children Bicycles		☑
2						☐

As you can see in the image above, we have defined a query parameter. This can be directly used as a variable further in the API flow to design the flow logic and set values.

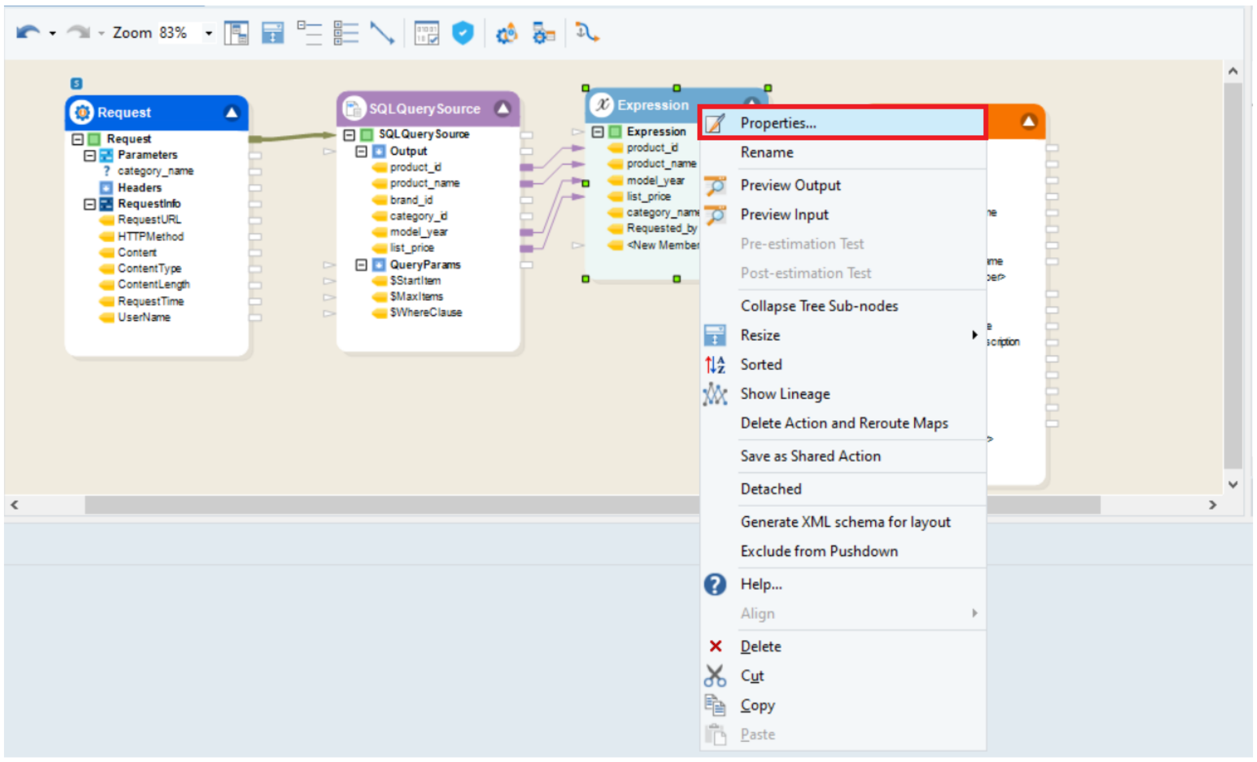
Following the request object, the flow uses a *SQL Query Source* object to read all products where the category name matches the request parameter value. Since a *SQL Query Source* requires a SQL query to be defined, let's see how we can use the incoming query parameter in the context of the SQL query.



```
Select * from production.products where category_id=(Select category_id from production.categories where category_name='{RestRequestContext.Parameters.category_name}')
```

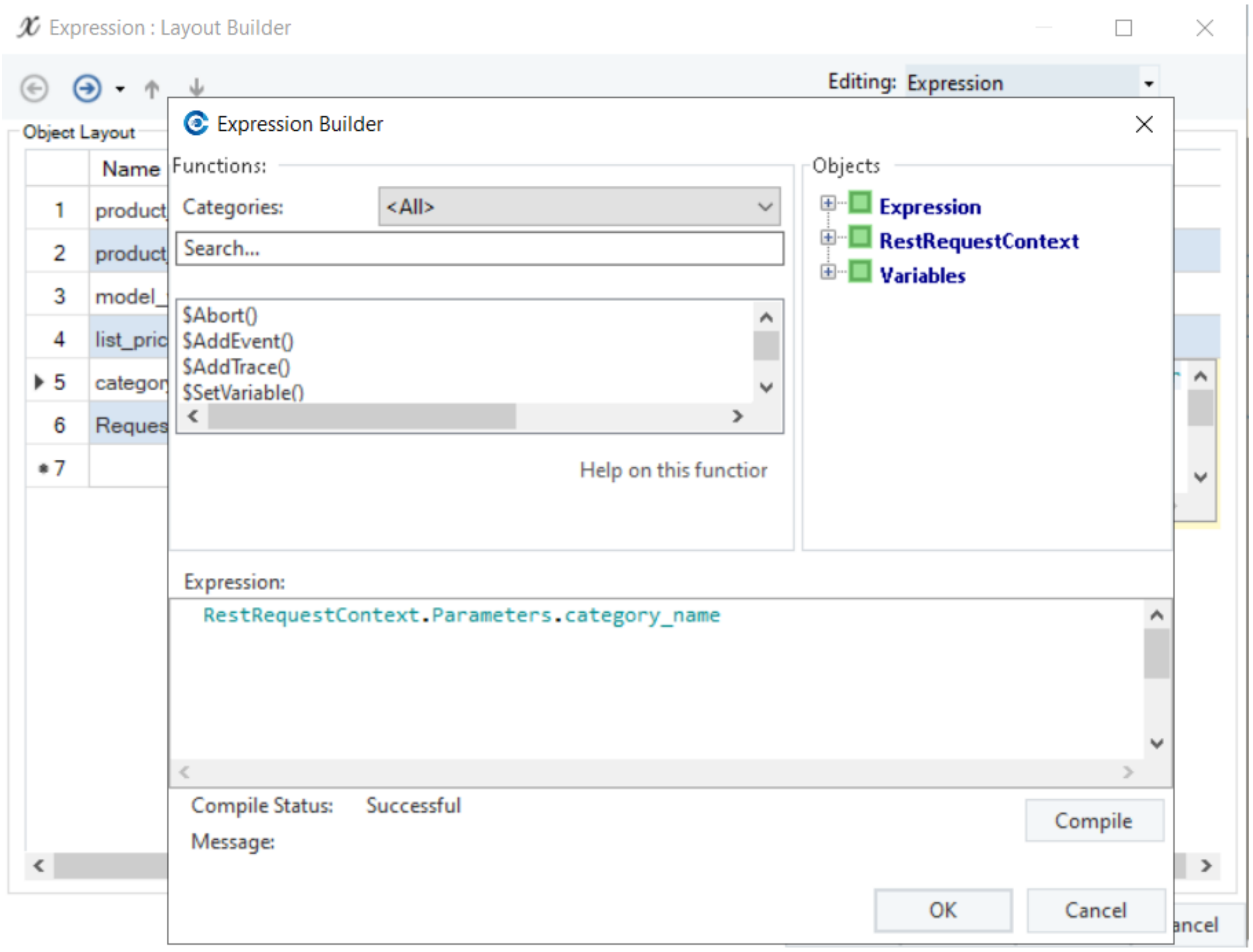
These Context Parameters can also be used in other objects as variables. As another example, we have used them to define expressions.

2. We will open the *Properties* of the *Expression* object.



This will open the configuration window for the object.

3. Click on a field and open the *Expression Builder*. Here, all request parameters are available and can be referenced to be used in expression functions.



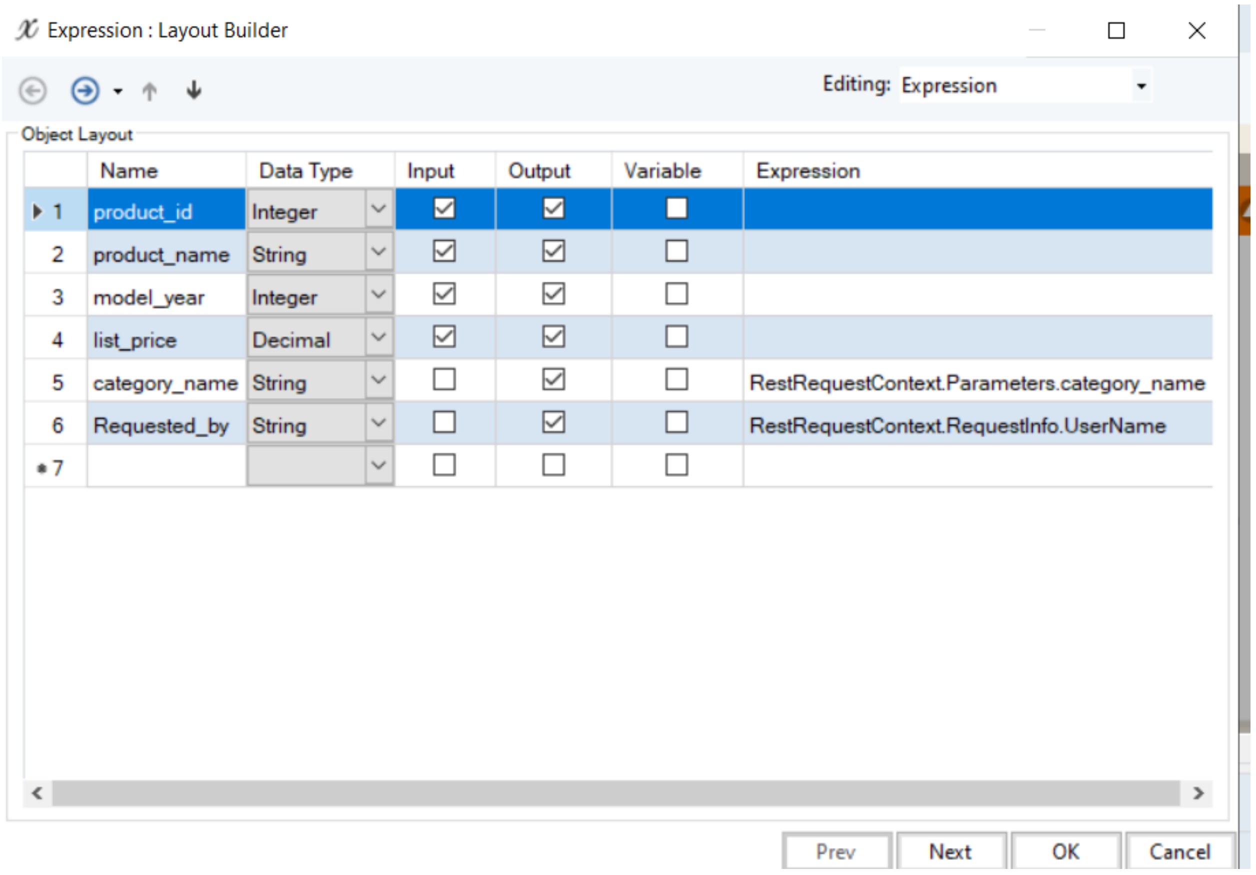
Expression:

```
RestRequestContext.Parameters.category_name
```

Compile Status: Successful

Message:

Here, you can see that we have defined Request Context Parameters using the values we had in the *REST Request* object.



	Name	Data Type	Input	Output	Variable	Expression
1	product_id	Integer	✓	✓	☐	
2	product_name	String	✓	✓	☐	
3	model_year	Integer	✓	✓	☐	
4	list_price	Decimal	✓	✓	☐	
5	category_name	String	☐	✓	☐	RestRequestContext.Parameters.category_name
6	Requested_by	String	☐	✓	☐	RestRequestContext.RequestInfo.UserName
7			☐	☐	☐	

These parameters can be used anywhere ahead in the entire API flow.

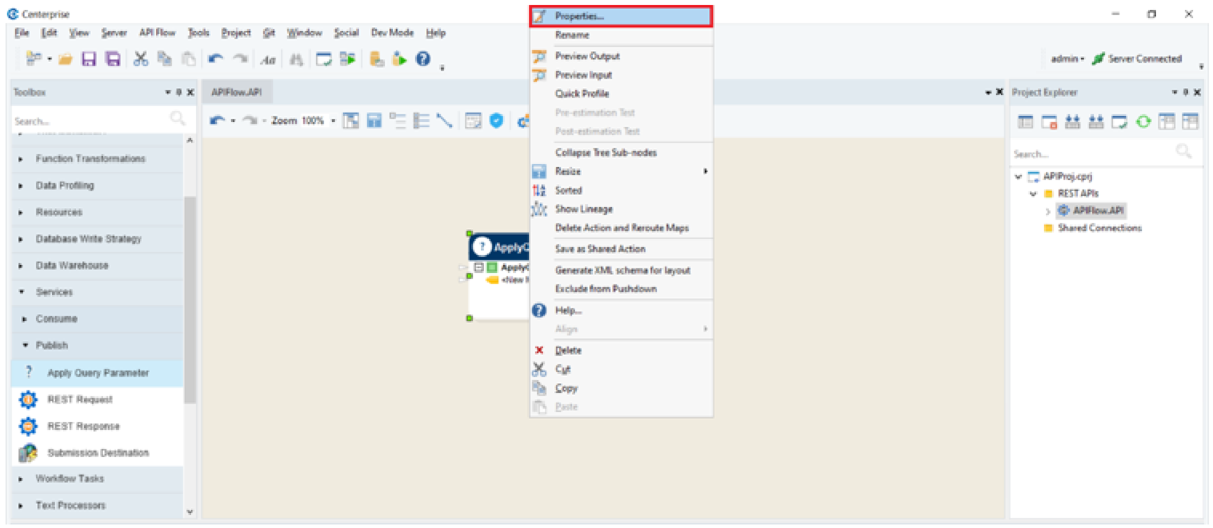
This concludes the working of the Request Context Parameters.

10.3 Configuring Sorting and Filtering in API Flows

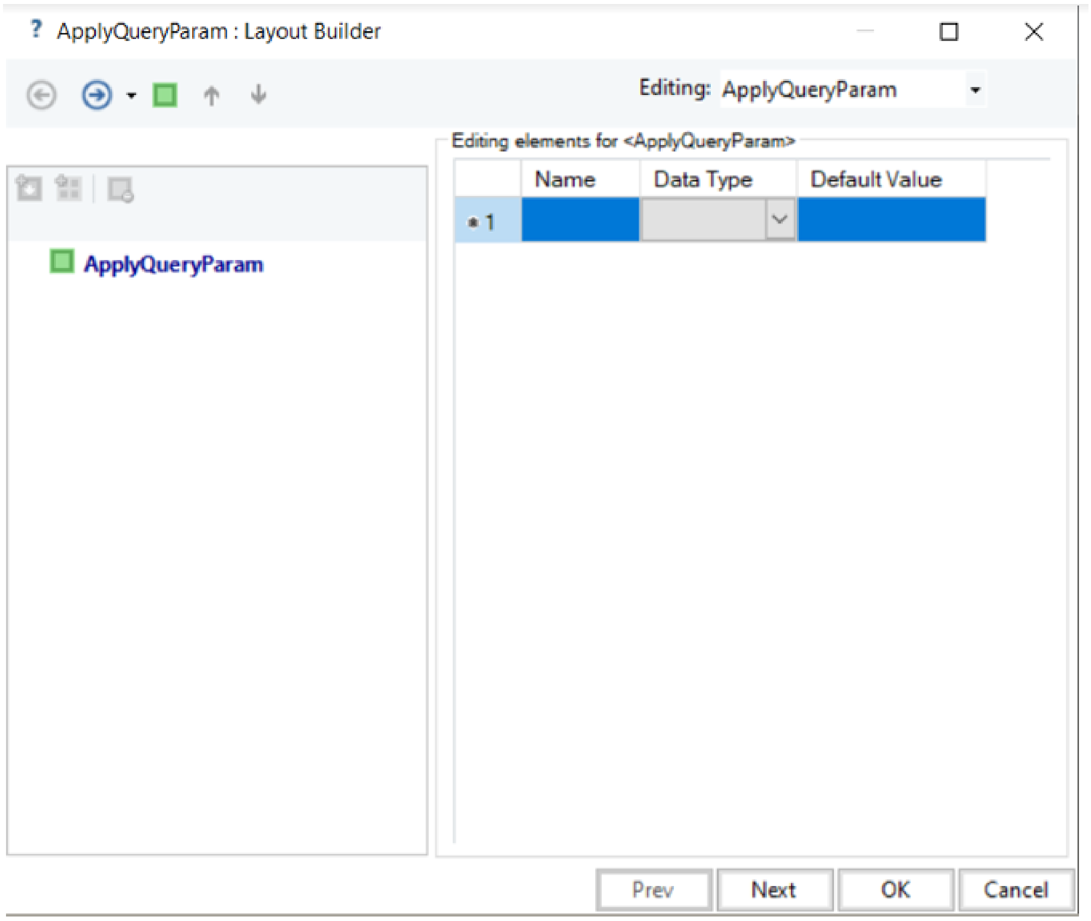
The *Apply Query Parameter* object is used to filter and sort data in an API flow in accordance with the user application. Its location in an API flow can depend on when sorting or filtering is required in the processing of the API request.

10.3.1 Configuring the Apply Query Parameter object

1. To start, right-click on the object and select *Properties* from the context menu.



The layout builder window will open.


ApplyQueryParam : Layout Builder
Editing: ApplyQueryParam
Editing elements for <ApplyQueryParam>
ApplyQueryParam
Prev
Next
OK
Cancel

	Name	Data Type	Default Value
1			

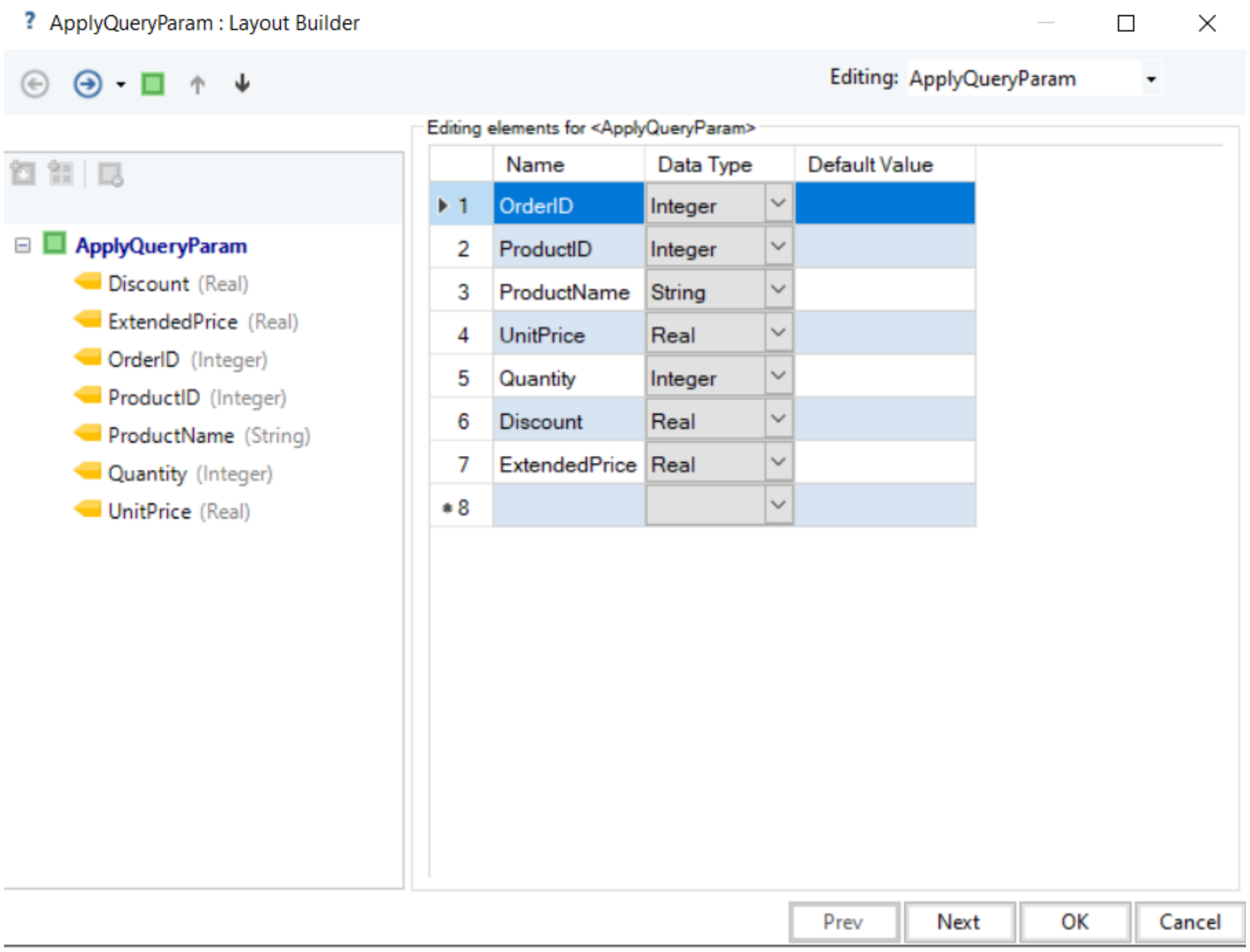
This is where the layout of the incoming data is going to be mapped. It can be automatically mapped from a preceding object.

Note: The left-side window shows the node hierarchy of the *Apply Query Parameter* object.

For our use case, we have a flow that fetches order items from the database for the given OrderID in the request. Now, to allow sorting and filter operations on this response data, we will add the *Apply Query Parameter* object right after the Database Lookup object and before the *REST Response* object.

The position of this object in a flow can depend on where it is required. It can be placed anywhere between objects in the API flow.

Thus, our layout builder is populated according to our flow.



ApplyQueryParam : Layout Builder

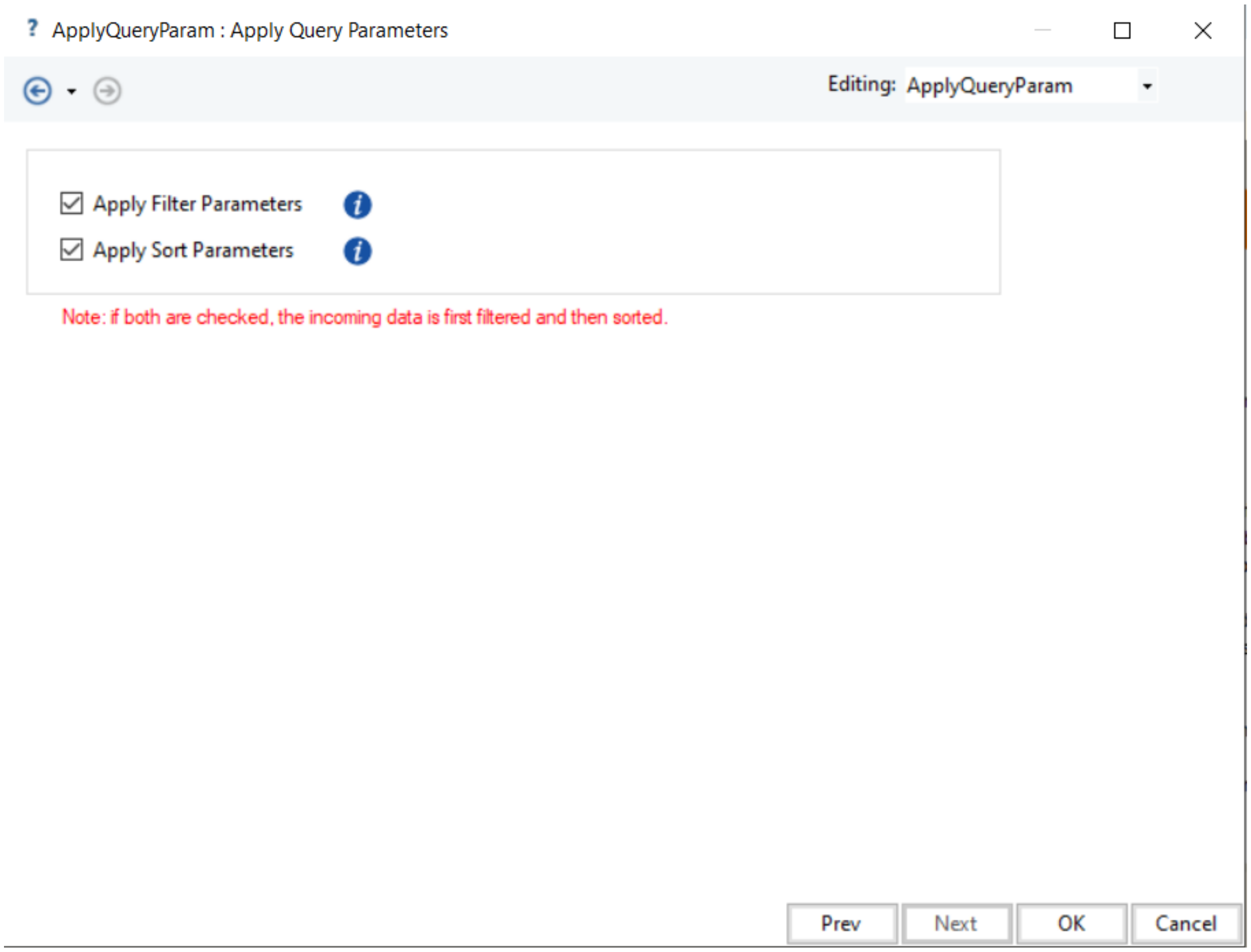
Editing: ApplyQueryParam

ApplyQueryParam
- Discount (Real)
- ExtendedPrice (Real)
- OrderID (Integer)
- ProductID (Integer)
- ProductName (String)
- Quantity (Integer)
- UnitPrice (Real)

Editing elements for <ApplyQueryParam>

	Name	Data Type	Default Value
1	OrderID	Integer	
2	ProductID	Integer	
3	ProductName	String	
4	UnitPrice	Real	
5	Quantity	Integer	
6	Discount	Real	
7	ExtendedPrice	Real	
8			

2. Once done, click *Next*. Here, you can enable filter and sort functions on the API response data.



[✓] Apply Filter Parameters

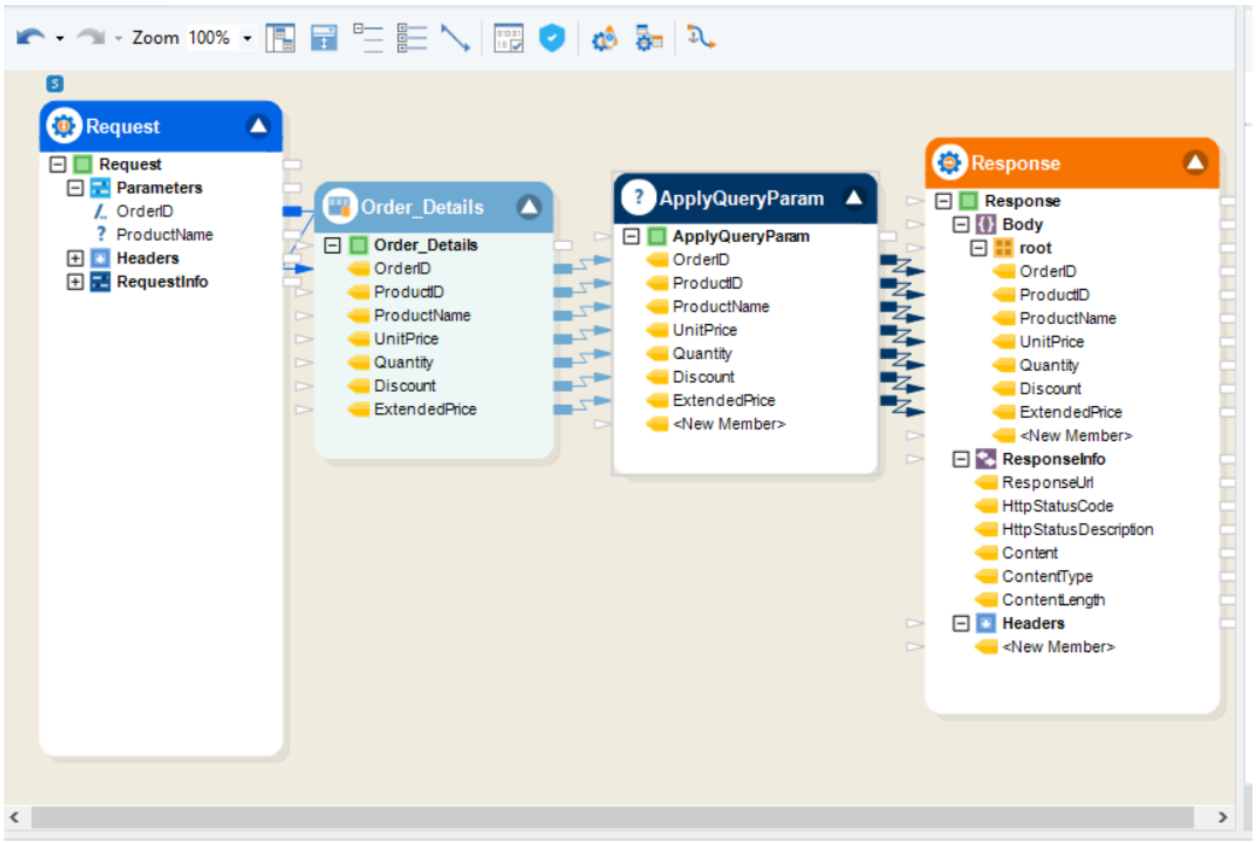
[✓] Apply Sort Parameters

Note: if both are checked, the incoming data is first filtered and then sorted.

Apply Filter Parameters: This allows users to send filter parameters of response layout fields in the request URL.

Apply Sort Parameters: This allows users to send sort_by parameter of response layout fields in the request URL.

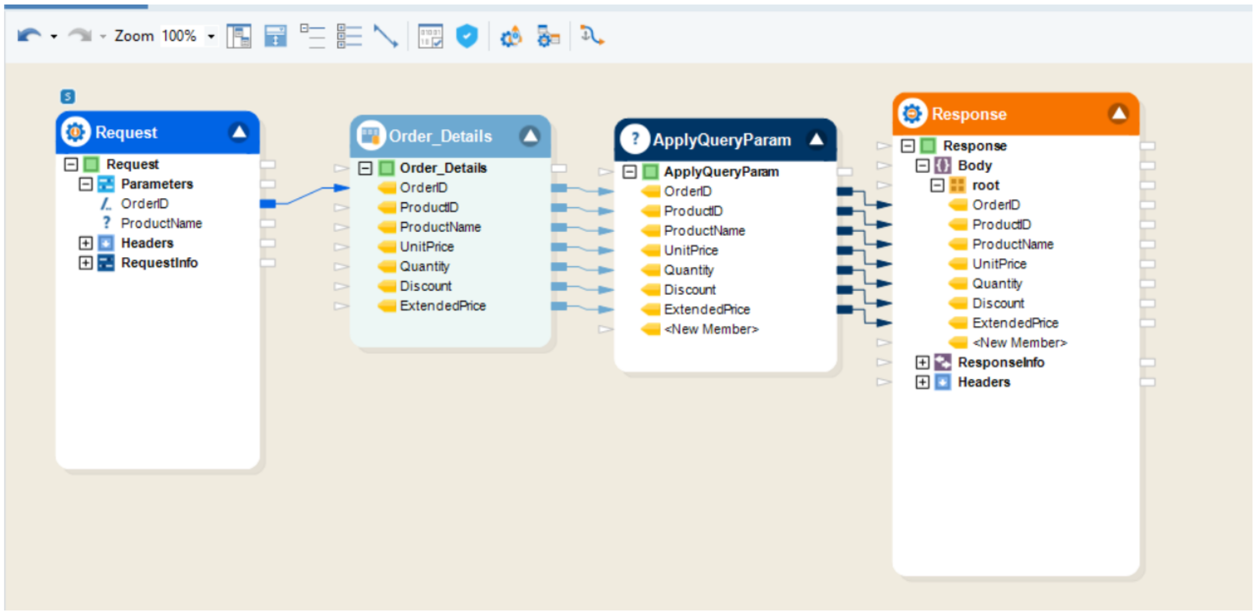
3. Select *Ok* after you are done with this window and the configuration will be completed.



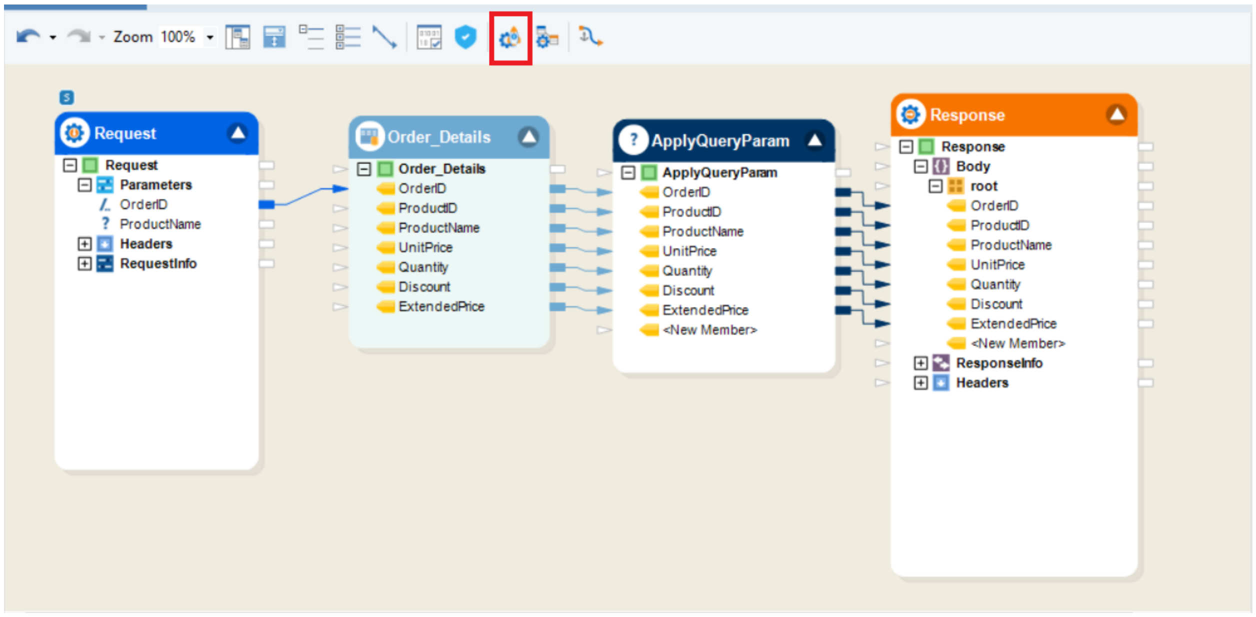
The *Apply Query Parameter* object has been configured to return responses as per the sort or filter parameters requested.

10.3.2 Applying Filter and Sort Parameters in Request

1. To further examine filter and sort functionalities, deploy the flow you have used the *Apply Query Parameter* object in.



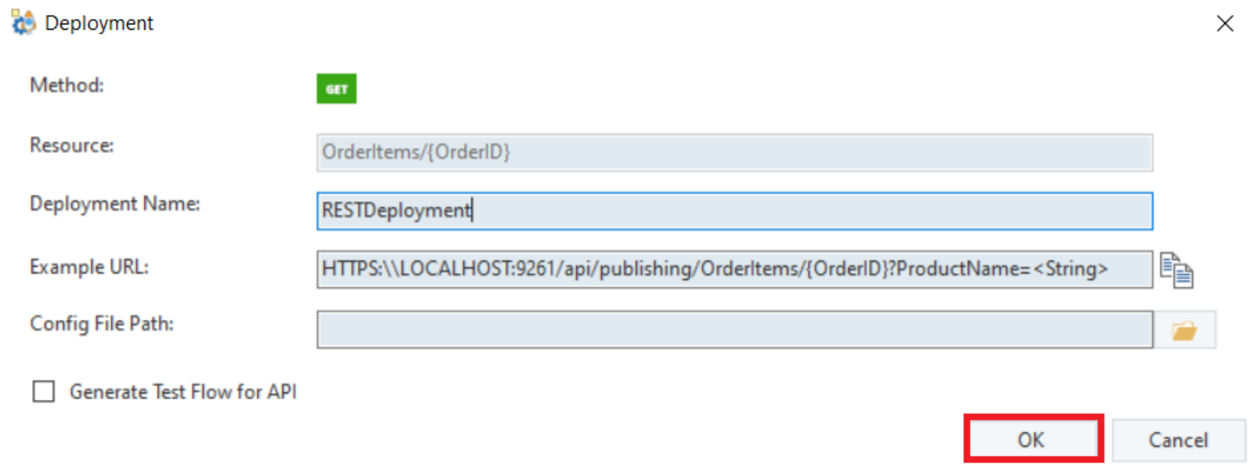
2. To deploy the API flow, select the third option from the right, on the designer toolbar



This will open a new screen,

3. Provide the *Deployment Name* and the *Config File Path*, if any, and click *Ok*.

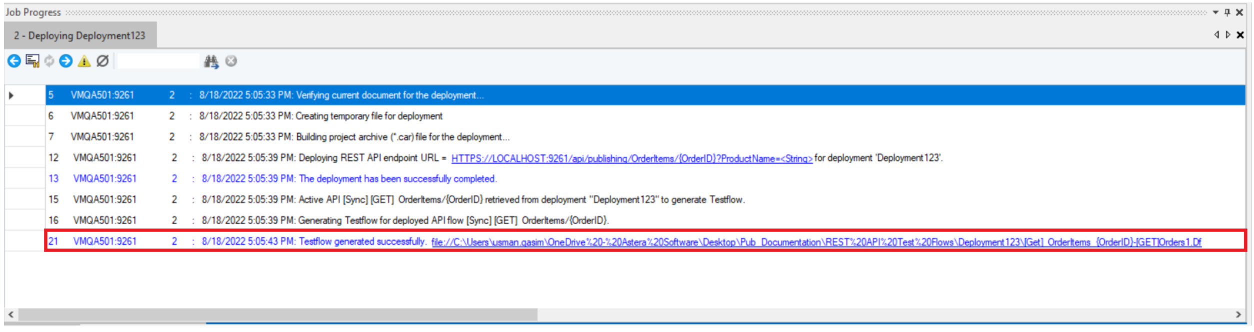
Generate Test Flow for API: Selecting this option will generate a flow to execute a test request for the API in run-time, to use after deployment.



Method:	GET
Resource:	OrderItems/{OrderID}
Deployment Name:	RESTDeployment
Example URL:	HTTPS:\\LOCALHOST:9261/api/publishing/OrderItems/{OrderID}?ProductName=<String>
Config File Path:	

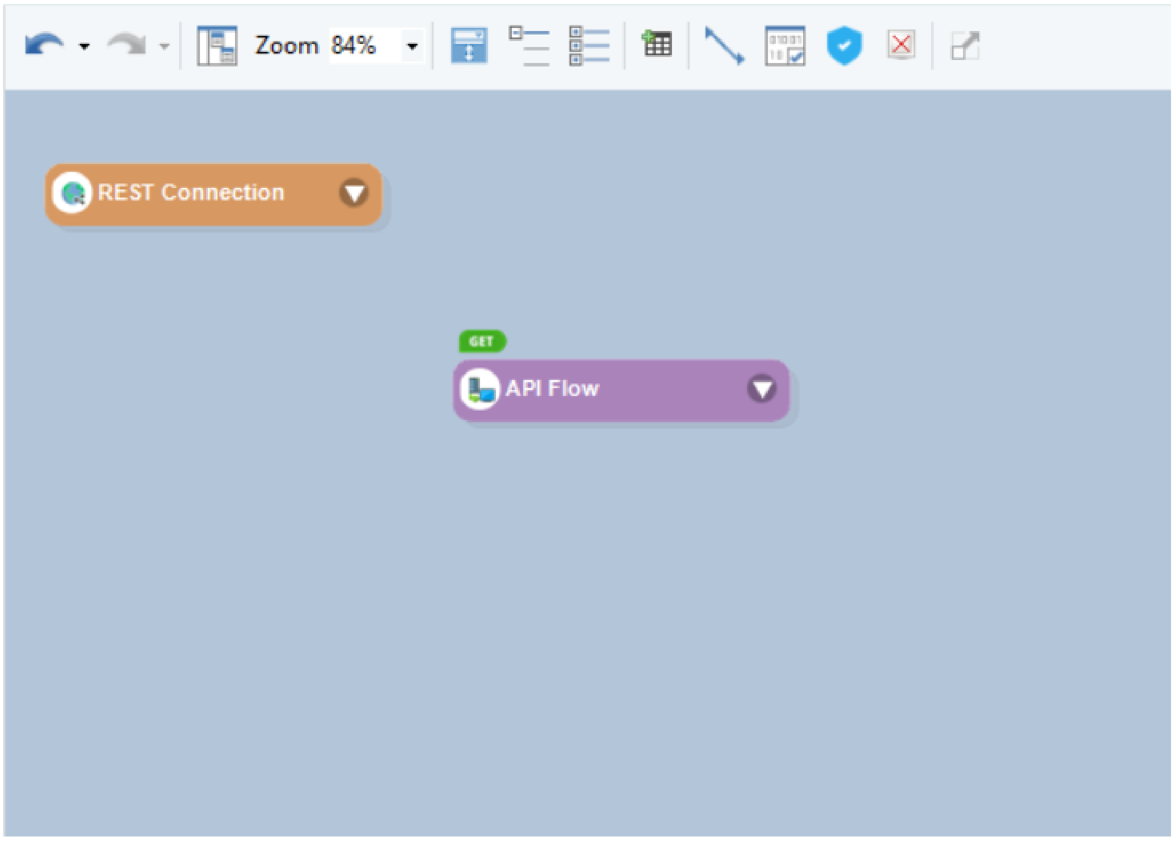
☐ Generate Test Flow for API

4. Once done, you can open the generated test flow from the job progress window.

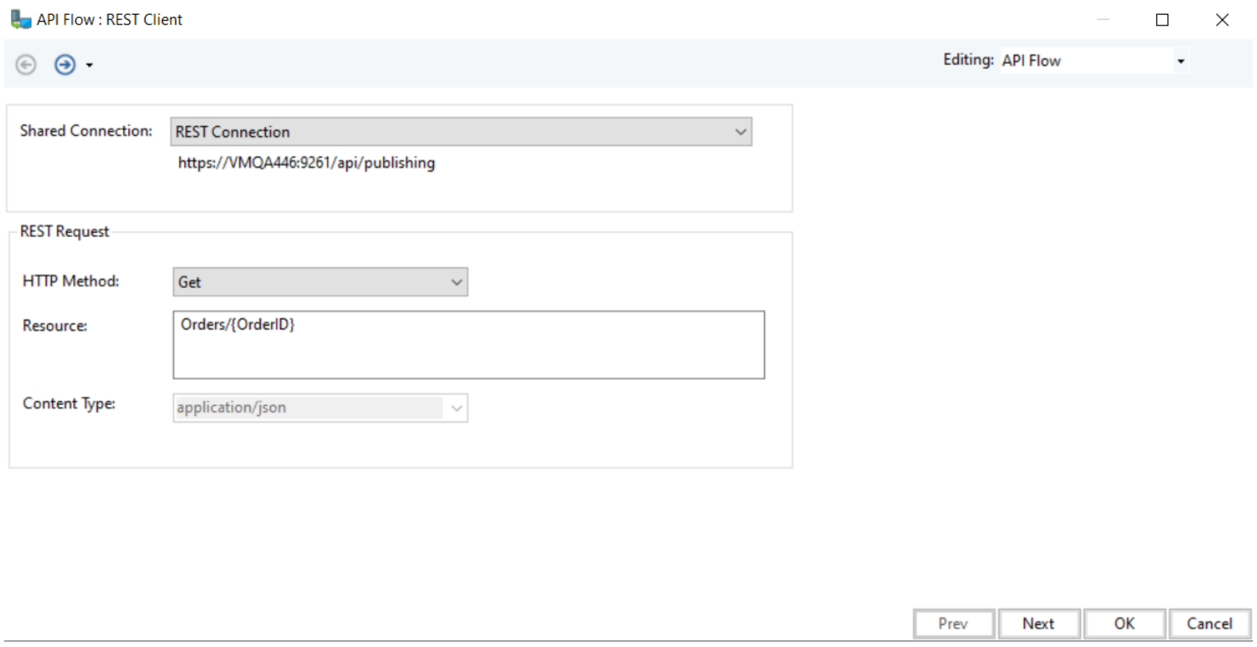


```
5   VMQA501:9261   2  :  8/18/2022 5:05:33 PM: Verifying current document for the deployment...
6   VMQA501:9261   2  :  8/18/2022 5:05:33 PM: Creating temporary file for deployment
7   VMQA501:9261   2  :  8/18/2022 5:05:33 PM: Building project archive (*.car) file for the deployment...
12  VMQA501:9261   2  :  8/18/2022 5:05:39 PM: Deploying REST API endpoint URL = HTTPS://LOCALHOST:9261/api/publishing/OrderItems/{OrderID}?ProductName=<String> for deployment 'Deployment123'.
13  VMQA501:9261   2  :  8/18/2022 5:05:39 PM: The deployment has been successfully completed.
15  VMQA501:9261   2  :  8/18/2022 5:05:39 PM: Active API [Sync] [GET]  OrderItems/{OrderID} retrieved from deployment "Deployment123" to generate Testflow.
16  VMQA501:9261   2  :  8/18/2022 5:05:39 PM: Generating Testflow for deployed API flow [Sync] [GET]  OrderItems/{OrderID}.
21  VMQA501:9261   2  :  8/18/2022 5:05:43 PM: Testflow generated successfully.  file://C:\Users\usman.qasim\OneDrive%20-%20Astera%20Software\Desktop\Pub_Documentation\REST%20API%20Test%20Flows\Deployment123\[Get]_OrderItems_{OrderID}-[GET]Orders1.Df
```

Note: Since the test flow was generated during the deployment, it already has a *REST Connection,* and *REST* Client auto-populated with the request configuration.



5. Right-click on the *REST Client* object and select *Properties* from the context menu.



Shared Connection: REST Connection
https://VMQA446:9261/api/publishing

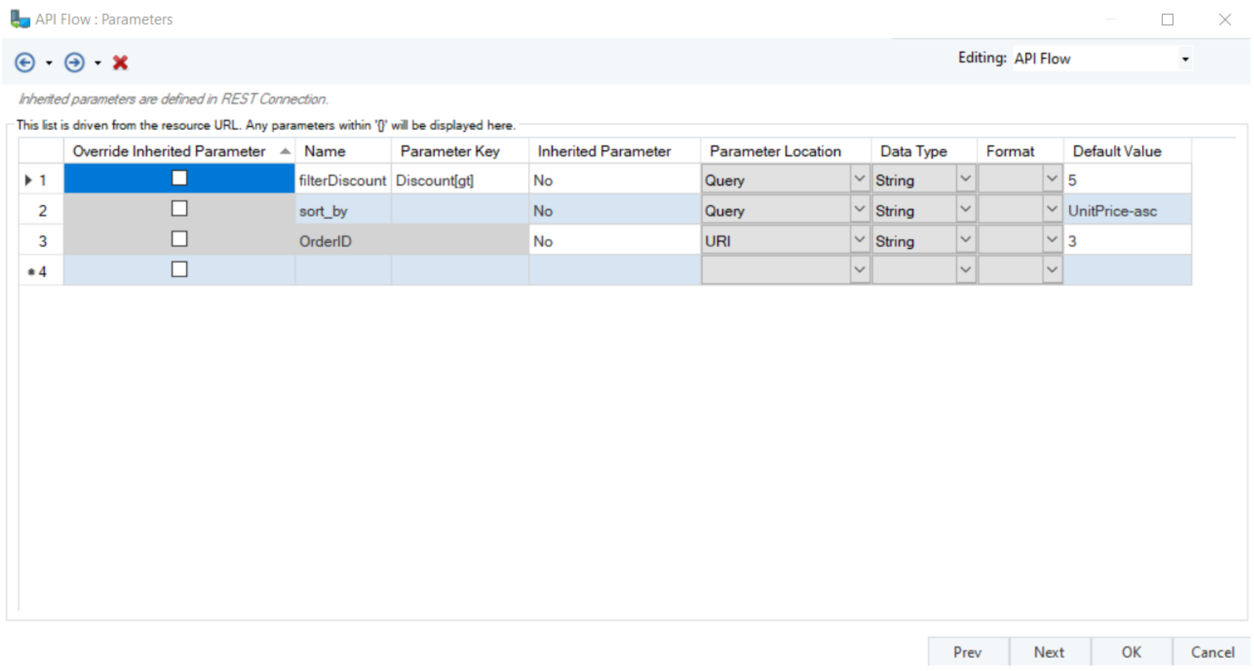
REST Request

HTTP Method: Get

Resource: Orders/{OrderID}

Content Type: application/json

6. Click *Next* and you will be led to the *Parameters* screen.



Inherited parameters are defined in REST Connection.

This list is driven from the resource URL. Any parameters within '{}' will be displayed here.

	Override Inherited Parameter	Name	Parameter Key	Inherited Parameter	Parameter Location	Data Type	Format	Default Value
1	☐	filterDiscount	Discount[gt]	No	Query	String		5
2	☐	sort_by		No	Query	String		UnitPrice-asc
3	☐	OrderID		No	URI	String		3
4	☐							

Here, you can use the *sort_by* parameter or the filter parameters, with the appropriate syntax, to obtain accurate data.

To sort the response data, we can define the query parameter ‘sort_by’. This parameter takes comma-separated values for multiple fields that need to be sorted. The syntax for each sort is as follows:

FieldName - SortOrder - Where the sort order can either be *asc* for ascending or *desc* for descending.

For example, a sort value as UnitPrice-asc, ProductName-desc would first sort the data by Unit Price in ascending order, then apply a second sort by Product Name in descending order.

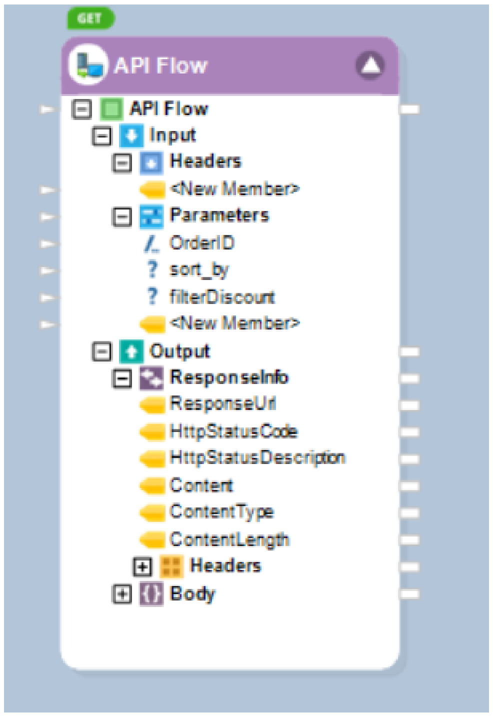
FieldName [Operator] - To add a filter on any of the response fields, you can define a query parameter with this syntax. The supported operators include,

- Equals to – eq
- Not equal to – neq
- Greater than – gt
- Greater than or equals to – gte
- Less than – lt
- Less than or equals to - lte

For example, to apply a filter on discount, we have defined a query parameter *Discount[gt]* as the parameter key and *5* as the value for this filter, implying to only show discount records greater than 5 in the response data. Additionally, more such filter parameters can also be added for other response fields.

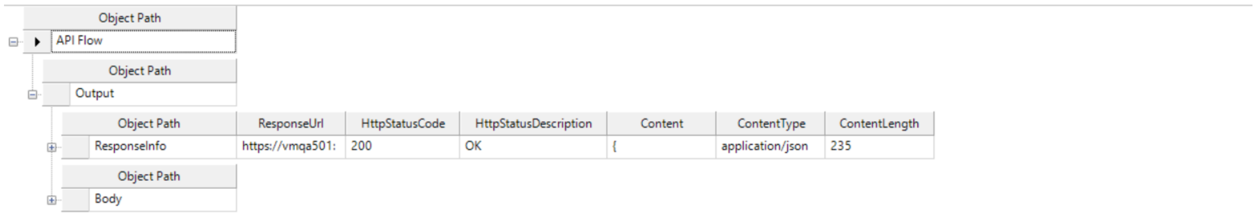
Note: This parameter name is defined as a Parameter Key because the name consists of special characters [] which are not allowed in the Name column. Parameter Key is used instead of Name in the actual API request.

7. Click *Ok*.



The parameters have been added.

8. Next, right-click on the *REST Client* object and select *Preview Output* to make an API call with the defined parameters.



Object Path
API Flow

Object Path
Output

Object Path	ResponseUrl	HttpStatusCode	HttpStatusDescription	Content	ContentType	ContentLength
ResponseInfo	https://vmqa501:	200	OK	{	application/json	235

Object Path
Body

The status code '200' shows that the API call was made successfully.

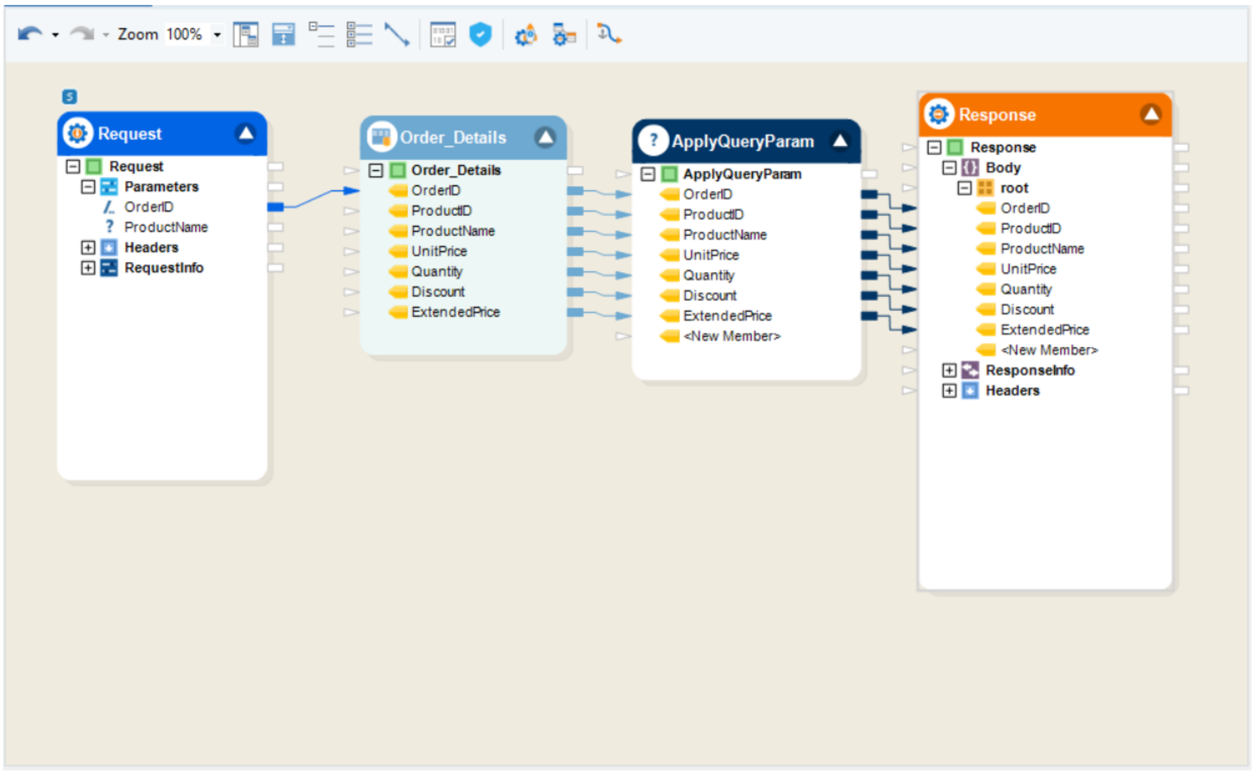
This concludes the usage of the *Apply Query Parameter* object after deployment, within an test flow.

10.4 Enable Pagination

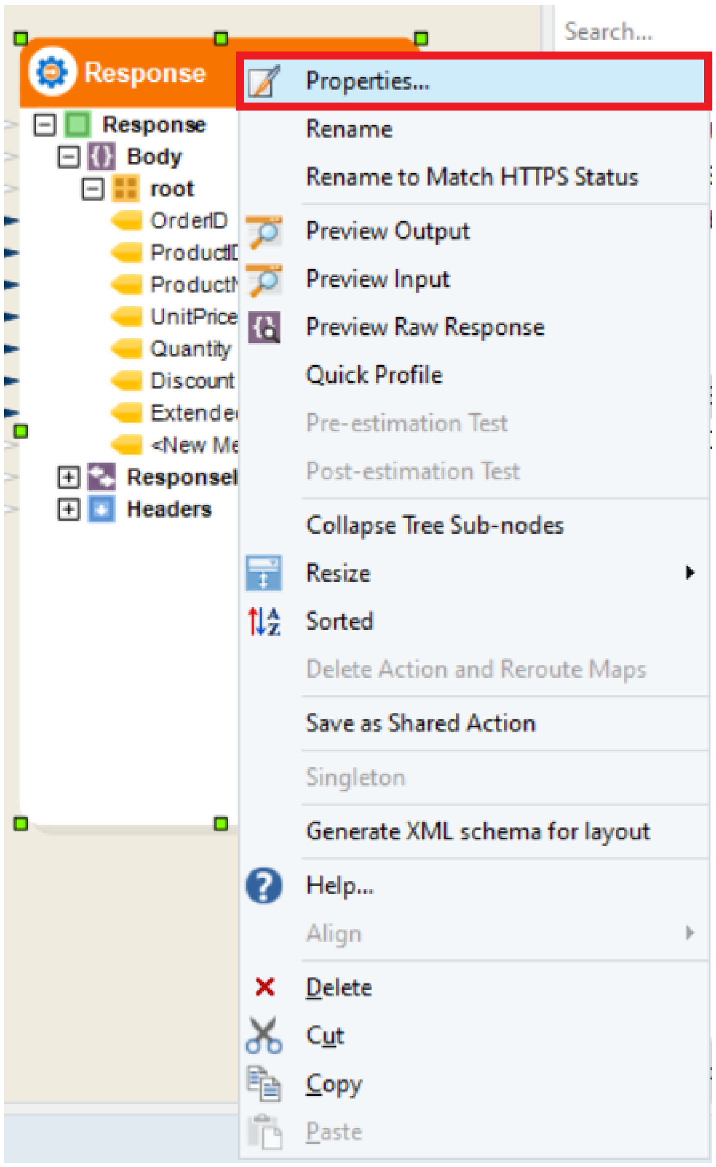
You can configure an API flow to paginate response data from the *REST Response* object.

Pagination is a process that is used to divide large data into smaller discrete pages, thus allowing for less clutter and better readability. It also means that server request processing will be faster as a small subset is to be returned. Hence, improving the overall API performance and readability.

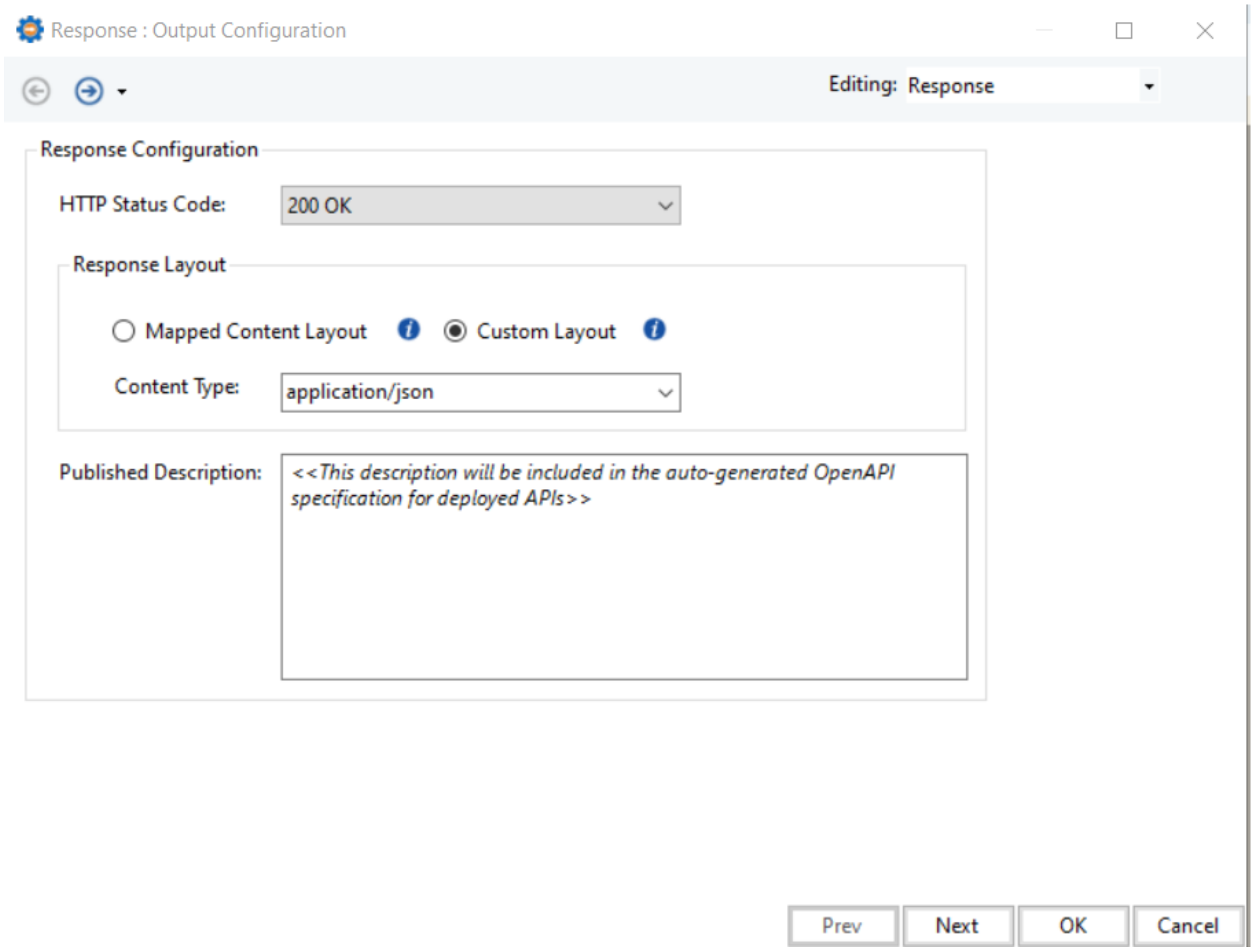
For our use case, we have an API flow that is configured to retrieve a collection of order items using a database lookup along with filter and sort functionalities enabled. Now, let us see how pagination can be configured in this flow.



1. Right-click on the *REST Response* object and select *Properties* from the context menu.



This will open the *Properties* window.


Response : Output Configuration

Editing: Response

Response Configuration

HTTP Status Code: 200 OK

Response Layout

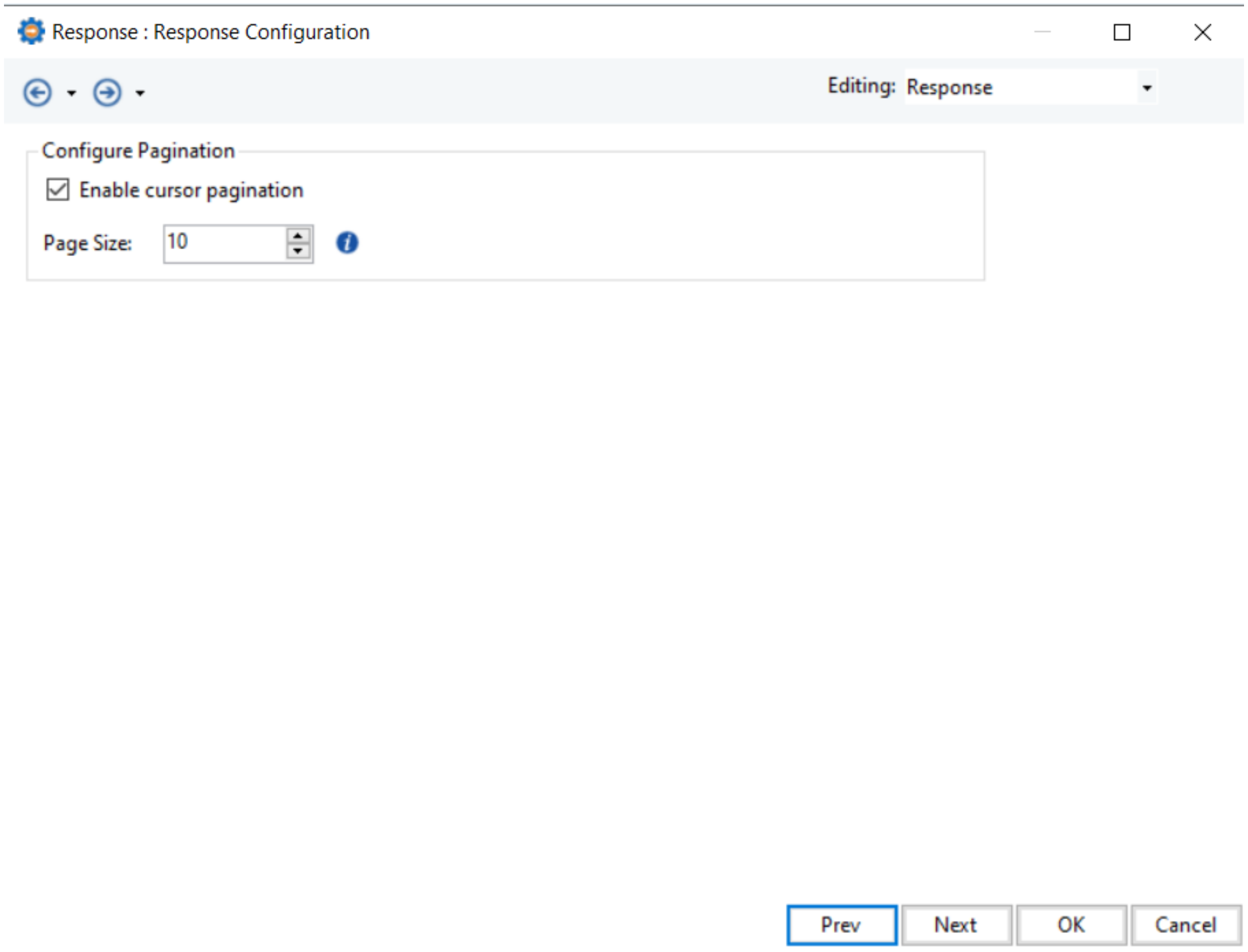
( ) Mapped Content Layout [✓ Custom Layout]

Content Type: application/json

Published Description: *<<This description will be included in the auto-generated OpenAPI specification for deployed APIs>>*

2. Click *Next* until you reach the *Response Configuration* window.

This is where you can set up pagination options to better structure the response data fetched for a successful 200-OK request.


Response : Response Configuration

Editing: Response

Configure Pagination

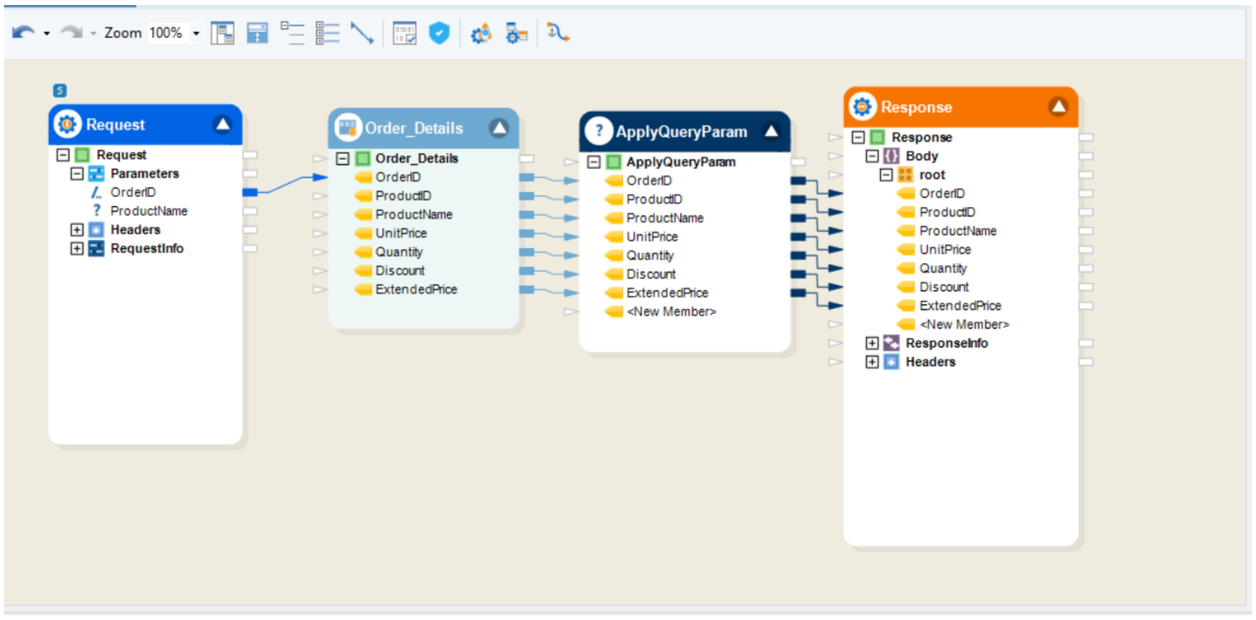
☑ Enable cursor pagination

Page Size: 10

Enable Cursor Pagination: Checking this box lets the incoming data be paginated by a cursor. Cursor-based pagination works by returning a pointer to the last item in the dataset page, using which the client can make successive requests to read the next set of records iteratively. Once all records have been read, the cursor value becomes null, thus indicating that no more records are left to be read.

Page Size: This counter determines the number of records returned in a single requested page.

3. Click *OK* and the *REST Response* object will be configured in accordance with the cursor pagination and page size.



Note: Any request made to this endpoint will return the first n (page size) records along with a ‘cursor’ field in the response payload. This cursor field can then be reiterated in the next request as a Query Parameter named ‘Cursor’ to fetch the next n (page size) records.

At runtime, these paginated calls are cached at the server. To learn more about it, click here.

10.5 Asynchronous API Request

In Astera API Management, we can process an API request either synchronously or asynchronously.

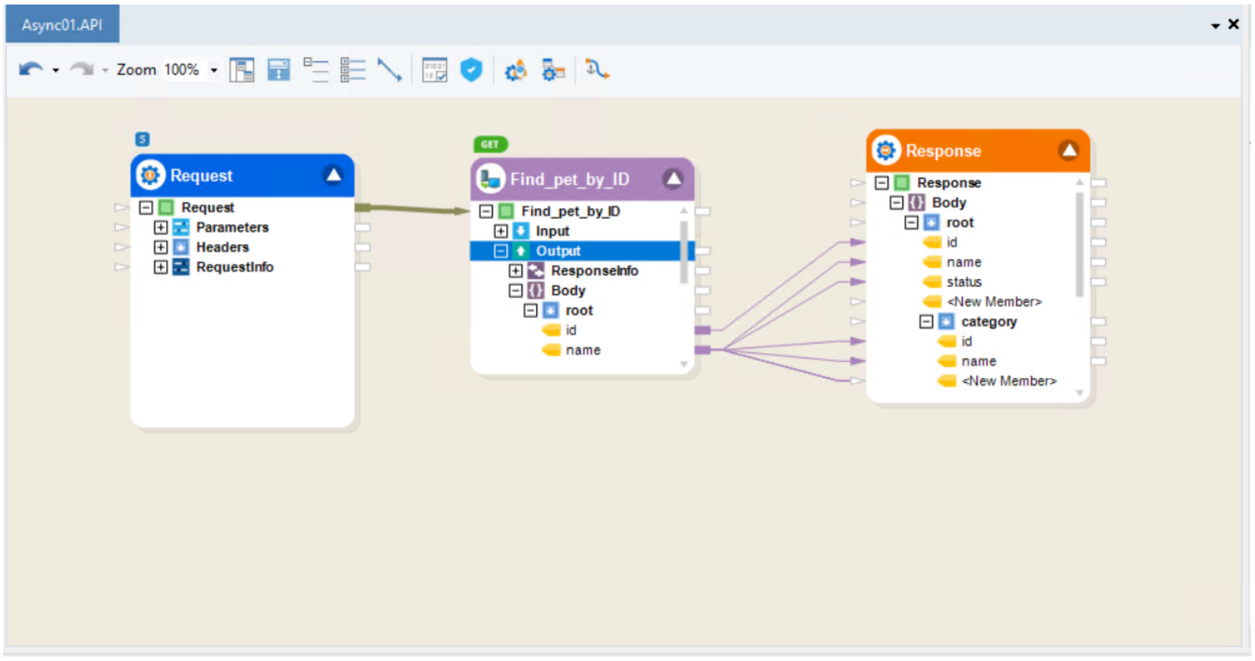
In Synchronous execution, the response to the API call does not return until the process has been completed or there has been an error.

In Asynchronous execution, the response to the API call is returned immediately with a polling URL while the request continues to be processed.

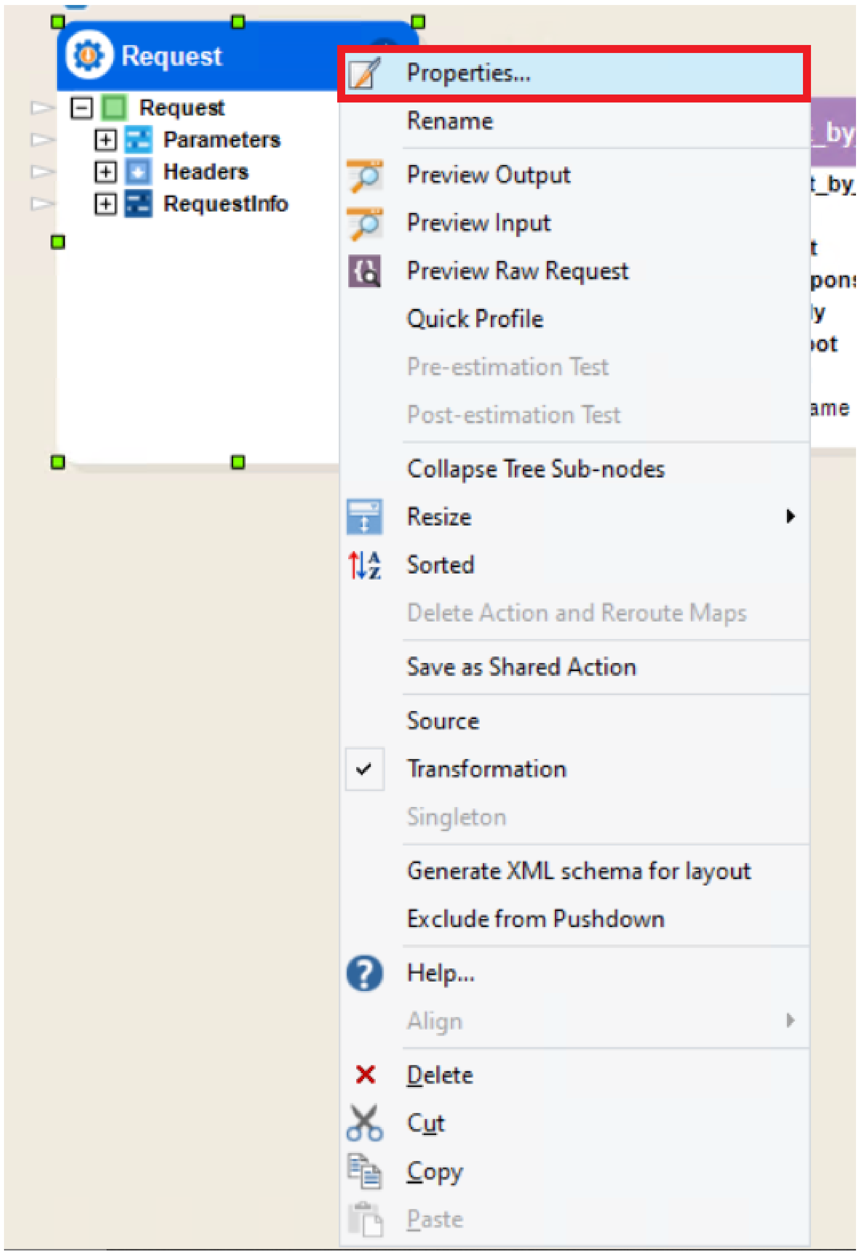
To check the functionality of such execution, we have created an API flow that will process the request Asynchronously.

10.5.1 Processing an API Request Asynchronously

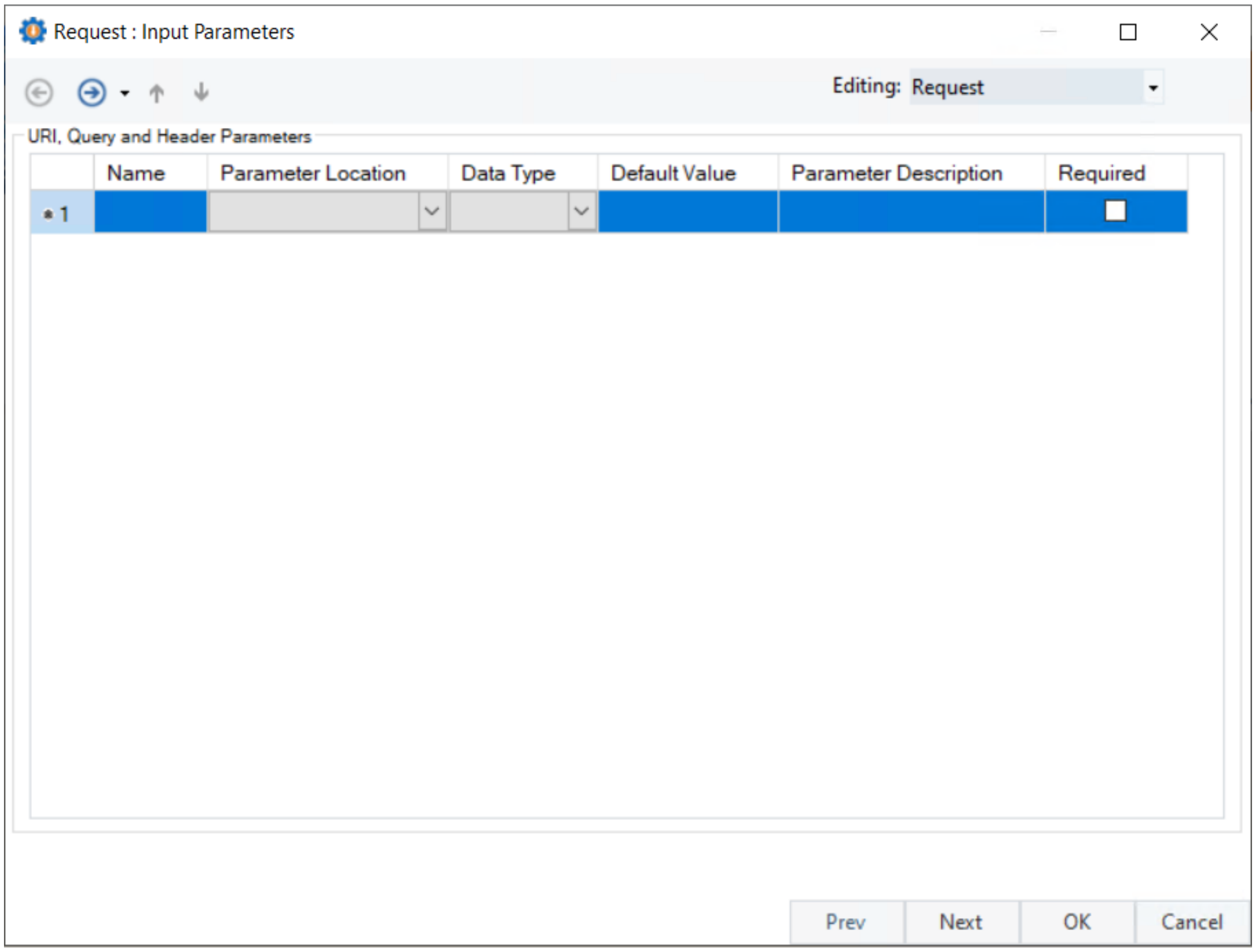
For our use case, we are deploying an API flow, which calls another 3rd party API and may take long to respond.



1. To enable Asynchronous execution, right-click on the *Request* object and select *Properties* from the context menu.



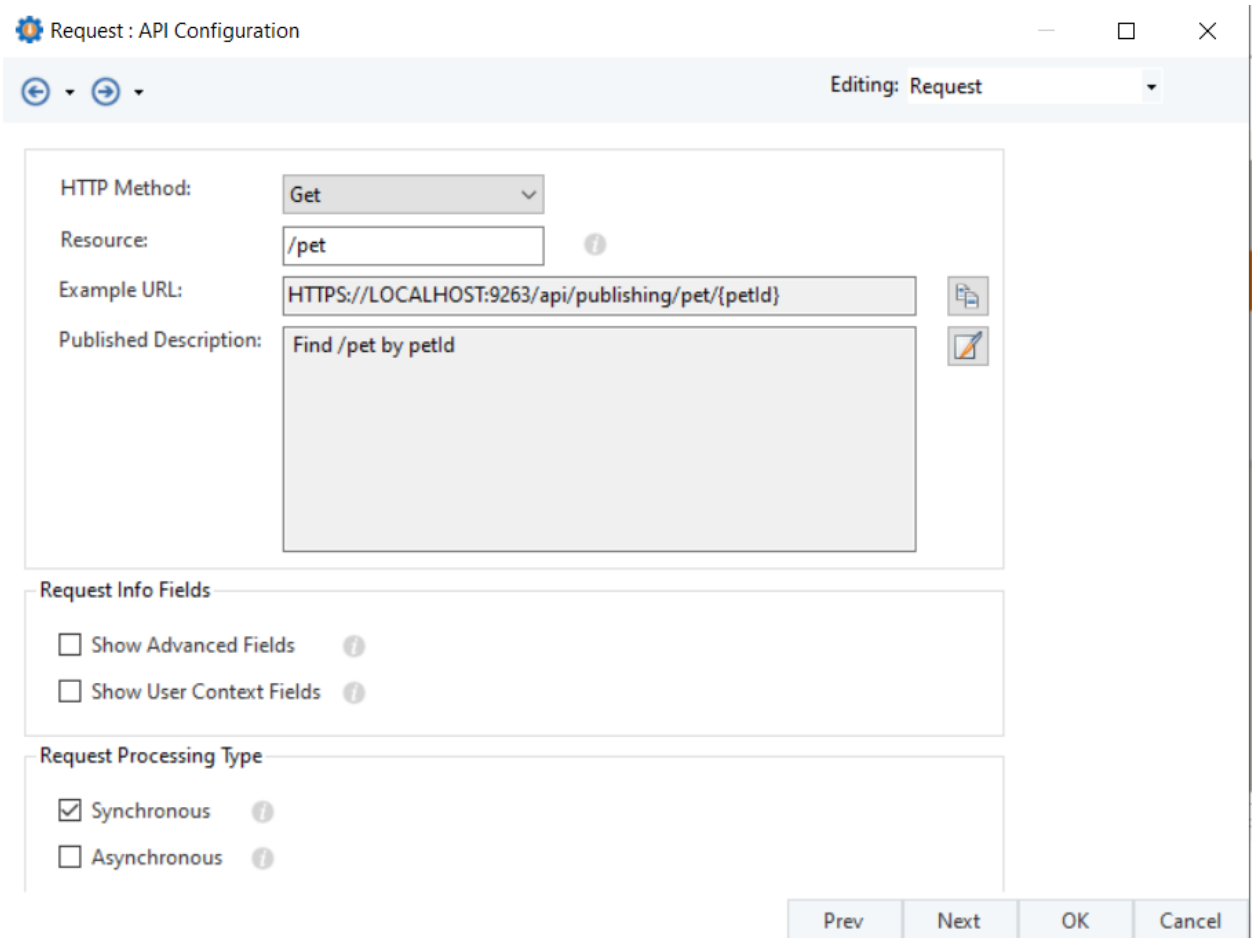
This will open the *Properties* window.



	Name	Parameter Location	Data Type	Default Value	Parameter Description	Required
1						☐

2. Click *Next* and you will be led to the *API Configuration* screen.

By default, the synchronous option has been selected,



HTTP Method: Get

Resource: /pet

Example URL: HTTPS://LOCALHOST:9263/api/publishing/pet/{petId}

Published Description: Find /pet by petId

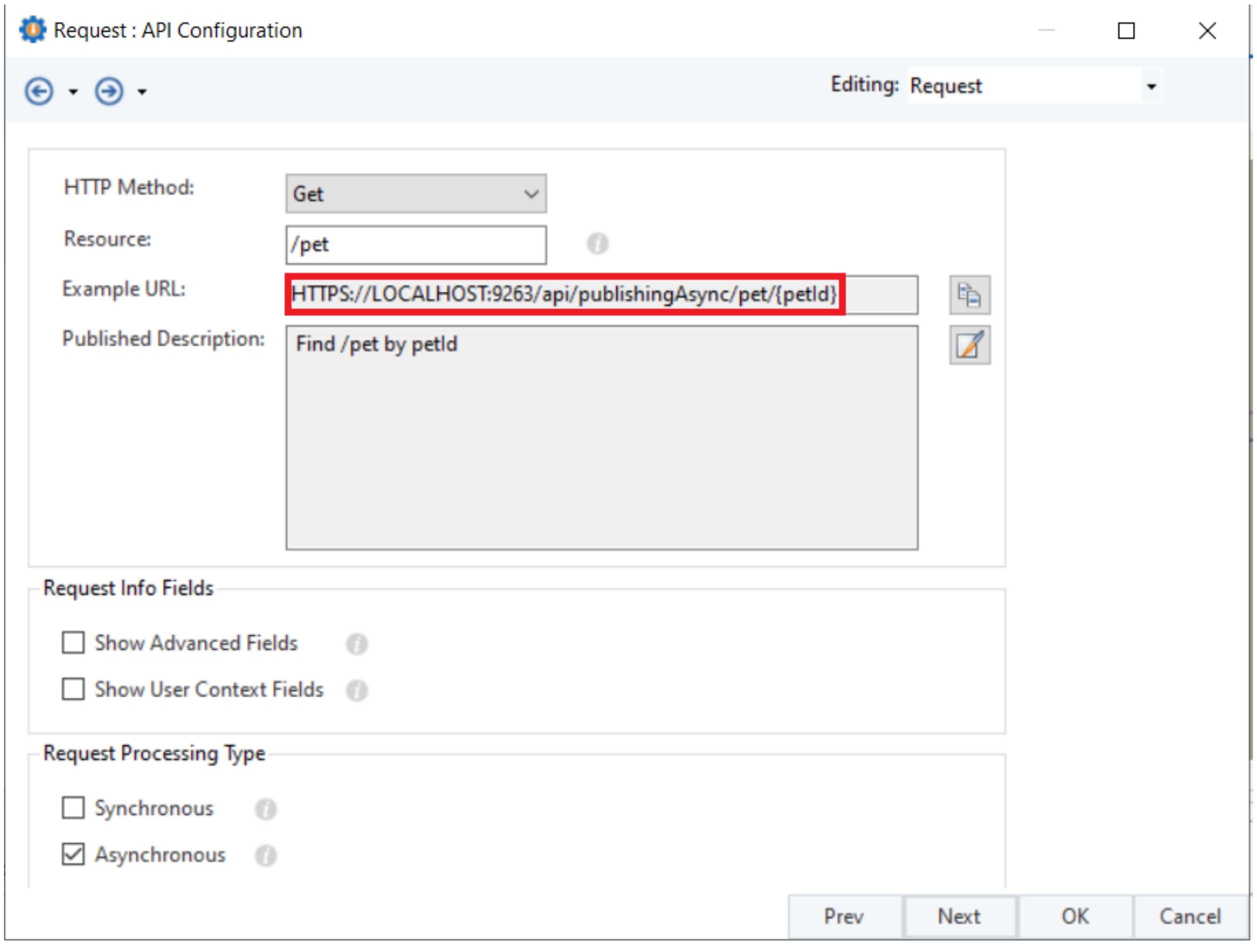
Request Info Fields

- [ ] Show Advanced Fields
- [ ] Show User Context Fields

Request Processing Type

- [x] Synchronous
- [ ] Asynchronous

After we select the *Asynchronous* checkbox, the API controller path in the *Example URL* also changes to ‘publishingAsync’.



HTTP Method: Get

Resource: /pet

Example URL: HTTPS://LOCALHOST:9263/api/publishingAsync/pet/{petId}

Published Description: Find /pet by petId

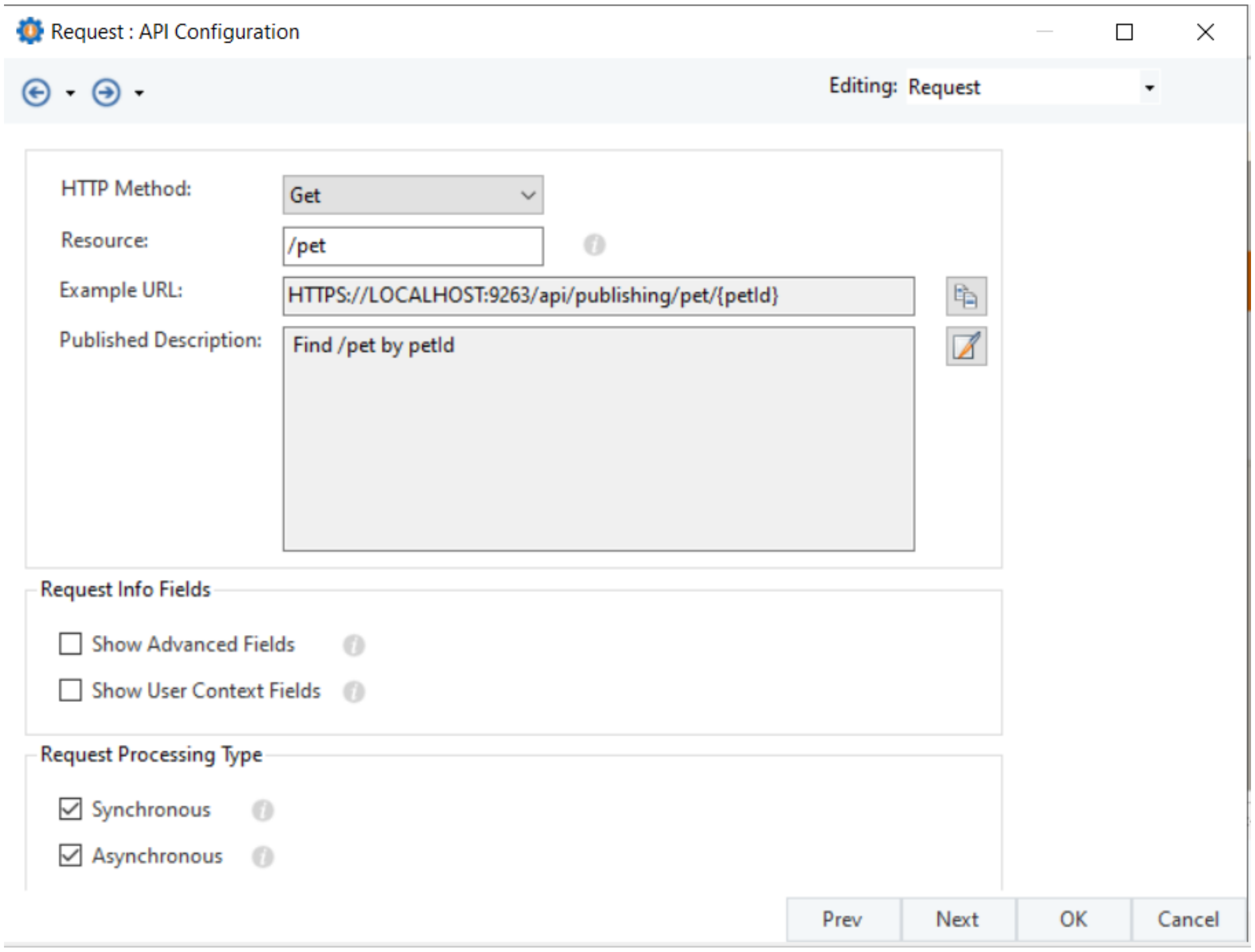
Request Info Fields

- [ ] Show Advanced Fields
- [ ] Show User Context Fields

Request Processing Type

- [ ] Synchronous
- [x] Asynchronous

If we plan to deploy both processing types, the *Example URL* just shows the Synchronous API example,



Request : API Configuration

Editing: Request

HTTP Method: Get

Resource: /pet

Example URL: HTTPS://LOCALHOST:9263/api/publishing/pet/{petId}

Published Description: Find /pet by petId

Request Info Fields

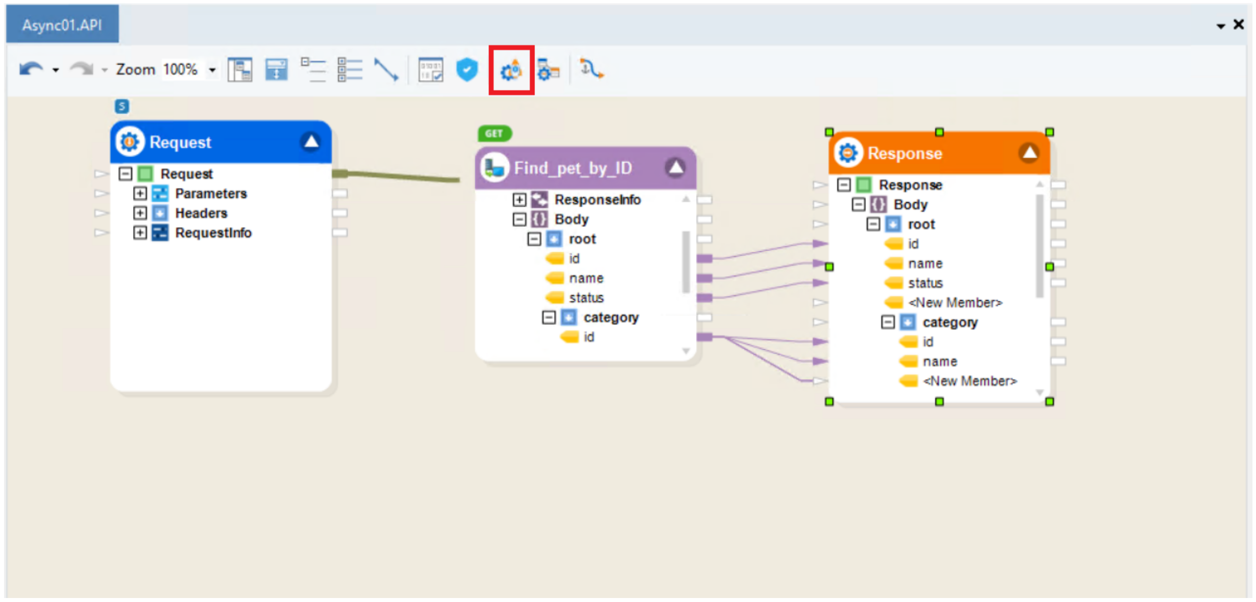
- [ ] Show Advanced Fields
- [ ] Show User Context Fields

Request Processing Type

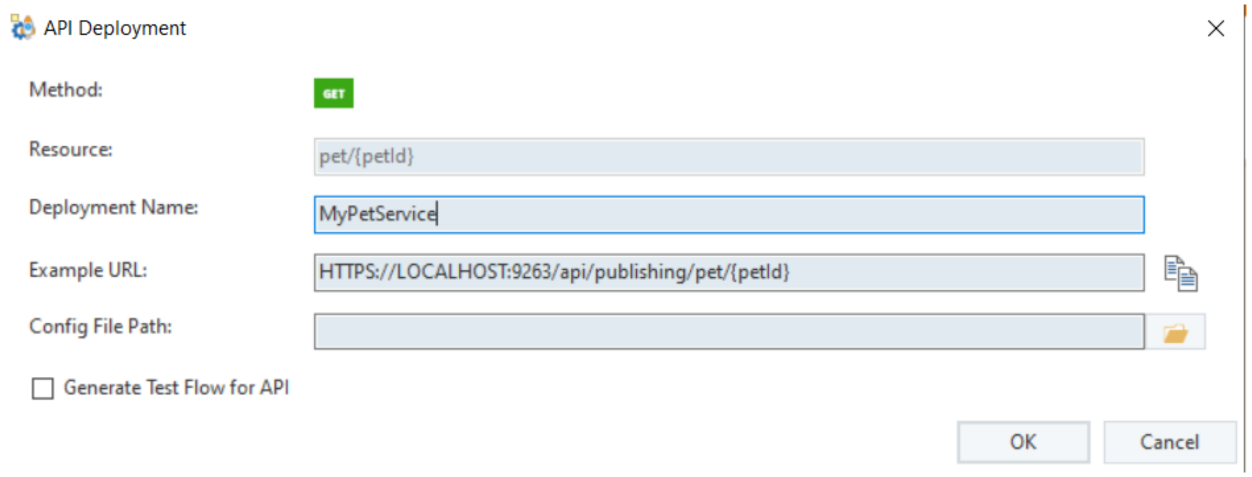
- [x] Synchronous
- [x] Asynchronous

Prev | Next | OK | Cancel

3. Click *OK* and deploy this flow through the option present in the API flow toolbar.



This will open a new window where the deployment name can be defined.



API Deployment

Method: GET

Resource: pet/{petId}

Deployment Name: MyPetService

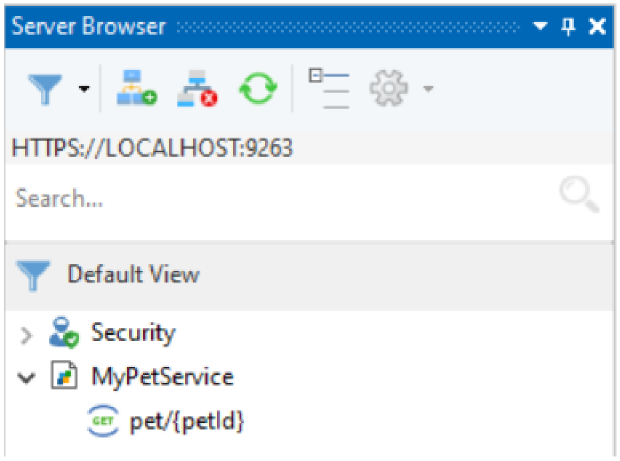
Example URL: HTTPS://LOCALHOST:9263/api/publishing/pet/{petId}

Config File Path:

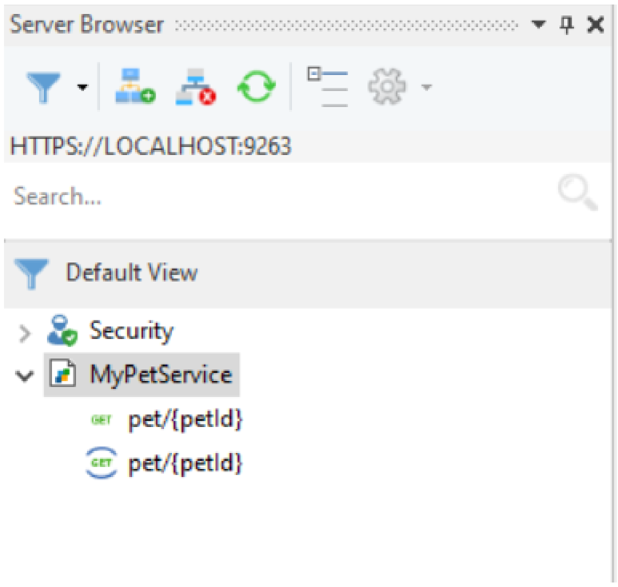
☐ Generate Test Flow for API

4. Once the deployment has been created, you can view it in the Server Browser.

As you can see, the GET endpoint icon here represents Asynchronous processing,

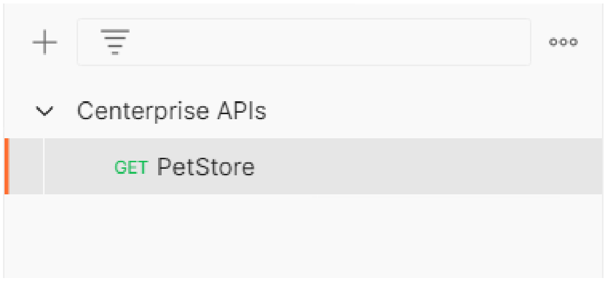


If we had selected both processing types, the endpoints would have looked like this,



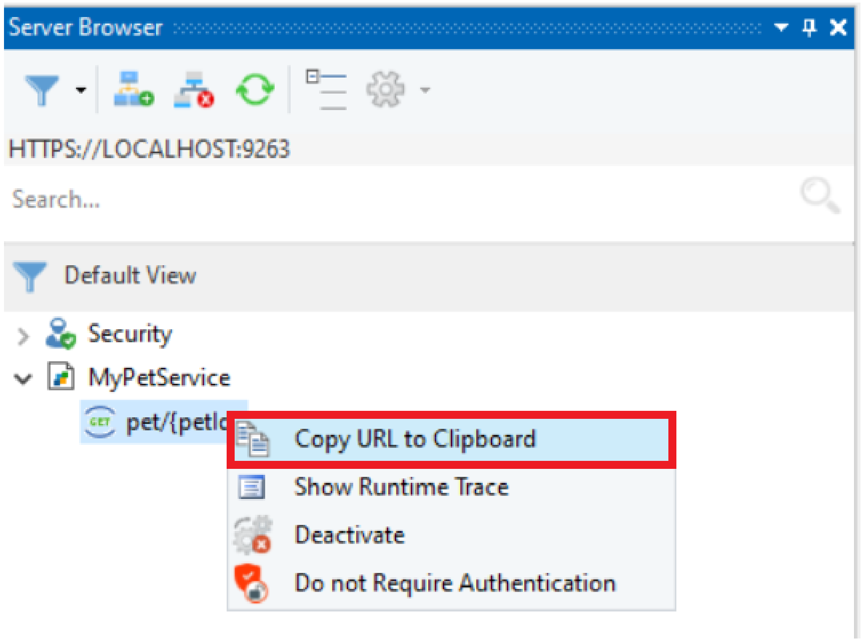
To show each of the steps associated with Asynchronous API requests, we will be using the Postman API Client to execute a request on the deployed Astera API.

5. Open your Postman Client, create a new collection and add a new request to it.

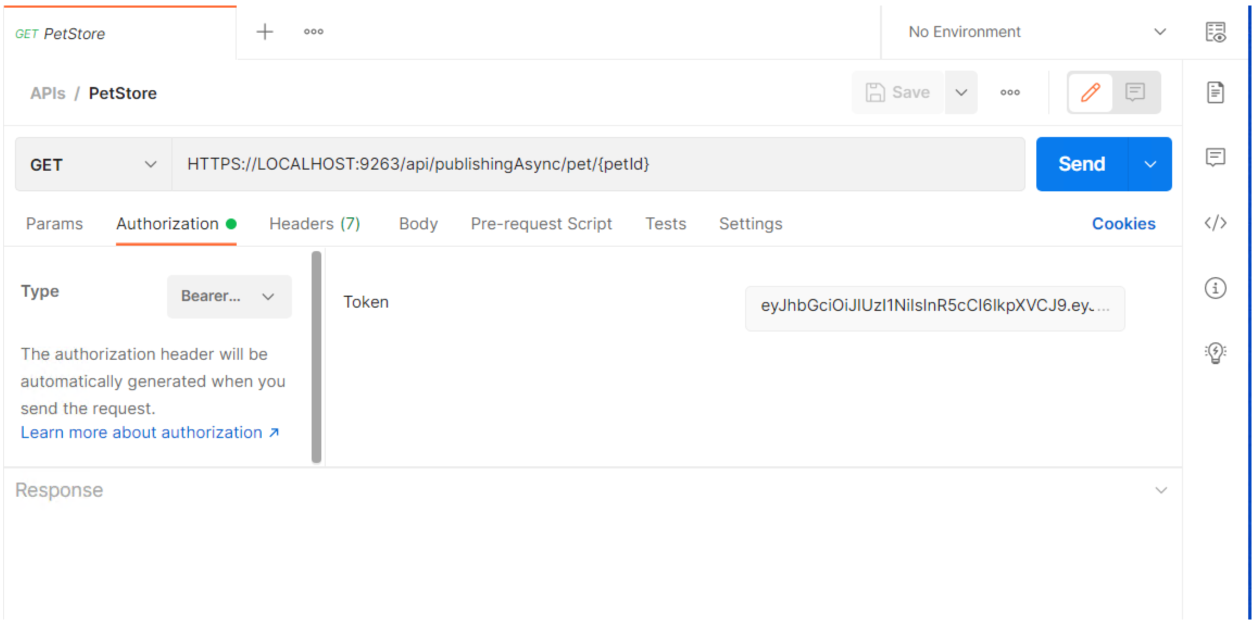


We have named our first request as ‘PetStore’

6. Select the appropriate method and enter the API URL copied from the API Browser,

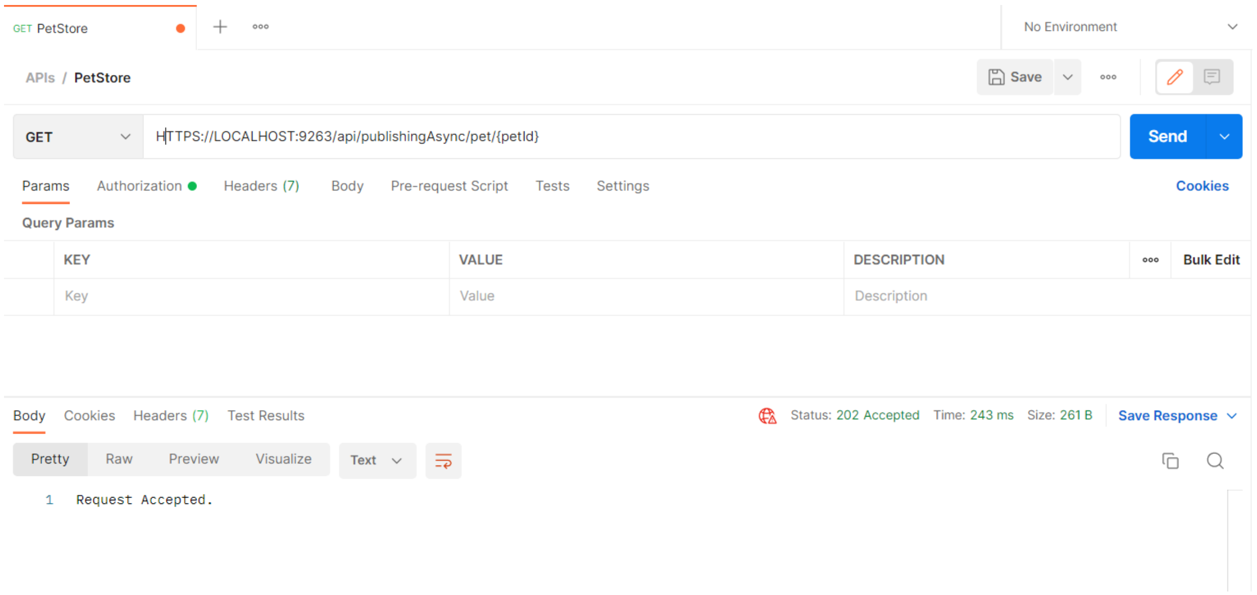


We can see the respective Postman request below,


GET PetStore
No Environment
APIs / PetStore
Save
GET
HTTPS://LOCALHOST:9263/api/publishingAsync/pet/{petId}
Send
Params
Authorization
Headers (7)
Body
Pre-request Script
Tests
Settings
Cookies
Type
Bearer...
The authorization header will be automatically generated when you send the request.
Learn more about authorization ↗
Token
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey...
Response

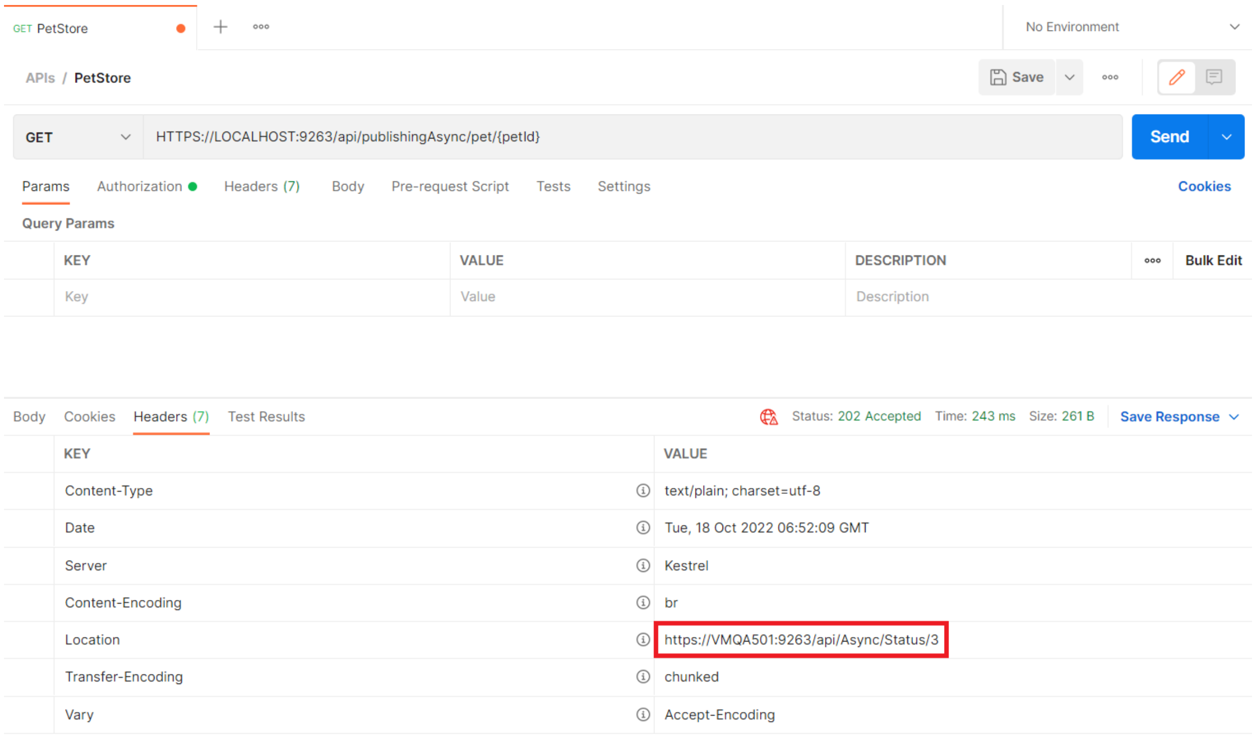
7. Click on Send.

It shows us that the request was accepted.



```
Request Accepted.
```

We have received a location response header and this parameter contains the successive status API's URL that can be used to inspect the API request's processing status.



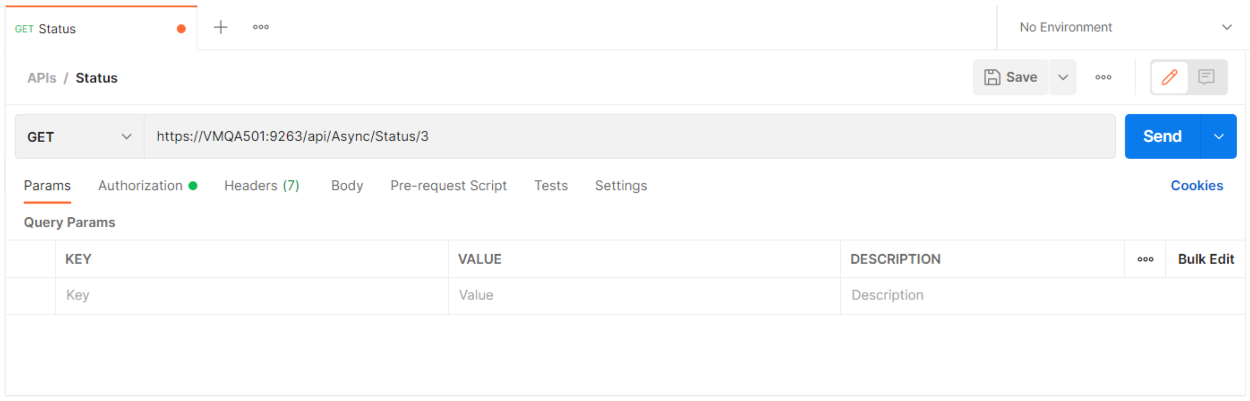
GET HTTPS://LOCALHOST:9263/api/publishingAsync/pet/{petId}

KEY	VALUE	DESCRIPTION
Key	Value	Description

Status: 202 Accepted Time: 243 ms Size: 261 B

KEY	VALUE
Content-Type	text/plain; charset=utf-8
Date	Tue, 18 Oct 2022 06:52:09 GMT
Server	Kestrel
Content-Encoding	br
Location	https://VMQA501:9263/api/Async/Status/3
Transfer-Encoding	chunked
Vary	Accept-Encoding

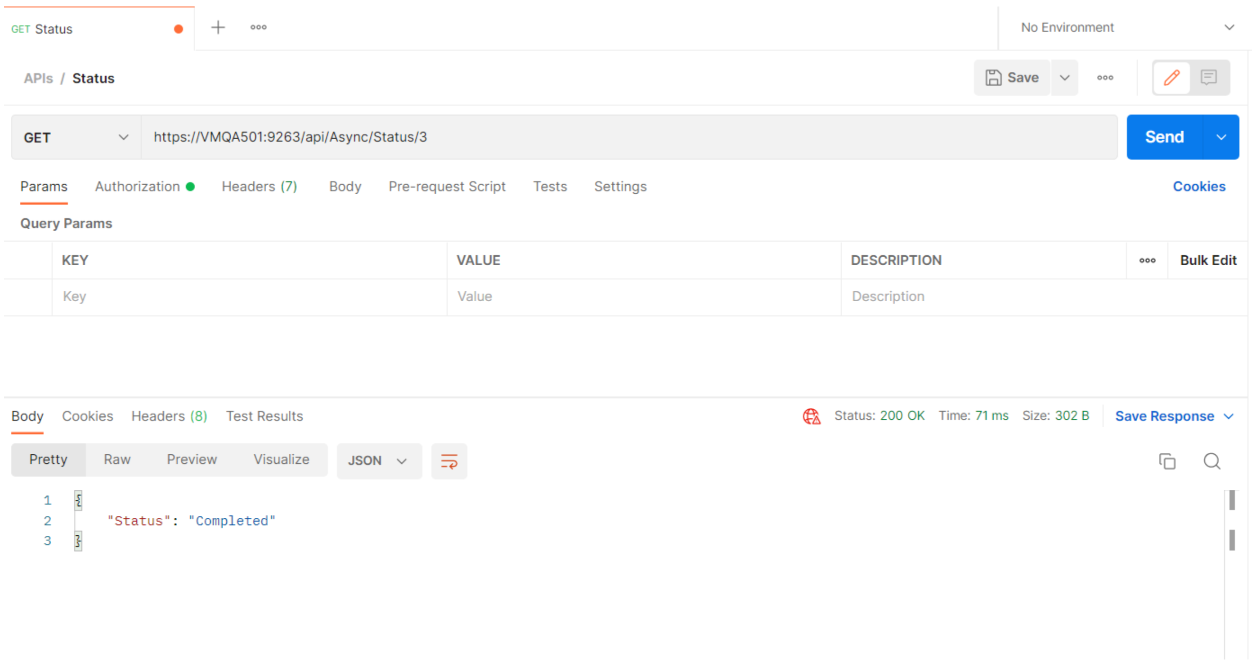
8. Next, send a follow-up request to the status API URL received in the location header.



GET https://VMQA501:9263/api/Async/Status/3

KEY	VALUE	DESCRIPTION
Key	Value	Description

9. Click Send and you will be able to view the status of the API call in the response body.



GET https://VMQA501:9263/api/Async/Status/3

KEY	VALUE	DESCRIPTION
Key	Value	Description

Status: 200 OK Time: 71 ms Size: 302 B

```
{
    "Status": "Completed"
}
```

‘*Status: Completed*’ means that the request we sent was completed.

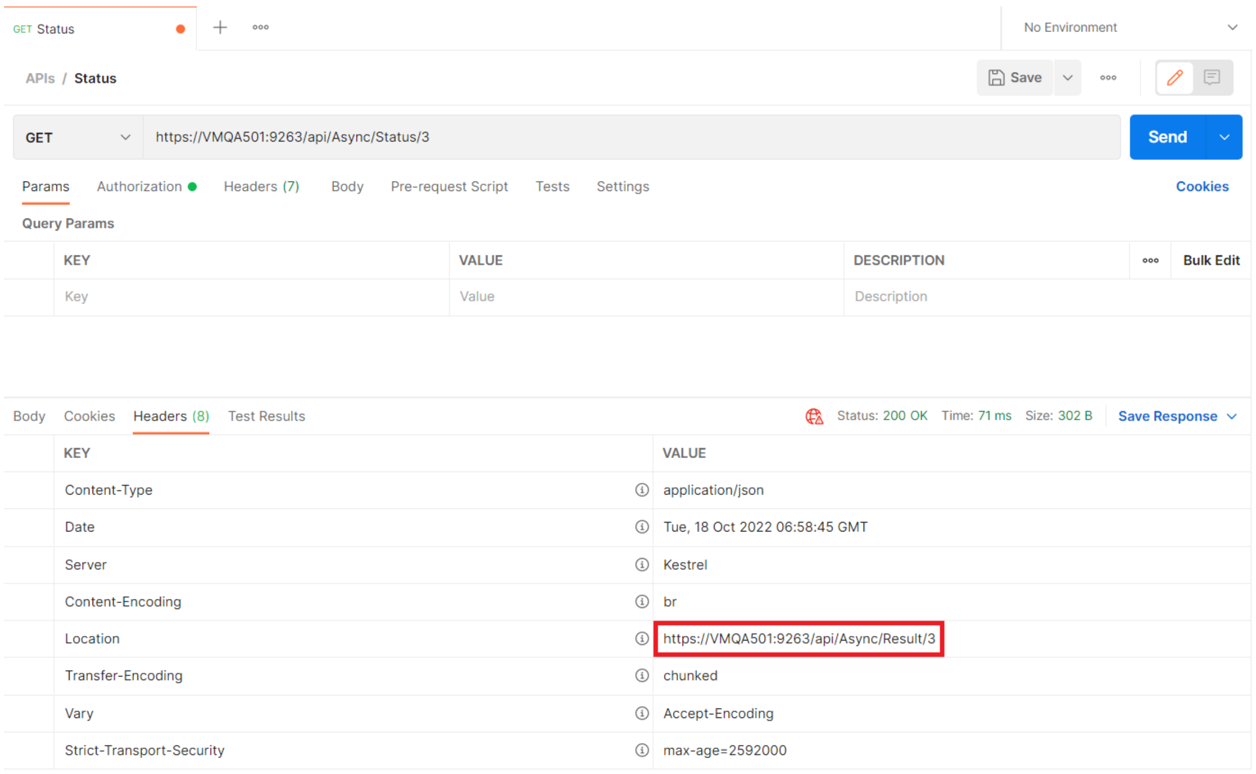
Apart from this, the other status possibilities are,

Running: The request is still being processed.

Error: The request processing has encountered an error.

Unknown: The request for the given ID was not found or purged.

10. The location response header received with a ‘Completed’ status API call is then used to make the successive API call to retrieve the API results.



GET https://VMQA501:9263/api/Async/Status/3

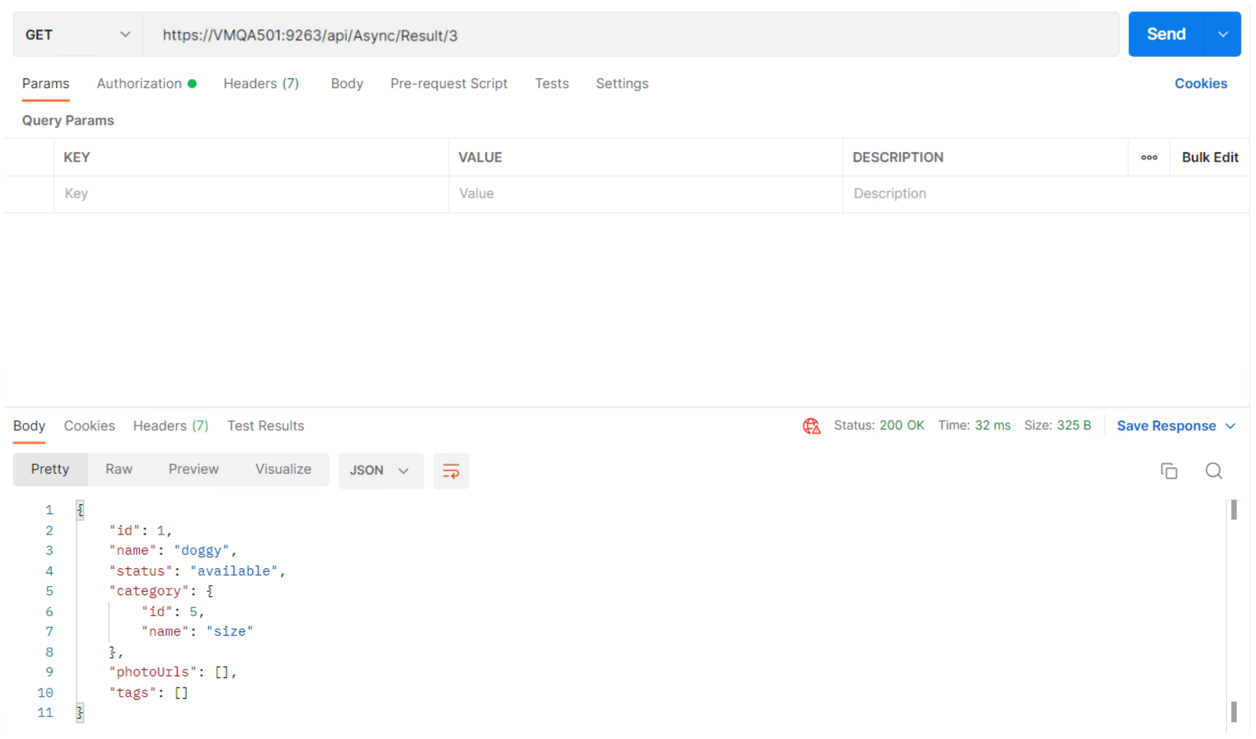
Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Status: 200 OK Time: 71 ms Size: 302 B

KEY	VALUE
Content-Type	application/json
Date	Tue, 18 Oct 2022 06:58:45 GMT
Server	Kestrel
Content-Encoding	br
Location	https://VMQA501:9263/api/Async/Result/3
Transfer-Encoding	chunked
Vary	Accept-Encoding
Strict-Transport-Security	max-age=2592000

11. Make a request to this result API to see the actual API response processed.



GET https://VMQA501:9263/api/Async/Result/3

Query Params

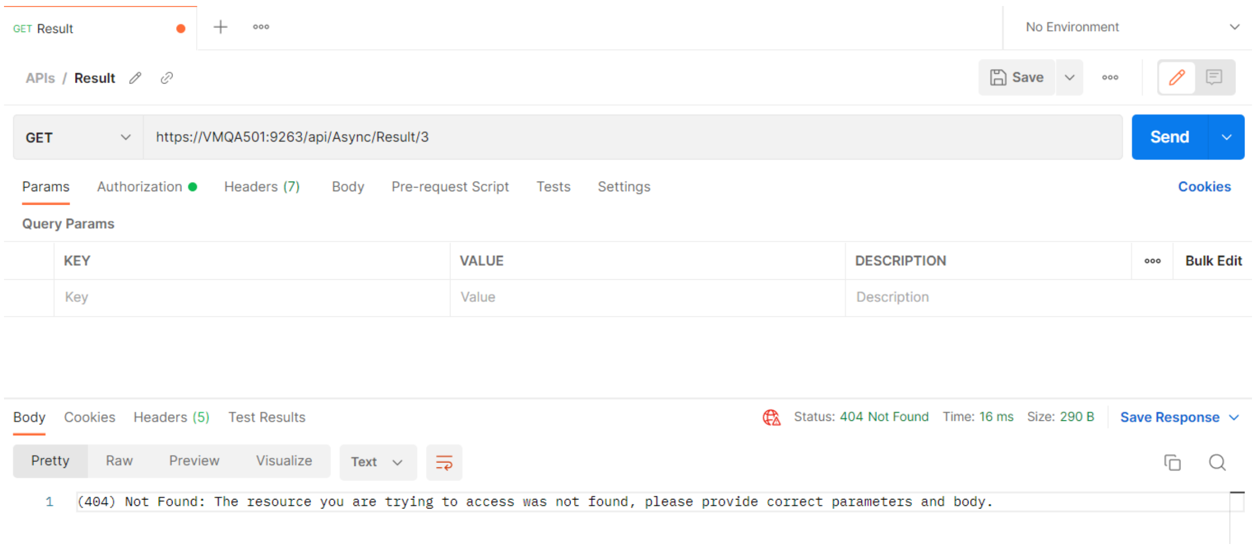
KEY	VALUE	DESCRIPTION
Key	Value	Description

Status: 200 OK Time: 32 ms Size: 325 B

```
{
    "id": 1,
    "name": "doggy",
    "status": "available",
    "category": {
        "id": 5,
        "name": "size"
    },
    "photoUrls": [],
    "tags": []
}
```

The result of an Asynchronous request is preserved for a duration of 24 hours after which it is purged/removed.

As you can see, after 24 hours, the status becomes ‘Unknown’ and the Status/Result APIs return a ‘404 Not Found’ response.



```
(404) Not Found: The resource you are trying to access was not found, please provide correct parameters and body.
```

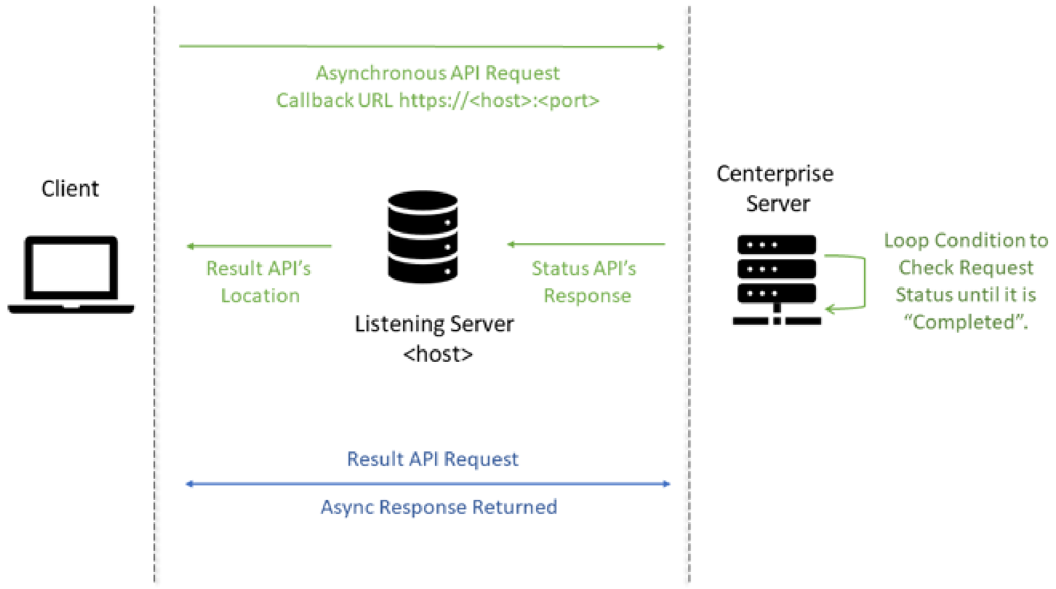
10.5.2 Callback URL

Attaching a Callback URL in an Async request allows the API client to get the response at the server specified in the URL, rather than polling for a response at various intervals.

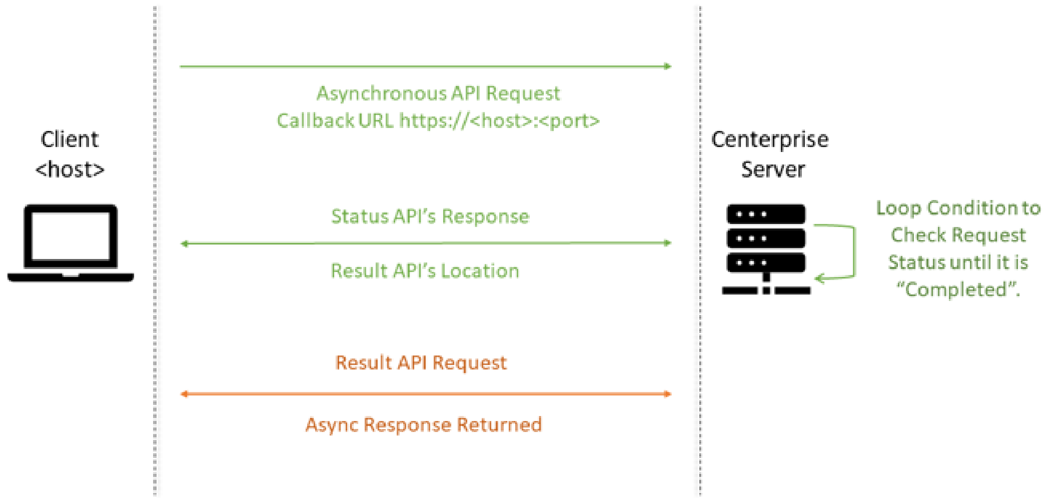
The Callback URL's functionality is implemented for the *Asynchronous API Requests*. A query parameter called *callbackUrl defining* the URL of the listening server is required in the Asynchronous request. Once the request is sent to the Astera server, it stores this call-back URL and periodically checks for the availability of the response. When the status is "completed" i.e., when the response is available, the Server sends it to the address that was specified in the URL.

The visual representation of the process will look like this,

1. Callback URL to the Listening Server,

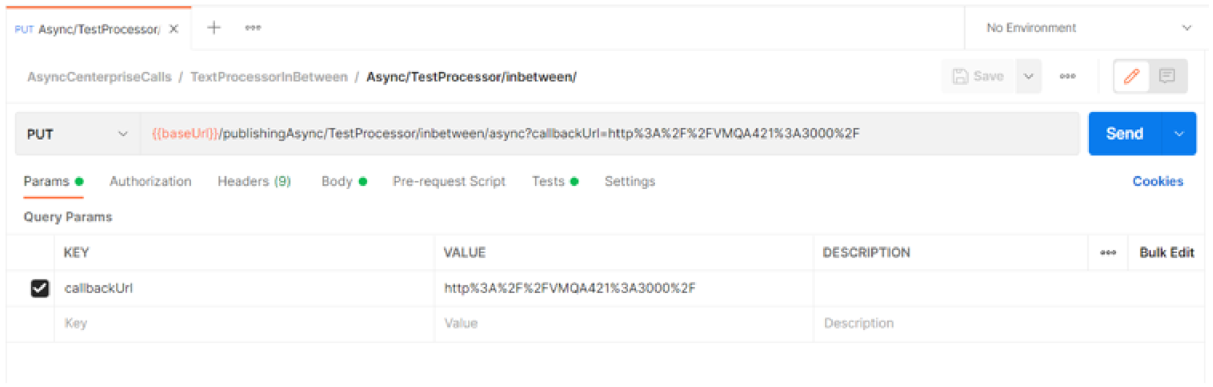


2. Callback URL to the Local Host,



Let us consider such a use-case in which a callback URL query parameter with the value "http::" is defined in the Asynchronous request.

It is necessary to encode the callback URL. After that the request will look something like this,



PUT {{baseUrl}}/publishingAsync/TestProcessor/inbetween/async?callbackUrl=http%3A%2F%2FVMQA421%3A3000%2F

	KEY	VALUE	DESCRIPTION
✓	callbackUrl	http%3A%2F%2FVMQA421%3A3000%2F	
	Key	Value	Description

Once the request is sent, the Astera server checks if the Status of the request is completed or not, and when it is, the response is sent, and it can be seen at the specified destination address.

Note: We have created a listening server at our end using the JavaScript Code. Its purpose was to continuously send requests till it receives the status response from the Astera Server and display its headers parameters on the terminal screen.

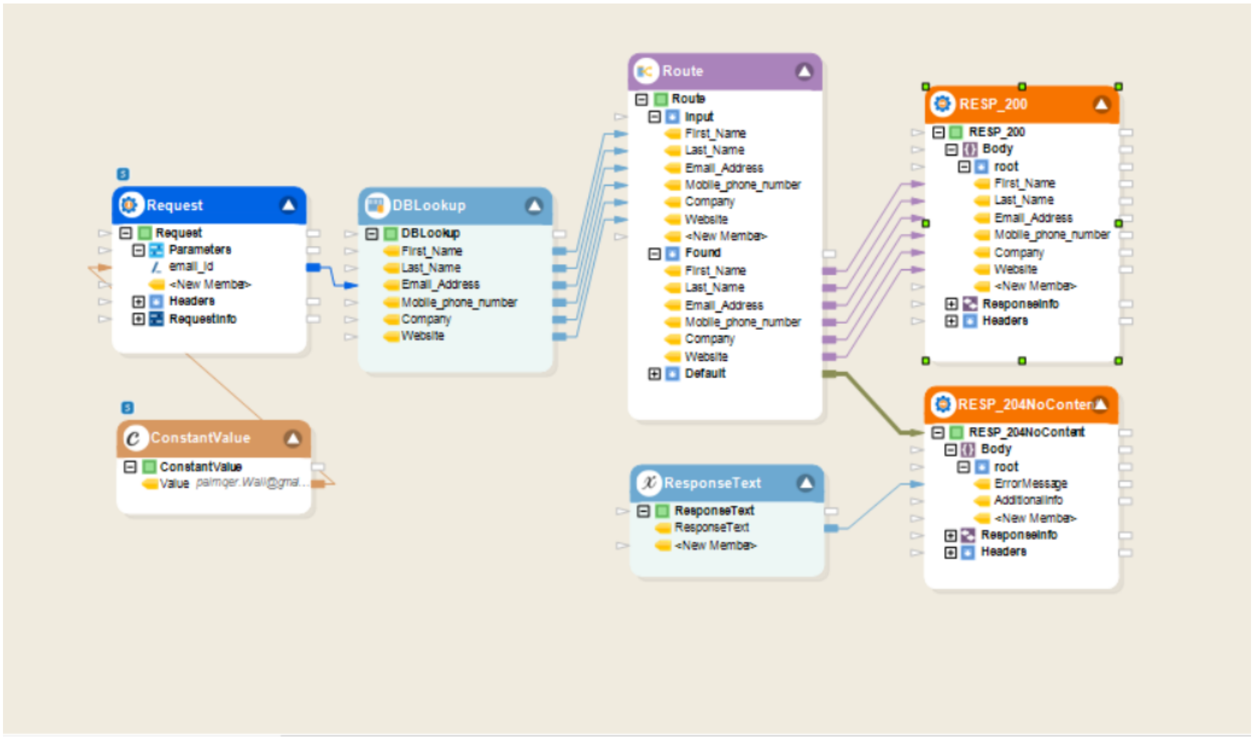
Now, the returned Result API's URL in the Location header parameter can be used to retrieve the response body of the initial Asynchronous call.

This concludes the article on Asynchronous API Request Execution and Callback URL with respect to Astera API Management.

10.6 Multiple Responses Using Conditional Route

An API flow can be conditioned to return a different response as per the designed flow. The server could return a successful response for a valid request or return a missing parameter response for an incomplete request.

To define an API with multiple responses, we have mapped two *REST Response* objects through a *Route Transformation* object conditioned on the request received. The *Route* conditions should be defined to take care of routing all the incoming data to either of the two responses at a time, avoiding any unexpected responses due to race conditions.



Since there is no data flowing for a 'No Content' response, such responses can be controlled using *Anchor Maps*. These are mapped with the *Route Transformation* outgoing node for the respective rule.

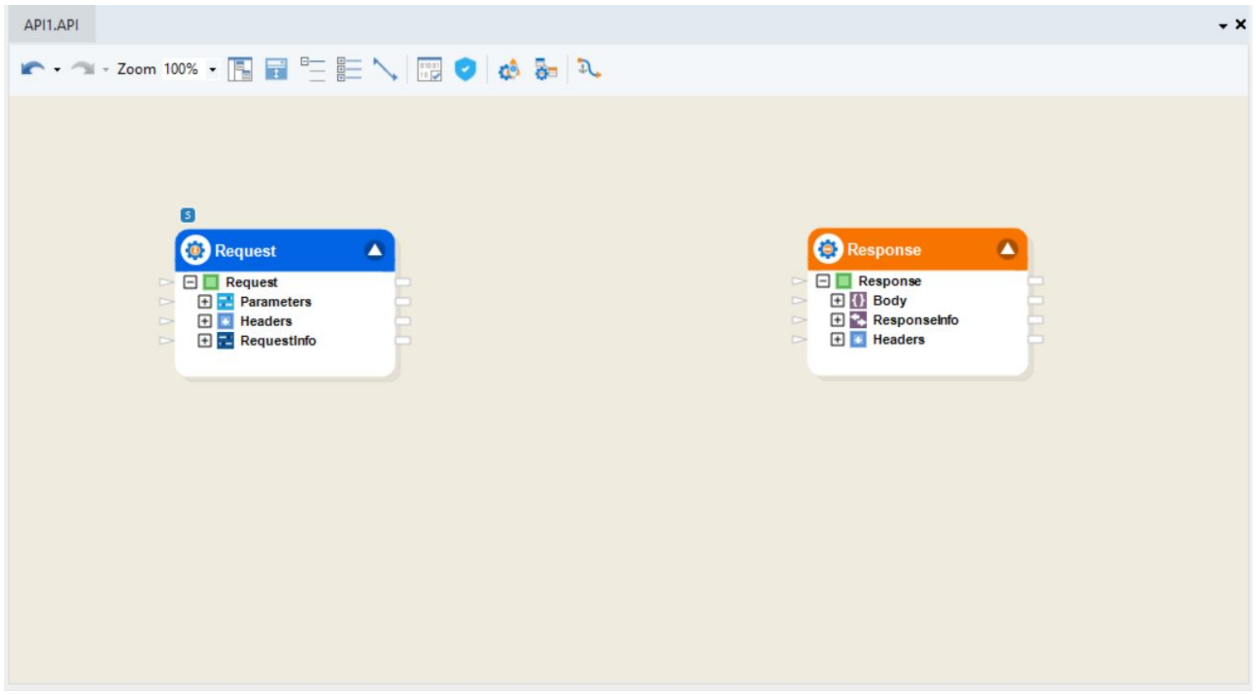
To create an anchor map, press the icon on the API flow toolbar and create a map from the UnderProcess rule node of the *Route Transformation* to the 'Resp_200_02' Response object.

This concludes the working of multiple responses in an API flow.

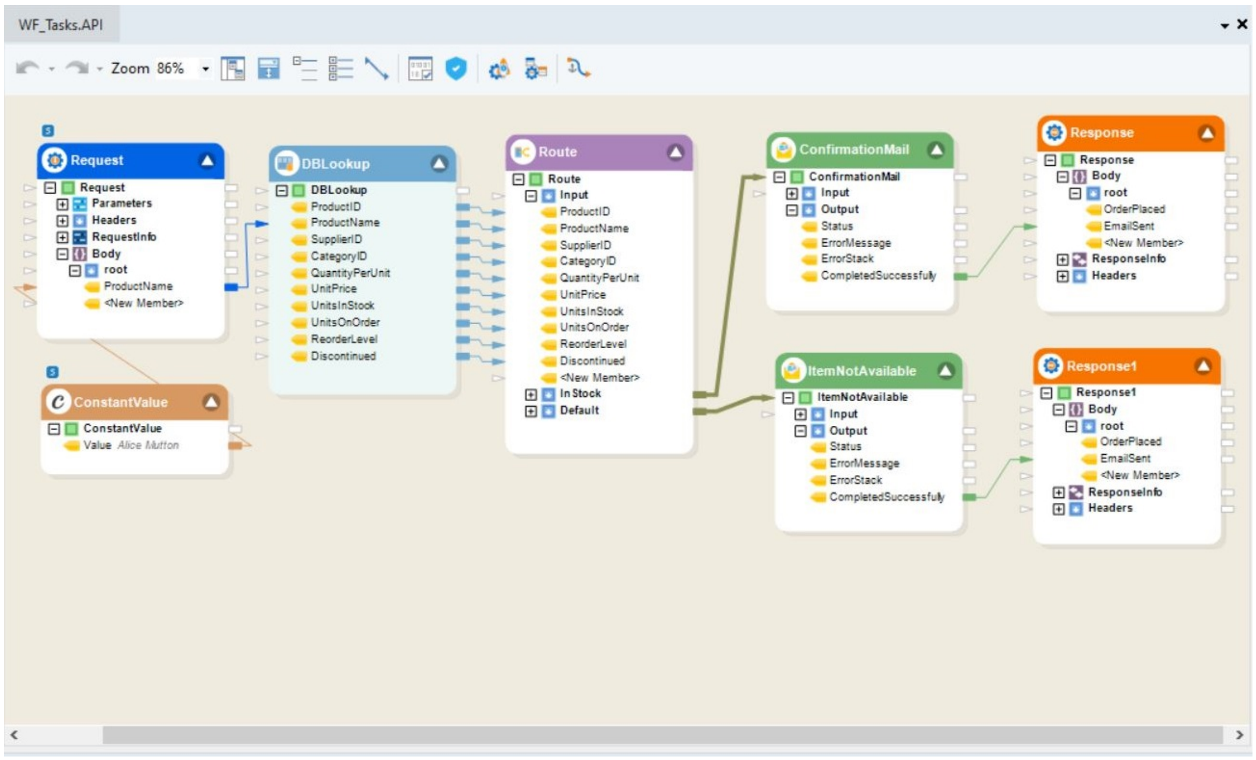
10.7 Workflow Tasks in an API Flow

An API flow orchestration allows the usage of various workflow tasks. These include tasks like *Send Mail*, *System File Actions*, *Run Exe Programs*, *Run Flow* tasks, FTP tasks, etc, which can be utilized in accordance with the API action.

Inside the toolbox, the *Workflow Tasks* tab lists all the available tasks that can be used when designing an API flow.



For our use case, we have designed an API flow for the ‘Get Product by name’ endpoint.



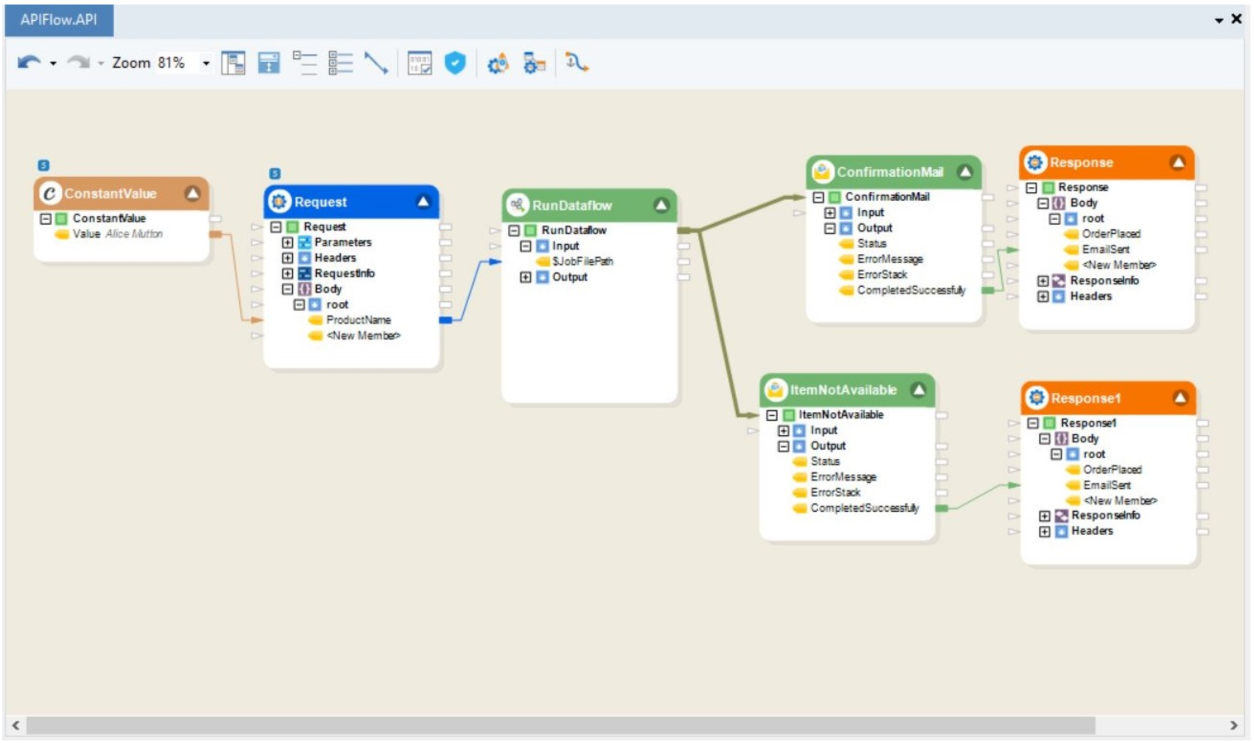
As you can see in the flow above, two *Send Mail* workflow tasks have been used. Once a request is received with the product name, it is sent as an input to the *Database Lookup* object to fetch a matching record. It is then passed through a *Route Transformation* which routes the data to send a confirmation mail for a successful match or a ‘Not Found’ mail notification otherwise.

The user will either receive a mail that says the item is available or a mail that states that the item is unavailable, and the API will return the respective responses.

Note: If no data mappings are available to orchestrate the flow, as in this case when using a *Send Mail* object, the user can make use of Anchor Maps to control the flow.

To learn more about how Anchor Maps are used, please refer to the respective document here.

Similarly, here is another way we have used workflow tasks within an API flow,



Instead of using the *DB Lookup* and *Route Transformation* object, the entire process has been replaced with a *Run Dataflow* object that can run any ETL pipeline.

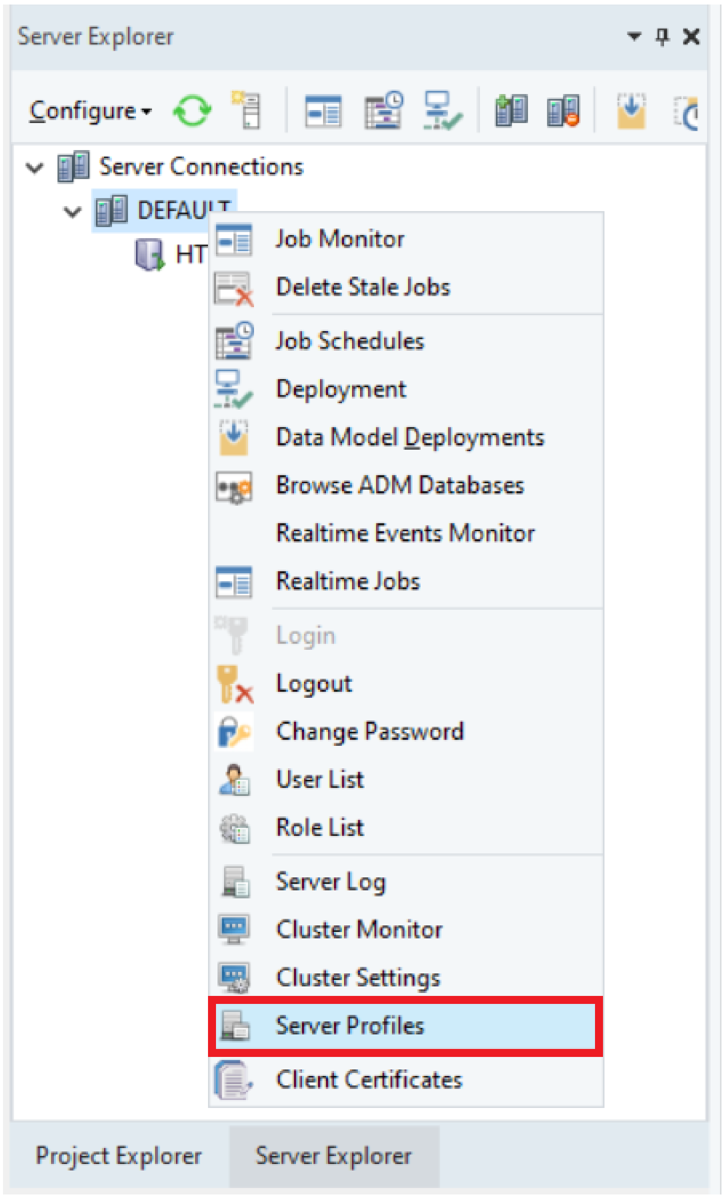
The dataflow which will be triggered in the API is doing the same work as the previous API flow. However, it is now less cluttered.

This concludes the functionality of Workflow tasks in an API flow.

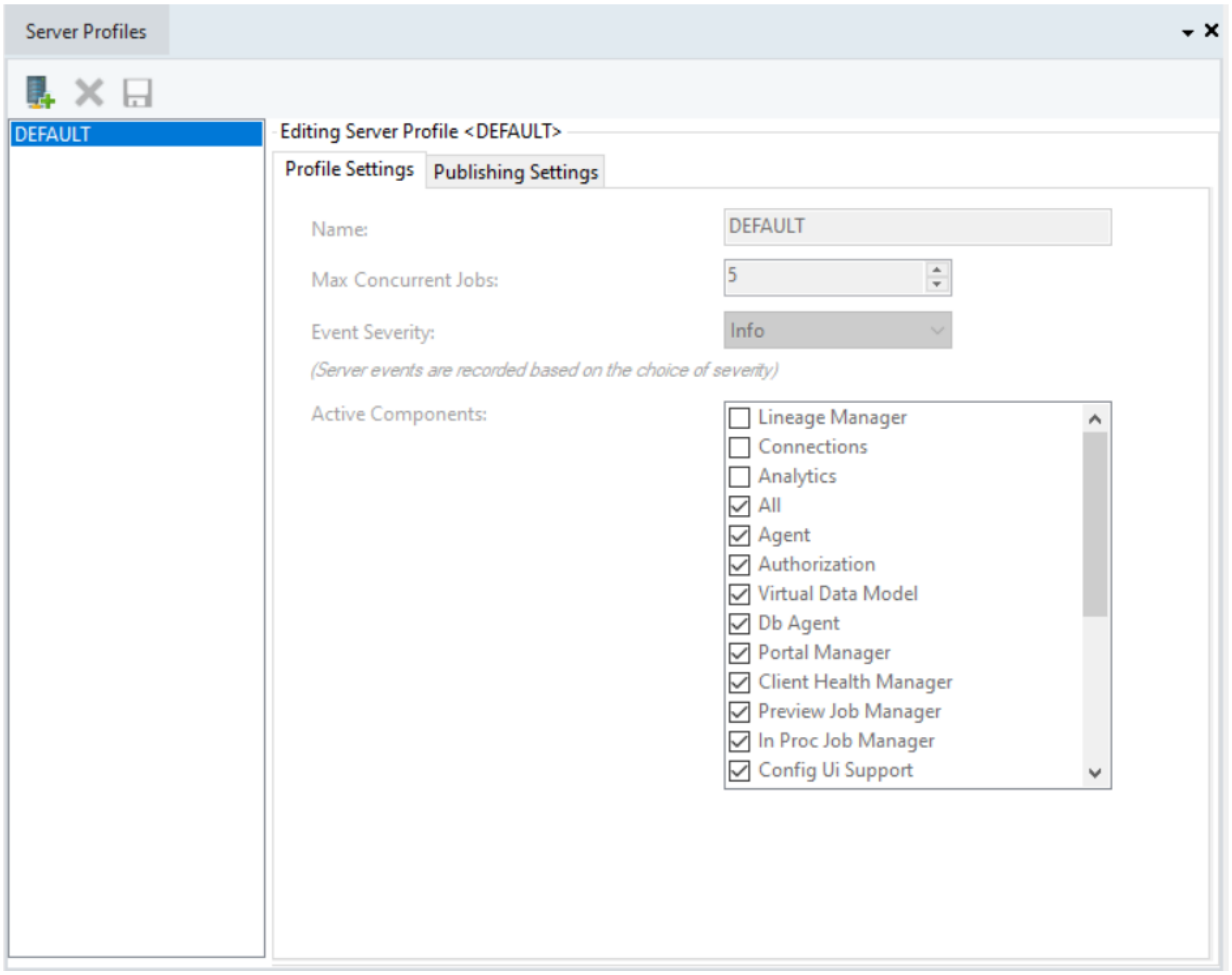
10.8 Enable File Download-Upload Through APIs

Users can utilize APIs to upload and download files to and from the Astera API Management server.

1. Head to the Server Explorer, right-click on the cluster node, and select *Server Profiles* from the context menu.



This will open a new window.



Server Profiles

DEFAULT

Editing Server Profile <DEFAULT>

Profile Settings | Publishing Settings

Name: DEFAULT

Max Concurrent Jobs: 5

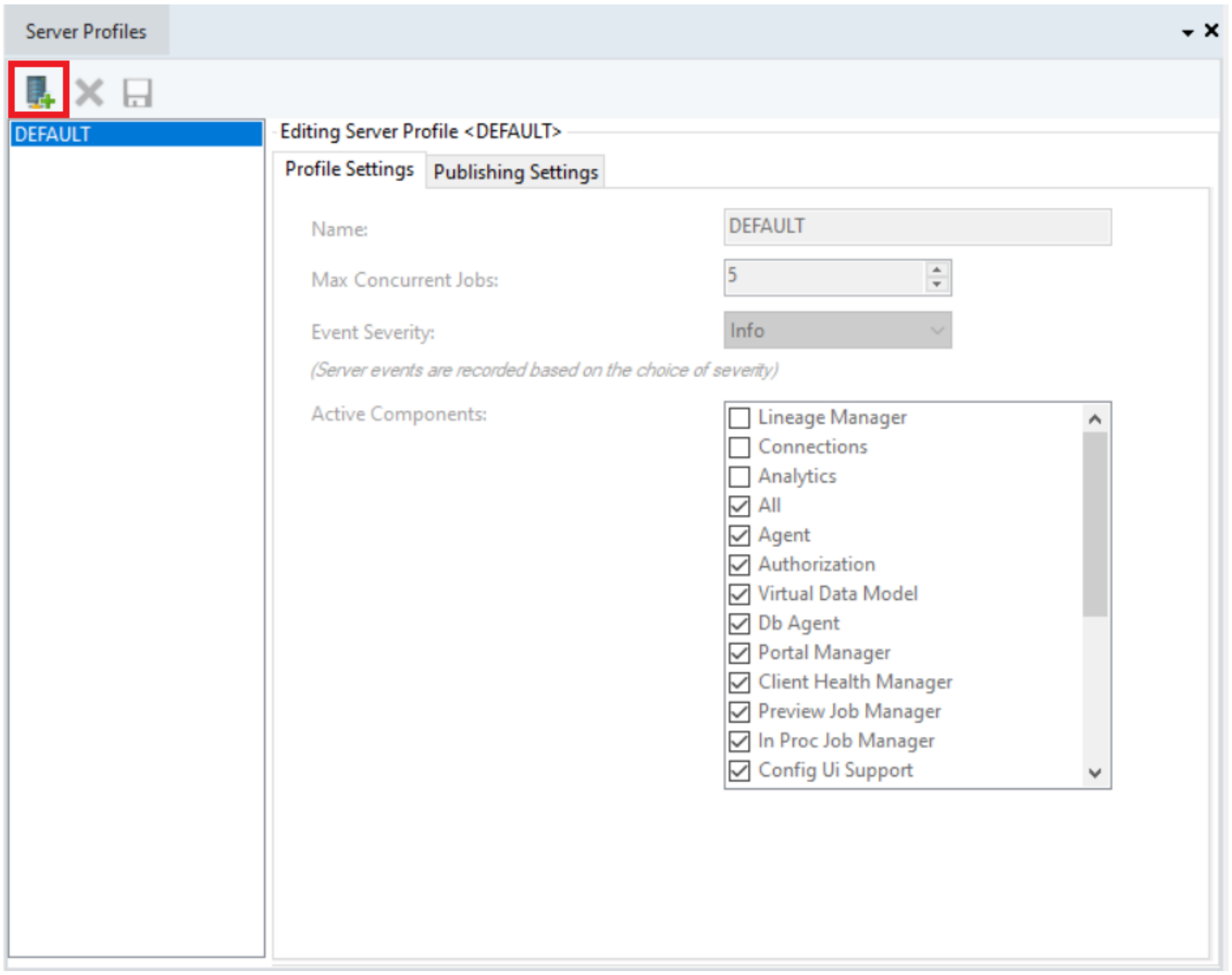
Event Severity: Info

(Server events are recorded based on the choice of severity)

Active Components:

- [ ] Lineage Manager
- [ ] Connections
- [ ] Analytics
- [x] All
- [x] Agent
- [x] Authorization
- [x] Virtual Data Model
- [x] Db Agent
- [x] Portal Manager
- [x] Client Health Manager
- [x] Preview Job Manager
- [x] In Proc Job Manager
- [x] Config Ui Support

2. Create a new profile by clicking on the following icon,



Name: DEFAULT

Max Concurrent Jobs: 5

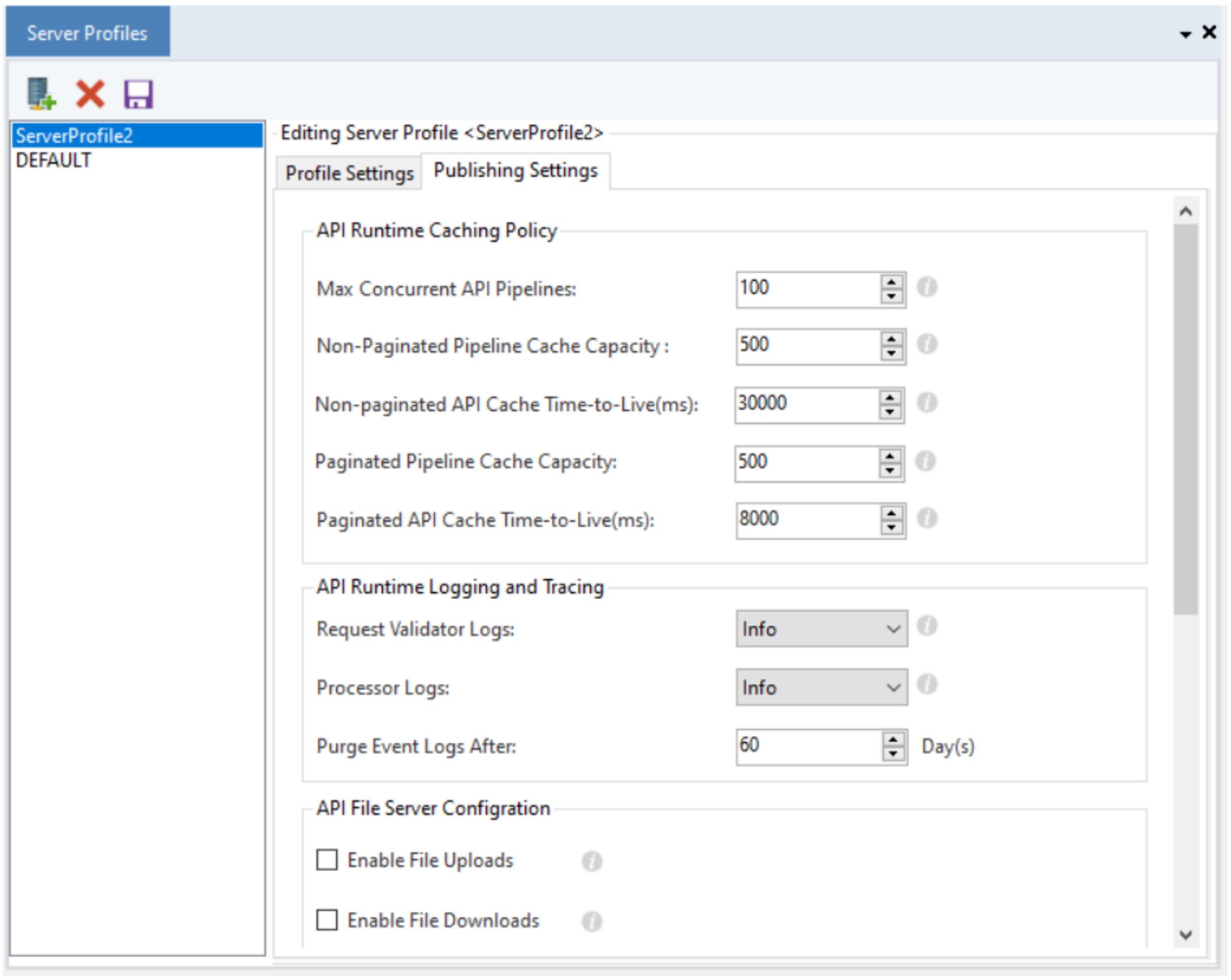
Event Severity: Info

(Server events are recorded based on the choice of severity)

Active Components:

- [ ] Lineage Manager
- [ ] Connections
- [ ] Analytics
- [x] All
- [x] Agent
- [x] Authorization
- [x] Virtual Data Model
- [x] Db Agent
- [x] Portal Manager
- [x] Client Health Manager
- [x] Preview Job Manager
- [x] In Proc Job Manager
- [x] Config Ui Support

3. Once the new profile is created, select the *Publishing Settings* tab,



Editing Server Profile <ServerProfile2>

API Runtime Caching Policy

Max Concurrent API Pipelines: 100

Non-Paginated Pipeline Cache Capacity : 500

Non-paginated API Cache Time-to-Live(ms): 30000

Paginated Pipeline Cache Capacity: 500

Paginated API Cache Time-to-Live(ms): 8000

API Runtime Logging and Tracing

Request Validator Logs: Info

Processor Logs: Info

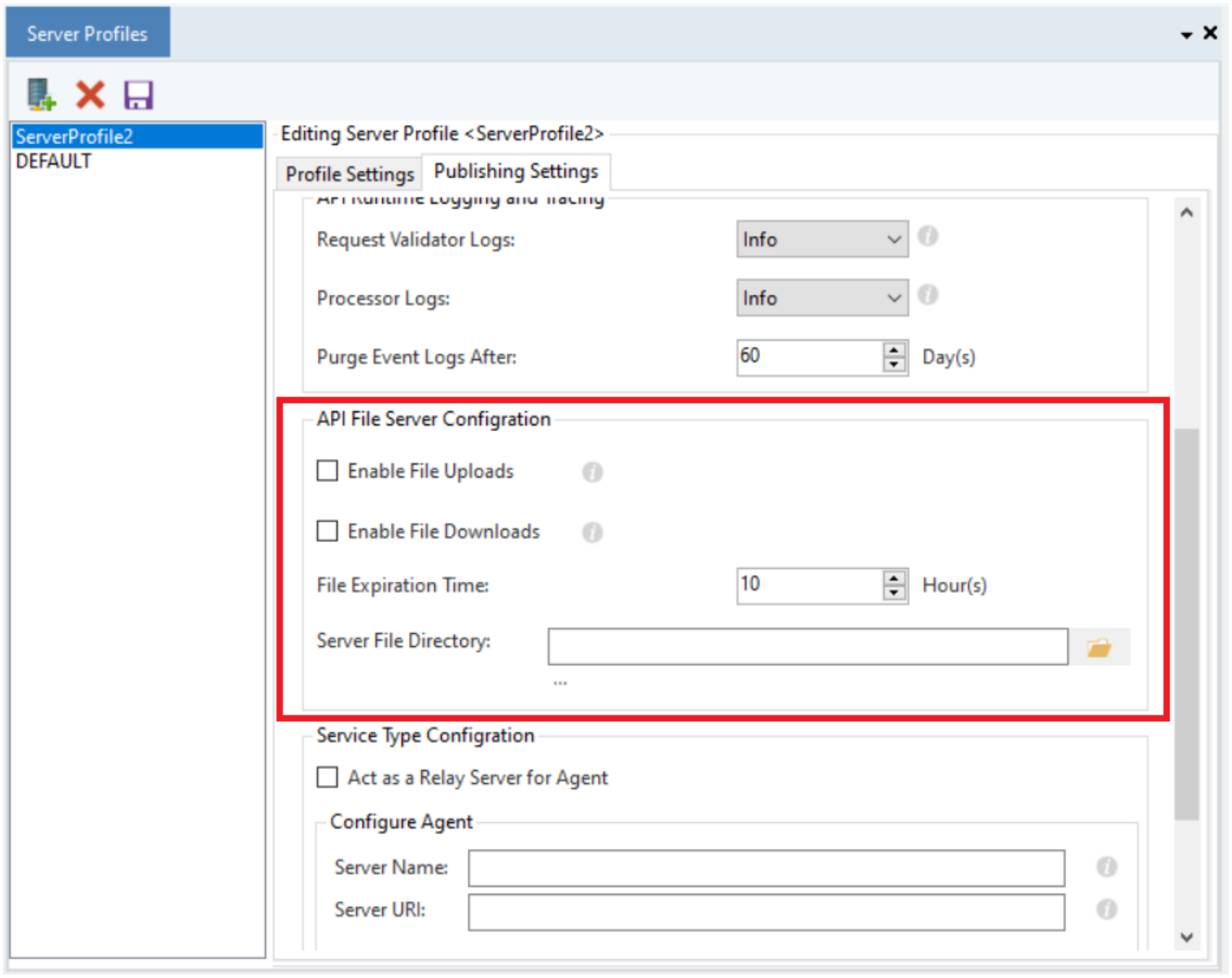
Purge Event Logs After: 60 Day(s)

API File Server Configration

☐ Enable File Uploads

☐ Enable File Downloads

4. Go to the *API File Server Configuration* section. Here, we can configure the file action functionalities.


Server Profiles

ServerProfile2
DEFAULT

Editing Server Profile <ServerProfile2>

Profile Settings | Publishing Settings

API Runtime Logging and Tracing

Request Validator Logs: Info

Processor Logs: Info

Purge Event Logs After: 60 Day(s)

API File Server Configration

☐ Enable File Uploads

☐ Enable File Downloads

File Expiration Time: 10 Hour(s)

Server File Directory:

Service Type Configration

☐ Act as a Relay Server for Agent

Configure Agent

Server Name:

Server URI:

Enable File Uploads: Selecting this checkbox will let the user upload files onto the specified server directory.

Enable File Downloads: Selecting this checkbox will let the user download files from the server.

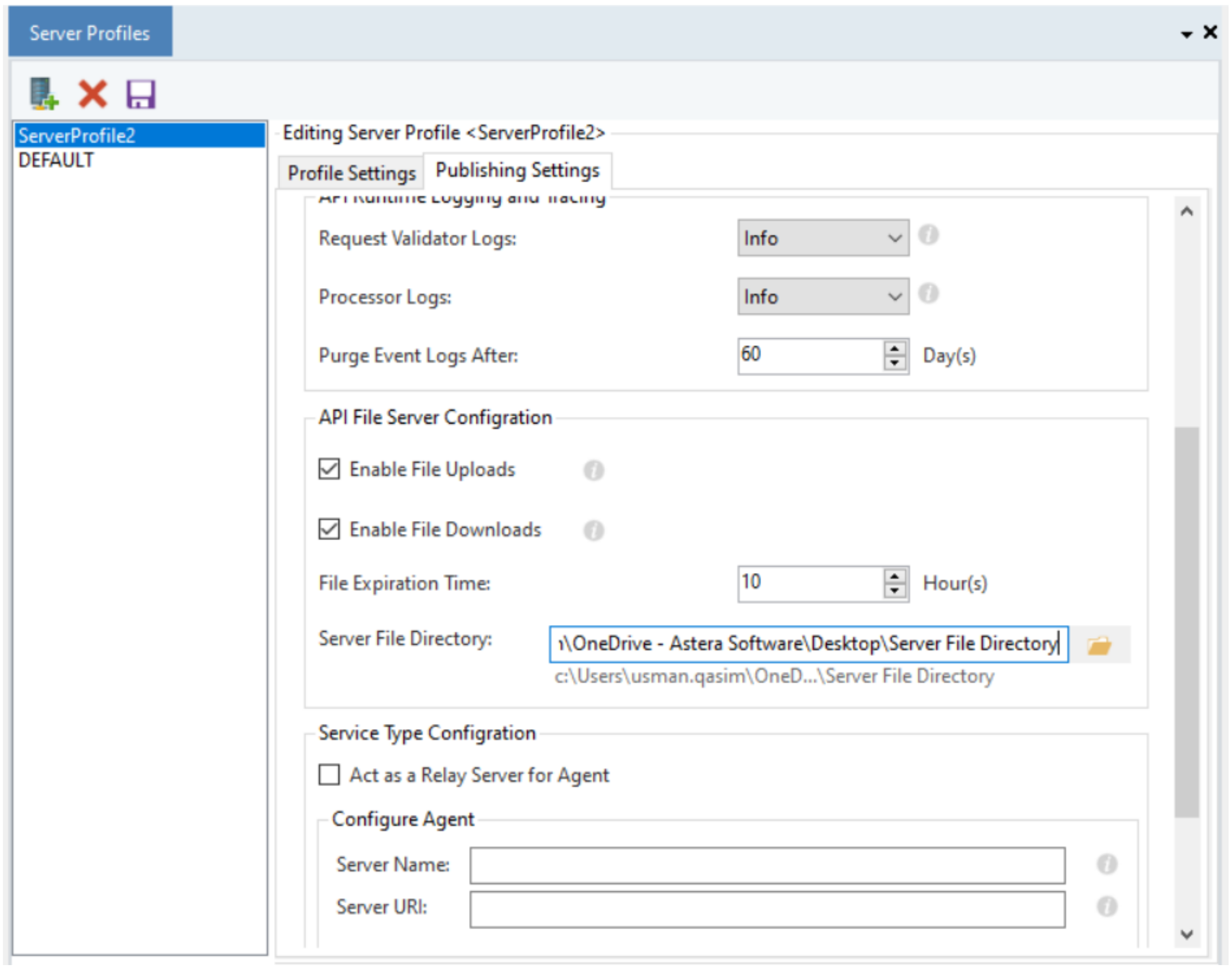
File Expiration Time: This counter determines how long the file will be kept in the Server File Directory before it is automatically removed.

Server File Directory: This is where the file path will be given to the Server File Directory. All the file uploads will be saved here, which can also be downloaded.

Note: We can download a file from anywhere on the server as long as that location is accessible by the server using the *Download Path Generator* object.

5. Select both checkboxes and provide a file path.

Note: The *Server File Directory* file path can be at any location that is accessible by the Astera server, be it on local or remote.



API Runtime Logging and Tracing

Request Validator Logs: Info

Processor Logs: Info

Purge Event Logs After: 60 Day(s)

API File Server Configration

☑ Enable File Uploads

☑ Enable File Downloads

File Expiration Time: 10 Hour(s)

Server File Directory: \OneDrive - Astera Software\Desktop\Server File Directory

c:\Users\usman.qasim\OneD...\Server File Directory

Service Type Configration

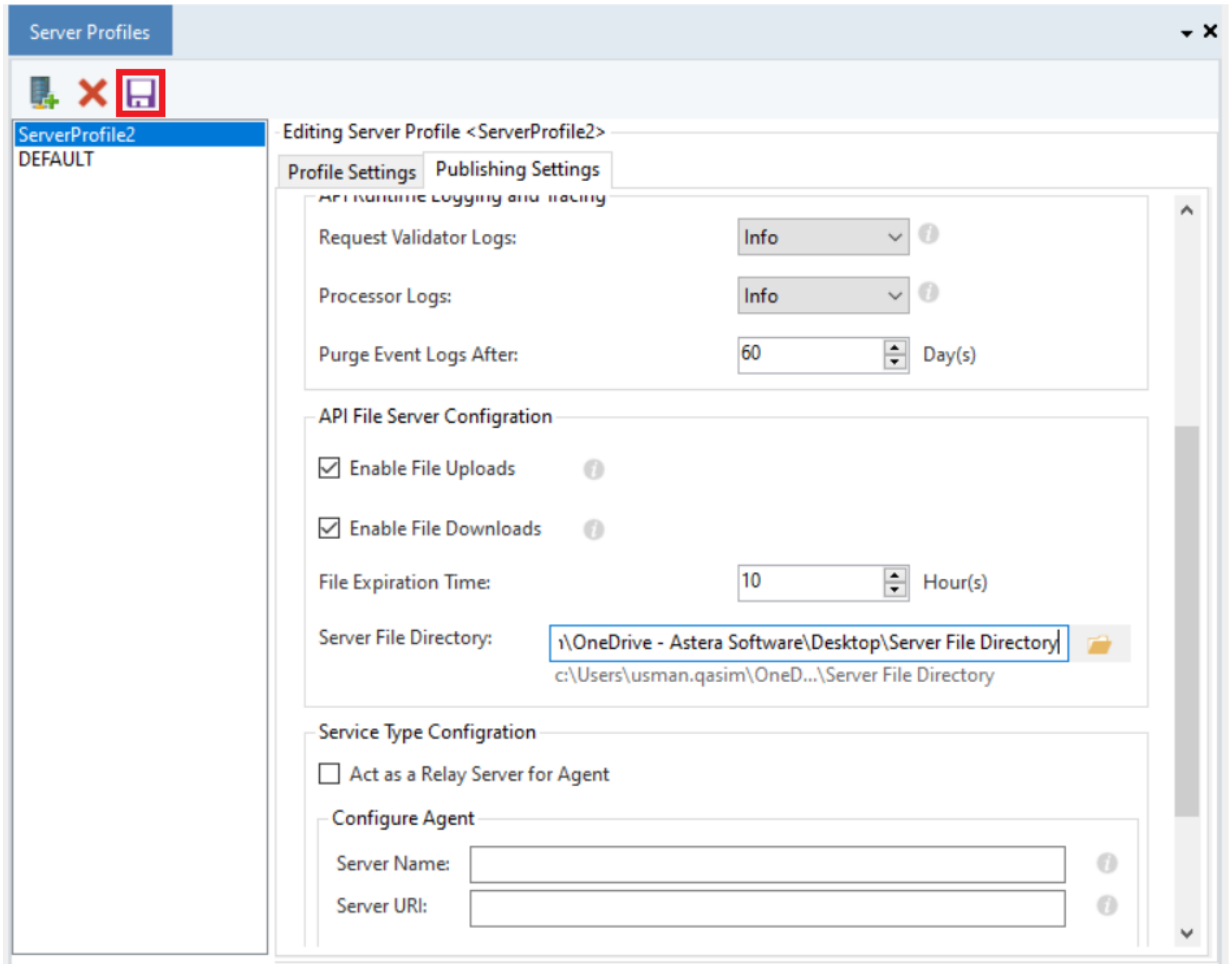
☐ Act as a Relay Server for Agent

Configure Agent

Server Name:

Server URI:

6. Once done, save the changes to the server profile.



Editing Server Profile <ServerProfile2>

API Runtime Logging and Tracing

Request Validator Logs: Info

Processor Logs: Info

Purge Event Logs After: 60 Day(s)

API File Server Configration

☑ Enable File Uploads

☑ Enable File Downloads

File Expiration Time: 10 Hour(s)

Server File Directory: \OneDrive - Astera Software\Desktop\Server File Directory

c:\Users\usman.qasim\OneD...\Server File Directory

Service Type Configration

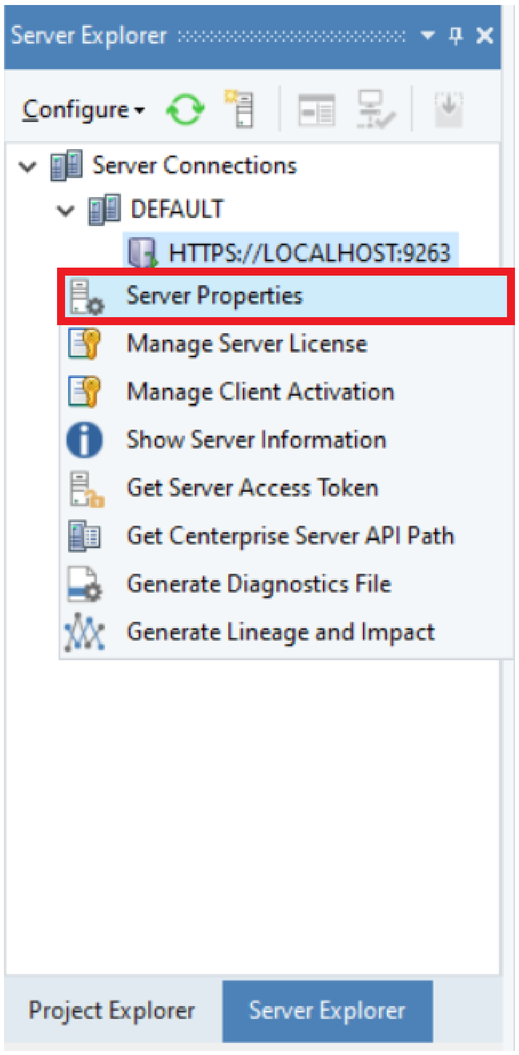
☐ Act as a Relay Server for Agent

Configure Agent

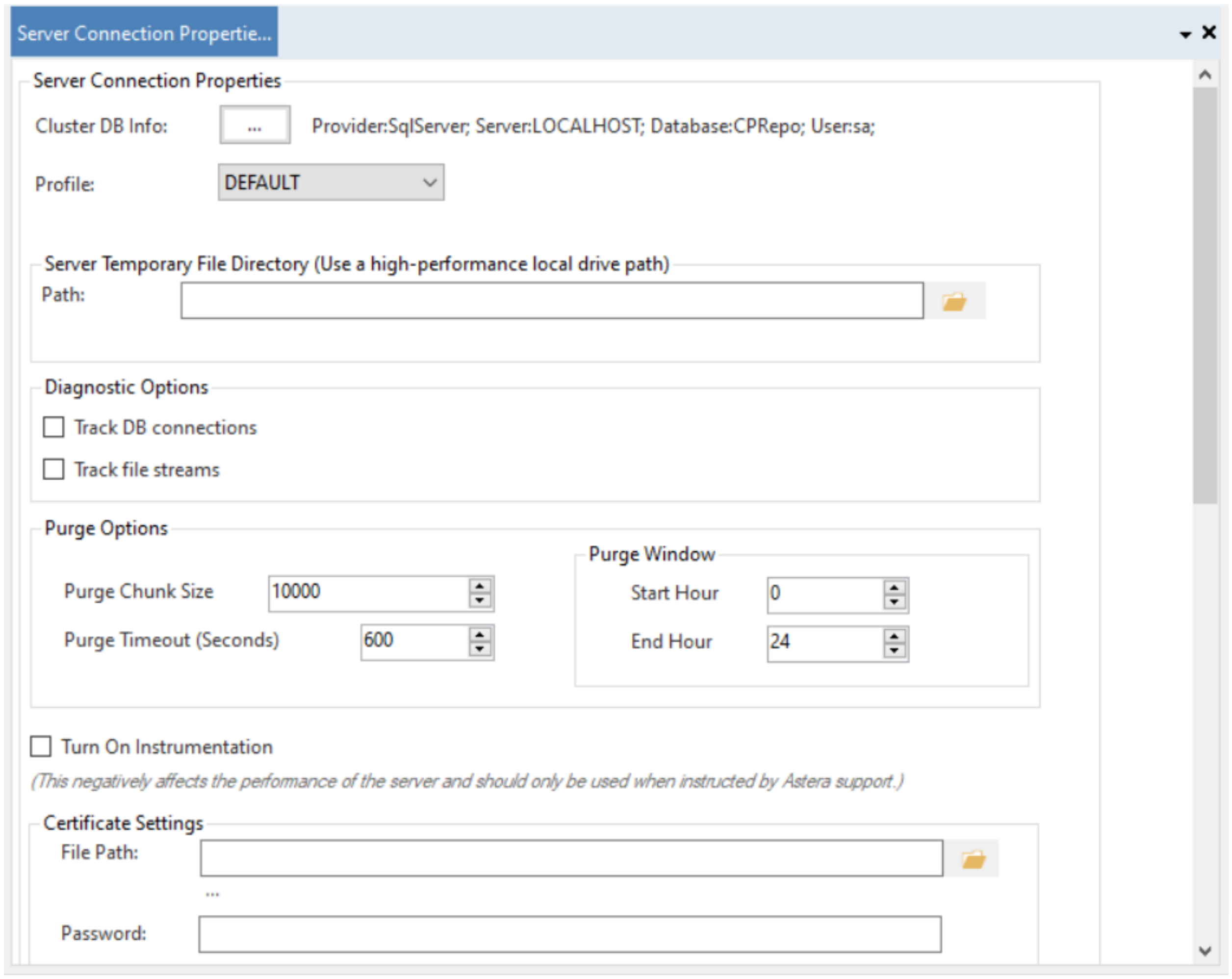
Server Name:

Server URI:

7. Next, right-click on the Server URL node in the Server Explorer and select *Server Properties* from the context menu.



This will open the *Server Properties* window.


Server Connection Propertie...

Server Connection Properties

Cluster DB Info: ... Provider:SqlServer; Server:LOCALHOST; Database:CPRepo; User:sa;

Profile: DEFAULT

Server Temporary File Directory (Use a high-performance local drive path)

Path:

Diagnostic Options

☐ Track DB connections

☐ Track file streams

Purge Options

Purge Chunk Size 10000

Purge Timeout (Seconds) 600

Purge Window

Start Hour 0

End Hour 24

☐ Turn On Instrumentation

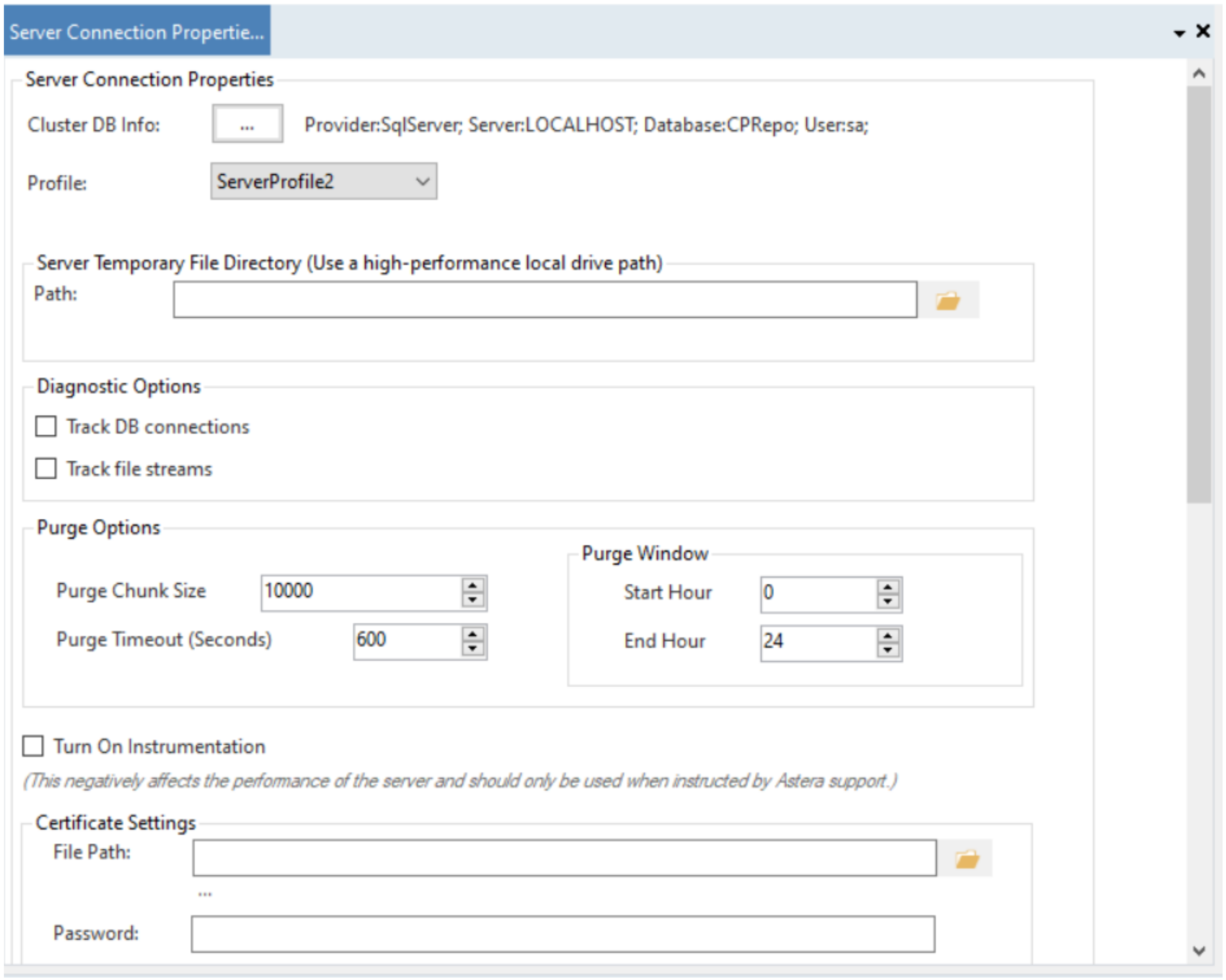
(This negatively affects the performance of the server and should only be used when instructed by Astera support.)

Certificate Settings

File Path:

Password:

8. Select the *Profile* that you have just configured and save it.



Cluster DB Info: ... Provider:SqlServer; Server:LOCALHOST; Database:CPRepo; User:sa;

Profile: ServerProfile2

Server Temporary File Directory (Use a high-performance local drive path)

Path:

Diagnostic Options

☐ Track DB connections

☐ Track file streams

Purge Options

Purge Chunk Size 10000

Purge Timeout (Seconds) 600

Purge Window

Start Hour 0

End Hour 24

☐ Turn On Instrumentation

(This negatively affects the performance of the server and should only be used when instructed by Astera support.)

Certificate Settings

File Path:

...

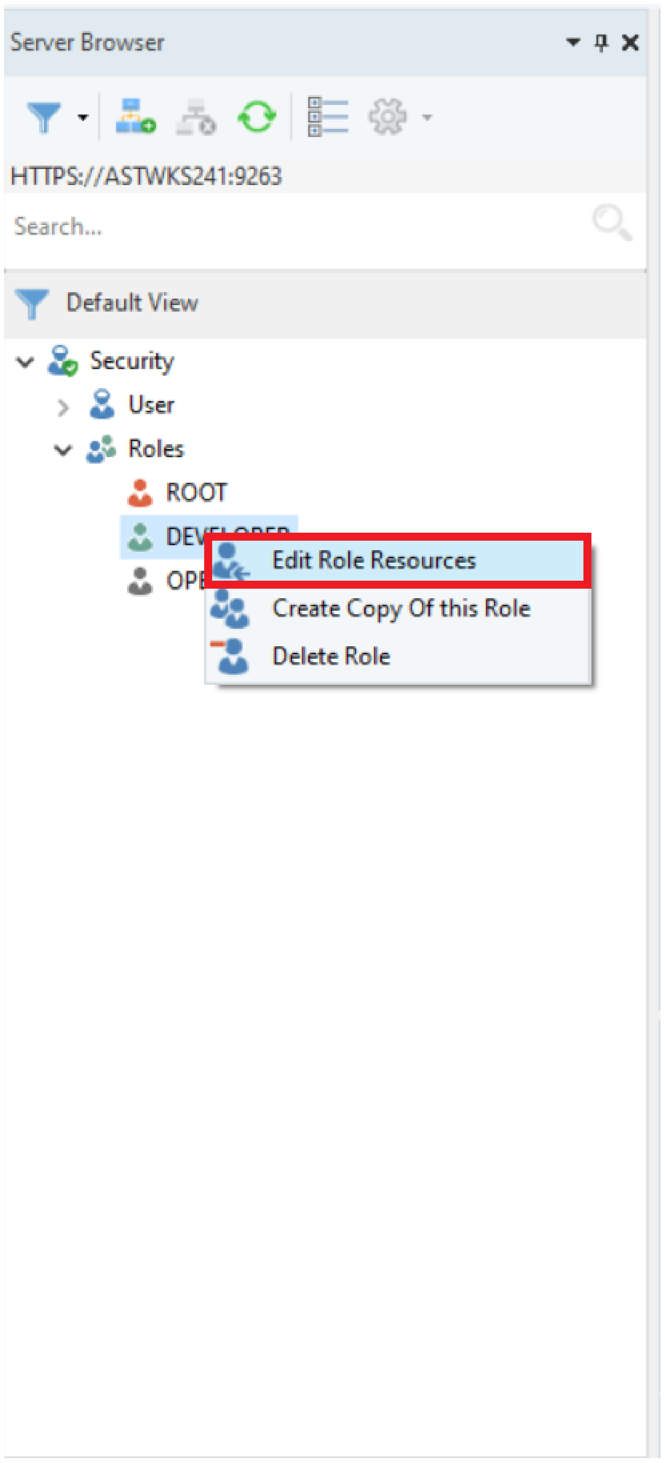
Password:

File upload and download functionalities have now been enabled with this profile on the server.

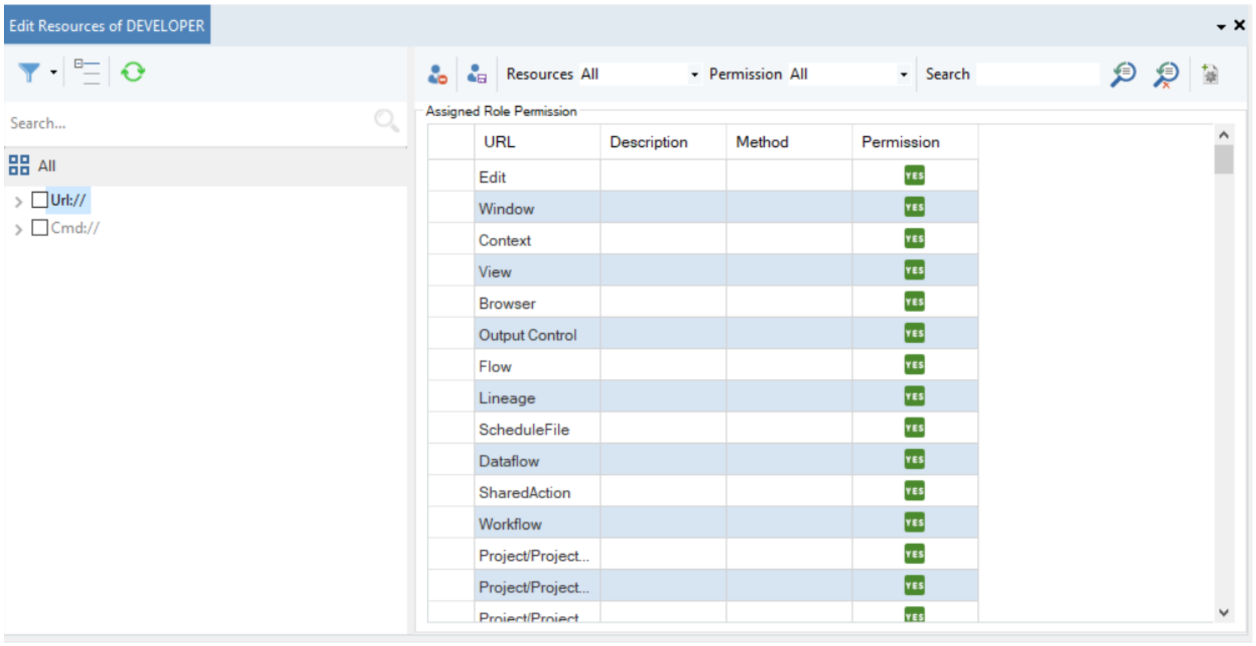
10.8.1 Enable Download-Upload for Non-Admin/Non-Root Users

For a non-admin or a non-root user, we must go to the user roles and enable the *Upload Download File APIs* option, otherwise, the user cannot proceed with the upload document.

1. Right-click on a non-root role and select *Edit Role Resources* from the context menu,

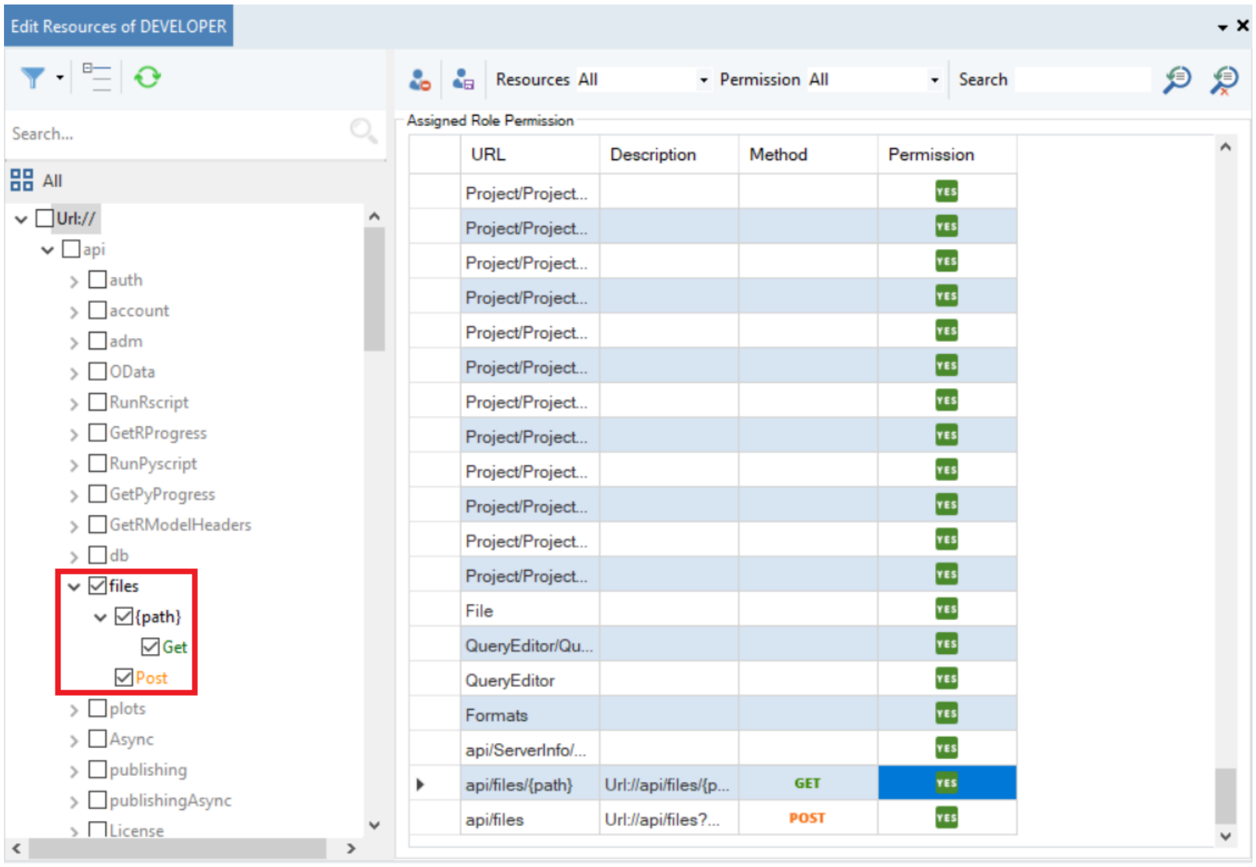


This will open a new window.



URL	Description	Method	Permission
Edit			YES
Window			YES
Context			YES
View			YES
Browser			YES
Output Control			YES
Flow			YES
Lineage			YES
ScheduleFile			YES
Dataflow			YES
SharedAction			YES
Workflow			YES
Project/Project...			YES
Project/Project...			YES
Project/Project			YES

2. Expand the *Url* node and select the *files* node under *api* to enable download-upload,

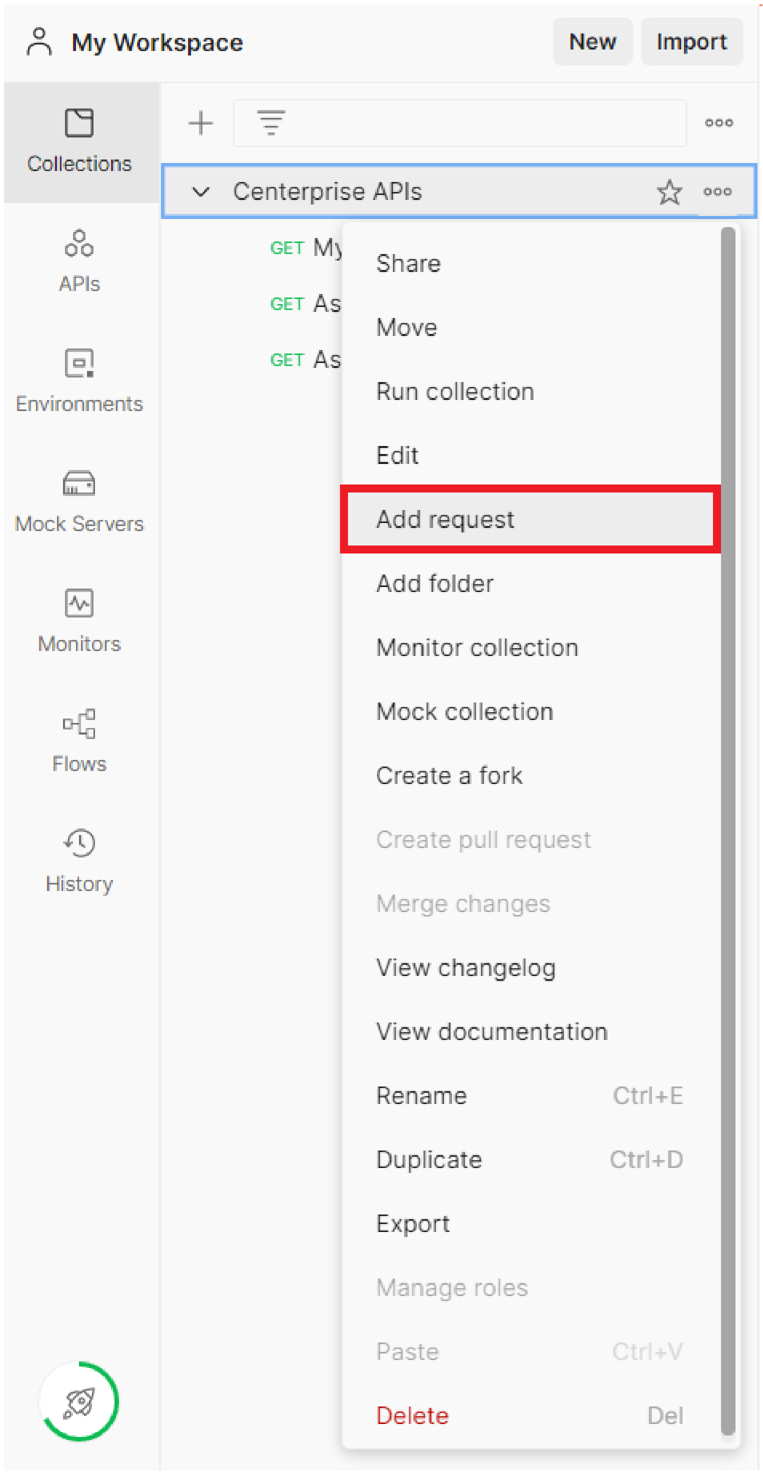


URL	Description	Method	Permission
Project/Project...			YES
Project/Project...			YES
Project/Project...			YES
Project/Project...			YES
Project/Project...			YES
Project/Project...			YES
Project/Project...			YES
Project/Project...			YES
Project/Project...			YES
Project/Project...			YES
Project/Project...			YES
Project/Project...			YES
File			YES
QueryEditor/Qu...			YES
QueryEditor			YES
Formats			YES
api/ServerInfo/...			YES
api/files/{path}	Url://api/files/{p...	GET	YES
api/files	Url://api/files?...	POST	YES

10.8.2 Uploading a File

For our use case, we will be using the Postman API Client from another remote machine.

1. Add a new request to your Postman collection by right-clicking on the collection and selecting *Add Request* from the context menu.

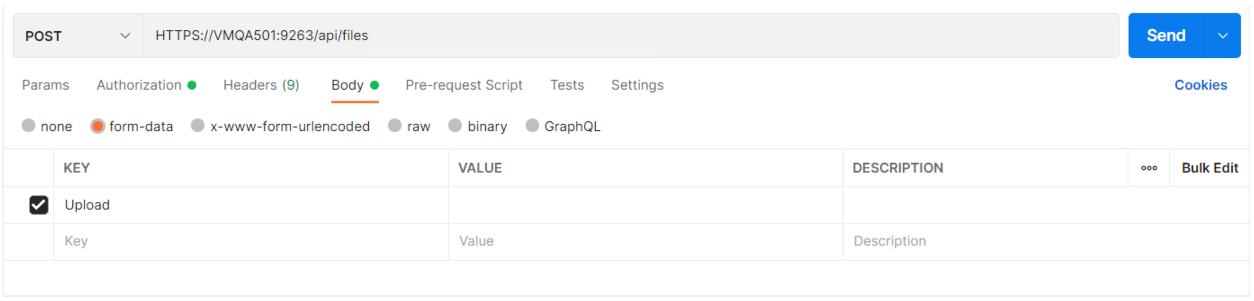


2. Select the HTTP method as *POST,* provide the file API URL deployed at the Astera Server and define a

multipart/form-date request body,

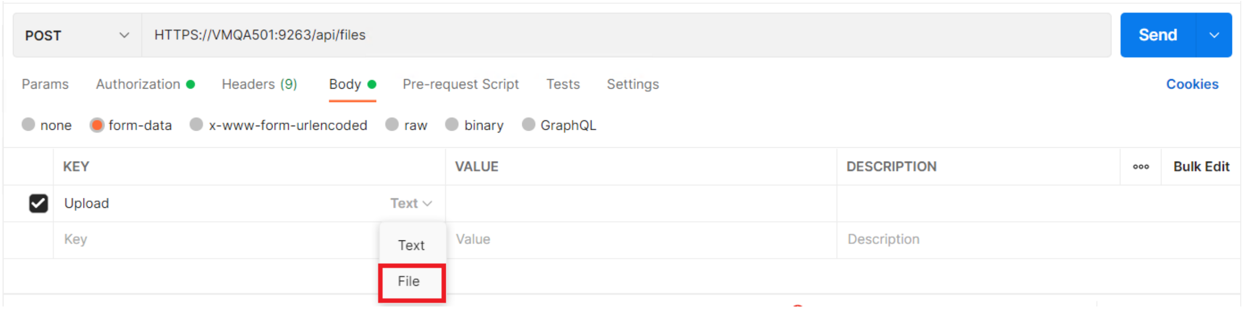
"HTTPS://ServerHostName:9263/api/files"

Note: ServerHostName is referring to the Server Machine Name for Astera API Management.



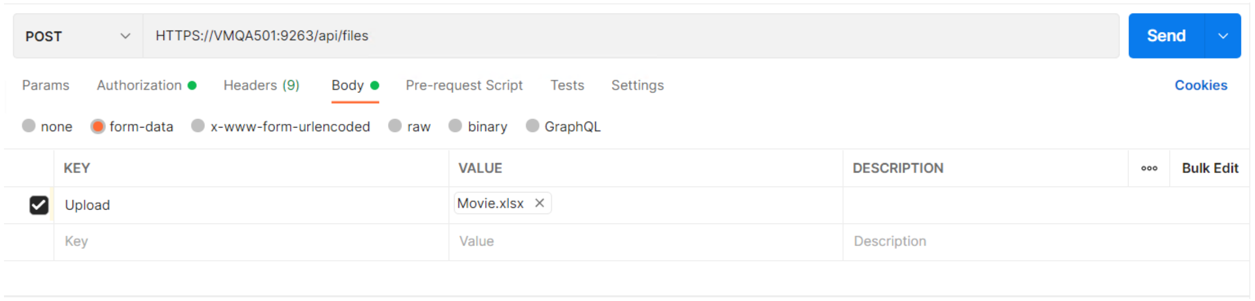
	KEY	VALUE	DESCRIPTION
✓	Upload		
	Key	Value	Description

3. Define a Key of your choice and select its value type as *File,*



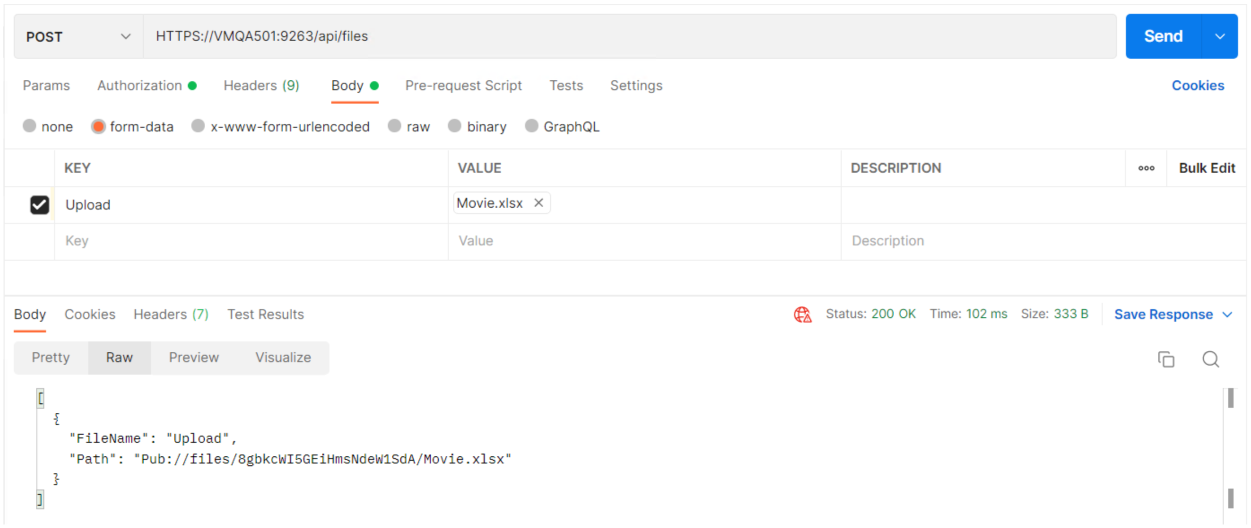
	KEY	VALUE	DESCRIPTION
✓	Upload		
	Key	Value	Description

4. Browse your desired file to the *VALUE*,



	KEY	VALUE	DESCRIPTION
✓	Upload	Movie.xlsx ×	
	Key	Value	Description

5. Click *Send* and the file will be uploaded to the specified file directory on the Astera Server.



POST HTTPS://VMQA501:9263/api/files

Params | Authorization ● | Headers (9) | Body ● | Pre-request Script | Tests | Settings

none [✓ form-data] x-www-form-urlencoded raw binary GraphQL

	KEY	VALUE	DESCRIPTION
✓	Upload	Movie.xlsx	
	Key	Value	Description

Body | Cookies | Headers (7) | Test Results — Status: 200 OK Time: 102 ms Size: 333 B

Pretty | Raw | Preview | Visualize

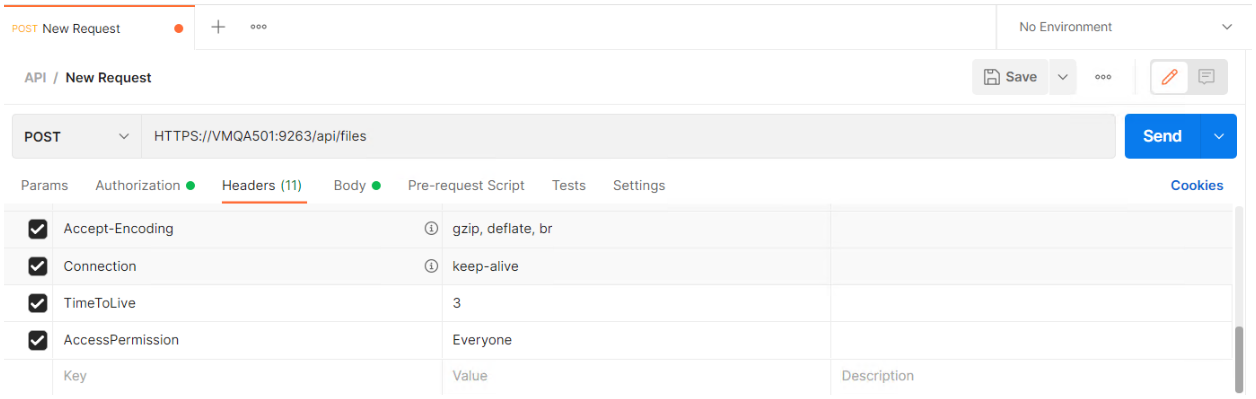
```
[
  {
    "FileName": "Upload",
    "Path": "Pub://files/8gbkcWI5GEiHmsNdeW1SdA/Movie.xlsx"
  }
]
```

As seen from the response above, the respective file has been uploaded.

The FileName is the key defined whereas the path is the relative path of the uploaded file on the Astera Server.

Note: Custom parameters can be defined with the same upload file call as well,

This allows the user to define custom values and overwrite predefined values from Astera API Management. 'Time-ToLive' refers to the time that the file is kept before it expires. 'AccessPermission' defines who has access permission other than the user.



POST New Request — API / New Request

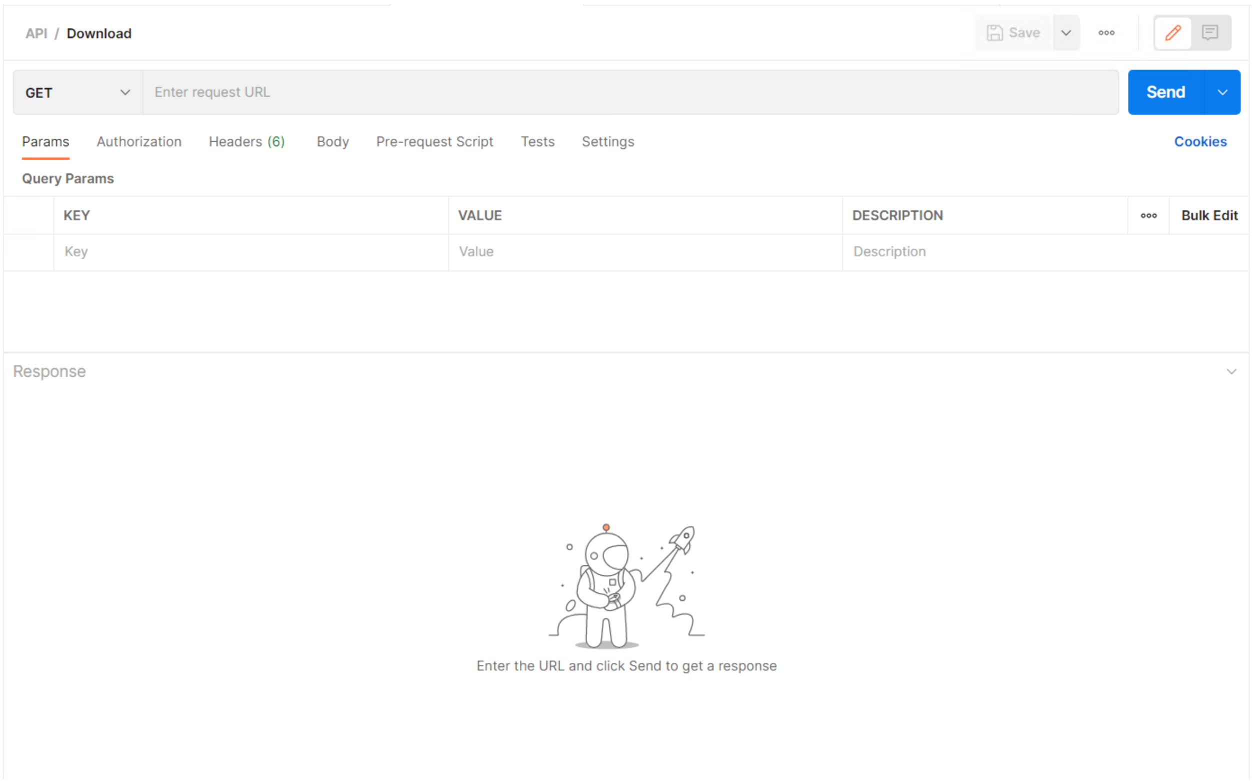
POST HTTPS://VMQA501:9263/api/files

Params | Authorization ● | Headers (11) | Body ● | Pre-request Script | Tests | Settings

✓	Accept-Encoding	gzip, deflate, br	
✓	Connection	keep-alive	
✓	TimeToLive	3	
✓	AccessPermission	Everyone	
	Key	Value	Description

10.8.3 Downloading a File

1. To download a file from the server, create a new request on Postman.



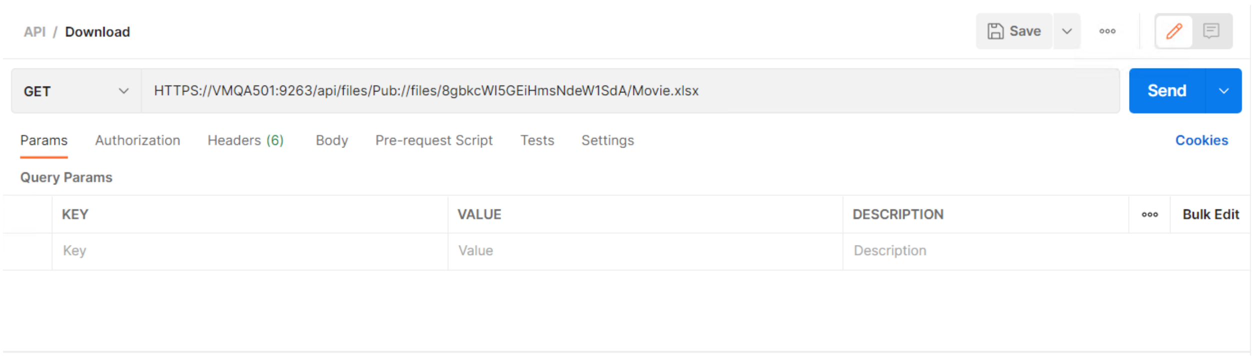
For our use case, we will be downloading the same file that we previously uploaded to the server.

2. Keep the HTTP method as *GET* and enter the request URL.

"HTTPS://ServerHostName:9263/api/files/{filepath}"

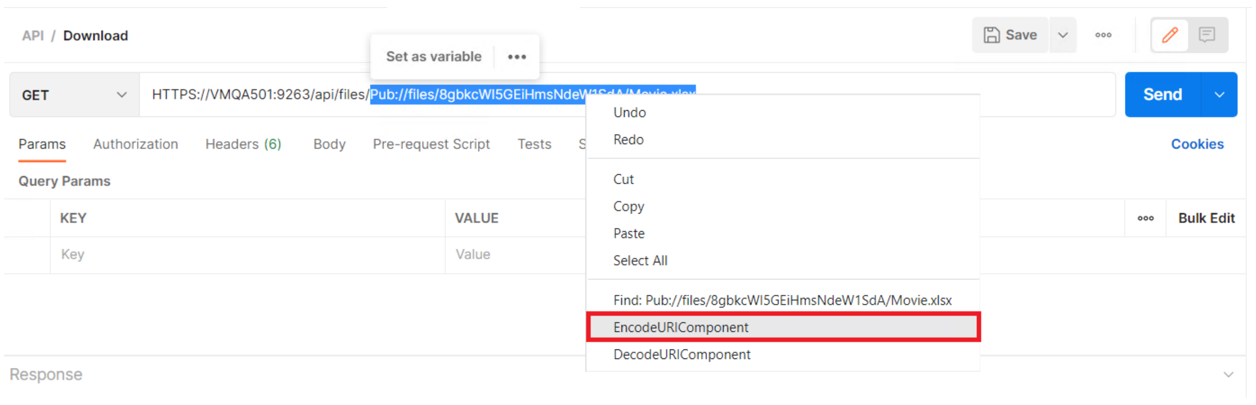
Note: ServerHostName is referring to the Server Machine Name for Astera API Management.

We are using the same file upload API resource as GET for the download function. However, the difference is that we provide the relative path of the uploaded file as a resource.



GET HTTPS://VMQA501:9263/api/files/Pub://files/8gbkcWI5GEiHmsNdeW1SdA/Movie.xlsx

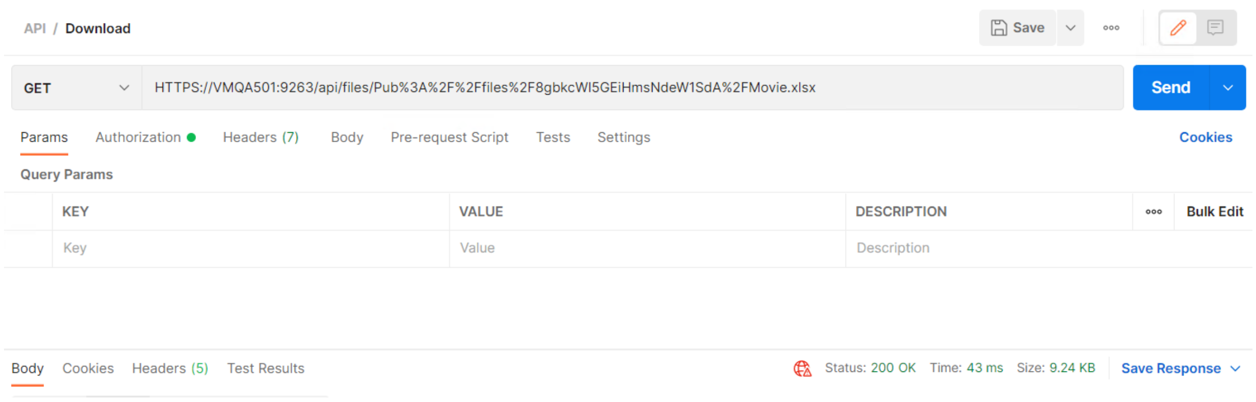
3. For the server to identify the relative file from the request URL, we need to encode the value.



```
GET  HTTPS://VMQA501:9263/api/files/Pub://files/8gbkcWI5GEiHmsNdeW1SdA/Movie.xlsx
```

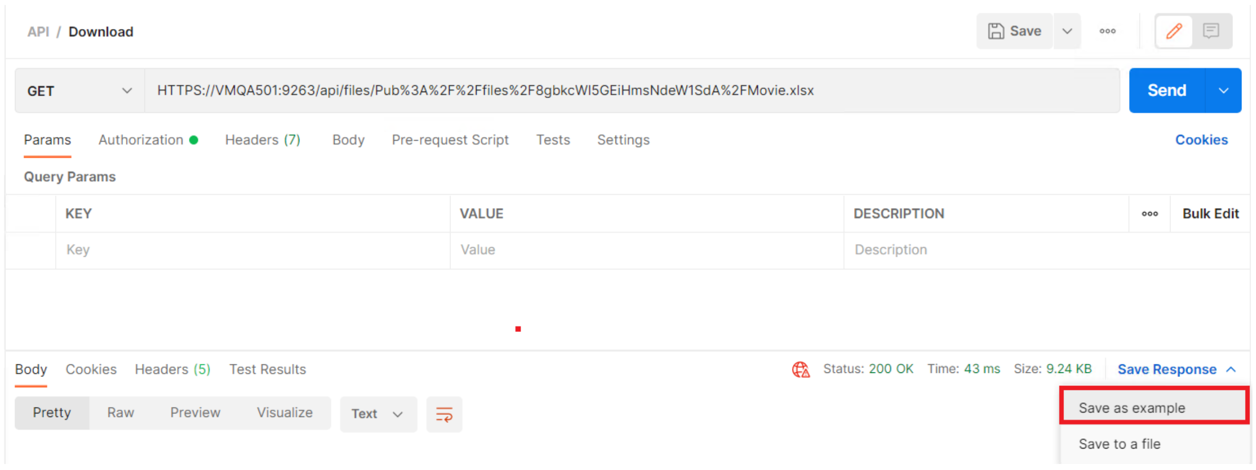
Note: If we send this request from the Astera *API Client,* then the object automatically encodes the resource (file path).

4. Click *Send* and the request will fetch the file’s content in the response body.



```
GET  HTTPS://VMQA501:9263/api/files/Pub%3A%2F%2Ffiles%2F8gbkcWI5GEiHmsNdeW1SdA%2FMovie.xlsx
Status: 200 OK   Time: 43 ms   Size: 9.24 KB
```

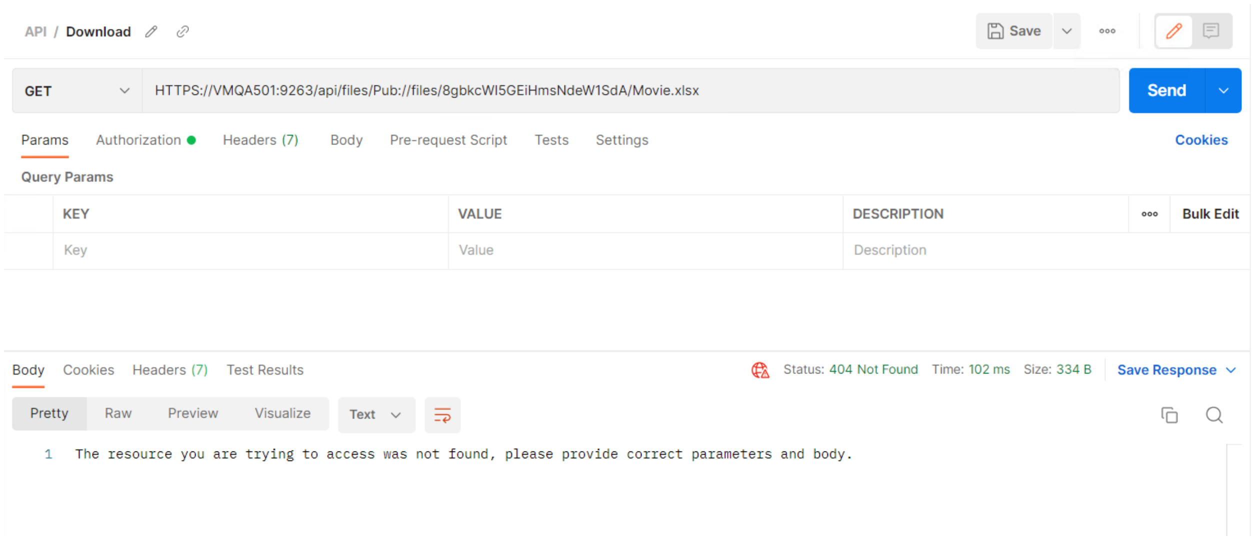
5. The response can then be saved to a file using the following option,



```
GET  HTTPS://VMQA501:9263/api/files/Pub%3A%2F%2Ffiles%2F8gbkcWI5GEiHmsNdeW1SdA%2FMovie.xlsx
Status: 200 OK   Time: 43 ms   Size: 9.24 KB
```

This is how a file can be downloaded from the Astera API Management server.

Note: On providing an invalid, wrong, non-encoded, and none-existing file’s file, the request will result in a ‘404 Not Found’ error with an appropriate message,



HTTPS://VMQA501:9263/api/files/Pub://files/8gbkcWI5GEiHmsNdeW1SdA/Movie.xlsx

KEY	VALUE	DESCRIPTION
Key	Value	Description

Status: 404 Not Found Time: 102 ms Size: 334 B

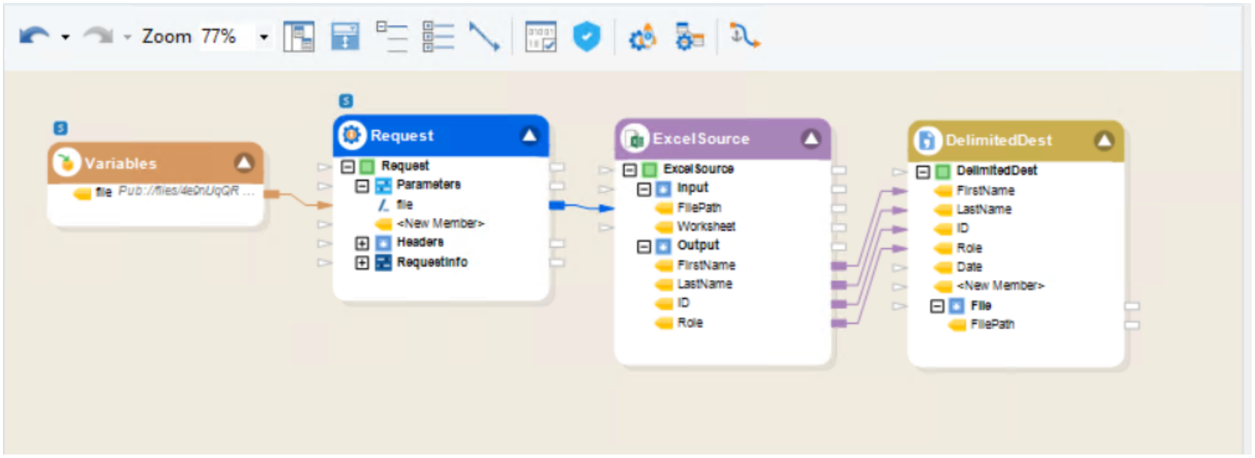
```
The resource you are trying to access was not found, please provide correct parameters and body.
```

10.8.4 Generating Downloadable path for files through Astera API Management

Astera API Management offers the user the ability to generate the downloadable path for any destination file using the *Download Path Generator* object.

This functionality can be seen within the scope of an API flow.

For our use case, we have the following API flow,



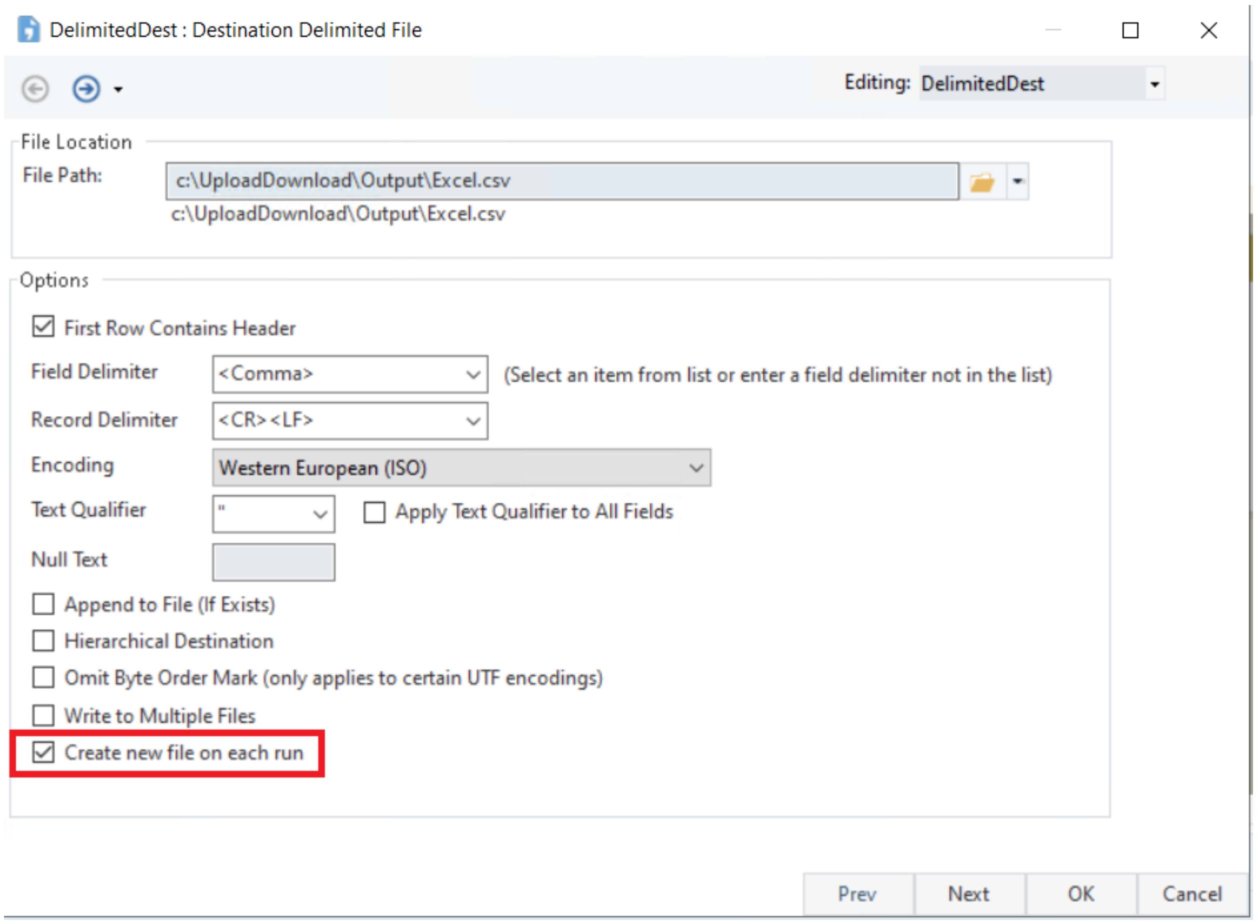
In the above flow, we can see that a file path has been given to an API *Request* object through a *Variables* object. We are trying to consume the uploaded file here using its relative path.

The flow then maps the request object to an *Excel Source* object, used as a transformation, and writes the records to a *Delimited Destination* object.

The *Delimited Destination* object has an additional *File* node. This enables us to create a new destination file on each run. Each file is created by appending a unique ID to the destination path given in the object, eliminating the chances of overwriting an existing file. This *FilePath* field outputs the unique path generated at runtime.

This can be enabled for any destination object.

1. Right-click on the *Delimited Destination* and select *Properties.* Next, check the *Create new file on each run* option. This will add the unique file path node to the destination object.



DelimitedDest : Destination Delimited File

Editing: DelimitedDest

File Location

File Path: c:\UploadDownload\Output\Excel.csv

c:\UploadDownload\Output\Excel.csv

Options

☑ First Row Contains Header

Field Delimiter: <Comma> (Select an item from list or enter a field delimiter not in the list)

Record Delimiter: <CR><LF>

Encoding: Western European (ISO)

Text Qualifier: " ☐ Apply Text Qualifier to All Fields

Null Text:

☐ Append to File (If Exists)

☐ Hierarchical Destination

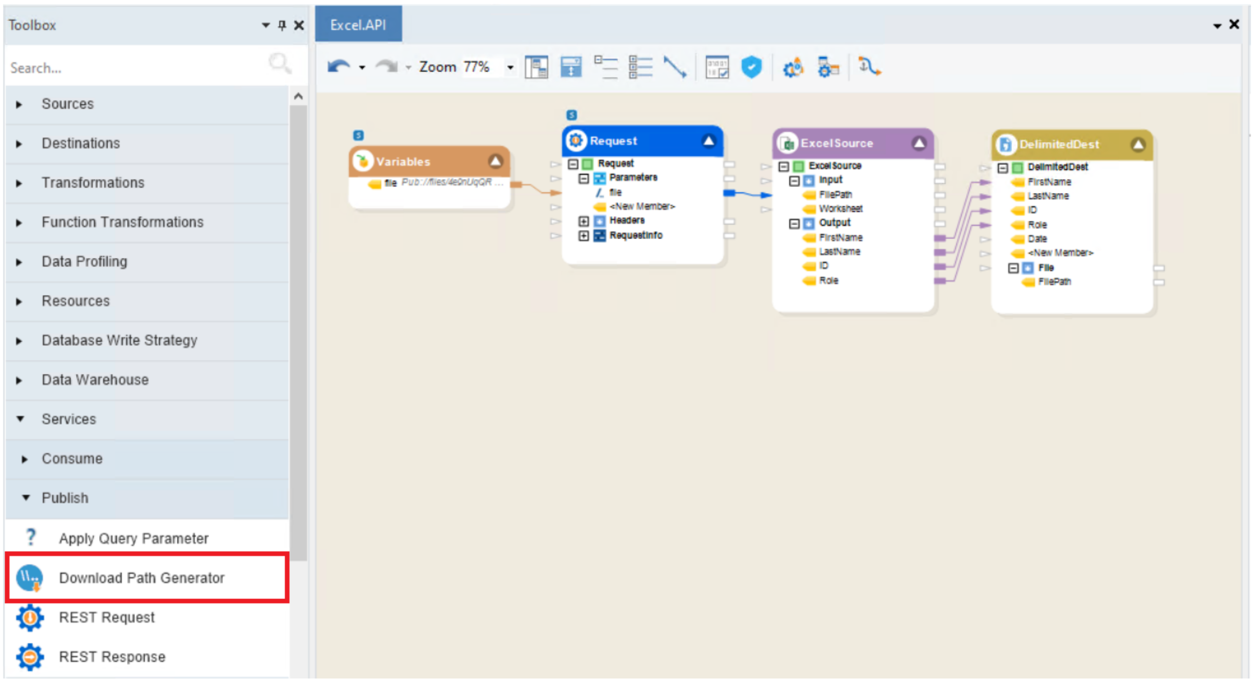
☐ Omit Byte Order Mark (only applies to certain UTF encodings)

☐ Write to Multiple Files

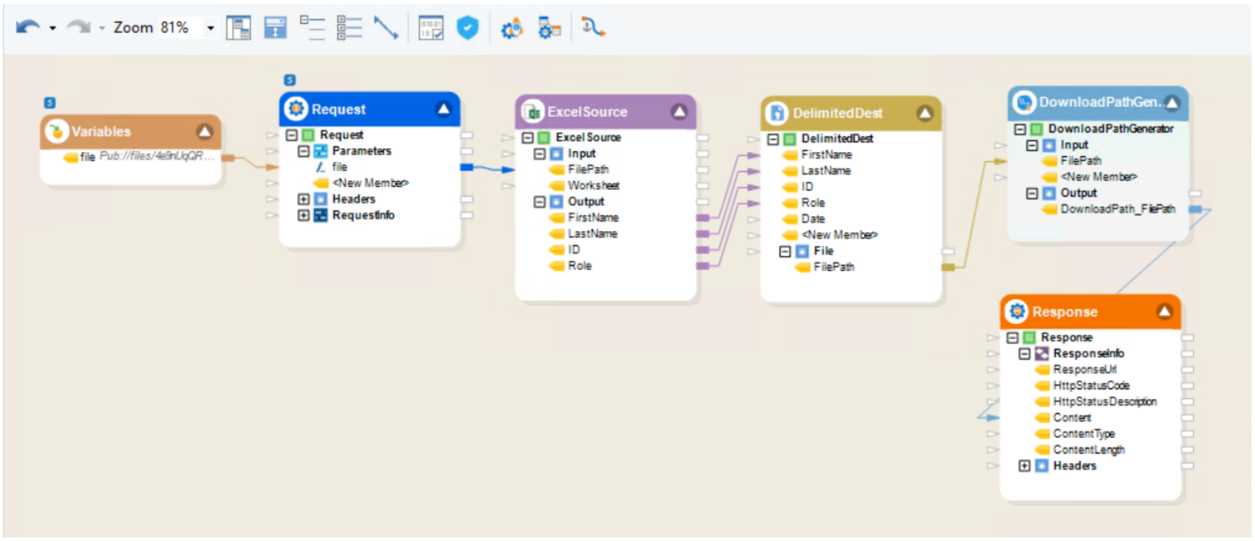
☑ Create new file on each run

2. Drag and drop a *Download Path Generator* object from the toolbox onto the API flow.

In order for us to obtain a downloadable path for our file, we require the use of this object.



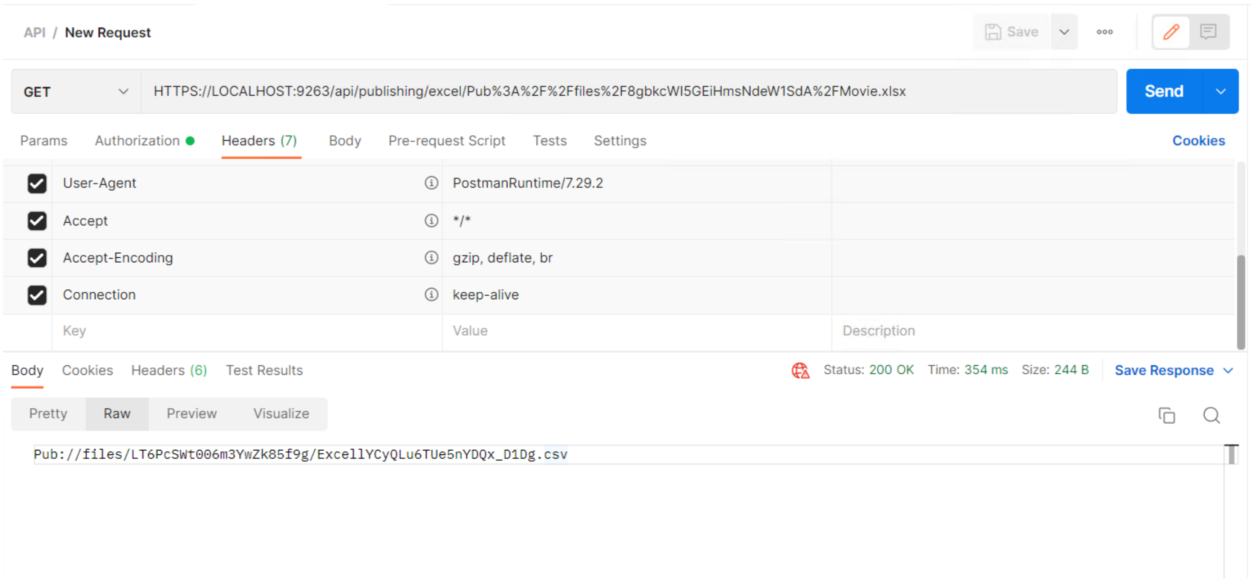
3. Map the input using the File Path field from the *Delimited Destination* object and map the output towards the API *Response* object.



This downloadable file path can now be used in further applications where the file is required.

Note: The *Download Path Generator* object cannot be previewed at design time because the downloadable path is generated at run-time.

Below, we can see the request being sent from Postman and the user receiving the downloadable path in the response.


API / New Request

GET HTTPS://LOCALHOST:9263/api/publishing/excel/Pub%3A%2F%2Ffiles%2F8gbkcWI5GEiHmsNdeW1SdA%2FMovie.xlsx

Params | Authorization | Headers (7) | Body | Pre-request Script | Tests | Settings

	Key	Value	Description
✓	User-Agent	PostmanRuntime/7.29.2	
✓	Accept	*/*	
✓	Accept-Encoding	gzip, deflate, br	
✓	Connection	keep-alive	

Body | Cookies | Headers (6) | Test Results — Status: 200 OK Time: 354 ms Size: 244 B

Pretty | Raw | Preview | Visualize

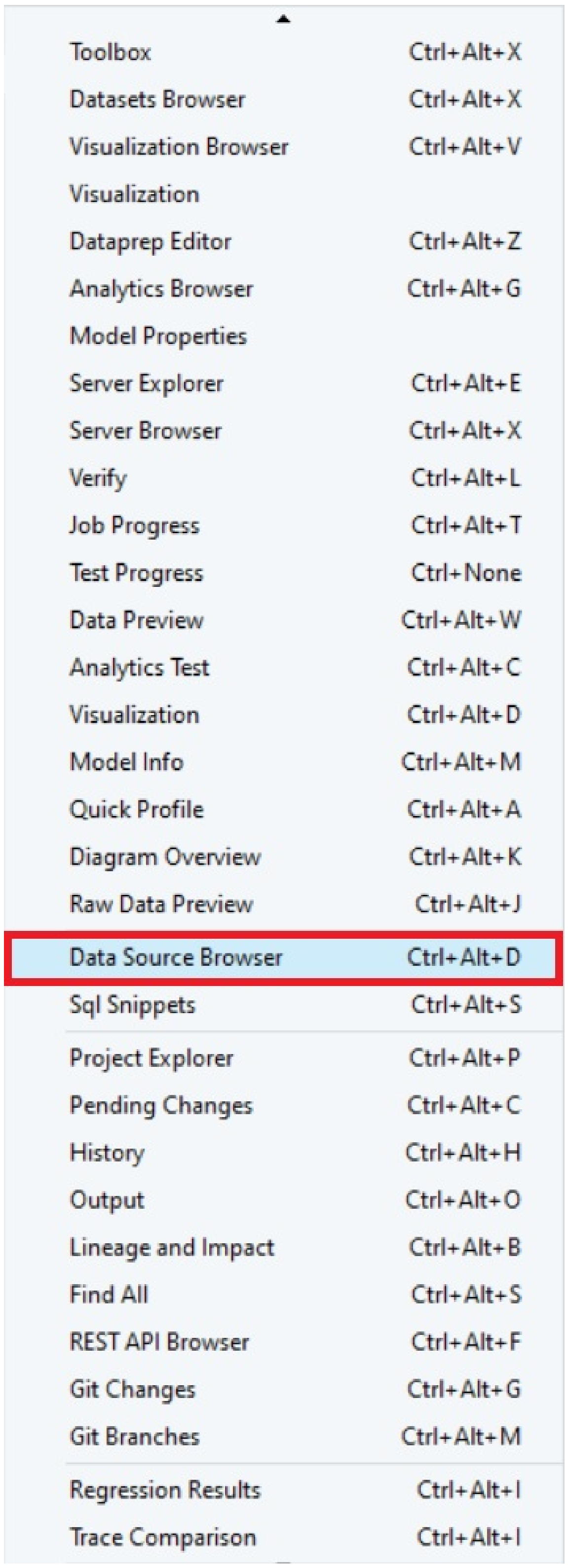
```
Pub://files/LT6PcSWt006m3YwZk85f9g/ExcellYCyQLu6TUe5nYDQx_D1Dg.csv
```

This concludes our document on enabling and using file download/upload in Astera API Management.

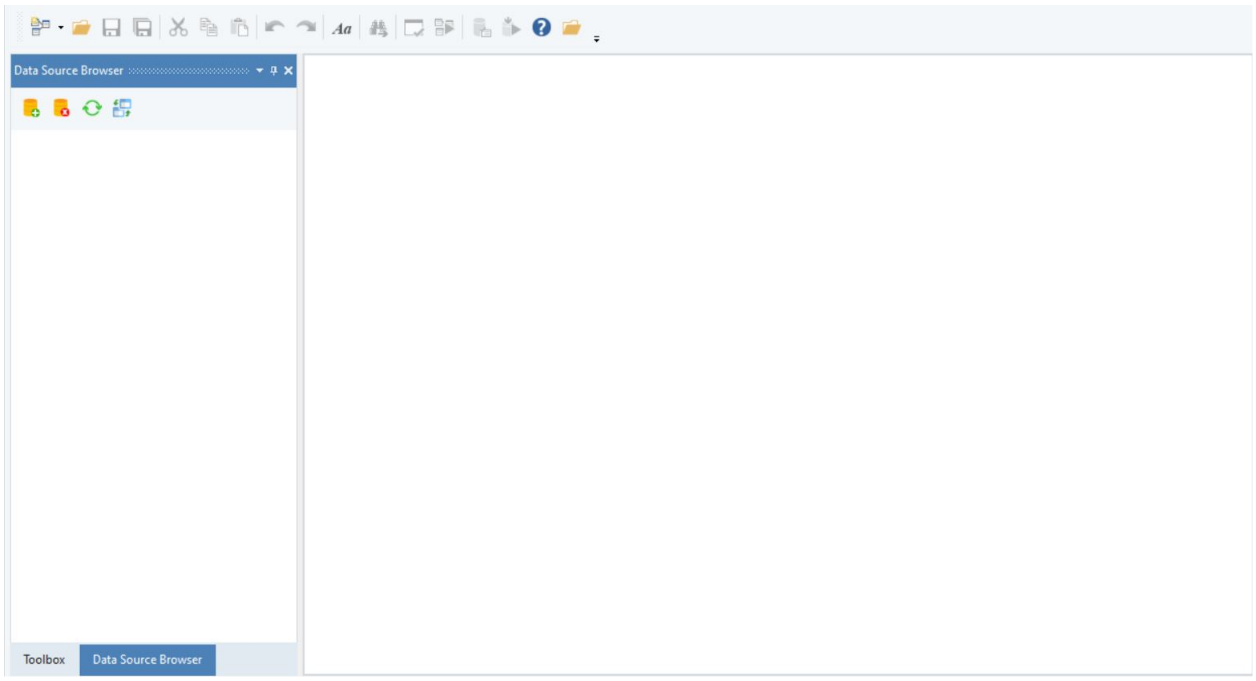
10.9 Database CRUD APIs Auto-Generation

Users can auto-generate CRUD API endpoints for any database using the Data Source Browser. CRUD APIs are meant for Create-Retrieve-Update-Delete operations on the database table records.

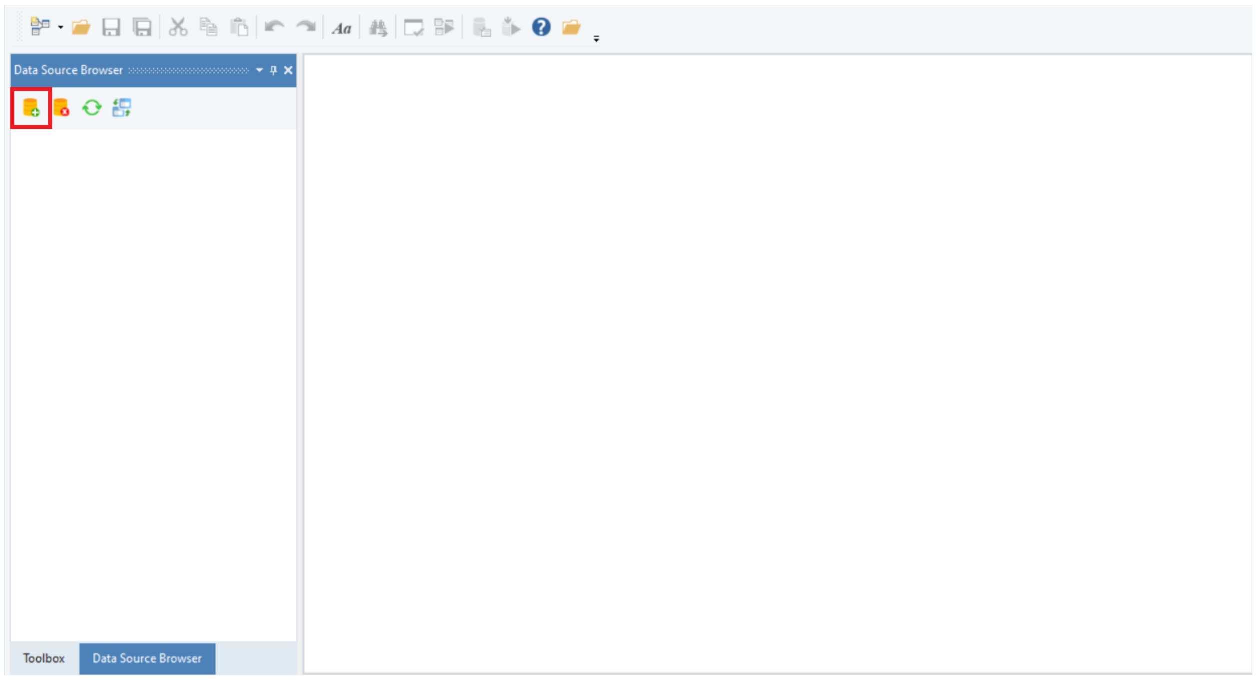
1. Click on *View* in the main menu bar and select *Data Source Browser* from the drop-down menu.


Toolbox Ctrl+Alt+X
Datasets Browser Ctrl+Alt+X
Visualization Browser Ctrl+Alt+V
Visualization
Dataprep Editor Ctrl+Alt+Z
Analytics Browser Ctrl+Alt+G
Model Properties
Server Explorer Ctrl+Alt+E
Server Browser Ctrl+Alt+X
Verify Ctrl+Alt+L
Job Progress Ctrl+Alt+T
Test Progress Ctrl+None
Data Preview Ctrl+Alt+W
Analytics Test Ctrl+Alt+C
Visualization Ctrl+Alt+D
Model Info Ctrl+Alt+M
Quick Profile Ctrl+Alt+A
Diagram Overview Ctrl+Alt+K
Raw Data Preview Ctrl+Alt+J
Data Source Browser Ctrl+Alt+D
Sql Snippets Ctrl+Alt+S
Project Explorer Ctrl+Alt+P
Pending Changes Ctrl+Alt+C
History Ctrl+Alt+H
Output Ctrl+Alt+O
Lineage and Impact Ctrl+Alt+B
Find All Ctrl+Alt+S
REST API Browser Ctrl+Alt+F
Git Changes Ctrl+Alt+G
Git Branches Ctrl+Alt+M
Regression Results Ctrl+Alt+I
Trace Comparison Ctrl+Alt+I

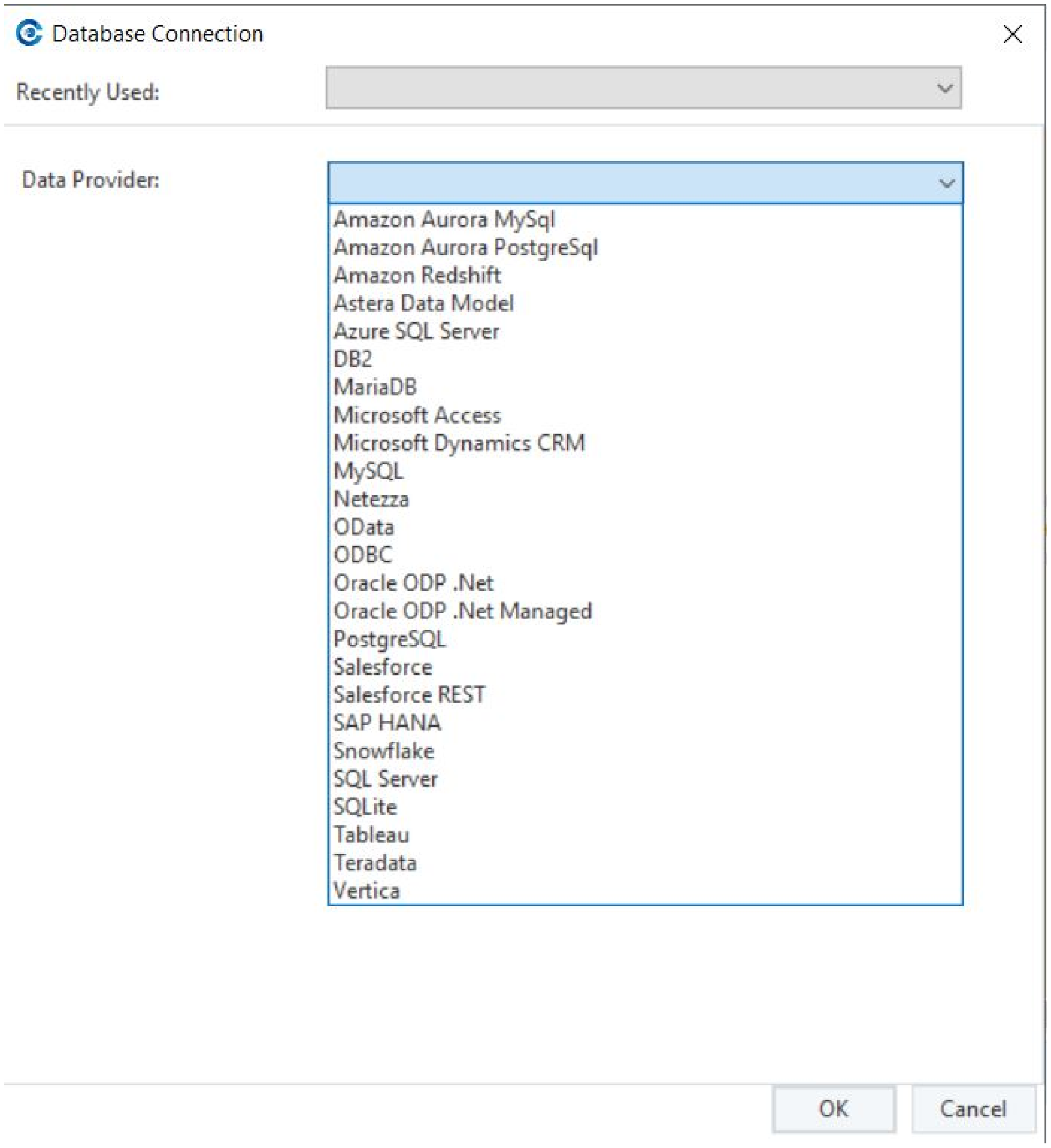
This will open the Data Source Browser.



2. Add a new database server by selecting the *Add Database Server* option.



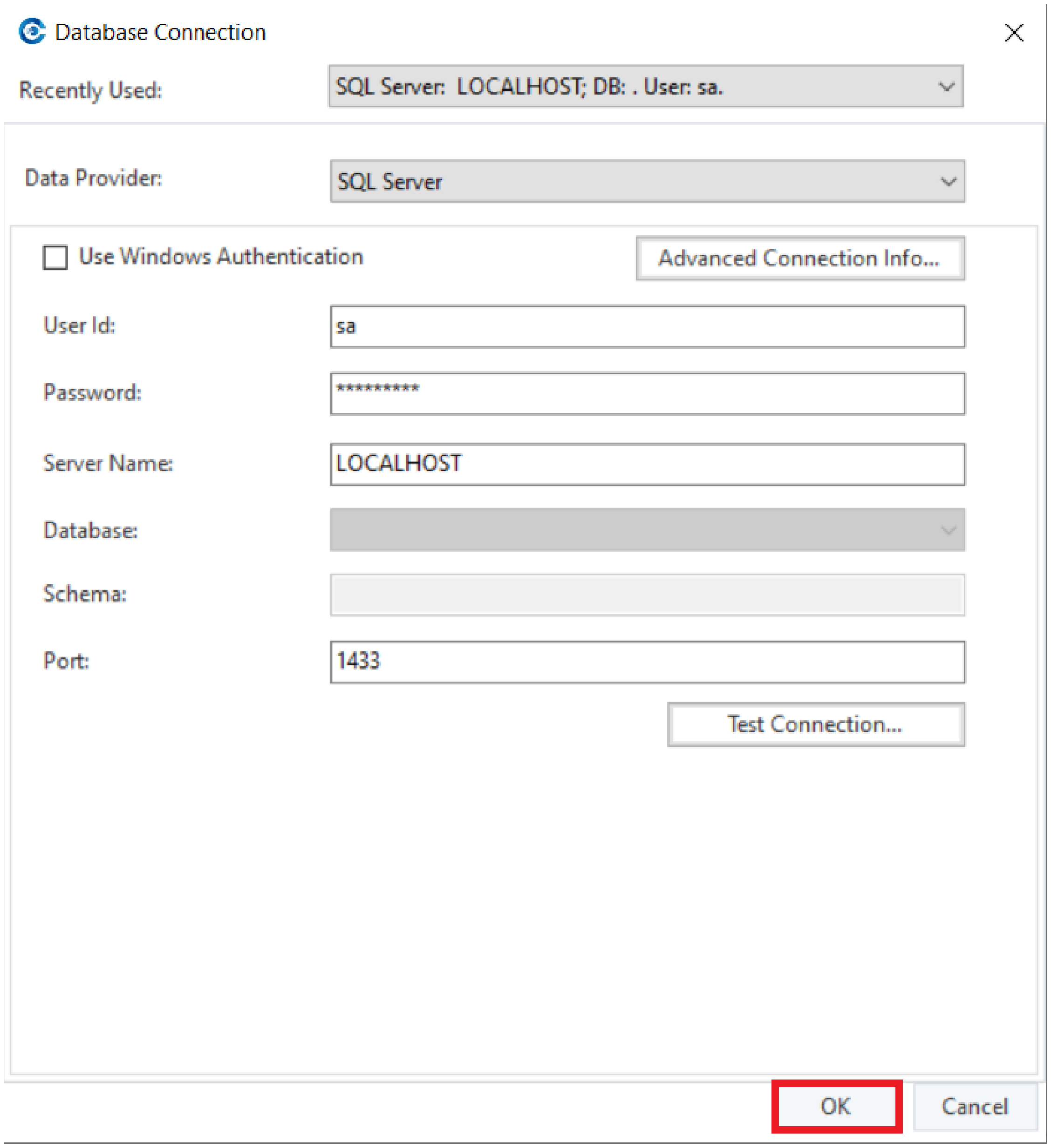
This will open a configuration window to define a database connection. A database server can be configured from any of the listed providers.



Data Provider:

Amazon Aurora MySql
Amazon Aurora PostgreSql
Amazon Redshift
Astera Data Model
Azure SQL Server
DB2
MariaDB
Microsoft Access
Microsoft Dynamics CRM
MySQL
Netezza
OData
ODBC
Oracle ODP .Net
Oracle ODP .Net Managed
PostgreSQL
Salesforce
Salesforce REST
SAP HANA
Snowflake
SQL Server
SQLite
Tableau
Teradata
Vertica

3. Add all the essential details to configure the database server connection and click *OK*.



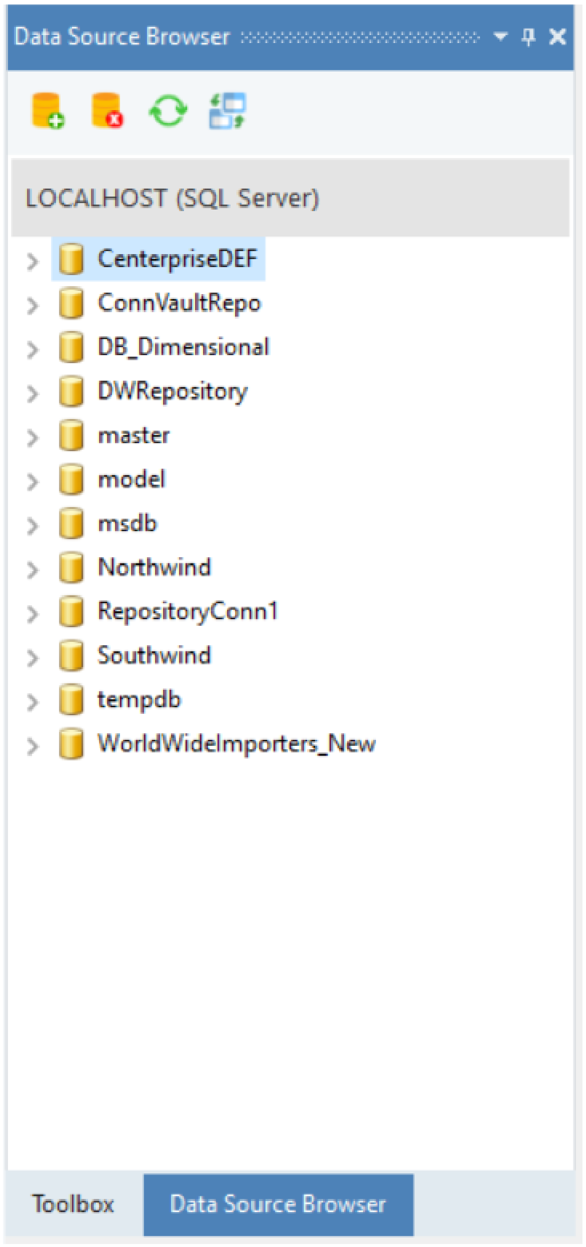
Recently Used: SQL Server: LOCALHOST; DB: . User: sa.

Data Provider: SQL Server

☐ Use Windows Authentication

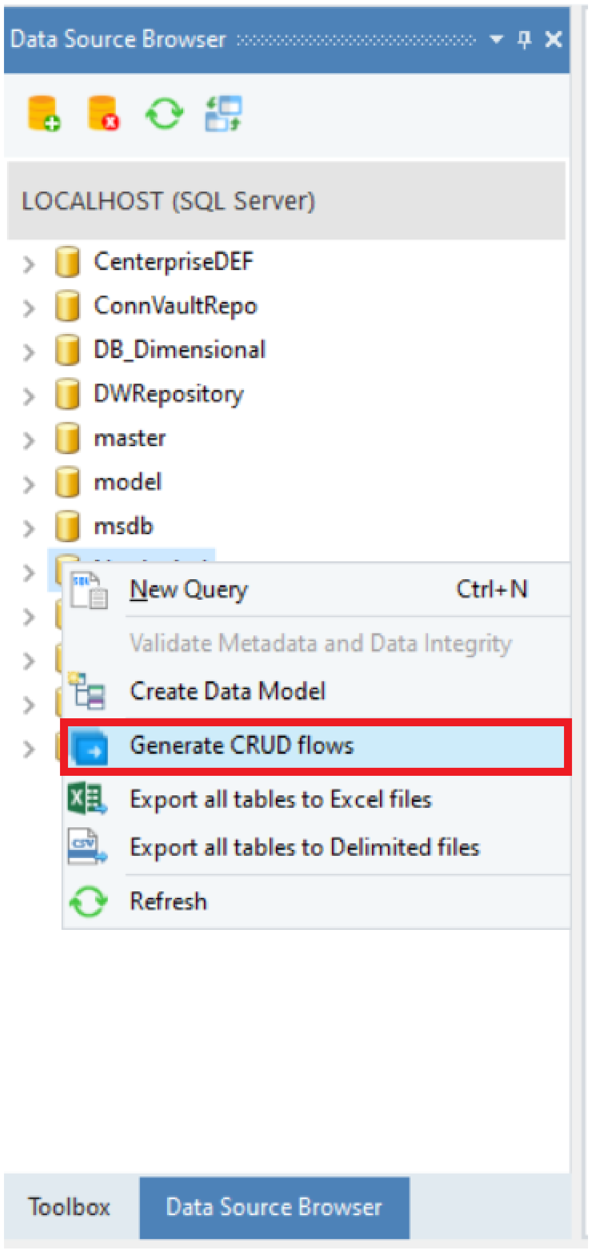
User Id:	sa
Password:	**********
Server Name:	LOCALHOST
Database:	
Schema:	
Port:	1433

Now, the Data Source Browser will be populated with all the databases from the connected server.



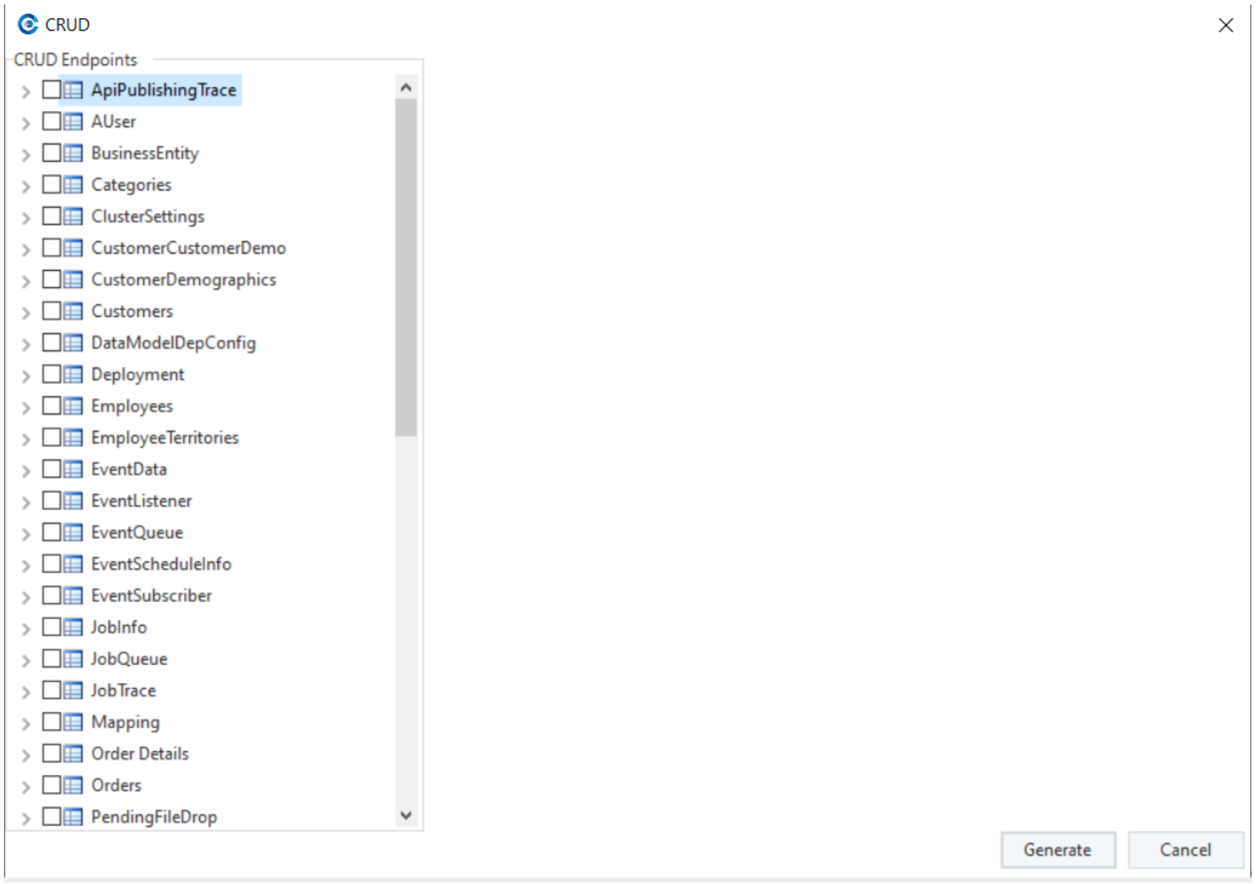
4. Right-click on any database and select *Generate CRUD flows* from the context menu.

Note: It is necessary for a project to be open when CRUD API flows are generated, since they are added under a CRUD folder created in the project.



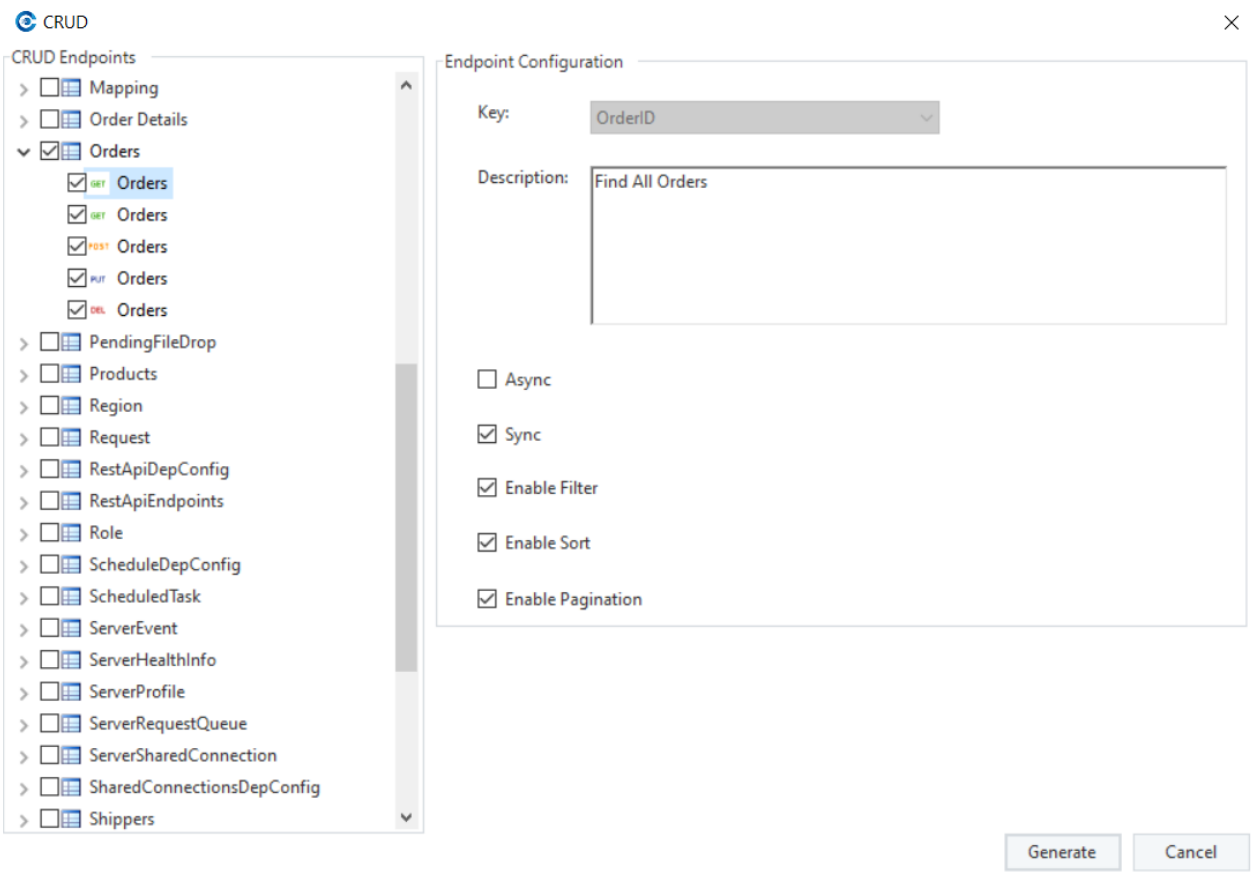
This will open a new window.

Here, you can select the tables and the respective CRUD operations to generate API flows.



For our use case, we will be selecting the *Orders* table. The following operations are available for each table:

1. *Find all records* – A Get method that fetches all the records
2. *Get record by ID* – A Get method along with a path parameter for a key that fetches the records based on the key.
3. *Create a new record* – Selecting this creates a new record.
4. *Update a record by ID* – Selecting this option lets the user update a record by ID
5. *Delete a record by ID* – Selecting this option lets the user delete a record by ID.



CRUD

CRUD Endpoints

- [ ] Mapping
- [ ] Order Details
- [✓] Orders
  - [✓] GET Orders
  - [✓] GET Orders
  - [✓] POST Orders
  - [✓] PUT Orders
  - [✓] DEL Orders
- [ ] PendingFileDrop
- [ ] Products
- [ ] Region
- [ ] Request
- [ ] RestApiDepConfig
- [ ] RestApiEndpoints
- [ ] Role
- [ ] ScheduleDepConfig
- [ ] ScheduledTask
- [ ] ServerEvent
- [ ] ServerHealthInfo
- [ ] ServerProfile
- [ ] ServerRequestQueue
- [ ] ServerSharedConnection
- [ ] SharedConnectionsDepConfig
- [ ] Shippers

Endpoint Configuration

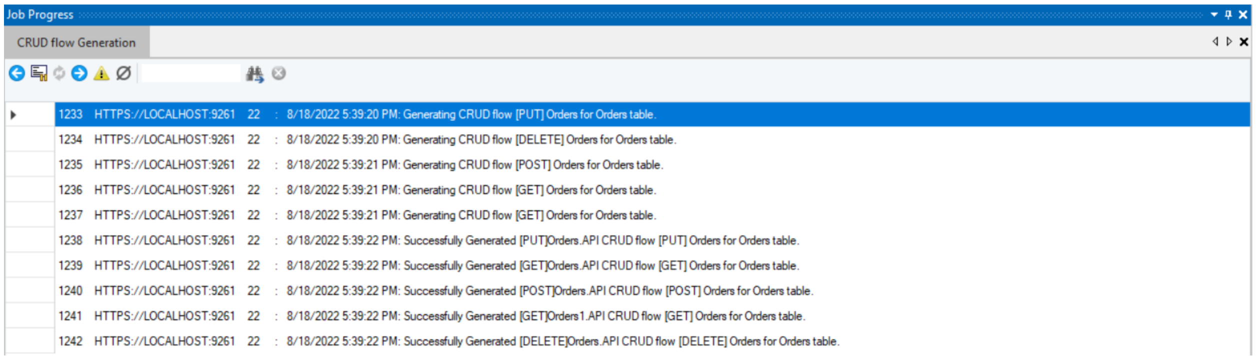
Key: OrderID

Description: Find All Orders

- [ ] Async
- [✓] Sync
- [✓] Enable Filter
- [✓] Enable Sort
- [✓] Enable Pagination

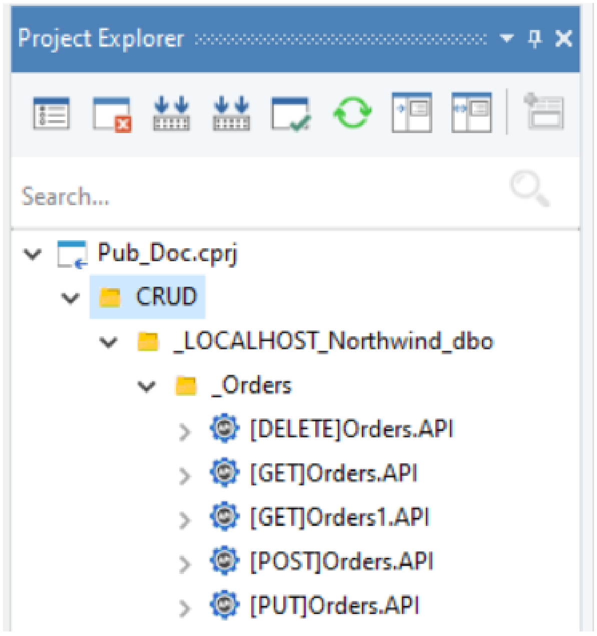
The user can even select configurations inside each endpoint, whether they want to enable sort or filter, or whether their execution type is Synchronous or Asynchronous.

5. Once done, click *Generate* and the CRUD flows will be generated.



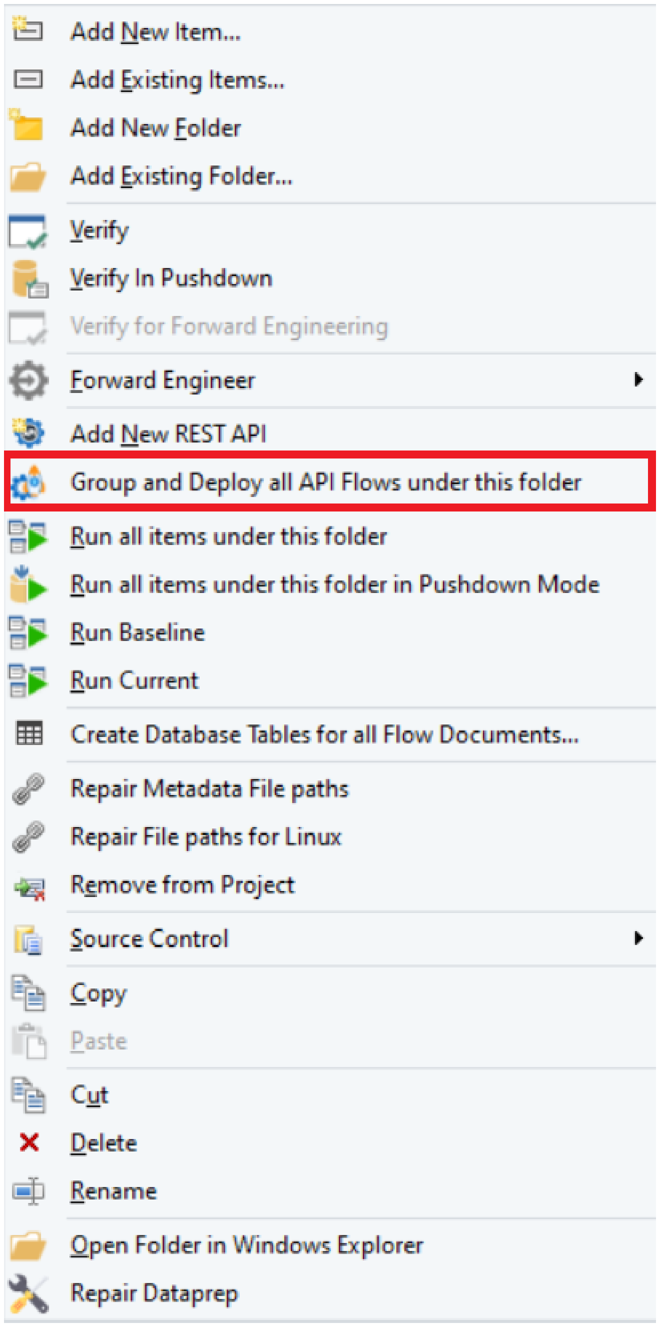
```
1233  HTTPS://LOCALHOST:9261  22  :  8/18/2022 5:39:20 PM: Generating CRUD flow [PUT] Orders for Orders table.
1234  HTTPS://LOCALHOST:9261  22  :  8/18/2022 5:39:20 PM: Generating CRUD flow [DELETE] Orders for Orders table.
1235  HTTPS://LOCALHOST:9261  22  :  8/18/2022 5:39:21 PM: Generating CRUD flow [POST] Orders for Orders table.
1236  HTTPS://LOCALHOST:9261  22  :  8/18/2022 5:39:21 PM: Generating CRUD flow [GET] Orders for Orders table.
1237  HTTPS://LOCALHOST:9261  22  :  8/18/2022 5:39:21 PM: Generating CRUD flow [GET] Orders for Orders table.
1238  HTTPS://LOCALHOST:9261  22  :  8/18/2022 5:39:22 PM: Successfully Generated [PUT]Orders.API CRUD flow [PUT] Orders for Orders table.
1239  HTTPS://LOCALHOST:9261  22  :  8/18/2022 5:39:22 PM: Successfully Generated [GET]Orders.API CRUD flow [GET] Orders for Orders table.
1240  HTTPS://LOCALHOST:9261  22  :  8/18/2022 5:39:22 PM: Successfully Generated [POST]Orders.API CRUD flow [POST] Orders for Orders table.
1241  HTTPS://LOCALHOST:9261  22  :  8/18/2022 5:39:22 PM: Successfully Generated [GET]Orders1.API CRUD flow [GET] Orders for Orders table.
1242  HTTPS://LOCALHOST:9261  22  :  8/18/2022 5:39:22 PM: Successfully Generated [DELETE]Orders.API CRUD flow [DELETE] Orders for Orders table.
```

You can then view the API endpoints in the Project Explorer.

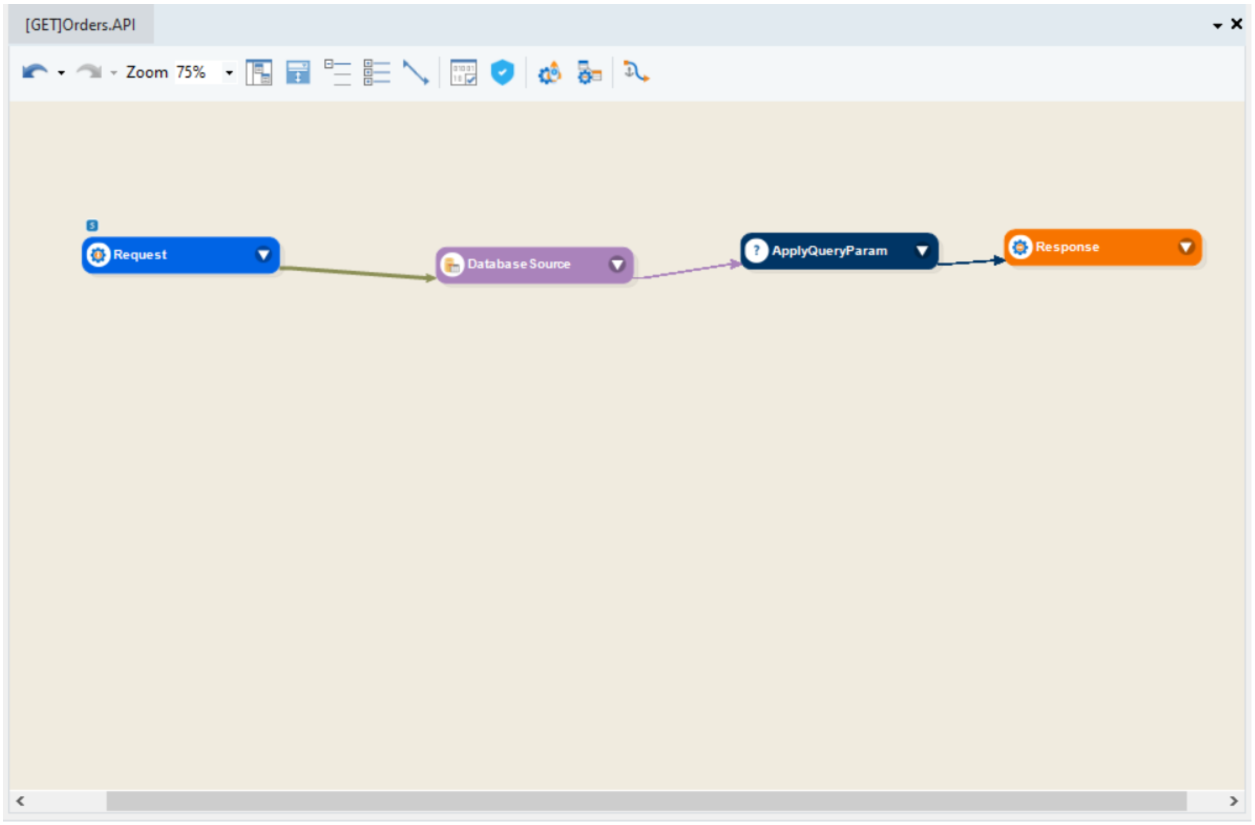


Pub_Doc.cprj
CRUD
_LOCALHOST_Northwind_dbo
_Orders
[DELETE]Orders.API
[GET]Orders.API
[GET]Orders1.API
[POST]Orders.API
[PUT]Orders.API

6. Now, you can directly group and deploy with a single click,


Add New Item...
Add Existing Items...
Add New Folder
Add Existing Folder...
Verify
Verify In Pushdown
Verify for Forward Engineering
Forward Engineer
Add New REST API
Group and Deploy all API Flows under this folder
Run all items under this folder
Run all items under this folder in Pushdown Mode
Run Baseline
Run Current
Create Database Tables for all Flow Documents...
Repair Metadata File paths
Repair File paths for Linux
Remove from Project
Source Control
Copy
Paste
Cut
Delete
Rename
Open Folder in Windows Explorer
Repair Dataprep

or open any of the API flows to see pre-configured API flows or make any changes.



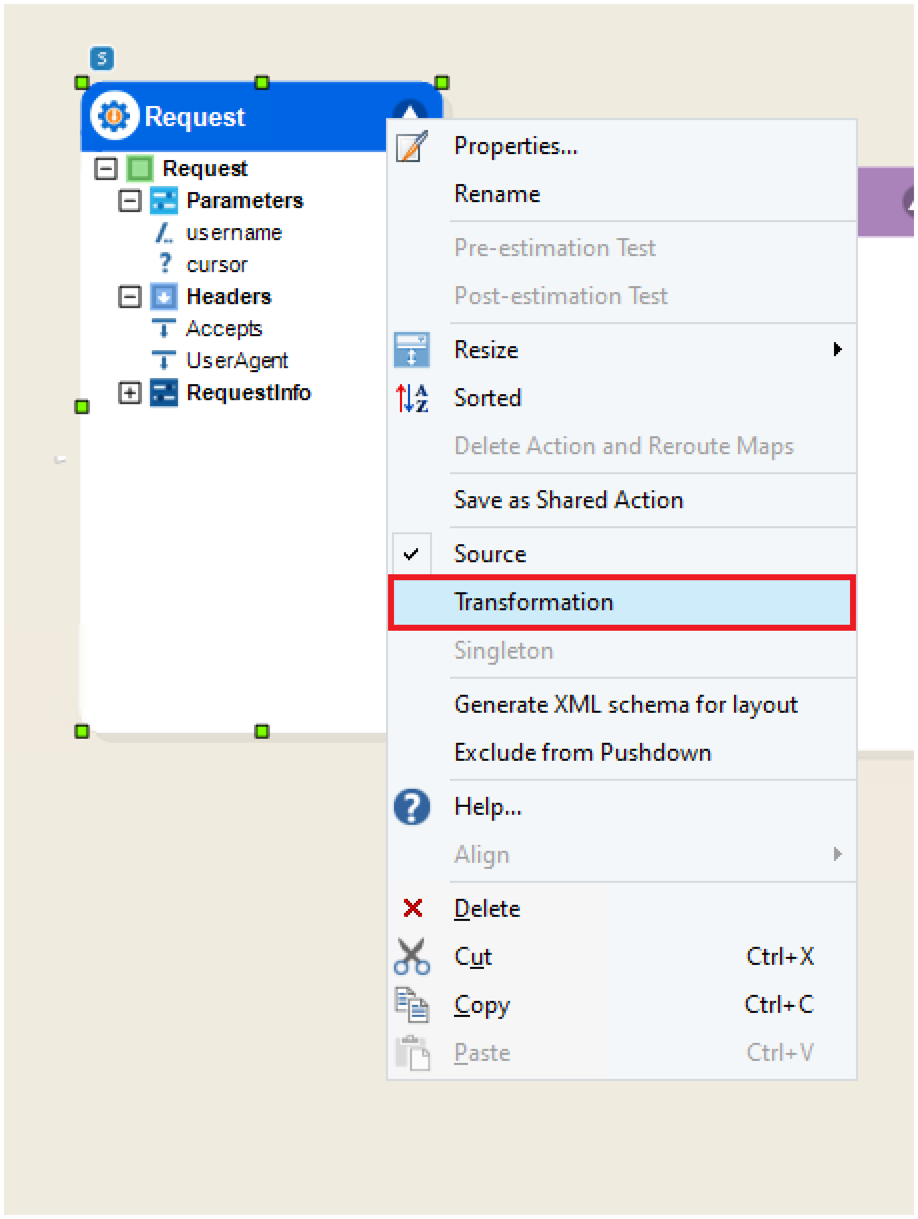
This concludes the working of the Database APIs CRUD auto-generation in Astera Centerprise.

10.10 Pre-deployment Testing and Verification of API flows

10.10.1 Instant Data Preview

When designing an API flow, users can benefit from the functionality to instantly preview and verify the input and output data for any action in the flow. Carrying out data-driven testing of the API functionality at design time helps identify any possible hindrances sooner.

To define test values for the API flow, the Request object must be set as a transformation and any test data can be mapped to it. Right-click on the Request object and select Transformation.



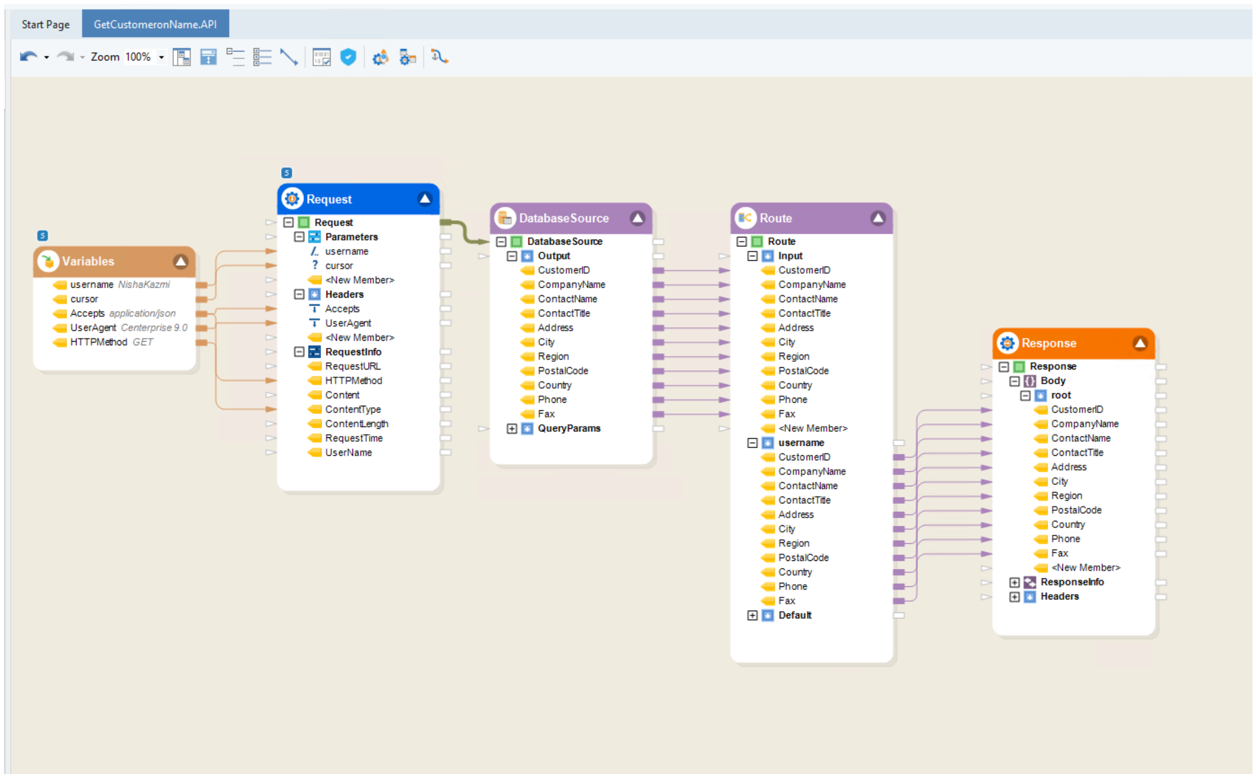
Since the request object is a Singleton object, only the first record is processed through the flow. This behavior compliments the runtime behavior of a single API call and provides ease in previewing the respective results. Let's preview the Request object to observe this.

10.10.2 Raw Request/Response Preview

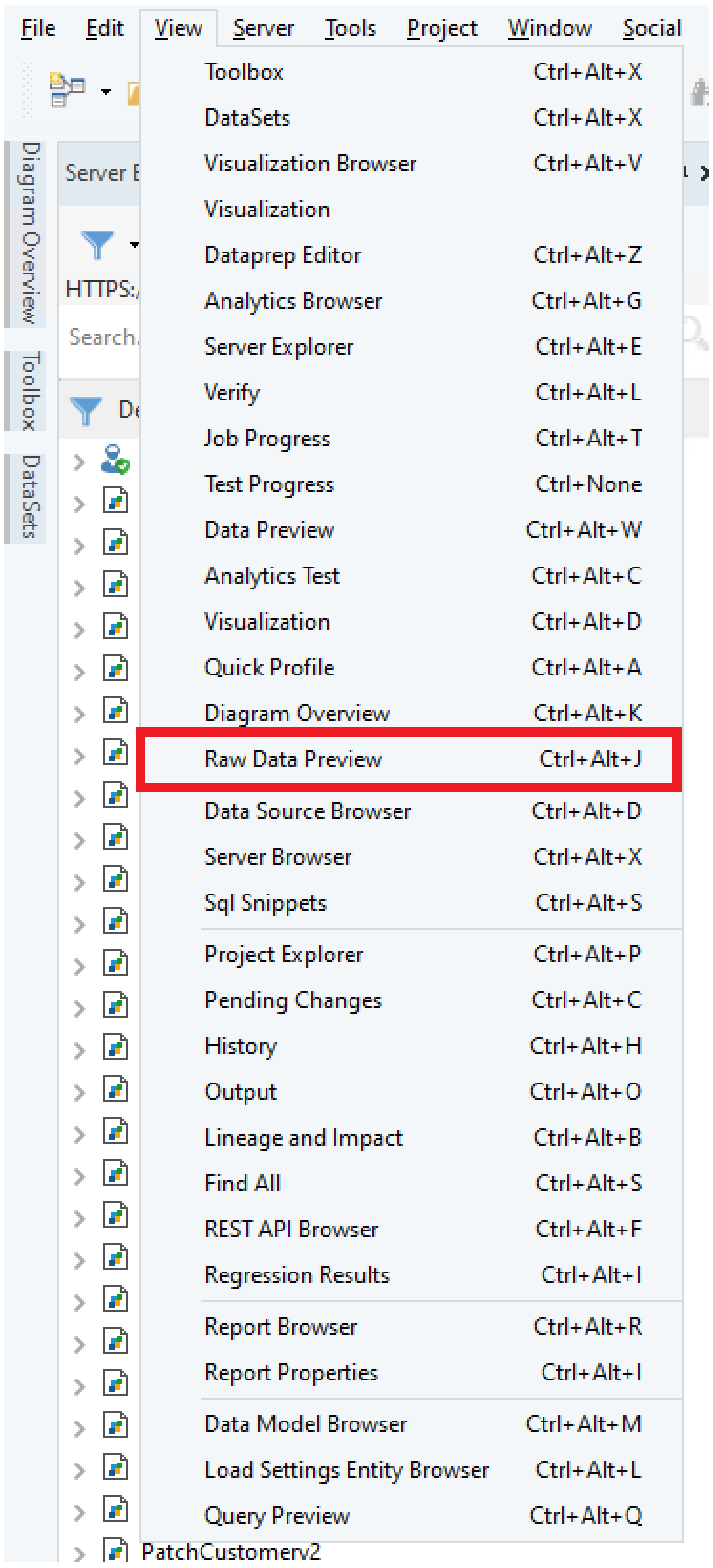
The *Preview Raw* option alternatively allows the users to view the API request and response in a raw unformatted form. This option is useful as it not only displays the data as a raw HTTPS packet but also gives us the benefit of copying, saving, or sharing the JSON body of both the request and response.

Let's take an example API flow and see how we can preview the raw request in Centerprise,

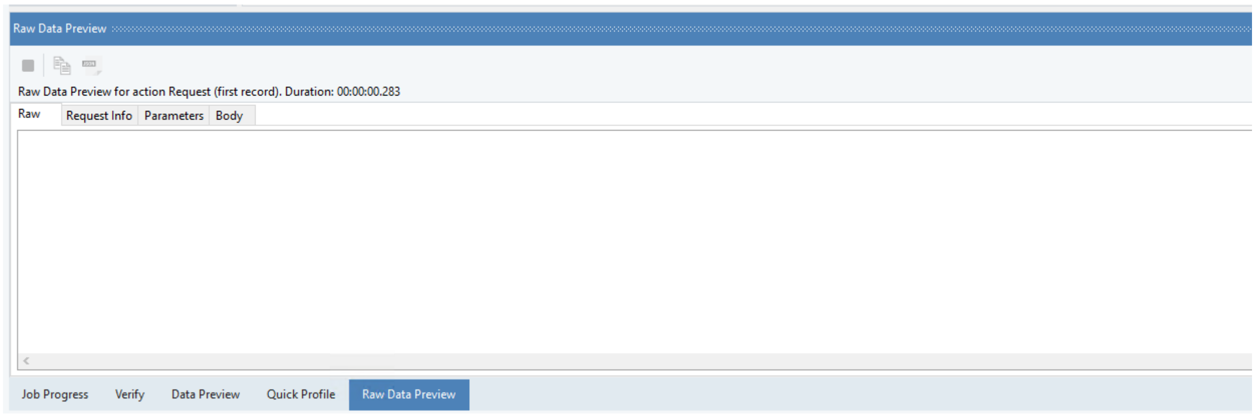
This API flow uses the *GET* HTTPS Method and allows the API user to view the Customers table's records based on the value of the *URI* parameter *username*. For demonstration, *Header* parameters i.e., *Accept*, *UserAgent*, *Query* parameter i.e., *cursor*, and *RequestInfo* parameters i.e., *HTTPMethod*, *Content-Type* are also defined.



1. The Raw Data Preview window will automatically open when a raw request/response is previewed. However, we can manually open the window as well. To do that, go to the *Menu bar* > *View* > *Raw Data Preview* or use the shortcut *Ctrl+Alt+J*.

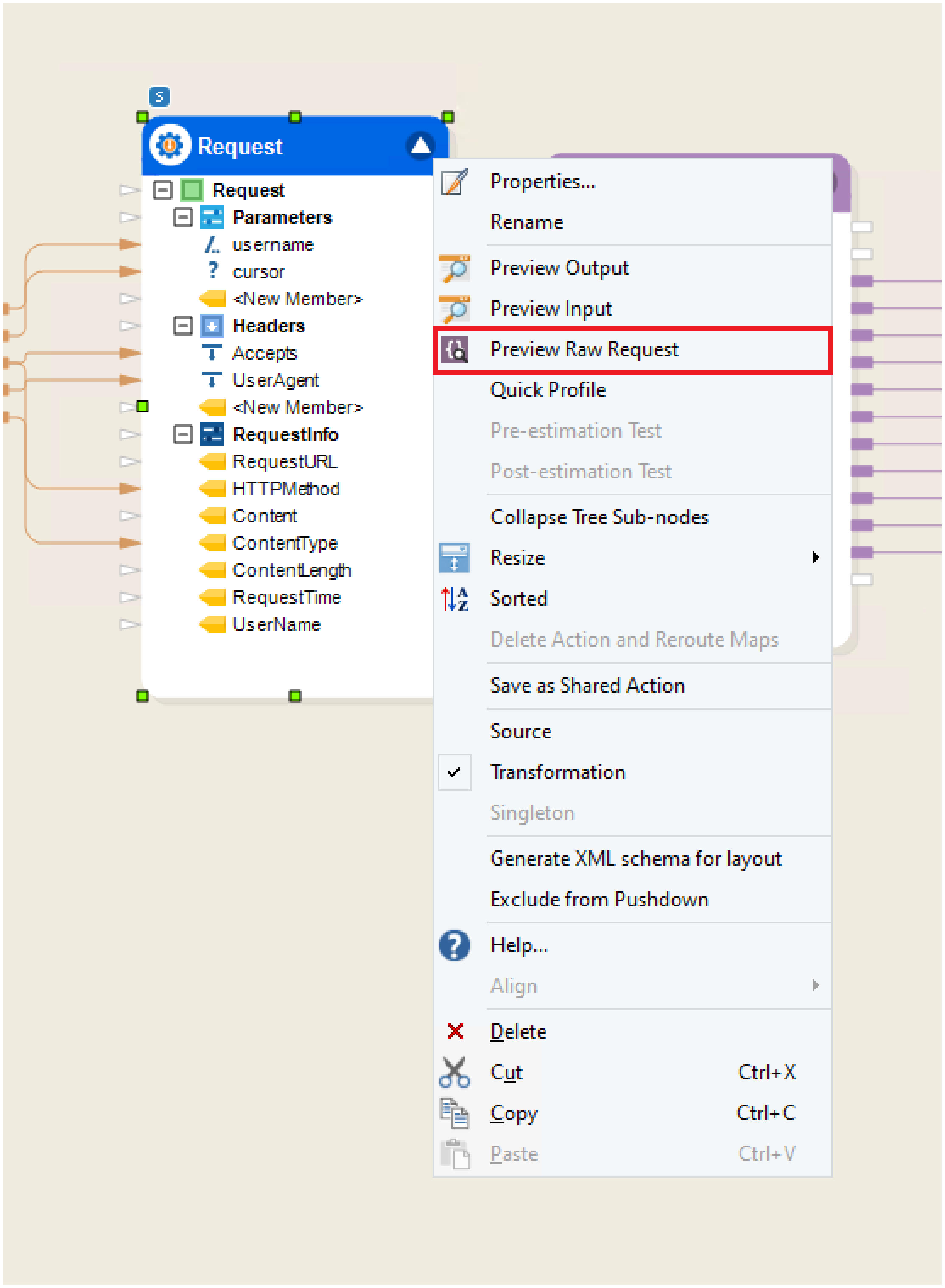


Toolbox	Ctrl+Alt+X
DataSets	Ctrl+Alt+X
Visualization Browser	Ctrl+Alt+V
Visualization	
Dataprep Editor	Ctrl+Alt+Z
Analytics Browser	Ctrl+Alt+G
Server Explorer	Ctrl+Alt+E
Verify	Ctrl+Alt+L
Job Progress	Ctrl+Alt+T
Test Progress	Ctrl+None
Data Preview	Ctrl+Alt+W
Analytics Test	Ctrl+Alt+C
Visualization	Ctrl+Alt+D
Quick Profile	Ctrl+Alt+A
Diagram Overview	Ctrl+Alt+K
Raw Data Preview	Ctrl+Alt+J
Data Source Browser	Ctrl+Alt+D
Server Browser	Ctrl+Alt+X
Sql Snippets	Ctrl+Alt+S
Project Explorer	Ctrl+Alt+P
Pending Changes	Ctrl+Alt+C
History	Ctrl+Alt+H
Output	Ctrl+Alt+O
Lineage and Impact	Ctrl+Alt+B
Find All	Ctrl+Alt+S
REST API Browser	Ctrl+Alt+F
Regression Results	Ctrl+Alt+I
Report Browser	Ctrl+Alt+R
Report Properties	Ctrl+Alt+I
Data Model Browser	Ctrl+Alt+M
Load Settings Entity Browser	Ctrl+Alt+L
Query Preview	Ctrl+Alt+Q



2. Right-click on the header of the *Request Publish* object. Select the *Preview Raw Request* option from the context menu.

Note: Preview would only work when Request has some incoming data mapped to it.

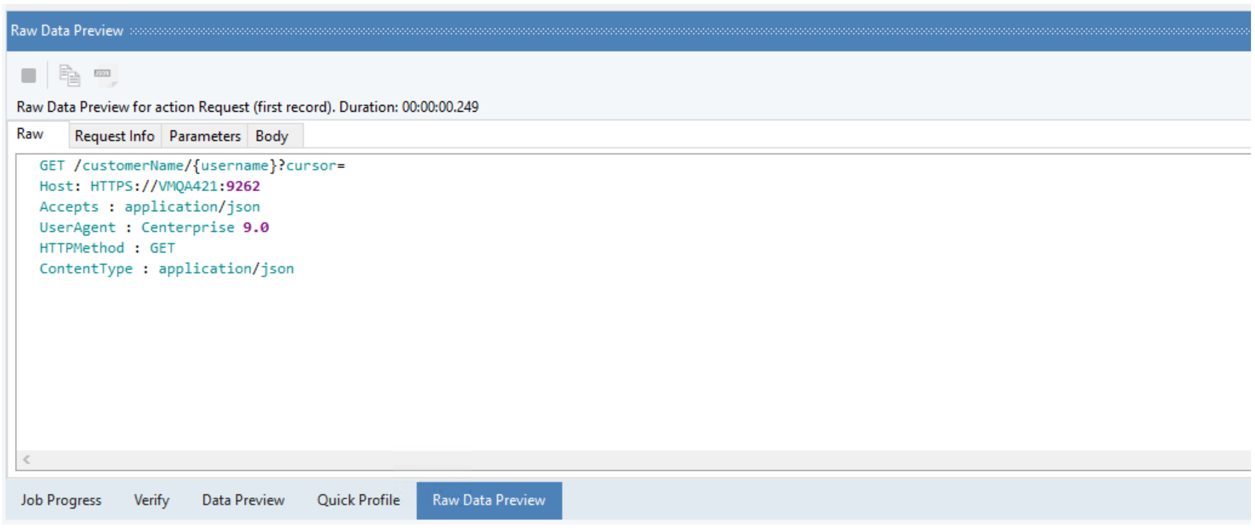


In a raw API Request, you can see:

- *URL:* Contains the *HTTPS Method*, *Resource, URI,* and *Query* parameters.*Host:* The server on which the API is deployed.
- *RequestInfo*: Default parameters containing information related to the server and request.

- *Header*: User-defined parameters containing meta-data associated with the request.
- *Body:* In case of a request other than GET, an input JSON body.

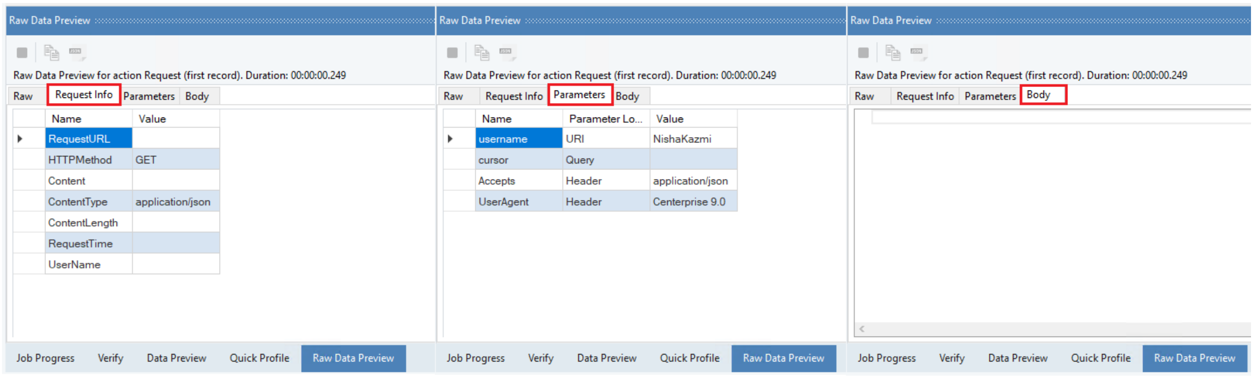
This is how a complete request looks in the Raw Data Preview window,



Raw Data Preview for action Request (first record). Duration: 00:00:00.249

```
GET /customerName/{username}?cursor=
Host: HTTPS://VMQA421:9262
Accepts : application/json
UserAgent : Centerprise 9.0
HTTPMethod : GET
ContentType : application/json
```

The *RequestInfo*, *Parameters*, and JSON *Body* are displayed in separate tabs.



Raw Data Preview for action Request (first record). Duration: 00:00:00.249

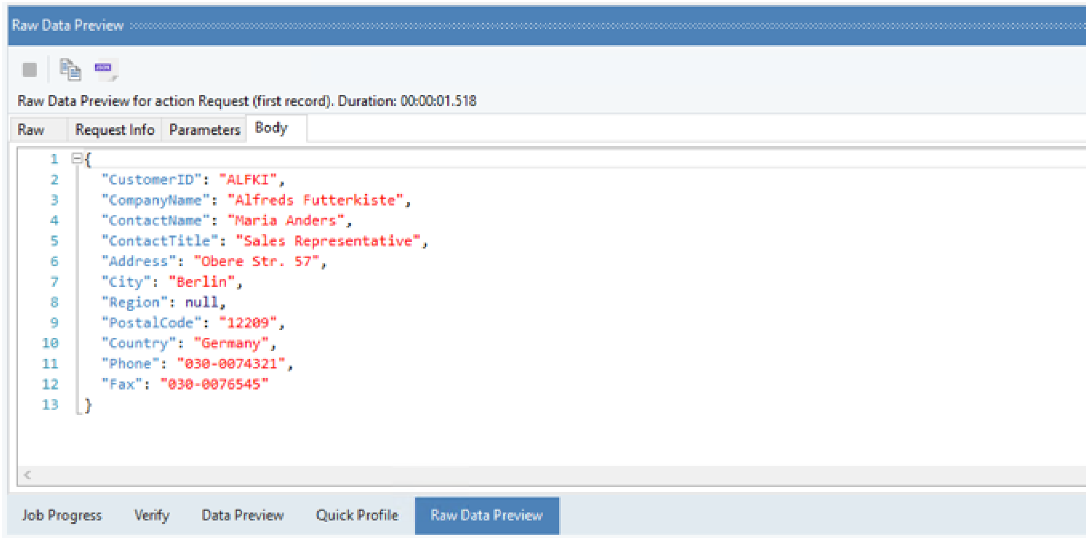
Name	Value
RequestURL	
HTTPMethod	GET
Content	
ContentType	application/json
ContentLength	
RequestTime	
UserName	

Raw Data Preview for action Request (first record). Duration: 00:00:00.249

Name	Parameter Lo...	Value
username	URI	NishaKazmi
cursor	Query	
Accepts	Header	application/json
UserAgent	Header	Centerprise 9.0

Raw Data Preview for action Request (first record). Duration: 00:00:00.249

In case of a request other than *GET,* we'll be able to see the Input JSON Body in the *Body* tab. Similar to this:



Raw Data Preview for action Request (first record). Duration: 00:00:01.518

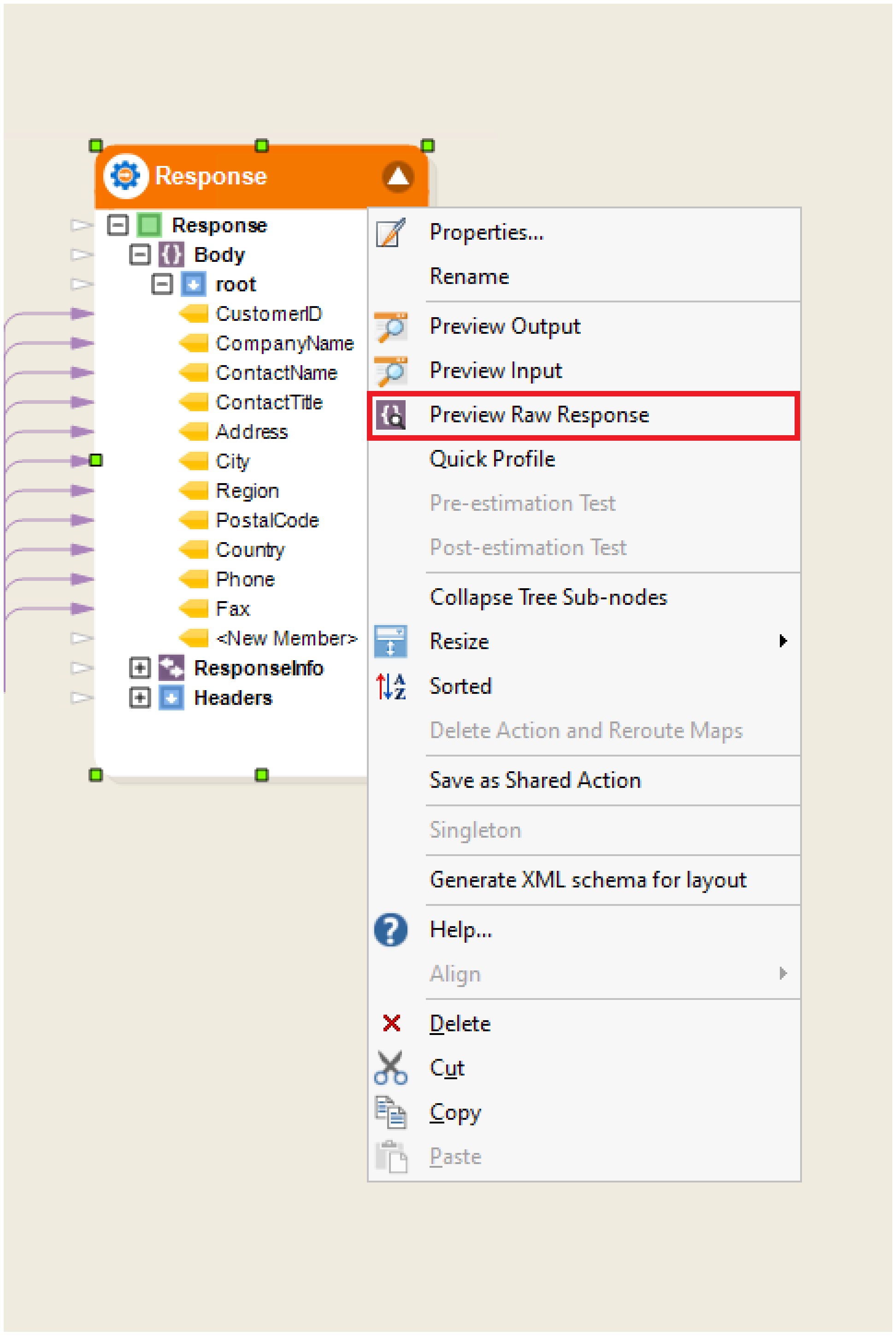
```
{
    "CustomerID": "ALFKI",
    "CompanyName": "Alfreds Futterkiste",
    "ContactName": "Maria Anders",
    "ContactTitle": "Sales Representative",
    "Address": "Obere Str. 57",
    "City": "Berlin",
    "Region": null,
    "PostalCode": "12209",
    "Country": "Germany",
    "Phone": "030-0074321",
    "Fax": "030-0076545"
}
```

In a raw API response, you can see:

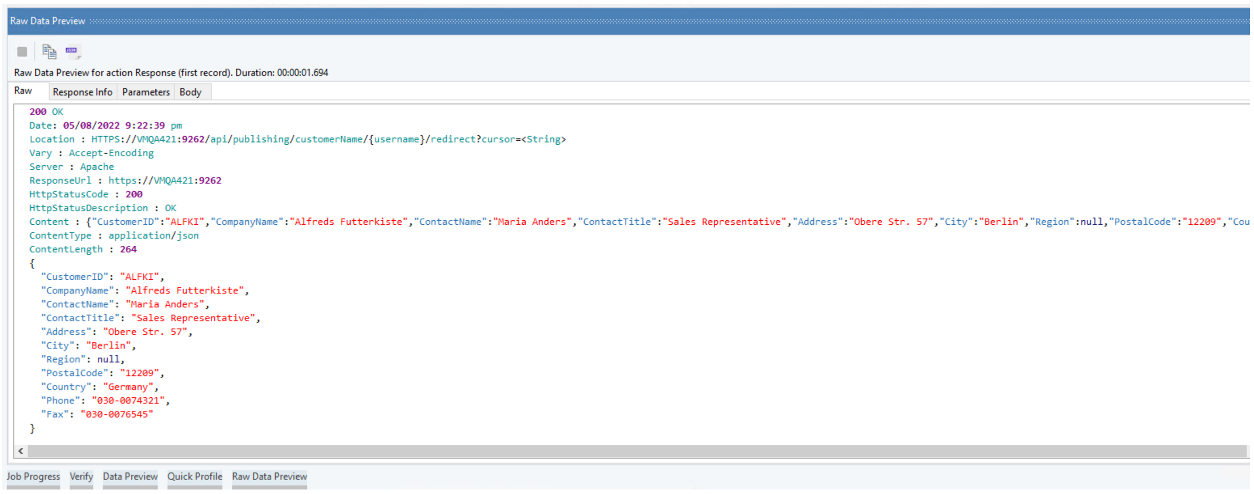
- *Date:* It specifies the date and time at which the client receives the response.
- *HTTPS Status Code:* It defines the standard response status expected from the executed flow i.e., 200 for OK or 400 Bad Request, etc.
- *HTTPS Status Description:* The standard response description matching the HTTPS Status code i.e., OK for a 200 code or BAD REQUEST for a 400 code, etc.
- *Header Parameters*: User-defined parameters containing meta-data associated with the response.
- *Content:* It contains the whole response body content in a string-like text.
- *Content-Type:* It describes the format type of the response body content.
- *Content-Length:* It specifies the number of bytes in the content of the response body.
- *Body:* It shows the response content parsed as per a defined *Custom Response Layout*.

Let's see how we can preview the raw response in Centerprise,

1. Right-click on the header of the *Response Publish* object. Select the *Preview Raw Response* option from the context menu.



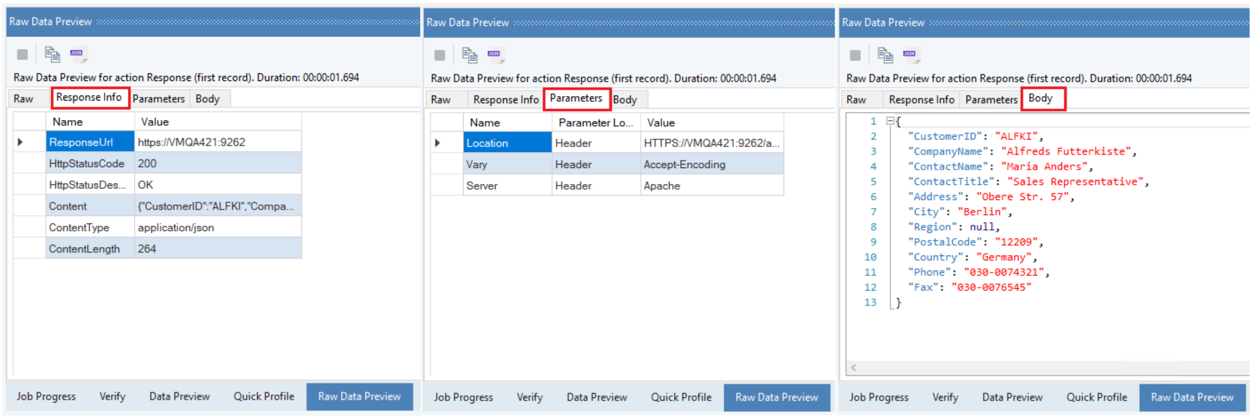
This is how the whole response looks in the Raw Data Preview,



Raw Data Preview for action Response (first record). Duration: 00:00:01.694

```
200 OK
Date: 05/08/2022 9:22:39 pm
Location : HTTPS://VMQA421:9262/api/publishing/customerName/{username}/redirect?cursor=<String>
Vary : Accept-Encoding
Server : Apache
ResponseUrl : https://VMQA421:9262
HttpStatusCode : 200
HttpStatusDescription : OK
Content : {"CustomerID":"ALFKI","CompanyName":"Alfreds Futterkiste","ContactName":"Maria Anders","ContactTitle":"Sales Representative","Address":"Obere Str. 57","City":"Berlin","Region":null,"PostalCode":"12209","Cou
ContentType : application/json
ContentLength : 264
{
  "CustomerID": "ALFKI",
  "CompanyName": "Alfreds Futterkiste",
  "ContactName": "Maria Anders",
  "ContactTitle": "Sales Representative",
  "Address": "Obere Str. 57",
  "City": "Berlin",
  "Region": null,
  "PostalCode": "12209",
  "Country": "Germany",
  "Phone": "030-0074321",
  "Fax": "030-0076545"
}
```

Similarly, the *RequestInfo*, *Header Parameters*, and JSON *Body* are displayed in separate tabs.



Raw Data Preview for action Response (first record). Duration: 00:00:01.694

Name	Value
ResponseUrl	https://VMQA421:9262
HttpStatusCode	200
HttpStatusDes...	OK
Content	{"CustomerID":"ALFKI","Compa...
ContentType	application/json
ContentLength	264

Raw Data Preview for action Response (first record). Duration: 00:00:01.694

Name	Parameter Lo...	Value
Location	Header	HTTPS://VMQA421:9262/a...
Vary	Header	Accept-Encoding
Server	Header	Apache

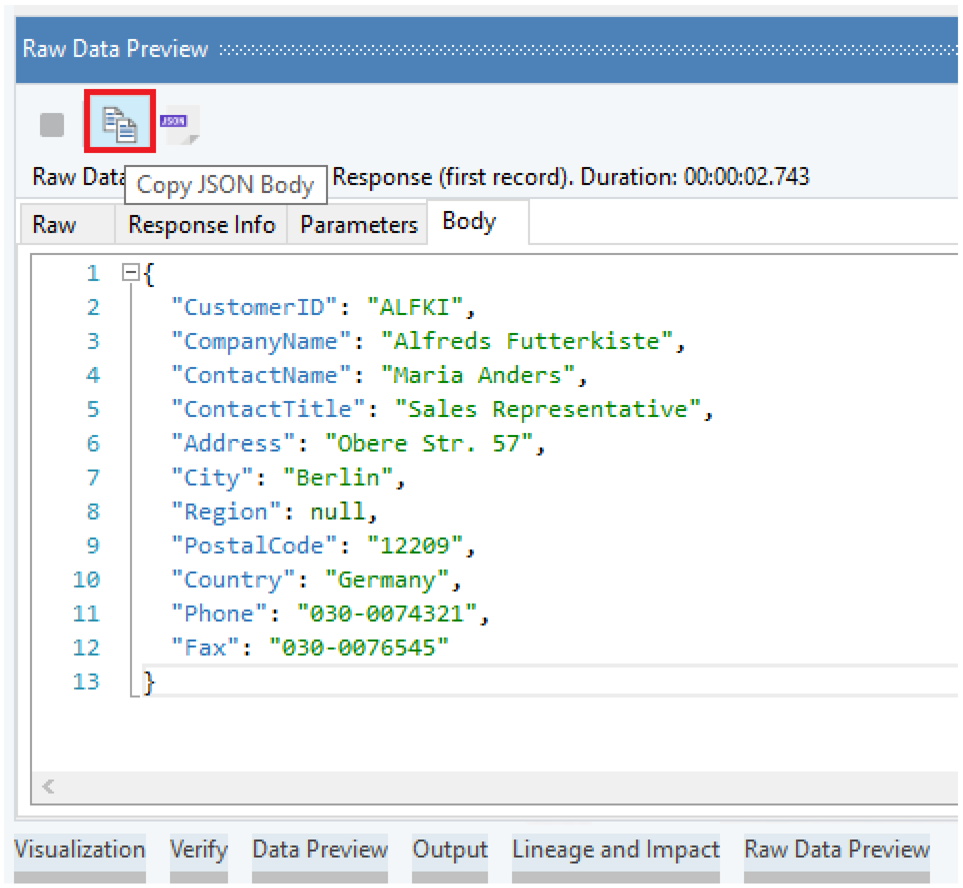
Raw Data Preview for action Response (first record). Duration: 00:00:01.694

```
{
  "CustomerID": "ALFKI",
  "CompanyName": "Alfreds Futterkiste",
  "ContactName": "Maria Anders",
  "ContactTitle": "Sales Representative",
  "Address": "Obere Str. 57",
  "City": "Berlin",
  "Region": null,
  "PostalCode": "12209",
  "Country": "Germany",
  "Phone": "030-0074321",
  "Fax": "030-0076545"
}
```

10.10.3 Save/Copy JSON

Using the *Preview Raw Response/Response* option, it is also possible to copy and save the JSON body of both the request and response.

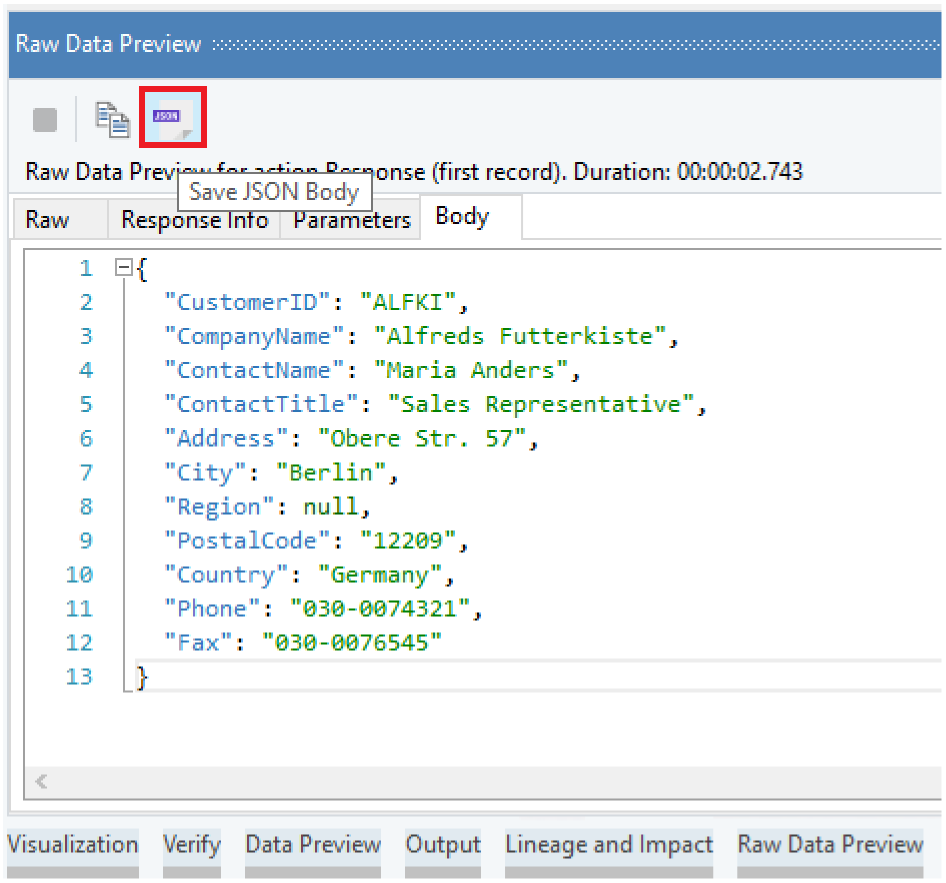
1. Click on the *Copy JSON Body* icon in the *Raw Data Preview* window.



Raw Data Response (first record). Duration: 00:00:02.743

```
{
  "CustomerID": "ALFKI",
  "CompanyName": "Alfreds Futterkiste",
  "ContactName": "Maria Anders",
  "ContactTitle": "Sales Representative",
  "Address": "Obere Str. 57",
  "City": "Berlin",
  "Region": null,
  "PostalCode": "12209",
  "Country": "Germany",
  "Phone": "030-0074321",
  "Fax": "030-0076545"
}
```

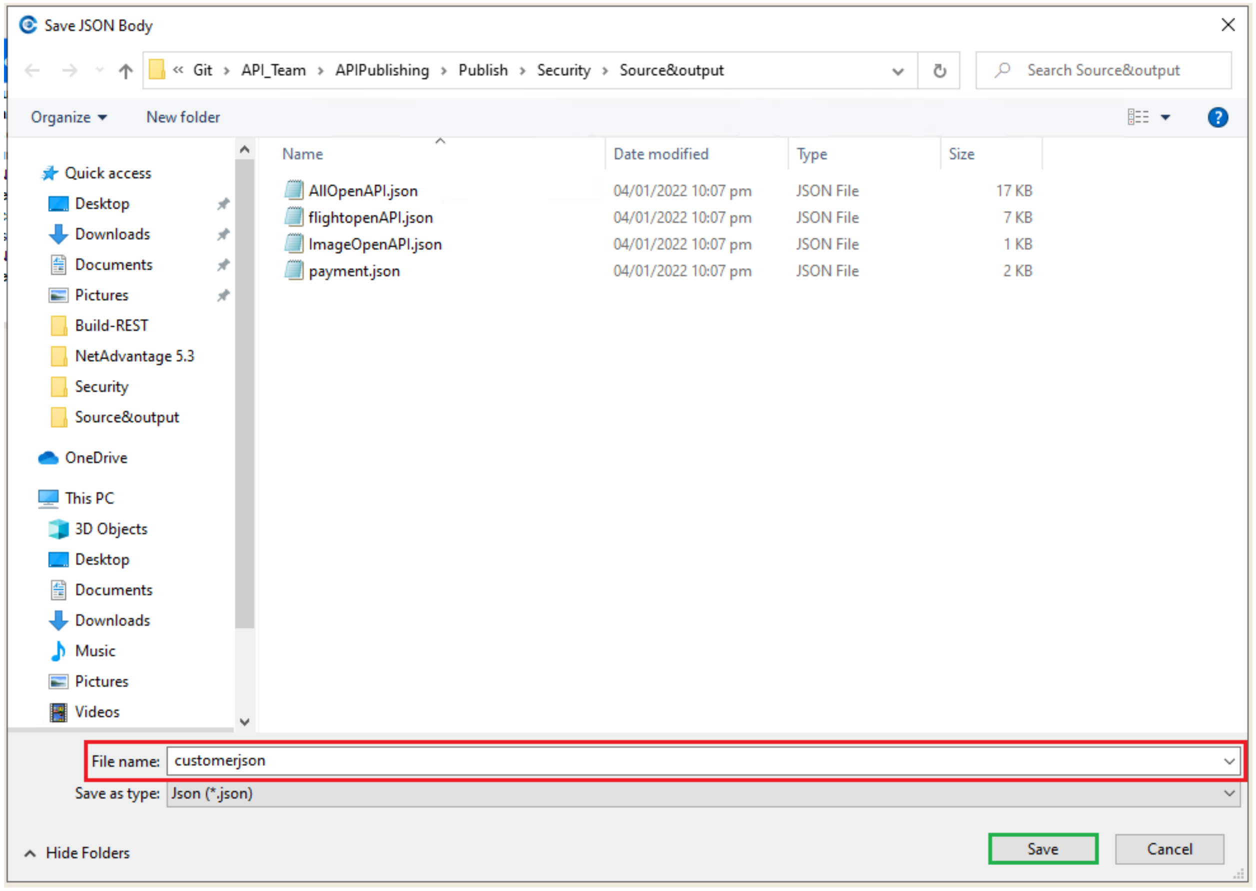
2. Similarly, click on the *Save JSON Body* icon to save the JSON body at the desired destination in a JSON format file.



Raw Data Preview for action Response (first record). Duration: 00:00:02.743

```
{
  "CustomerID": "ALFKI",
  "CompanyName": "Alfreds Futterkiste",
  "ContactName": "Maria Anders",
  "ContactTitle": "Sales Representative",
  "Address": "Obere Str. 57",
  "City": "Berlin",
  "Region": null,
  "PostalCode": "12209",
  "Country": "Germany",
  "Phone": "030-0074321",
  "Fax": "030-0076545"
}
```

Enter the desired destination in the *Save JSON Body* window and click *Save* to store the file.

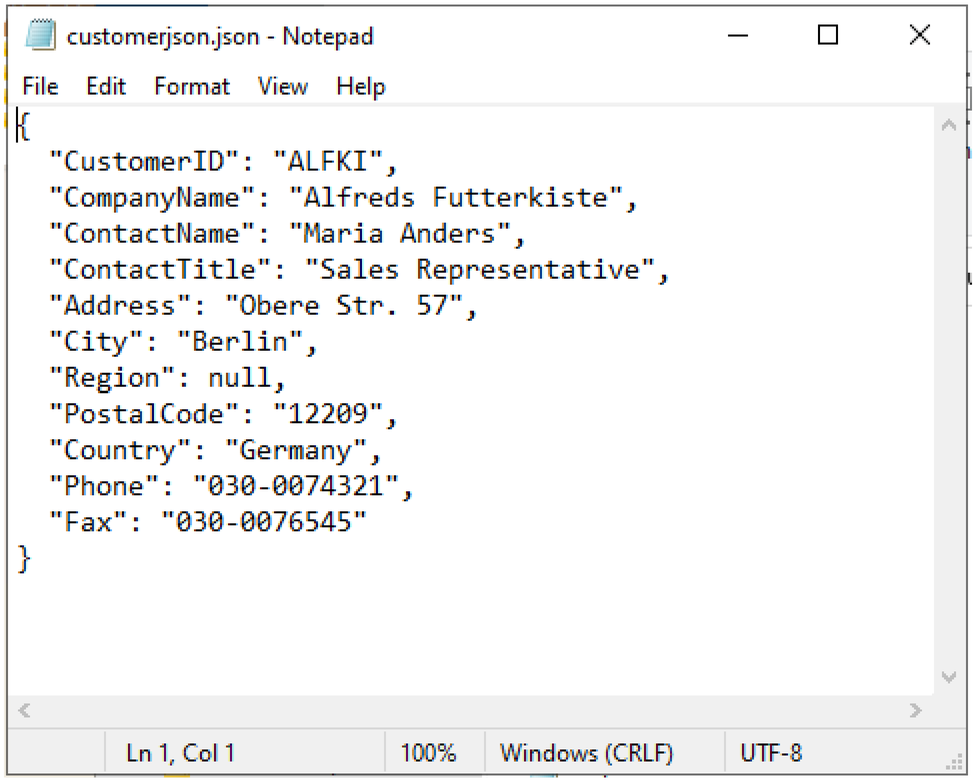


Name	Date modified	Type	Size
AllOpenAPI.json	04/01/2022 10:07 pm	JSON File	17 KB
flightopenAPI.json	04/01/2022 10:07 pm	JSON File	7 KB
ImageOpenAPI.json	04/01/2022 10:07 pm	JSON File	1 KB
payment.json	04/01/2022 10:07 pm	JSON File	2 KB

File name: customerjson

Save as type: Json (*.json)

This is what the save JSON file looks like,

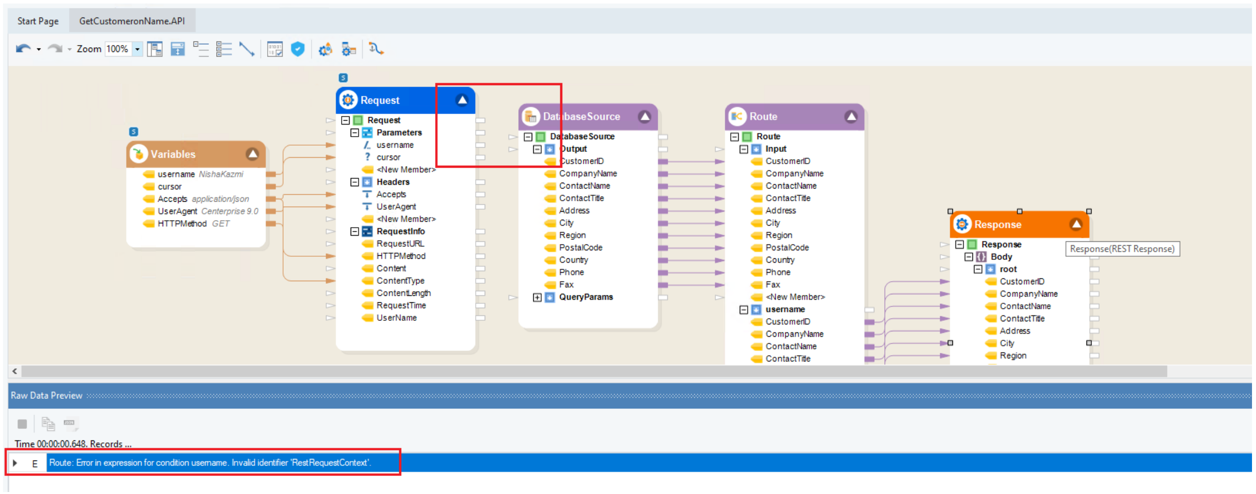


```
{
  "CustomerID": "ALFKI",
  "CompanyName": "Alfreds Futterkiste",
  "ContactName": "Maria Anders",
  "ContactTitle": "Sales Representative",
  "Address": "Obere Str. 57",
  "City": "Berlin",
  "Region": null,
  "PostalCode": "12209",
  "Country": "Germany",
  "Phone": "030-0074321",
  "Fax": "030-0076545"
}
```

10.10.4 Flow Verification

If the API flow contains any errors or warnings that affect the flow of data, they are displayed in the Raw/Data Preview window. In other words, if any obstacles block the flow of data from the Request to Response, the error is shown prominently on the window.

For example, we can see that the flow of data has been broken between the *Request* and *Database Source* object, resulting in an error state i.e., the *Route* is unable to identify a parameter, and as we preview the object we can see the error message in the window.



```
Time 00:00:00.648. Records ...
E  Route: Error in expression for condition username. Invalid identifier 'RestRequestContext'.
```

As for the whole API flow's verification, it is advised to use the *Verify Pushdown Job* option. To learn more about pushdown verification in API flows, click here.

This concludes our discussion on pre-deployment testing and verification of API flows.

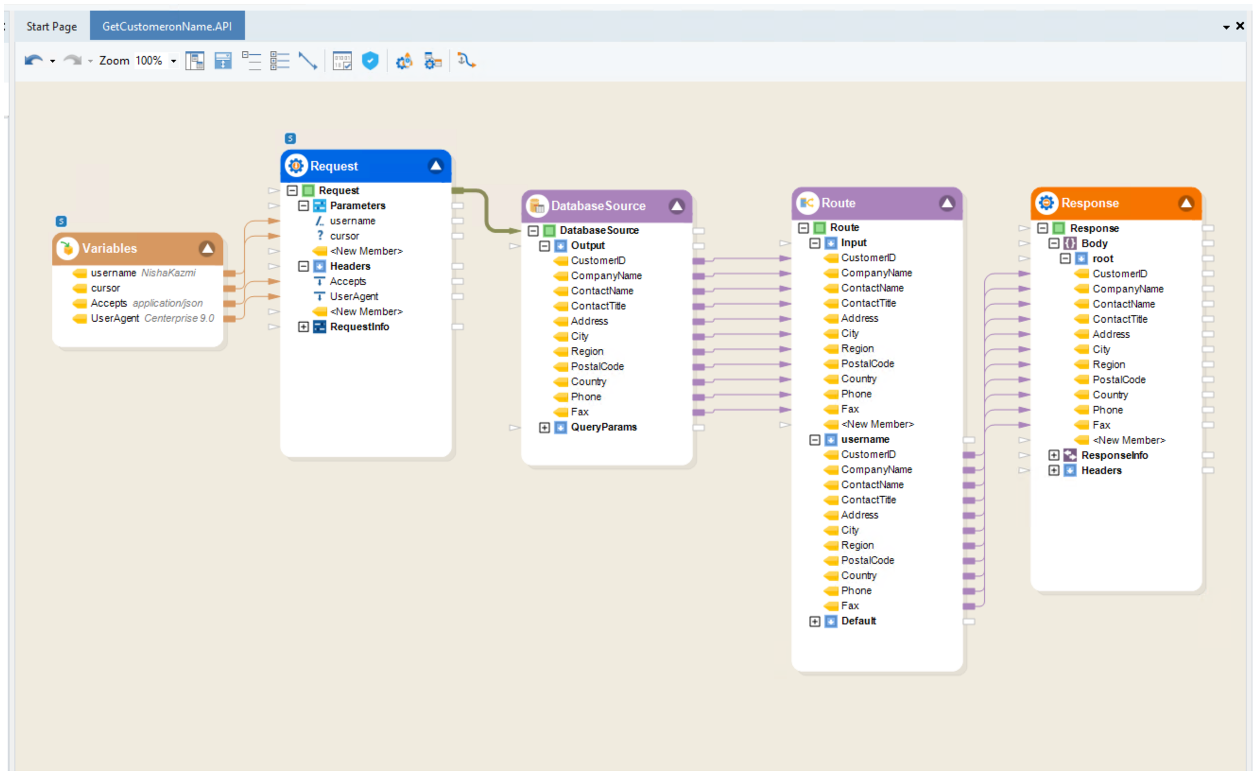
10.11 API Deployment

Using Astera API Management, users can design a complete set of API (Application Programming Interface) endpoint flows in a drag-and-drop interface. These APIs can then be deployed to the Astera Server before they can be consumed.

In this document, we will learn how to deploy API flows on the Astera Server. API flows can be deployed individually or as a group set in folder hierarchies.

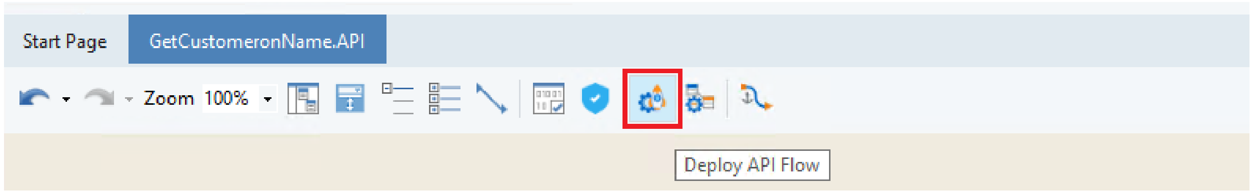
10.11.1 Deploying a single API Flow

Here, we have a pre-designed API flow,

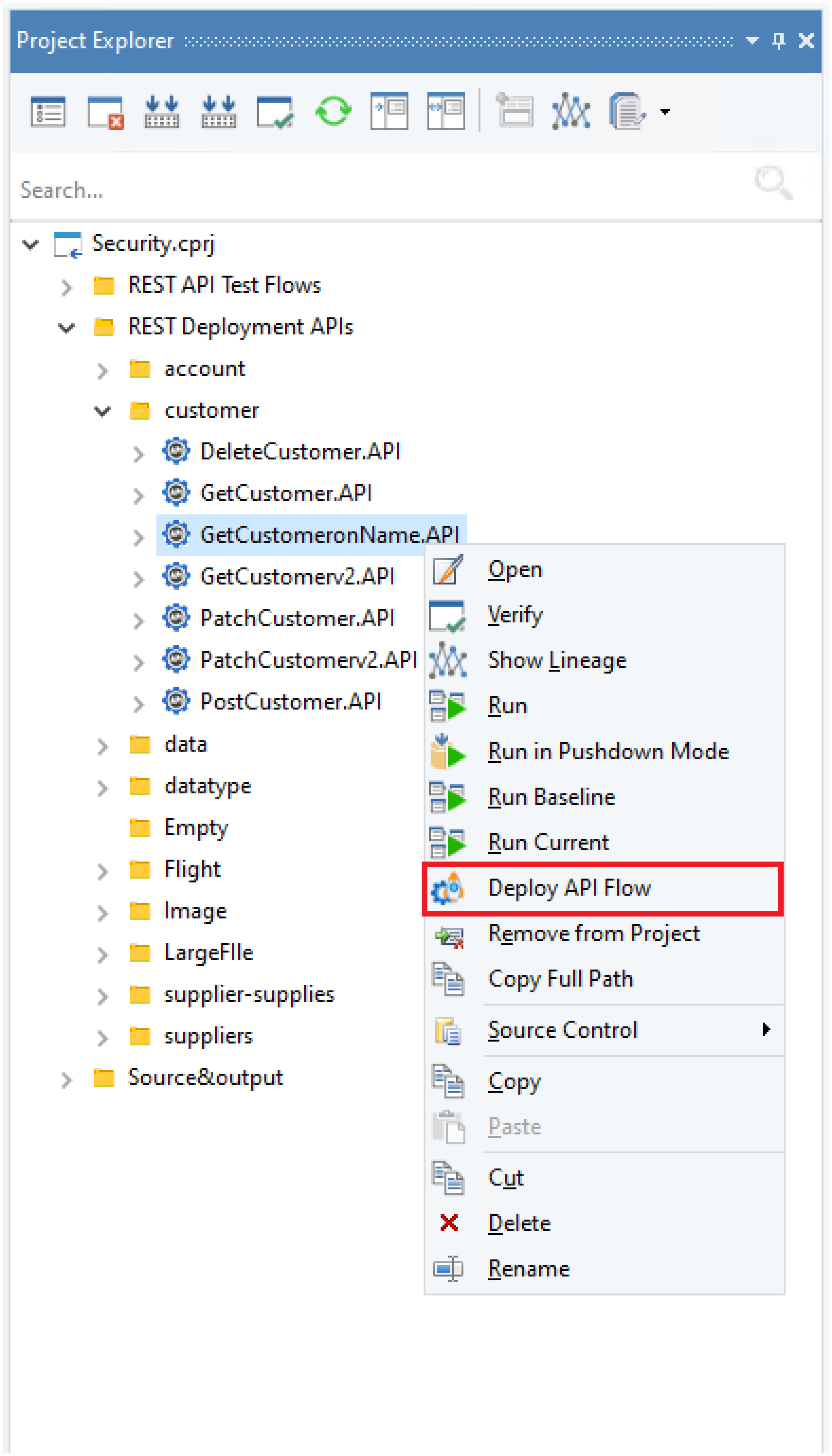


Let’s see how we can deploy this.

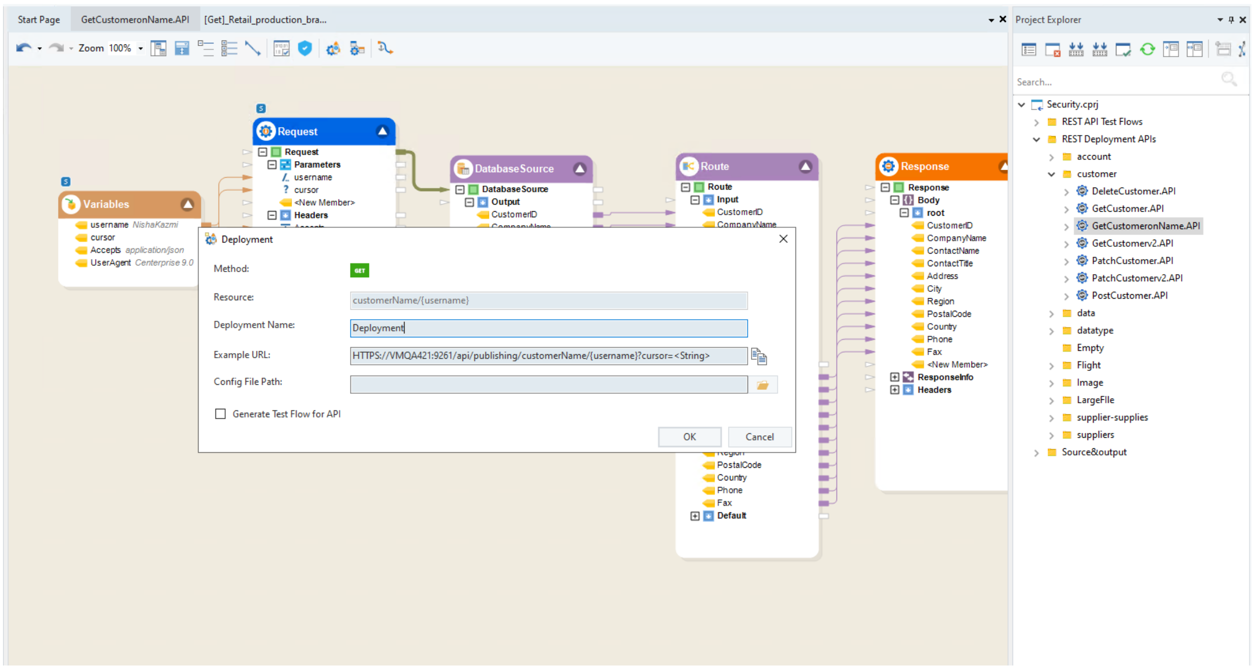
1. Click the *Deploy API Flow* icon on the API flow toolbar.



Similarly, the *Deploy API Flow* option from the Project Explorer context menu can also be used. This option is only available for API flows.



The Deployment window will appear like this,



Deployment

Method: GET

Resource: customerName/{username}

Deployment Name: Deployment

Example URL: HTTPS://VMQA421:9261/api/publishing/customerName/{username}?cursor=<String>

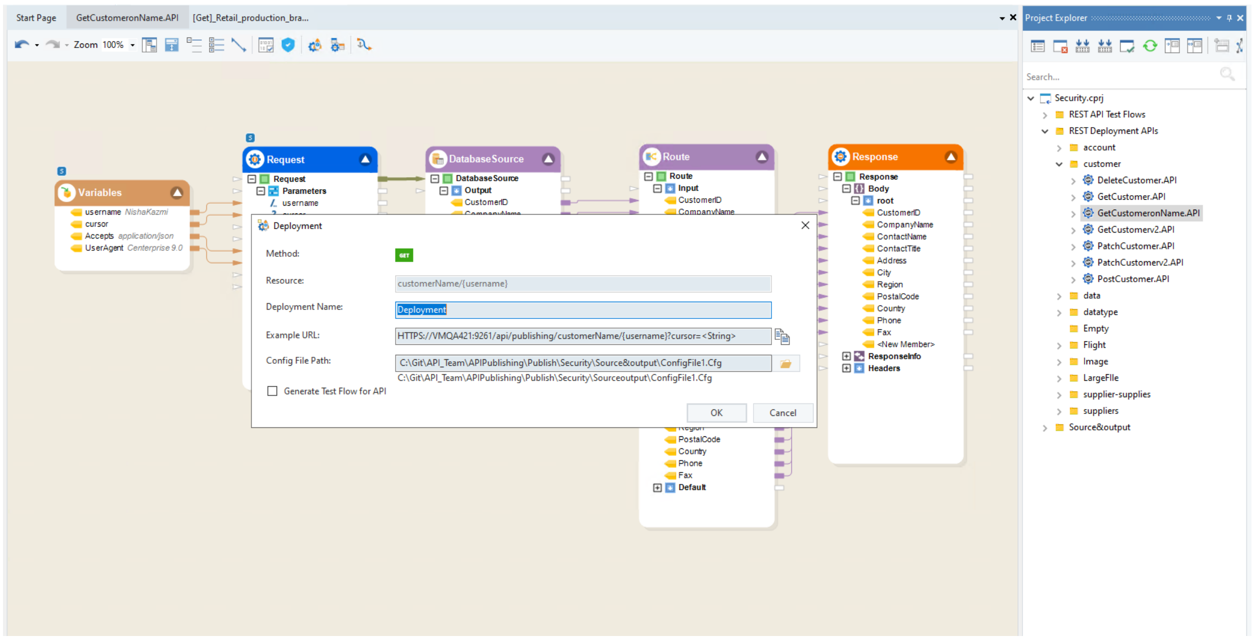
Config File Path:

☐ Generate Test Flow for API

OK Cancel

- *Method:* It's the HTTPS method selected in the *Request* object in the API flow.
- *Resource:* The endpoint defined in the *Request* object.
- *Deployment Name:* The name used to refer to this deployment in the Server Browser.
- *Example URL*: The complete URL that will be used to make the request. It includes the Base URL, Resource, URI, and Query parameters.
- *Config File Path:* The path of an optional deployment config file to define runtime variables used in the API flows.
- *Generate Test Flow for API:* Its functionality detail is given *here*.

2. Define the *Deployment Name* and the *Config File Path* (optional). Click *OK*.



Deployment

Method: GET

Resource: customerName/{username}

Deployment Name: Deployment

Example URL: HTTPS://VMQA421:9261/api/publishing/customerName/{username}?cursor=<String>

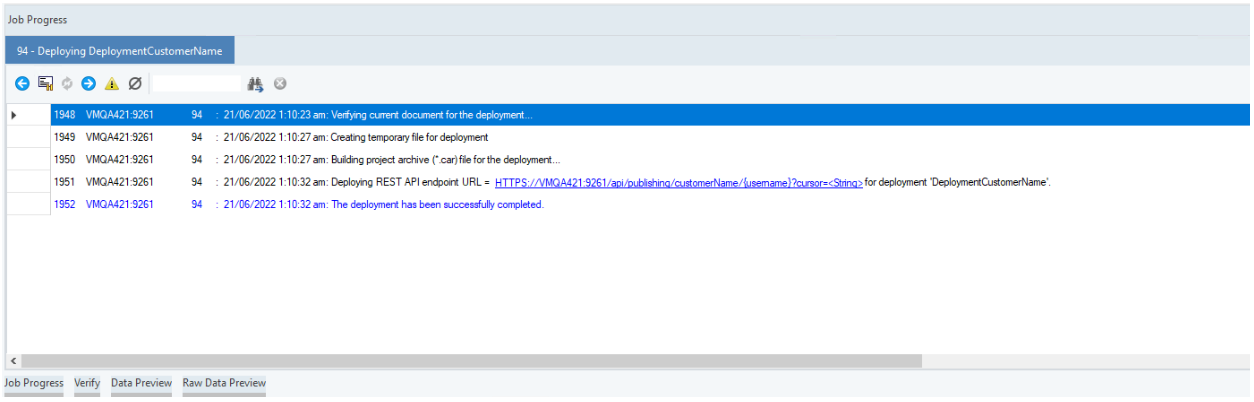
Config File Path: C:\Git\API_Team\APIPublishing\Publish\Security\Source&output\ConfigFile1.Cfg

C:\Git\API_Team\APIPublishing\Publish\Security\Sourceoutput\ConfigFile1.Cfg

☐ Generate Test Flow for API

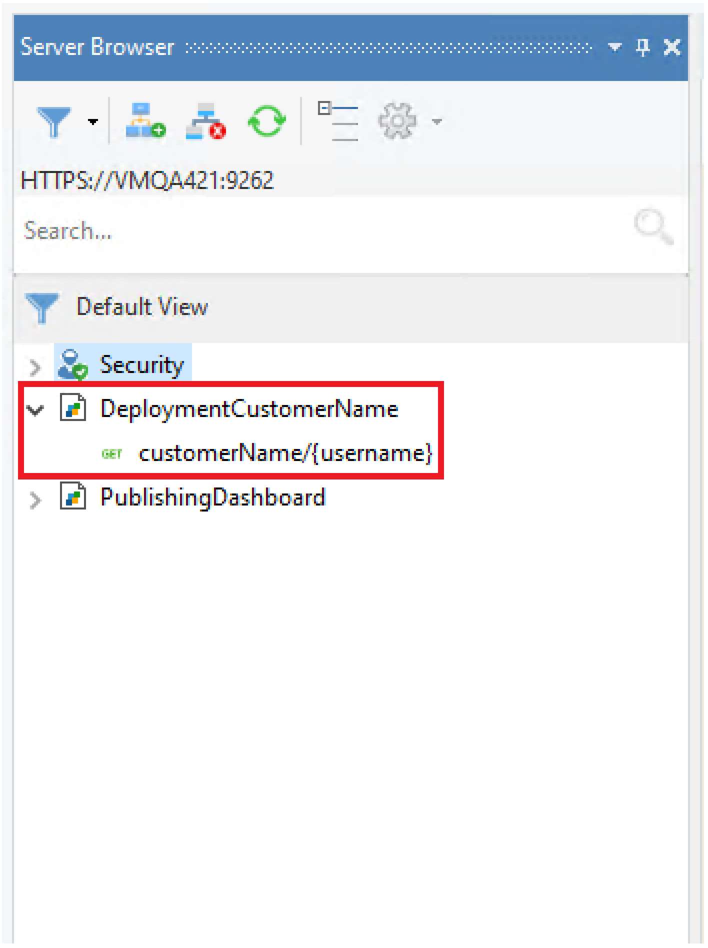
OK Cancel

The API Flow is verified in pushdown mode before the creation of the deployment. In case of any verification errors, the deployment will not be created, and the success or failure of deployment status will appear in the Job Progress window. In this case, as you can see, the deployment is completed successfully.



```
1948  VMQA421:9261  94  : 21/06/2022 1:10:23 am: Verifying current document for the deployment...
1949  VMQA421:9261  94  : 21/06/2022 1:10:27 am: Creating temporary file for deployment
1950  VMQA421:9261  94  : 21/06/2022 1:10:27 am: Building project archive (*.car) file for the deployment...
1951  VMQA421:9261  94  : 21/06/2022 1:10:32 am: Deploying REST API endpoint URL = HTTPS://VMQA421:9261/api/publishing/customerName/{username}?cursor=<String> for deployment 'DeploymentCustomerName'.
1952  VMQA421:9261  94  : 21/06/2022 1:10:32 am: The deployment has been successfully completed.
```

Once the deployment is successfully created, it becomes available in the Server Browser.



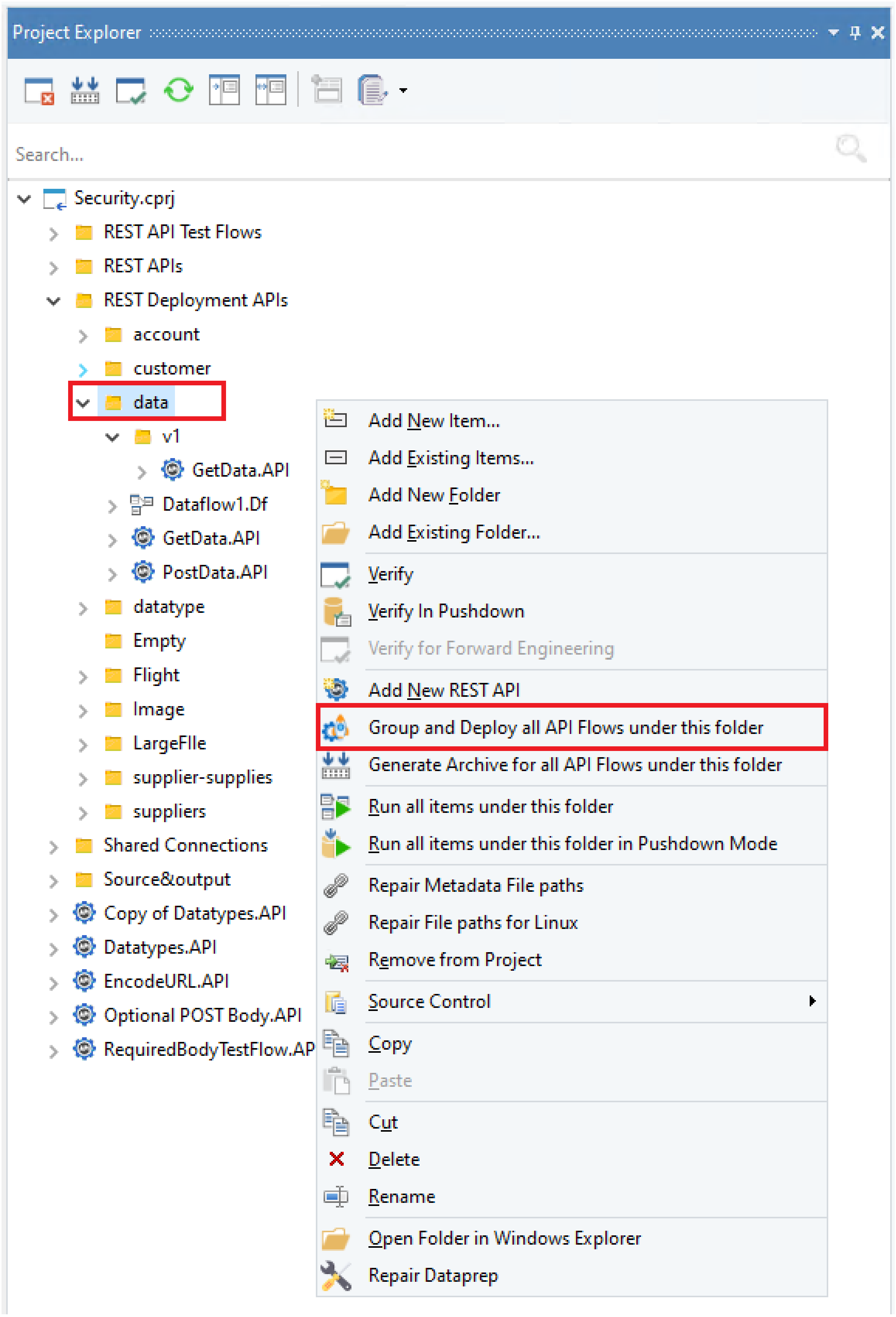
This is how you can deploy an API flow. Now, let's see how we can group and deploy API flows at the folder level.

10.11.2 Group and Deploy APIs

We can group and deploy API flow(s) contained under a folder. All the folder nodes present under the project have the *Group and Deploy All API Flows under this Folder* option, including the parent .cprj project node. Only the API flows shall be verified and deployed whereas all the other artifacts will not be considered in the deployment process.

1. Right-click on the desired folder and select *Group and Deploy All API Flows under this Folder* option.

Note: It is recommended to first verify all flows in pushdown and resolve any errors before proceeding to deployment.

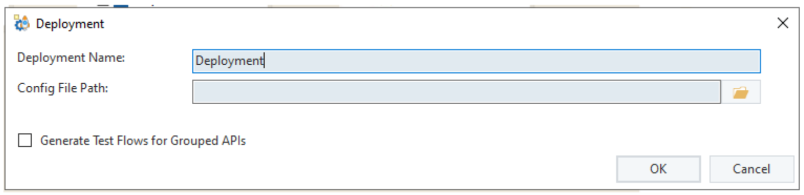

Project Explorer
Search...
Security.cprj
REST API Test Flows
REST APIs
REST Deployment APIs
account
customer
data
v1
GetData.API
Dataflow1.Df
GetData.API
PostData.API
datatype
Empty
Flight
Image
LargeFIle
supplier-supplies
suppliers
Shared Connections
Source&output
Copy of Datatypes.API
Datatypes.API
EncodeURL.API
Optional POST Body.API
RequiredBodyTestFlow.AP
Add New Item...
Add Existing Items...
Add New Folder
Add Existing Folder...
Verify
Verify In Pushdown
Verify for Forward Engineering
Add New REST API
Group and Deploy all API Flows under this folder
Generate Archive for all API Flows under this folder
Run all items under this folder
Run all items under this folder in Pushdown Mode
Repair Metadata File paths
Repair File paths for Linux
Remove from Project
Source Control
Copy
Paste
Cut
Delete
Rename
Open Folder in Windows Explorer
Repair Dataprep

In this example, we have grouped and deployed all API flows under the folder called “Data”. It’s noticeable that the folder, “Data”, has a nested folder, “v1”, under it. In the case of deploying from a folder that contains nested/child folder(s), the name of the nested folder(s) will be appended to the API’s URL as a part of its Resource. For example, ‘’{base URL}/Nested Folder Name/Flow Resource/Parameters” such as ‘’https://localhost:9621/v1/{Resource}/{Parameters}.”

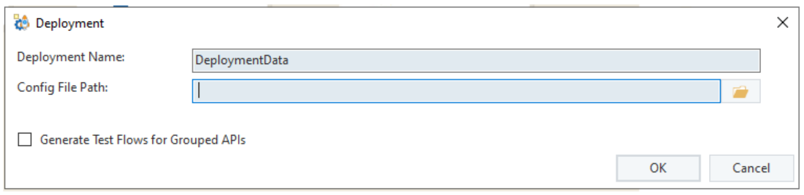
Similarly, we can see a “Dataflow1.Df” artifact under the folder as well. As explained before, this dataflow will not be considered during the deployment process.

Note: Please note that during group and deploy, the parent folder’s name is not considered as part of the resource.

After selecting the option, the Deployment window will appear like this,

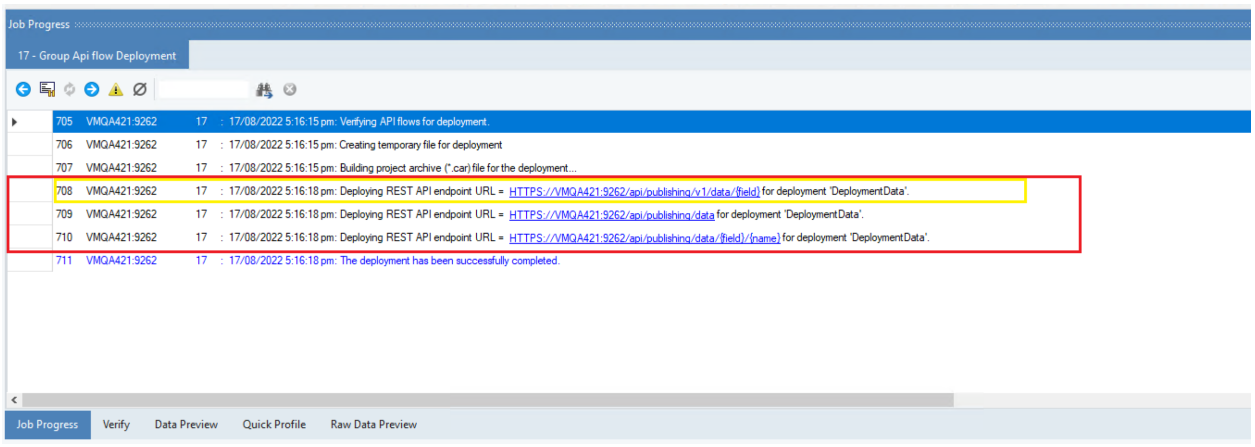


2. Write the *Deployment Name* and set the path of the *Config File* (optional). Click *OK*.



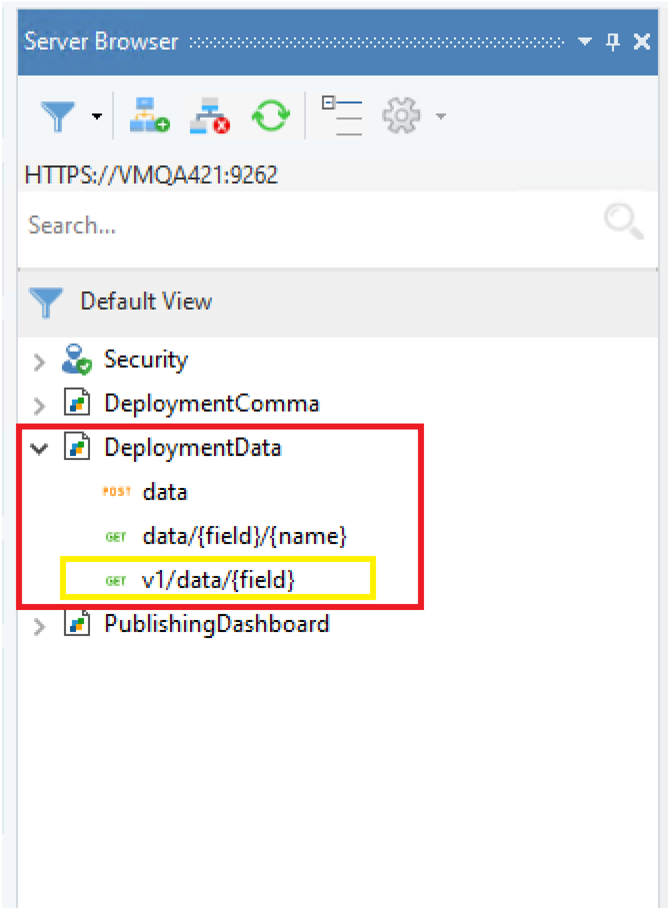
Note: All API Flows are verified in Pushdown mode before the deployment is created.

Success or Failure of the deployment will appear in the Job Progress window. In case of any errors, the verification window can be used to identify and fix errors. As the verification was successful, the API endpoints are visible in the trace.



```
705   VMQA421:9262   17  : 17/08/2022 5:16:15 pm: Verifying API flows for deployment.
706   VMQA421:9262   17  : 17/08/2022 5:16:15 pm: Creating temporary file for deployment
707   VMQA421:9262   17  : 17/08/2022 5:16:15 pm: Building project archive (*.car) file for the deployment...
708   VMQA421:9262   17  : 17/08/2022 5:16:18 pm: Deploying REST API endpoint URL = HTTPS://VMQA421:9262/api/publishing/v1/data/{field} for deployment 'DeploymentData'.
709   VMQA421:9262   17  : 17/08/2022 5:16:18 pm: Deploying REST API endpoint URL = HTTPS://VMQA421:9262/api/publishing/data for deployment 'DeploymentData'.
710   VMQA421:9262   17  : 17/08/2022 5:16:18 pm: Deploying REST API endpoint URL = HTTPS://VMQA421:9262/api/publishing/data/{field}/{name} for deployment 'DeploymentData'.
711   VMQA421:9262   17  : 17/08/2022 5:16:18 pm: The deployment has been successfully completed.
```

The successfully created deployment is visible in the Server Browser.



Note: Notice endpoint annotated in yellow. We can see here the nested folder name has been appended as a resource.

This is how you can group deploy the API(s) at the folder level.

10.12 Test Flow Generation

10.12.1 Generating a Test Flow

The *Generate Test Flow* option auto creates post-deployment test flows. These dataflows can be used to make live requests to the deployed API endpoints using the *REST Client* and *REST Connection* objects.

The *REST Connection* object contains the base URL of the server where the APIs are deployed and is configured with an Access *Token* for Authentication.

The *REST Client* object encapsulates the entire API flow's logic, starting from the *Request* object to the *Response Publish* object, including request parameters, request and response content bodies, and pagination configurations.

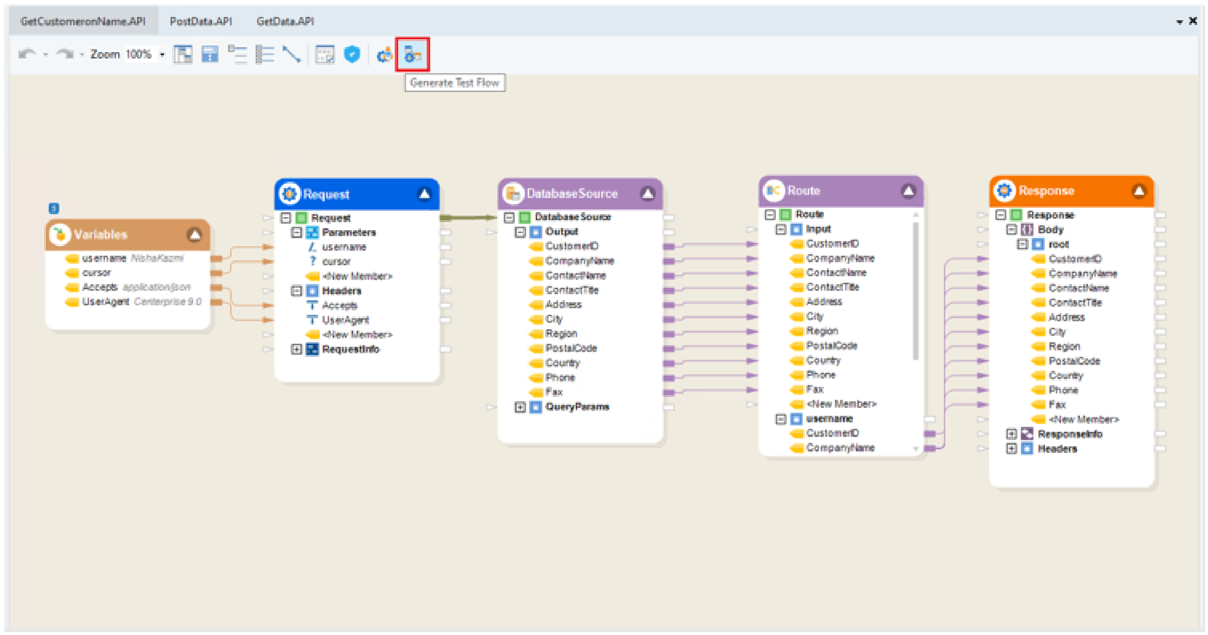
Other objects that are mapped either to the *Request* object or from the *Response* object will not be encapsulated in the API deployment and shall remain as it is in the test flow generated.

However, any Workflow Tasks shall not be made part of the testflow.

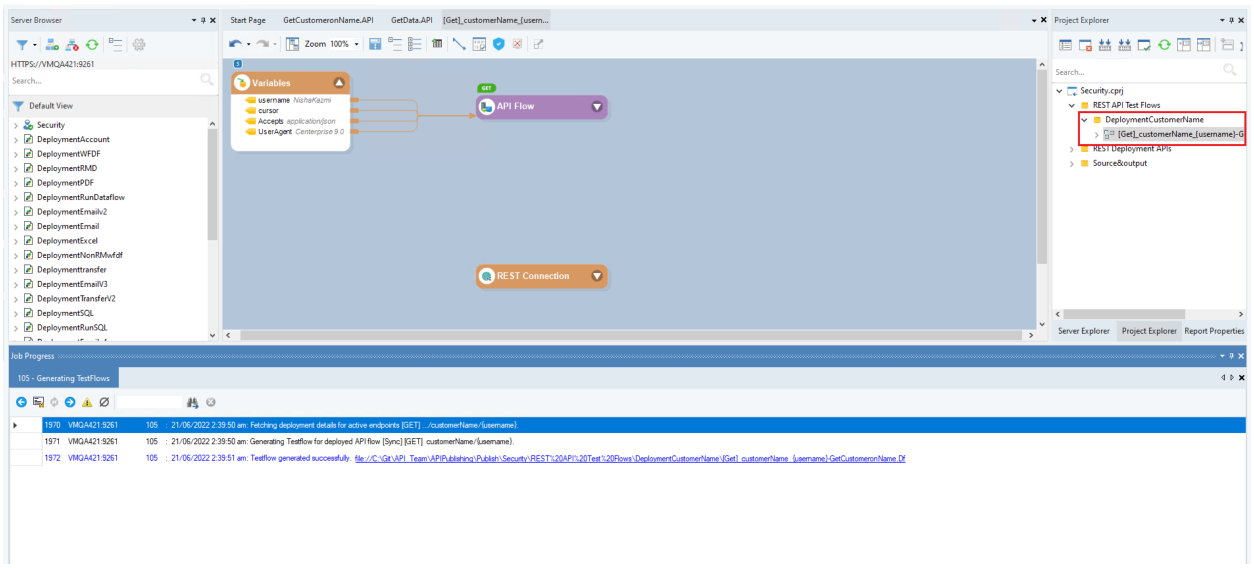
Flow Level

Generate Test Flow Icon

At the flow level, use the *Generate Test Flow* icon in the API flow toolbar to create the test flow for a deployed API.



Check the Job Progress to see if the test case generation resulted in a failure or success. Here, it is successful. This is the generated test flow for the API.

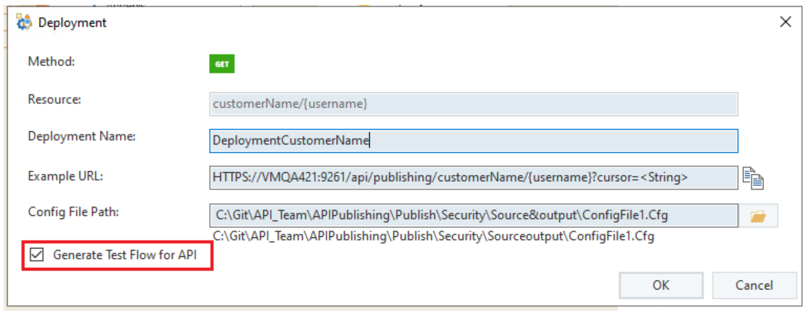


```
1970  VMQA421:9261  105  : 21/06/2022 2:39:50 am: Fetching deployment details for active endpoints [GET] .../customerName/{username}.
1971  VMQA421:9261  105  : 21/06/2022 2:39:50 am: Generating Testflow for deployed API flow [Sync] [GET]  customerName/{username}.
1972  VMQA421:9261  105  : 21/06/2022 2:39:51 am: Testflow generated successfully. file://C:\Git\API_Team\APIPublishing\Publish\Security\REST%20API%20Test%20Flows\DeploymentCustomerName\[Get]_customerName_{username}-GetCustomeronName.Df
```

You can run this test dataflow to check the behavior and assess the performance and functionality of the designed API.

Generate Test Flow for API checkbox

At the flow level, we can also check the *Generate Test Flow for API* checkbox on the deployment window.

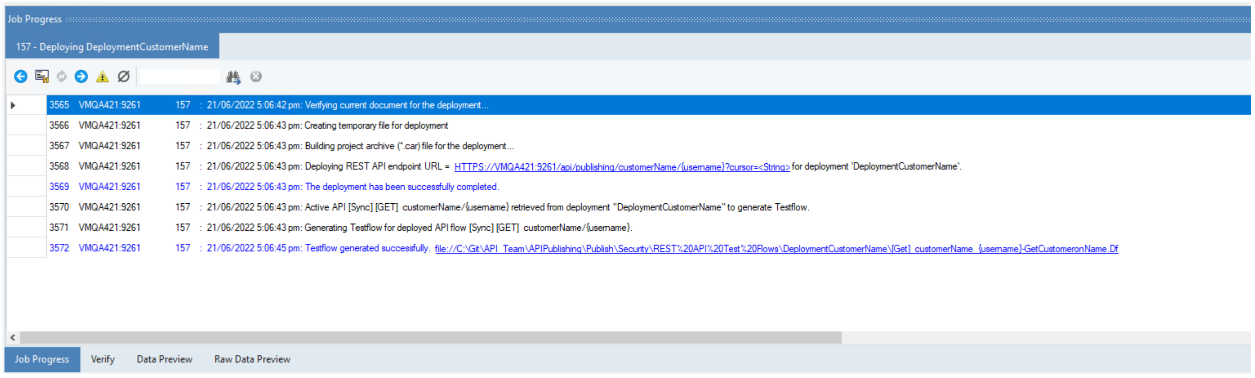


Method:	GET
Resource:	customerName/{username}
Deployment Name:	DeploymentCustomerName
Example URL:	HTTPS://VMQA421:9261/api/publishing/customerName/{username}?cursor=<String>
Config File Path:	C:\Git\API_Team\APIPublishing\Publish\Security\Source&output\ConfigFile1.Cfg

C:\Git\API_Team\APIPublishing\Publish\Security\Sourceoutput\ConfigFile1.Cfg

[✓] Generate Test Flow for API

This creates the test flow after the creation of the deployment. However, only when the API flow's verification is successful, the test flow is created. Otherwise, the entire process results in an error.



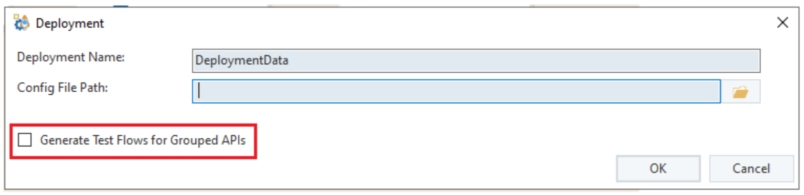
```
157 - Deploying DeploymentCustomerName

3565  VMQA421:9261  157  : 21/06/2022 5:06:42 pm: Verifying current document for the deployment...
3566  VMQA421:9261  157  : 21/06/2022 5:06:43 pm: Creating temporary file for deployment
3567  VMQA421:9261  157  : 21/06/2022 5:06:43 pm: Building project archive (*.car) file for the deployment...
3568  VMQA421:9261  157  : 21/06/2022 5:06:43 pm: Deploying REST API endpoint URL = HTTPS://VMQA421:9261/api/publishing/customerName/{username}?cursor=<String> for deployment 'DeploymentCustomerName'.
3569  VMQA421:9261  157  : 21/06/2022 5:06:43 pm: The deployment has been successfully completed.
3570  VMQA421:9261  157  : 21/06/2022 5:06:43 pm: Active API [Sync] [GET]  customerName/{username} retrieved from deployment "DeploymentCustomerName" to generate Testflow.
3571  VMQA421:9261  157  : 21/06/2022 5:06:43 pm: Generating Testflow for deployed API flow [Sync] [GET]  customerName/{username}.
3572  VMQA421:9261  157  : 21/06/2022 5:06:45 pm: Testflow generated successfully. file://C:\Git\API_Team\APIPublishing\Publish\Security\REST%20API%20Test%20Flows\DeploymentCustomerName\[Get]_customerName_{username}-GetCustomeronName.Df
```

Folder Level

Generate Test Flows for grouped APIs checkbox

For the Folder level test flow generation, check the *Generate Test Flows for Grouped APIs* check box while deploying the APIs.

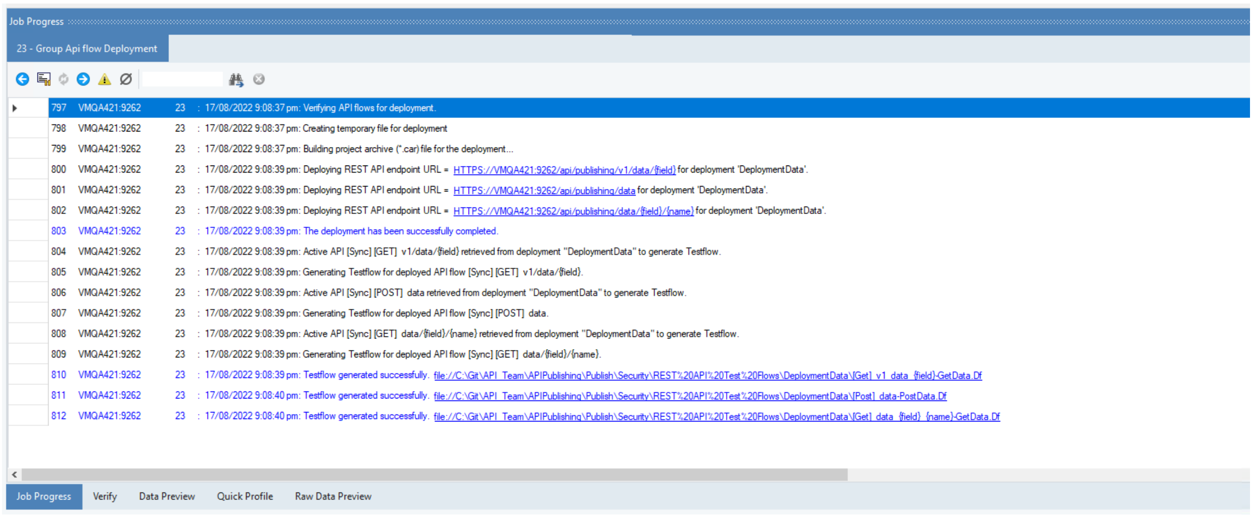


Deployment Name: DeploymentData

Config File Path:

☐ Generate Test Flows for Grouped APIs

Check the Job Progress to see if the verification of the API flows and the test case generation resulted in a failure or success along with the deployment creation job traces. Here, the test flow creation was successful.

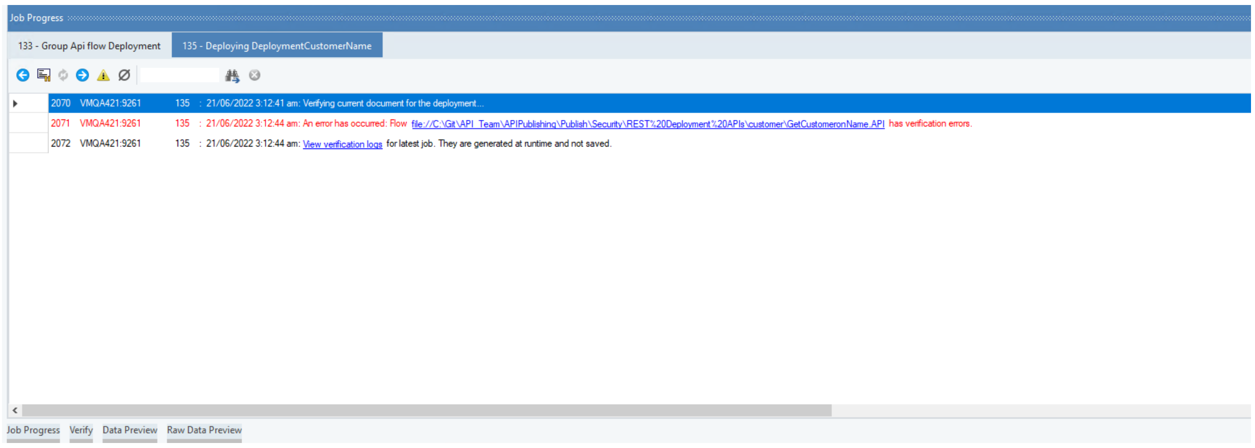


```
797  VMQA421:9262  23  : 17/08/2022 9:08:37 pm: Verifying API flows for deployment.
798  VMQA421:9262  23  : 17/08/2022 9:08:37 pm: Creating temporary file for deployment
799  VMQA421:9262  23  : 17/08/2022 9:08:37 pm: Building project archive (*.car) file for the deployment...
800  VMQA421:9262  23  : 17/08/2022 9:08:39 pm: Deploying REST API endpoint URL = HTTPS://VMQA421:9262/api/publishing/v1/data/{field} for deployment 'DeploymentData'.
801  VMQA421:9262  23  : 17/08/2022 9:08:39 pm: Deploying REST API endpoint URL = HTTPS://VMQA421:9262/api/publishing/data for deployment 'DeploymentData'.
802  VMQA421:9262  23  : 17/08/2022 9:08:39 pm: Deploying REST API endpoint URL = HTTPS://VMQA421:9262/api/publishing/data/{field}/{name} for deployment 'DeploymentData'.
803  VMQA421:9262  23  : 17/08/2022 9:08:39 pm: The deployment has been successfully completed.
804  VMQA421:9262  23  : 17/08/2022 9:08:39 pm: Active API [Sync] [GET]  v1/data/{field} retrieved from deployment "DeploymentData" to generate Testflow.
805  VMQA421:9262  23  : 17/08/2022 9:08:39 pm: Generating Testflow for deployed API flow [Sync] [GET]  v1/data/{field}.
806  VMQA421:9262  23  : 17/08/2022 9:08:39 pm: Active API [Sync] [POST]  data retrieved from deployment "DeploymentData" to generate Testflow.
807  VMQA421:9262  23  : 17/08/2022 9:08:39 pm: Generating Testflow for deployed API flow [Sync] [POST]  data.
808  VMQA421:9262  23  : 17/08/2022 9:08:39 pm: Active API [Sync] [GET]  data/{field}/{name} retrieved from deployment "DeploymentData" to generate Testflow.
809  VMQA421:9262  23  : 17/08/2022 9:08:39 pm: Generating Testflow for deployed API flow [Sync] [GET]  data/{field}/{name}.
810  VMQA421:9262  23  : 17/08/2022 9:08:39 pm: Testflow generated successfully. file://C:\Git\API_Team\APIPublishing\Publish\Security\REST%20API%20Test%20Flows\DeploymentData\[Get]_v1_data_{field}-GetData.Df
811  VMQA421:9262  23  : 17/08/2022 9:08:40 pm: Testflow generated successfully. file://C:\Git\API_Team\APIPublishing\Publish\Security\REST%20API%20Test%20Flows\DeploymentData\[Post]_data-PostData.Df
812  VMQA421:9262  23  : 17/08/2022 9:08:40 pm: Testflow generated successfully. file://C:\Git\API_Team\APIPublishing\Publish\Security\REST%20API%20Test%20Flows\DeploymentData\[Get]_data_{field}_{name}-GetData.Df
```

10.12.2 Verification of the API Flows

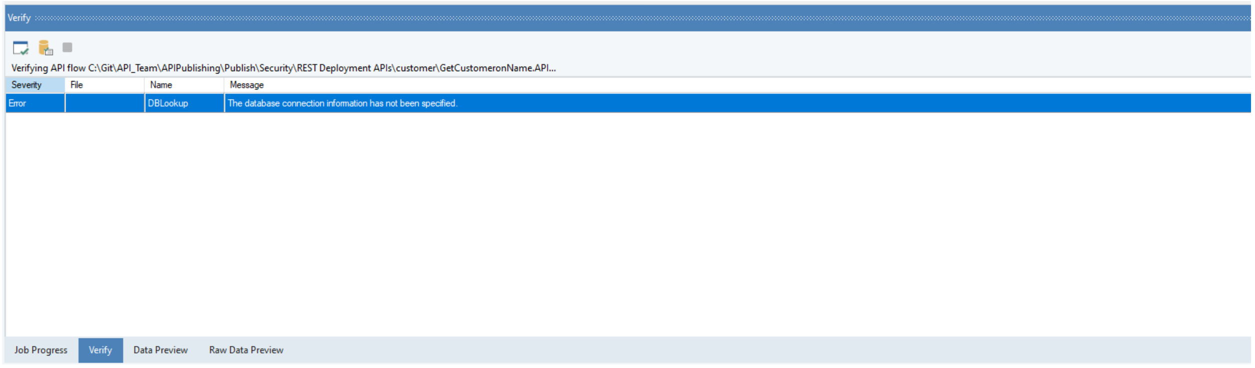
The initial process before the creation of deployment is the verification of the API Flow(s). By default, the deployment is verified in pushdown mode. If the flows are not pushdown-able, they are verified in the non-pushdown mode. To learn about pushdown mode, click here.

If the API deployment contains any errors or warnings, the deployment process is terminated with a link provided in the Job Progress window.


Job Progress
133 - Group Api flow Deployment
135 - Deploying DeploymentCustomerName
Job Progress
Verify
Data Preview
Raw Data Preview

```
2070  VMQA421:9261    135  : 21/06/2022 3:12:41 am: Verifying current document for the deployment...
2071  VMQA421:9261    135  : 21/06/2022 3:12:44 am: An error has occurred: Flow file://C:\Git\API_Team\APIPublishing\Publish\Security\REST%20Deployment%20APIs\customer\GetCustomeronName.API has verification errors.
2072  VMQA421:9261    135  : 21/06/2022 3:12:44 am: View verification logs for latest job. They are generated at runtime and not saved.
```

Clicking on this *View Verification Logs* link opens the Verify window. Here, we can see the verification logs. Its shows the *Severity* i.e., Error or Warning, the *Name* of the object which contains the issue, and the *Message* which is the description of the error/warning.



Verifying API flow C:\Git\API_Team\APIPublishing\Publish\Security\REST Deployment APIs\customer\GetCustomeronName.API...

Severity	File	Name	Message
Error		DBLookup	The database connection information has not been specified.

Please note that the verification process for both the Flow level and Group Level deployment is the same.

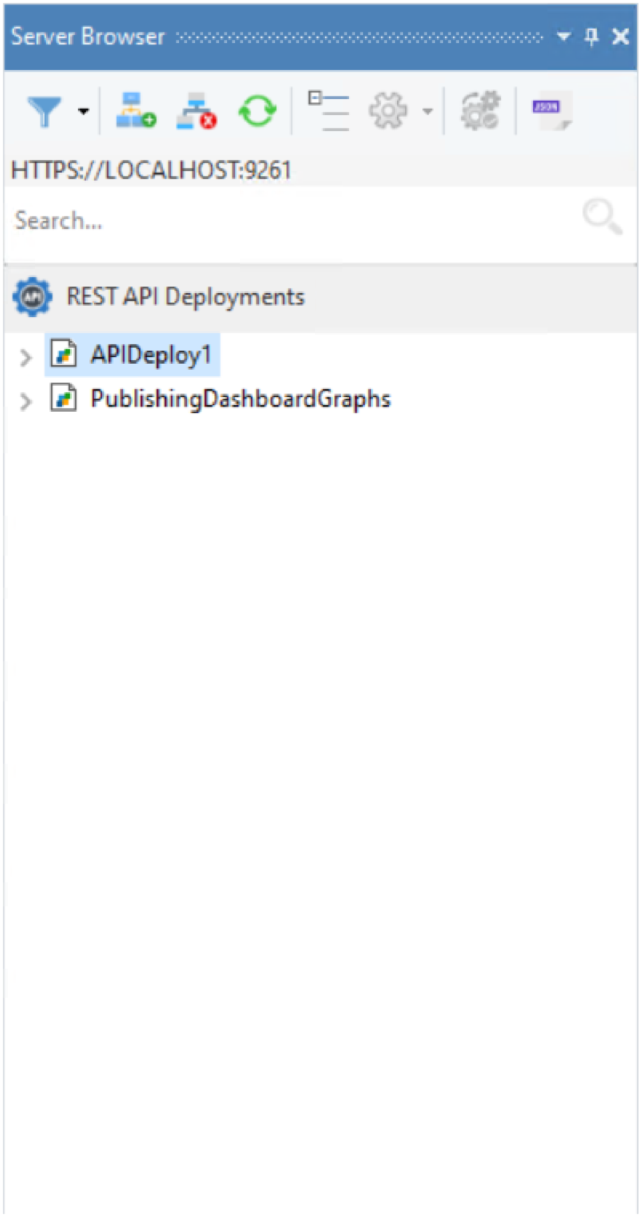
This concludes our discussion on Test flow Generation.

10.13 Server Browser Functionalities for API Publishing

The Server Browser in Astera API Management can be used to see all the Deployments/APIs/API services that the user has deployed onto the Astera Integration Server.

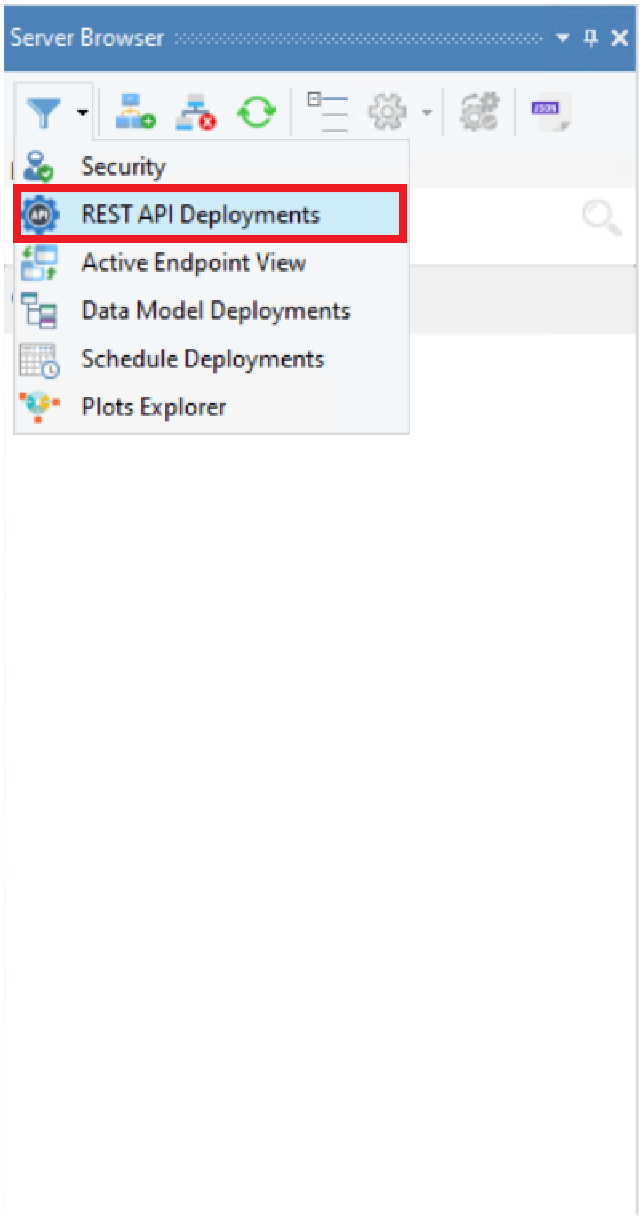
10.13.1 REST API Deployment View

Once we have deployed our API flows, we can see the deployment in the Server Browser,

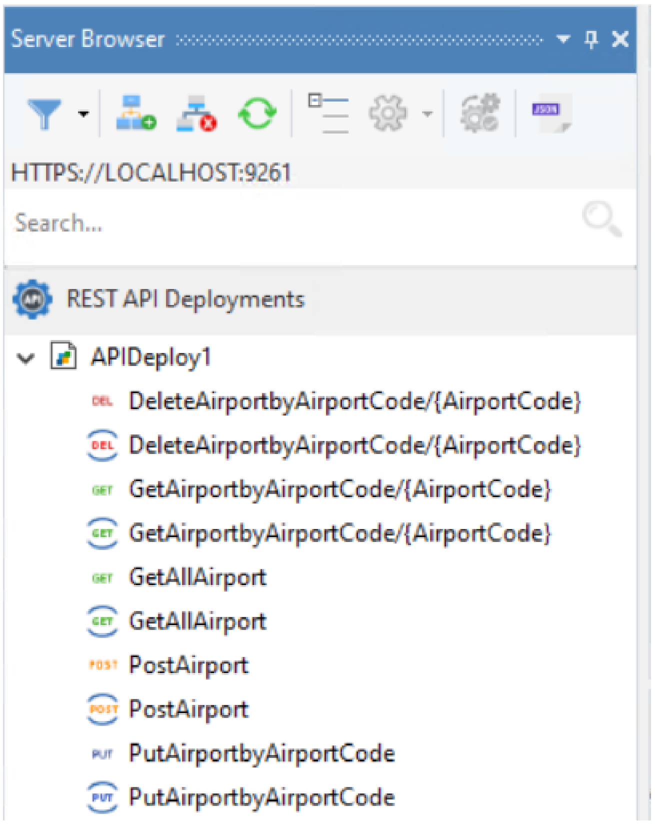


We can select a different view if we click on the *Select Deployment View - Filter* option (the filter icon on the left) in the Server Browser toolbar.

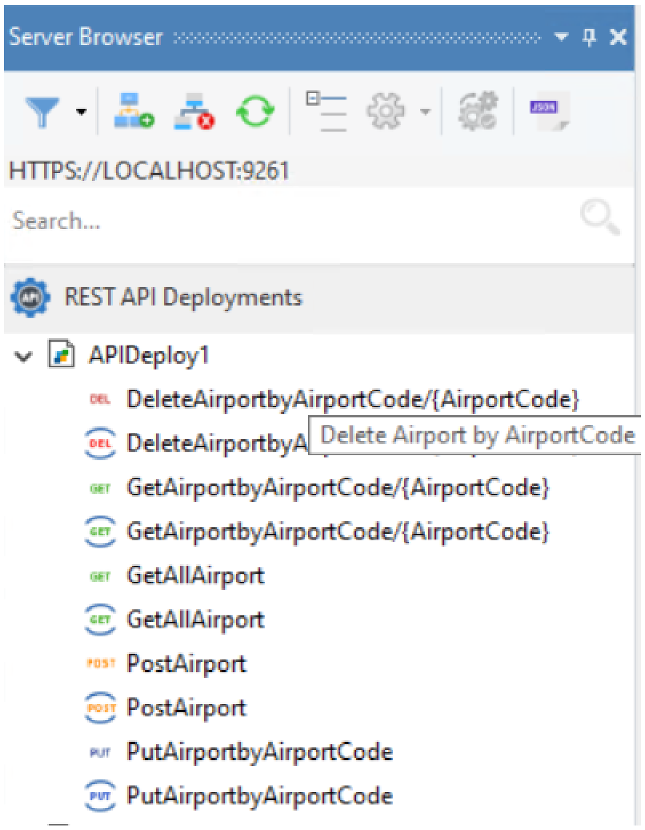
Let us select *REST API Deployments* from the drop-down menu.



This will open a new view for the user. Here, you can directly see the deployments that have been made by the user(s). If we expand a deployment, we can see all the available endpoints under it.



When we deploy the API flow, a description is automatically added, for each endpoint/API flow. This is with respect to the action performed by the API flow.


Server Browser

HTTPS://LOCALHOST:9261

Search...

REST API Deployments

APIDeploy1

DeleteAirportbyAirportCode/{AirportCode}

DeleteAirportbyA Delete Airport by AirportCode

GetAirportbyAirportCode/{AirportCode}

GetAirportbyAirportCode/{AirportCode}

GetAllAirport

GetAllAirport

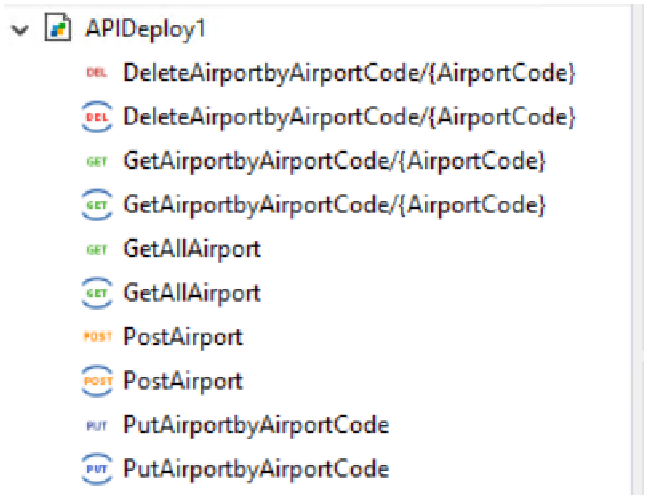
PostAirport

PostAirport

PutAirportbyAirportCode

PutAirportbyAirportCode

Note: We can see two entries for each endpoint because each endpoint can be processed Synchronously and Asynchronously.


APIDeploy1

DeleteAirportbyAirportCode/{AirportCode}

DeleteAirportbyAirportCode/{AirportCode}

GetAirportbyAirportCode/{AirportCode}

GetAirportbyAirportCode/{AirportCode}

GetAllAirport

GetAllAirport

PostAirport

PostAirport

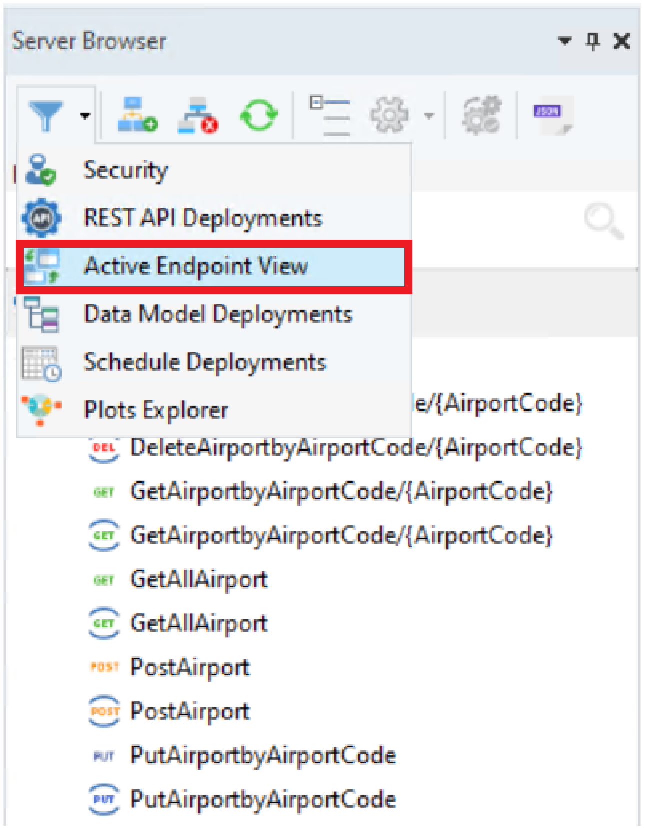
PutAirportbyAirportCode

PutAirportbyAirportCode

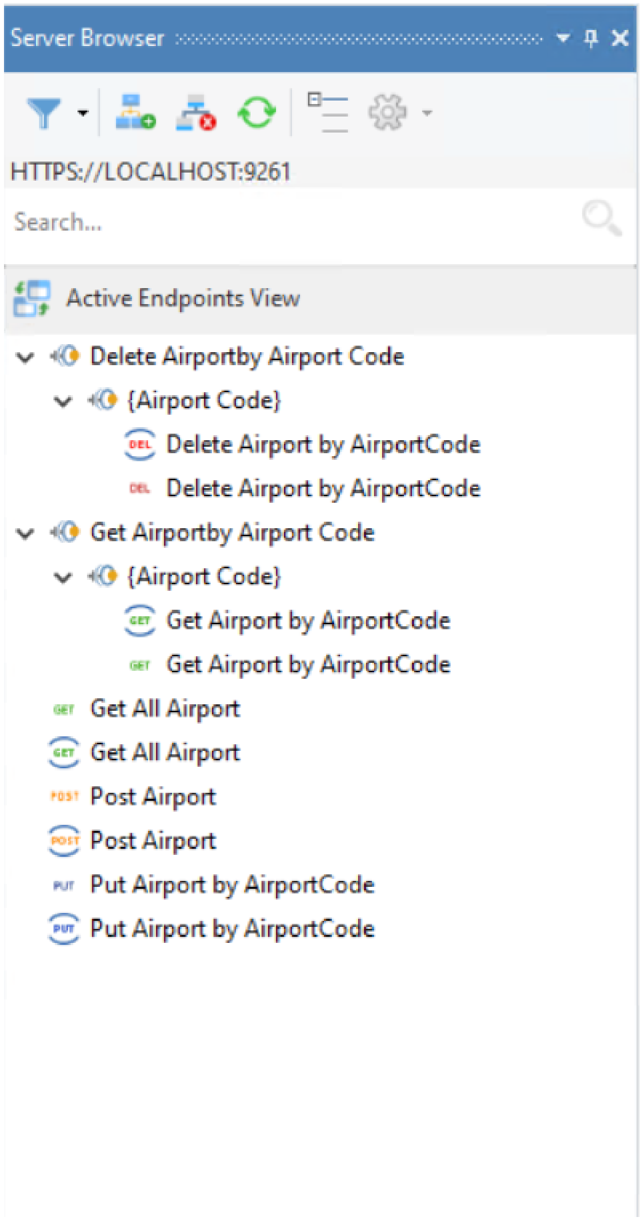
Synchronously processed endpoints can be seen with the HTTP method on its own and Asynchronously processed endpoints can be seen with the HTTP method encircled with blue curves.

10.13.2 Active Endpoint View

We can see a consolidated view containing only the active endpoints from each deployment using the following option.

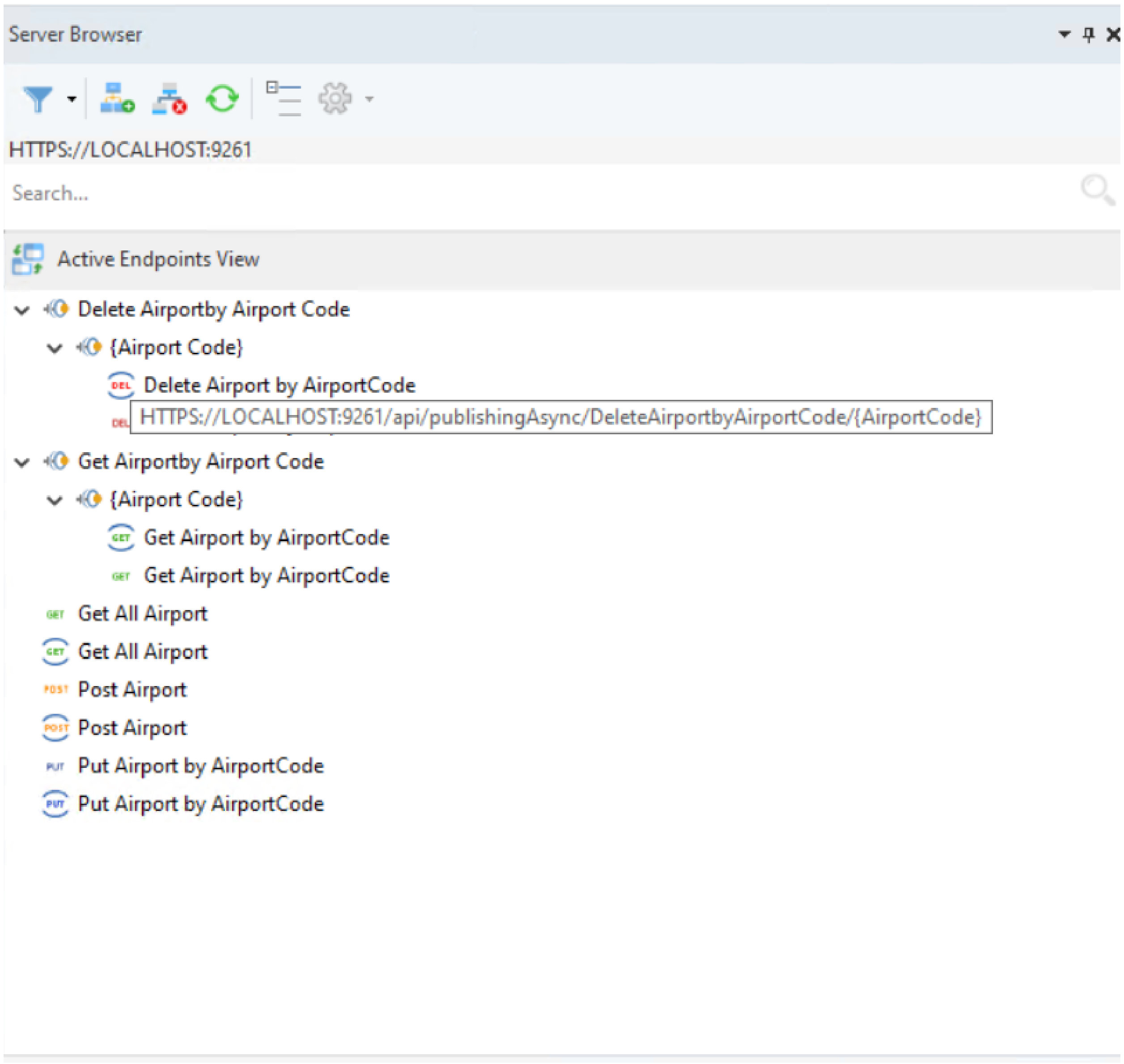


This will show the user a different view,



Delete Airportby Airport Code
{Airport Code}
Delete Airport by AirportCode
Delete Airport by AirportCode
Get Airportby Airport Code
{Airport Code}
Get Airport by AirportCode
Get Airport by AirportCode
Get All Airport
Get All Airport
Post Airport
Post Airport
Put Airport by AirportCode
Put Airport by AirportCode

We can see the endpoints in a tree-like or hierarchical structure. Each endpoint is characterized based on its resource. Upon hovering over each endpoint, their *Request URL* can be seen,

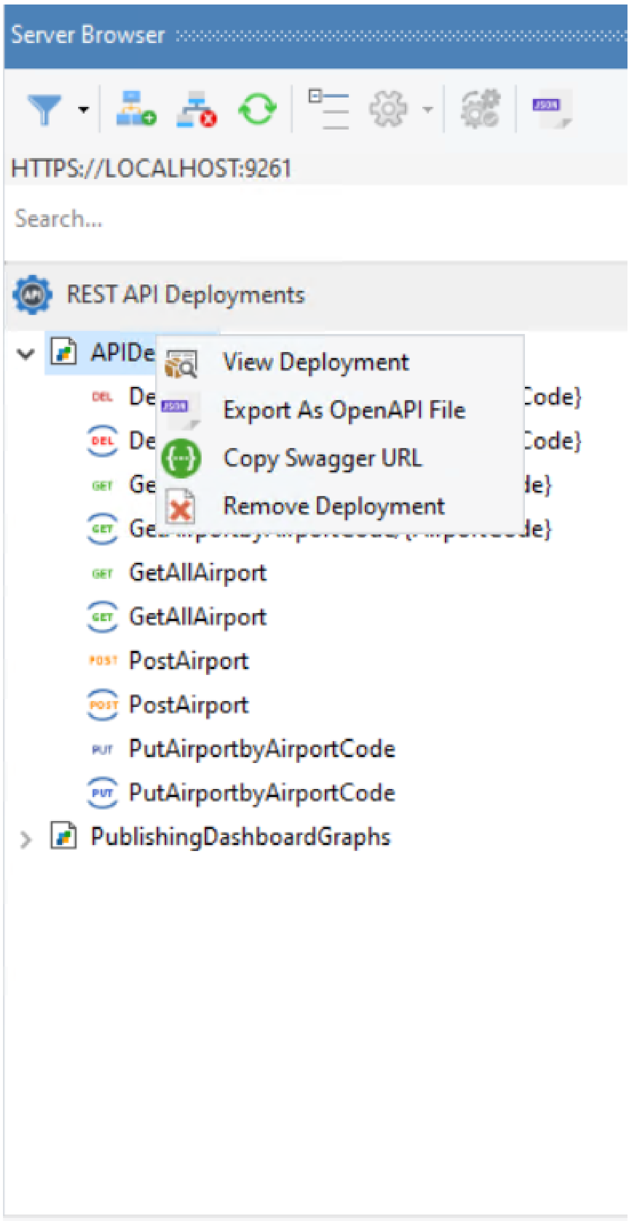


Delete Airportby Airport Code
{Airport Code}
Delete Airport by AirportCode
HTTPS://LOCALHOST:9261/api/publishingAsync/DeleteAirportbyAirportCode/{AirportCode}
Get Airportby Airport Code
{Airport Code}
Get Airport by AirportCode
Get Airport by AirportCode
Get All Airport
Get All Airport
Post Airport
Post Airport
Put Airport by AirportCode
Put Airport by AirportCode

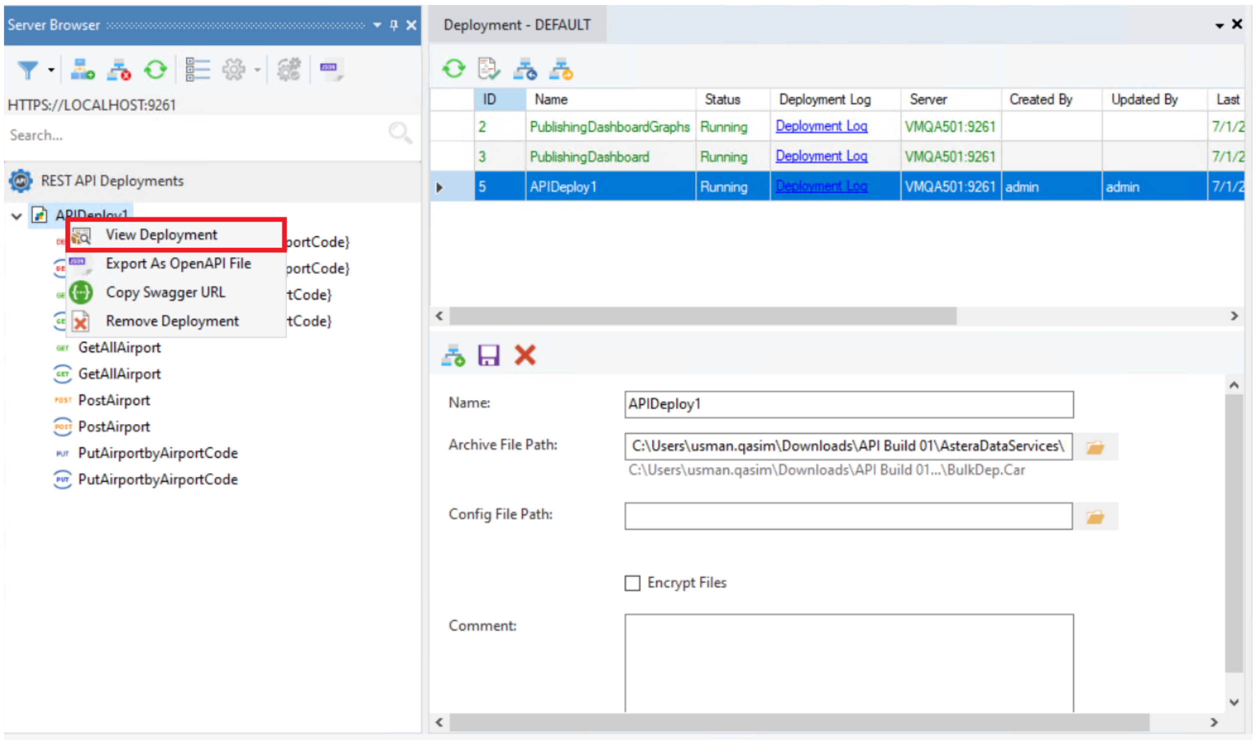
10.13.3 Context Options

Deployment Context Options

If we move back to the REST API Deployment View, we can see some options in the context menu of each of the deployments.



View Deployment: Selecting this option will let the user view the deployment in the deployment manager.



ID	Name	Status	Deployment Log	Server	Created By	Updated By	Last
2	PublishingDashboardGraphs	Running	Deployment Log	VMQA501:9261			7/1/2
3	PublishingDashboard	Running	Deployment Log	VMQA501:9261			7/1/2
5	APIDeploy1	Running	Deployment Log	VMQA501:9261	admin	admin	7/1/2

Name: APIDeploy1

Archive File Path: C:\Users\usman.qasim\Downloads\API Build 01\AsteraDataServices\
C:\Users\usman.qasim\Downloads\API Build 01...\BulkDep.Car

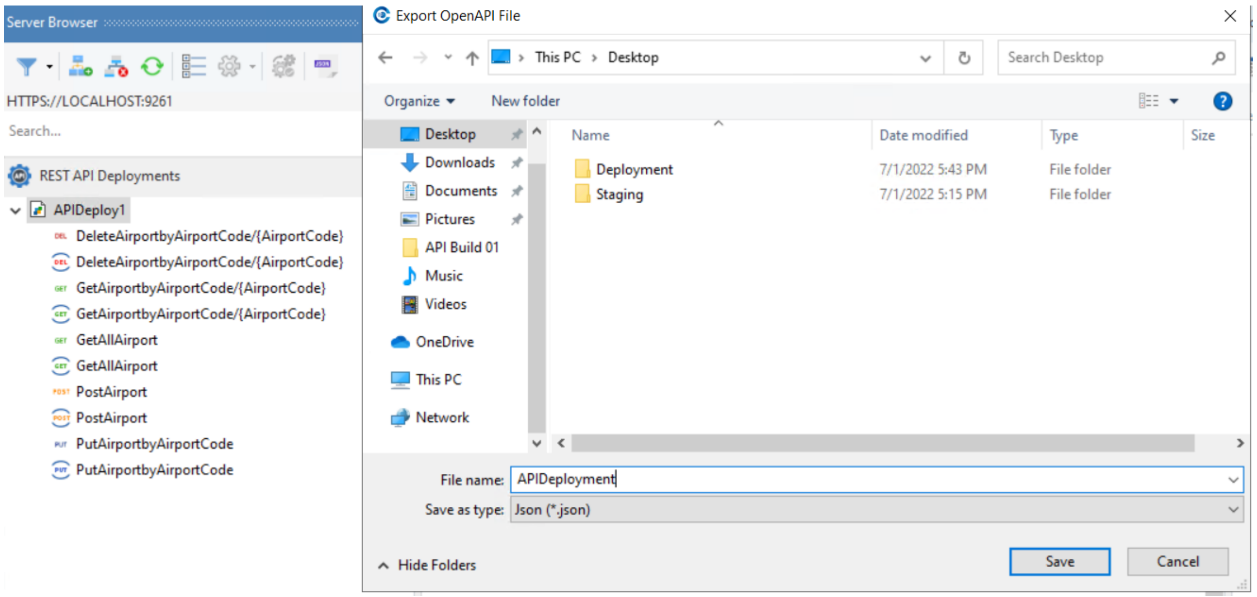
Config File Path:

☐ Encrypt Files

Comment:

Export as OpenAPI file: Selecting this option will allow the user to generate an Open API specification JSON file which we can export/save to the desired location, be it local or on a network.

This file can be used to import the API collection to any third-party tool i.e., Postman, Insomnia, etc., for consumption.

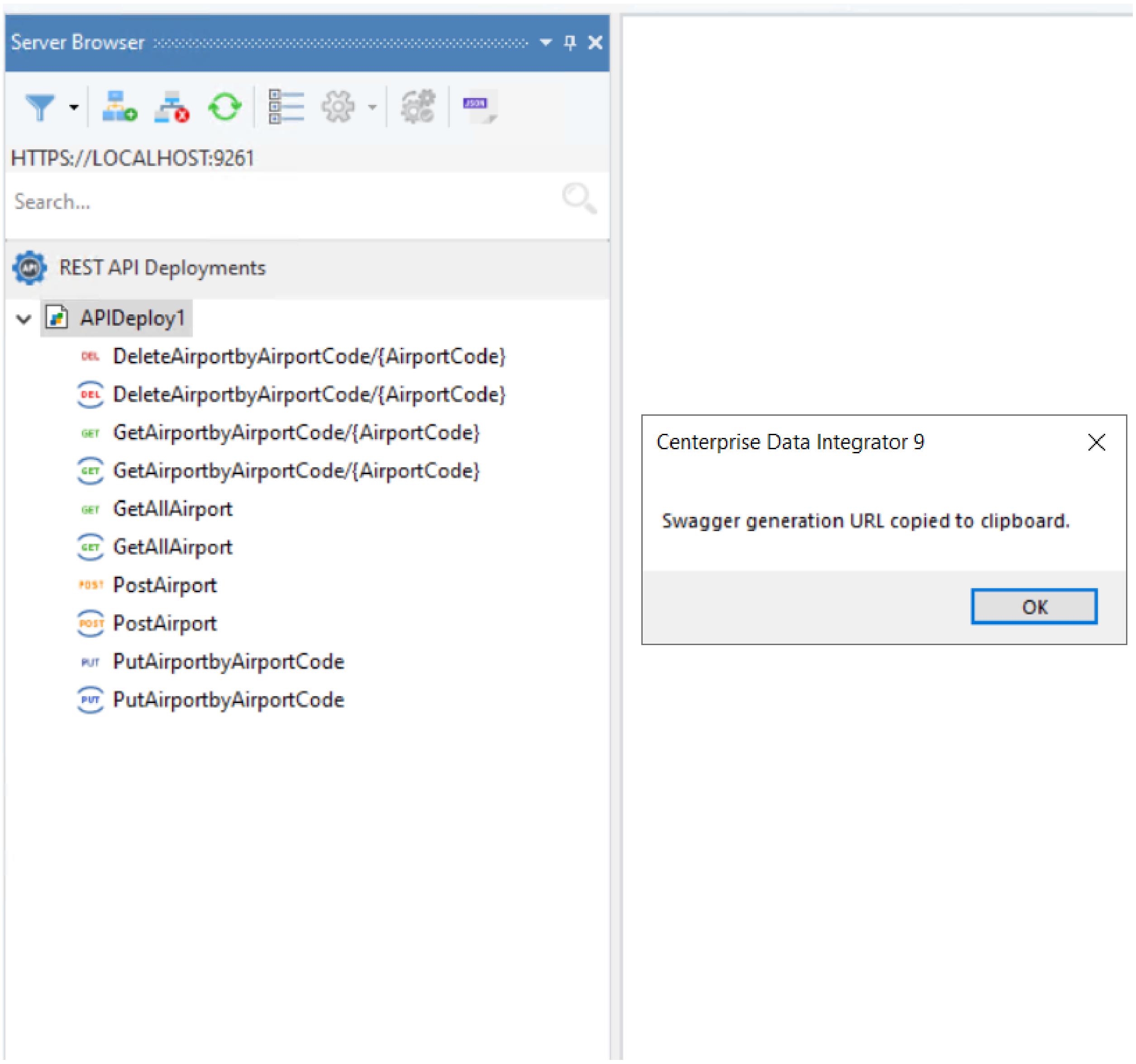


Name	Date modified	Type	Size
Deployment	7/1/2022 5:43 PM	File folder	
Staging	7/1/2022 5:15 PM	File folder	

File name: APIDeployment

Save as type: Json (*.json)

Copy Swagger URL: This option lets the user copy the Swagger URL for the deployment. We can use this URL to generate the swagger definition file for the API collection.

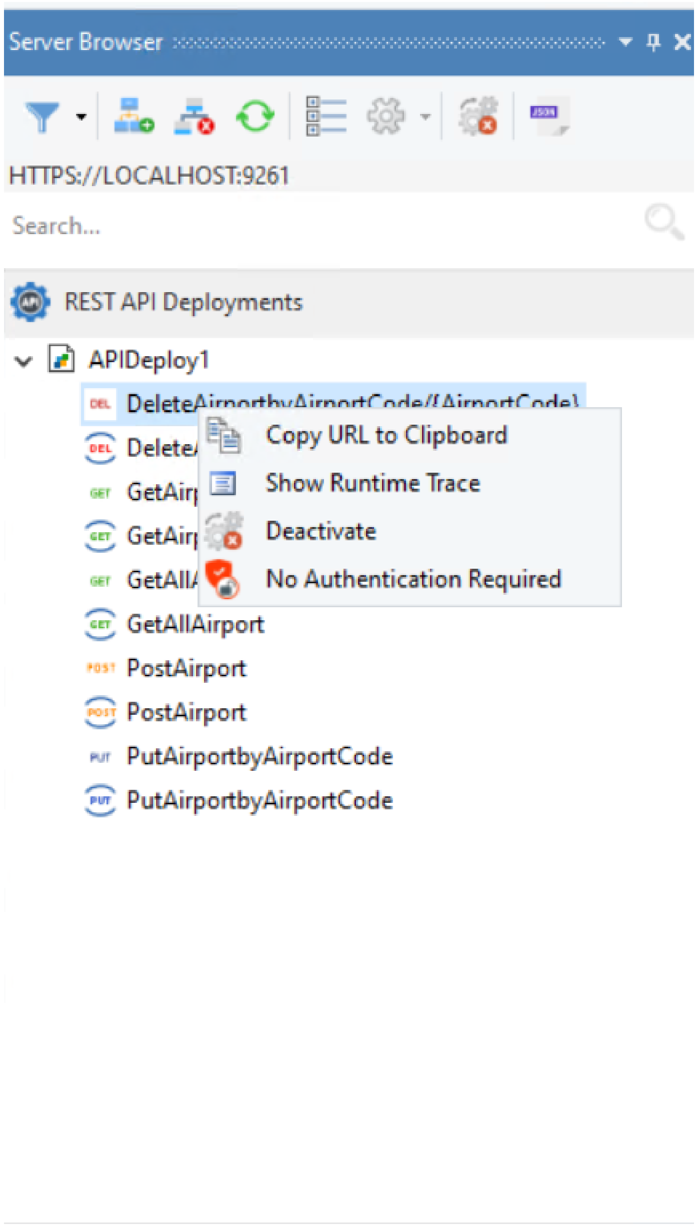

Server Browser
HTTPS://LOCALHOST:9261
Search...
REST API Deployments
APIDeploy1
DeleteAirportbyAirportCode/{AirportCode}
DeleteAirportbyAirportCode/{AirportCode}
GetAirportbyAirportCode/{AirportCode}
GetAirportbyAirportCode/{AirportCode}
GetAllAirport
GetAllAirport
PostAirport
PostAirport
PutAirportbyAirportCode
PutAirportbyAirportCode
Centerprise Data Integrator 9
OK

Swagger generation URL copied to clipboard.

Remove Deployment: Selecting this option will remove the selected deployment from the Server Browser.

Endpoint Context Options

Each deployment can either have a single endpoint or multiple ones. Similar to deployment, a context menu is available for each endpoint as well.



Copy URL to Clipboard: This option allows the user to copy the endpoint's request URL to the clipboard.

Show Runtime Trace: Selecting this option will show the runtime trace for that endpoint.

Deactivate: Selecting this option will deactivate a particular endpoint.

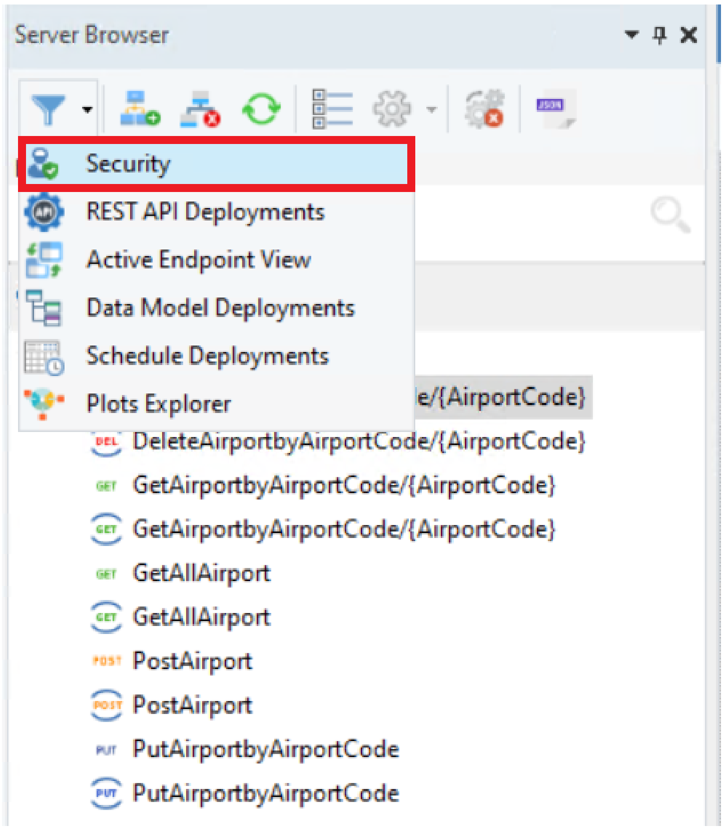
Note: Select the *Activate* option to re-activate the service.

No Authentication Required: Selecting this option will disable the authentication required by this endpoint. To enable the authentication, open the context menu again and select the *Authentication Required* option.

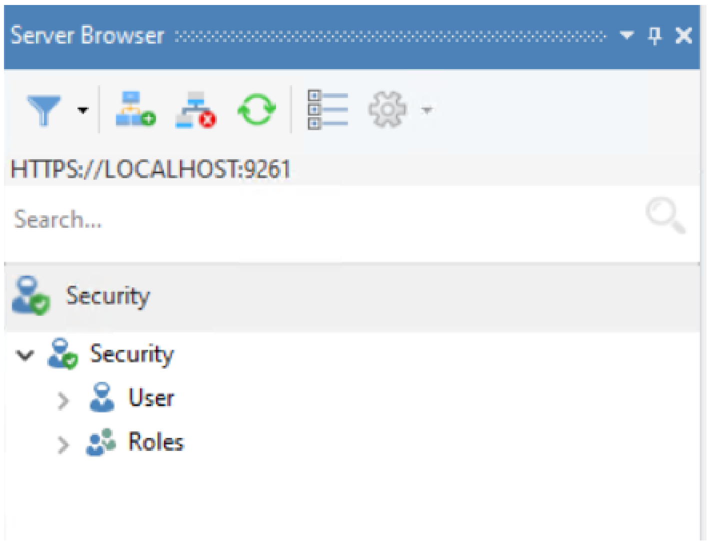
10.13.4 Security

In terms of security, Astera Centerprise gives the user the ability to define roles and provide resources to each role.

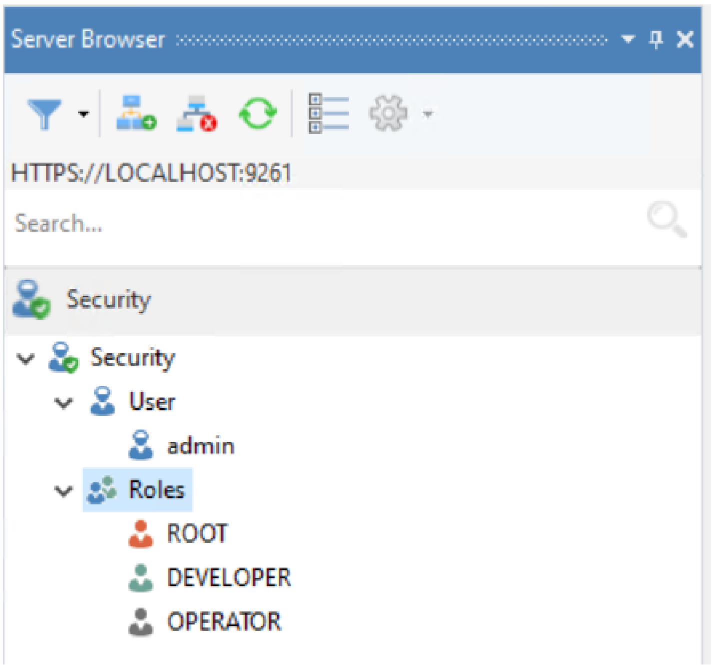
The security view in the Server Browser can be selected from the Server Browser drop-down menu,



This will open a new view,



Expanding the *User* and *Roles* nodes shows us the centerprise client's users and the available roles respectively.



Note:

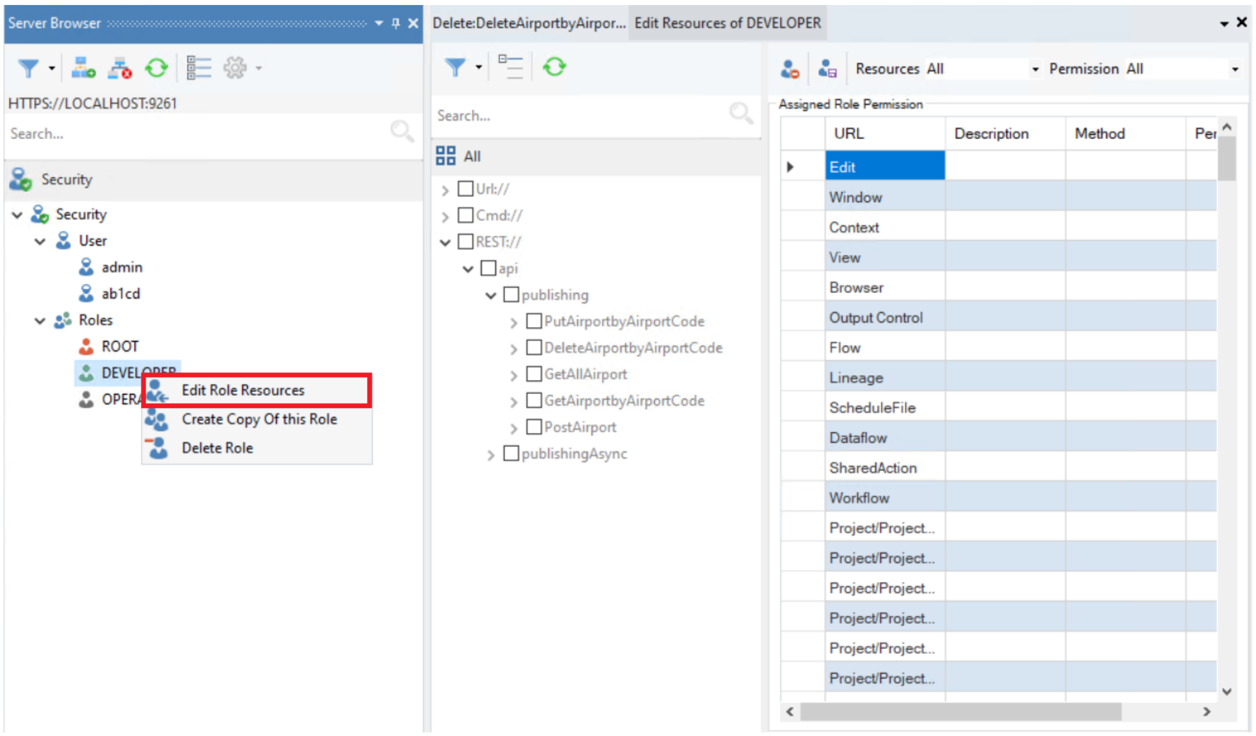
- New users can be added by right-clicking on the *User* option and selecting *Register User* from the context menu.
- New roles can be added by right-clicking on the *Roles* option and selecting *Add New Role* from the context menu.

Resources of each role can be allocated by right-clicking on the role and selecting *Edit Role Resources* from the context menu.

Here, the resources available below are,

- *URL*
- *Cmd*
- *REST – Publishing and Publishing Async*

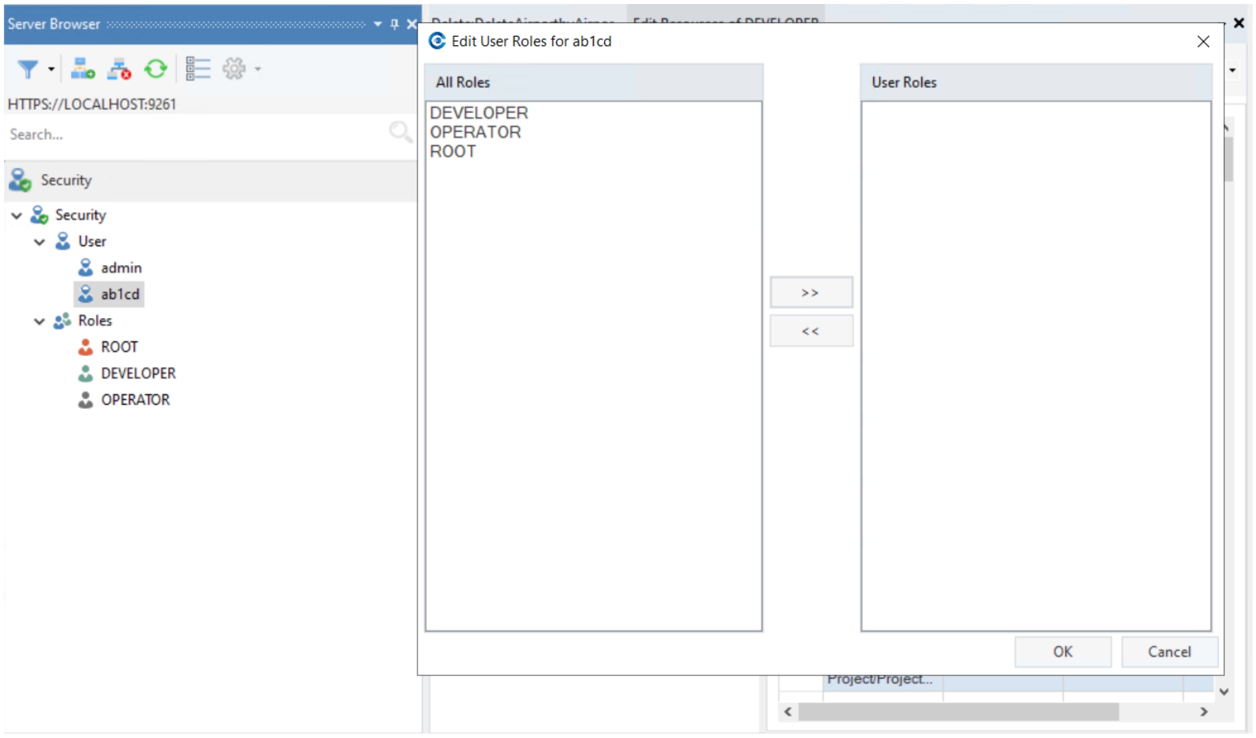
Note: A user needs to have the REST API resources enabled for the deployed API to avail the API services. Otherwise, they might never be able to use the deployed APIs.



URL	Description	Method	Per
Edit			
Window			
Context			
View			
Browser			
Output Control			
Flow			
Lineage			
ScheduleFile			
Dataflow			
SharedAction			
Workflow			
Project/Project...			
Project/Project...			
Project/Project...			
Project/Project...			
Project/Project...			
Project/Project...			

Note: Users will only be able to access the resources that have been allocated to their role.

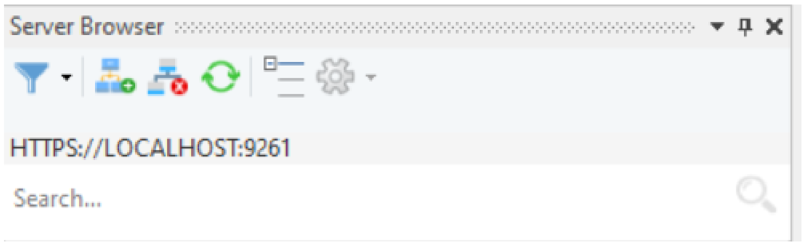
To assign a role to a user, right-click on the user and select *Edit User Roles* from the context menu.



Note: Selecting the specific role and then clicking the right-facing arrow will assign the role to the user. Clicking the left-facing arrow after checking a role will remove the role from the user.

10.13.5 Additional Server Browser Options

Apart from the deployment view options, these are the following options also present on the main menu bar of the Server Browser.



Add Deployment: Adds a new deployment

Remove All Deployments: Removes all deployments present on the Server Browser

Expand All: Expands each of the nodes of all the deployments.

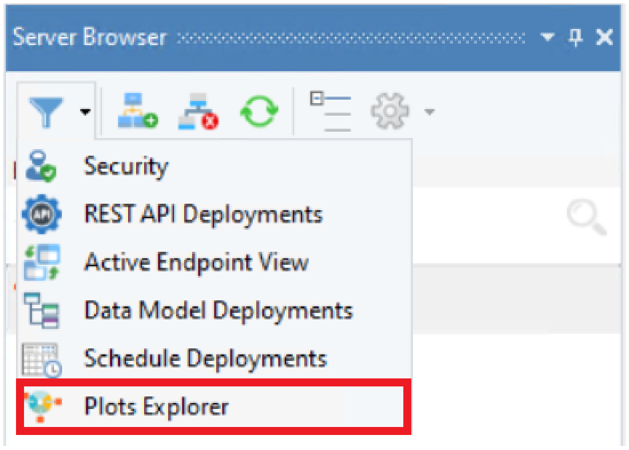
Search Bar: Here, you can write a name to search for any specific deployment or an endpoint.

This concludes the Server Browser functionalities for API Publishing in Astera Centerprise.

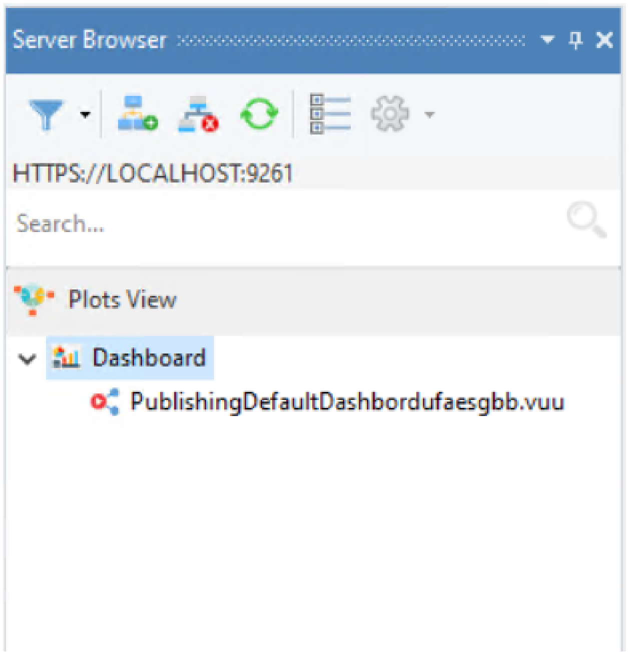
10.14 API Monitoring

Astera API Management lets users monitor live metrics for all deployed APIs using a Visualization Dashboard.

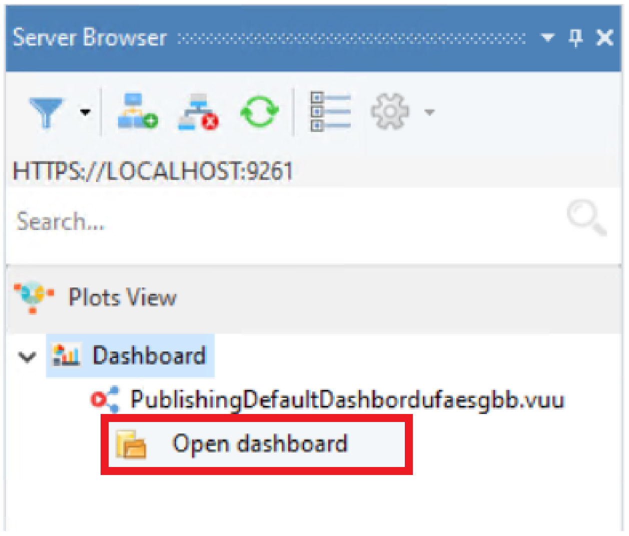
Select the *Plots Explorer* view in the Server Browser.



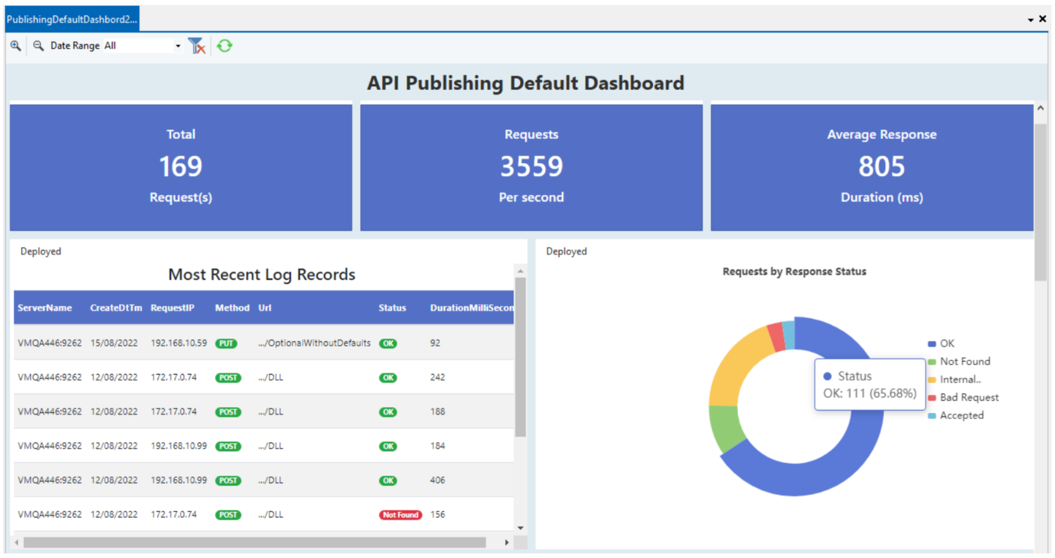
This will open a new view.



Right-click on the option present underneath the *Dashboard* node and select *Open Dashboard* from the context menu.



Selecting this will open the dashboard window.



Most Recent Log Records

ServerName	CreateDtTm	RequestIP	Method	Url	Status	DurationMilliSecon
VMQA446:9262	15/08/2022	192.168.10.59	PUT	.../OptionalWithoutDefaults	OK	92
VMQA446:9262	12/08/2022	172.17.0.74	POST	.../DLL	OK	242
VMQA446:9262	12/08/2022	172.17.0.74	POST	.../DLL	OK	188
VMQA446:9262	12/08/2022	192.168.10.99	POST	.../DLL	OK	184
VMQA446:9262	12/08/2022	192.168.10.99	POST	.../DLL	OK	406
VMQA446:9262	12/08/2022	172.17.0.74	POST	.../DLL	Not Found	156

The dashboard shows various performance metrics and graphs which can be used to monitor the deployed APIs. These include,

Total number of requests

Requests per second - Measures the throughput of the API server, gauging the number of requests the server can handle in a time unit of a second.

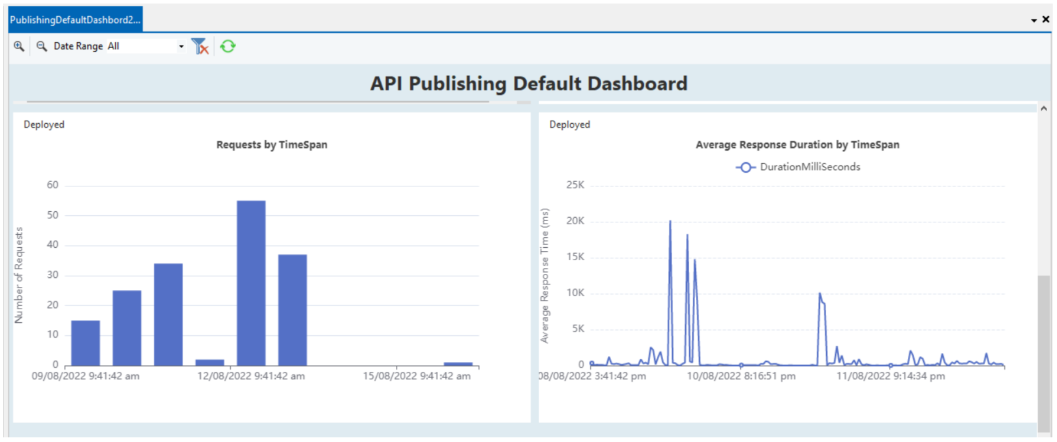
Average response duration - This is a critical KPI that signifies the average time taken for an API to respond.

Most Recent Logs Record - Lists the 10 most recent requests catered along with detailed information about the server and client.

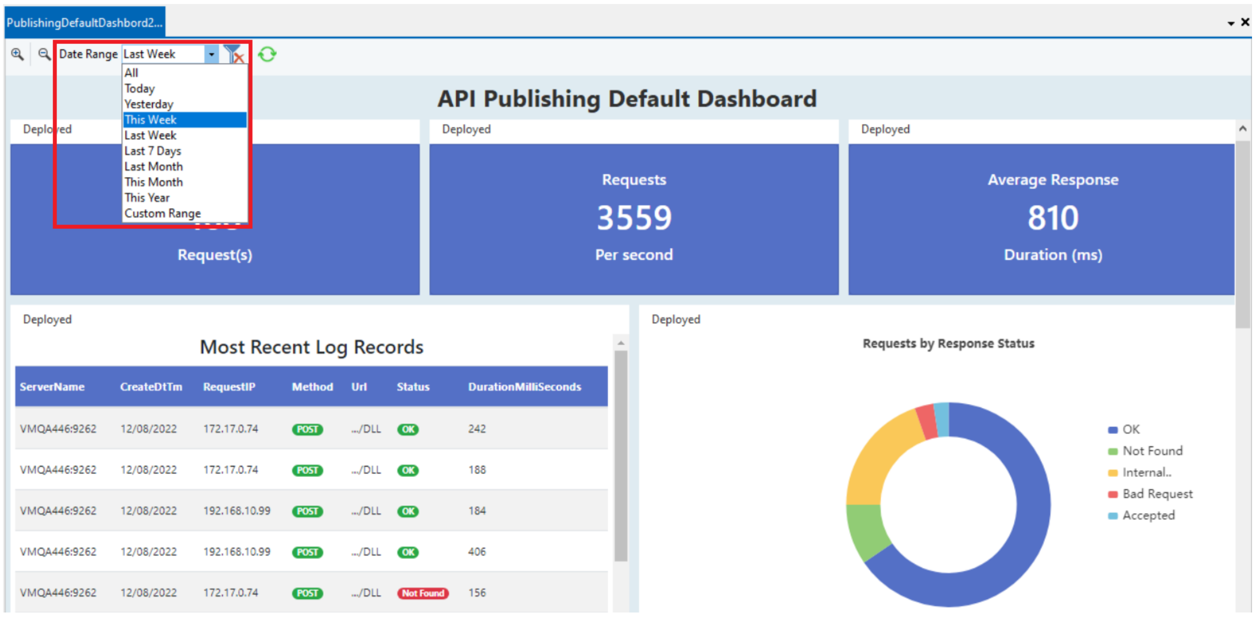
Requests by Response Status - This shows the percentage of each of the different responses for the deployed APIs.

Requests by Timespan - A bar graph that highlights the traffic received as per the number of requests received with time.

Average Response Duration by Timespan - A line graph showing the average response durations with time.



Using the Data Range filter, all these metrics and graphs can also be filtered by a data view.



API Publishing Default Dashboard

Deployed — Request(s)

Deployed — Requests 3559 Per second

Deployed — Average Response 810 Duration (ms)

Deployed — Most Recent Log Records

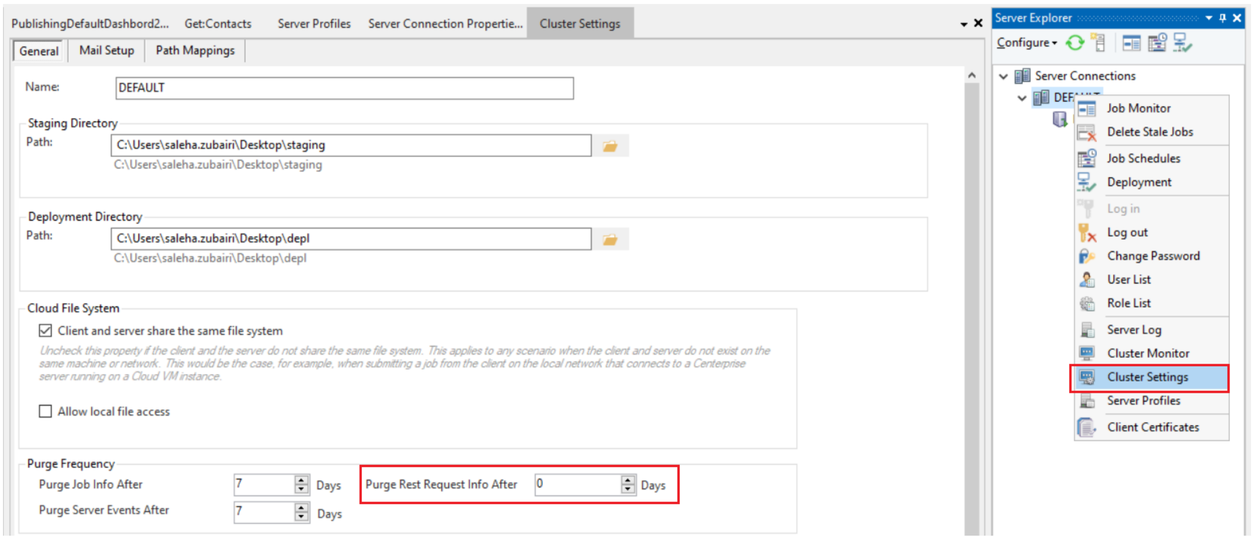
ServerName	CreateDtTm	RequestIP	Method	Url	Status	DurationMilliSeconds
VMQA446:9262	12/08/2022	172.17.0.74	POST	.../DLL	OK	242
VMQA446:9262	12/08/2022	172.17.0.74	POST	.../DLL	OK	188
VMQA446:9262	12/08/2022	192.168.10.99	POST	.../DLL	OK	184
VMQA446:9262	12/08/2022	192.168.10.99	POST	.../DLL	OK	406
VMQA446:9262	12/08/2022	172.17.0.74	POST	.../DLL	Not Found	156

Deployed — Requests by Response Status

Note: The source table for the dashboard can be configured to purge.

From the Server Explorer, right-click on the cluster node and open *Cluster Settings*. Here, you can set the *Purge REST Request Info After* value to enable purging the source table.

A value of 0 signifies that the table would never be purged.



Name: DEFAULT

Staging Directory
Path: C:\Users\saleha.zubairi\Desktop\staging
C:\Users\saleha.zubairi\Desktop\staging

Deployment Directory
Path: C:\Users\saleha.zubairi\Desktop\depl
C:\Users\saleha.zubairi\Desktop\depl

Cloud File System
☑ Client and server share the same file system
Uncheck this property if the client and the server do not share the same file system. This applies to any scenario when the client and server do not exist on the same machine or network. This would be the case, for example, when submitting a job from the client on the local network that connects to a Centerprise server running on a Cloud VM instance.
☐ Allow local file access

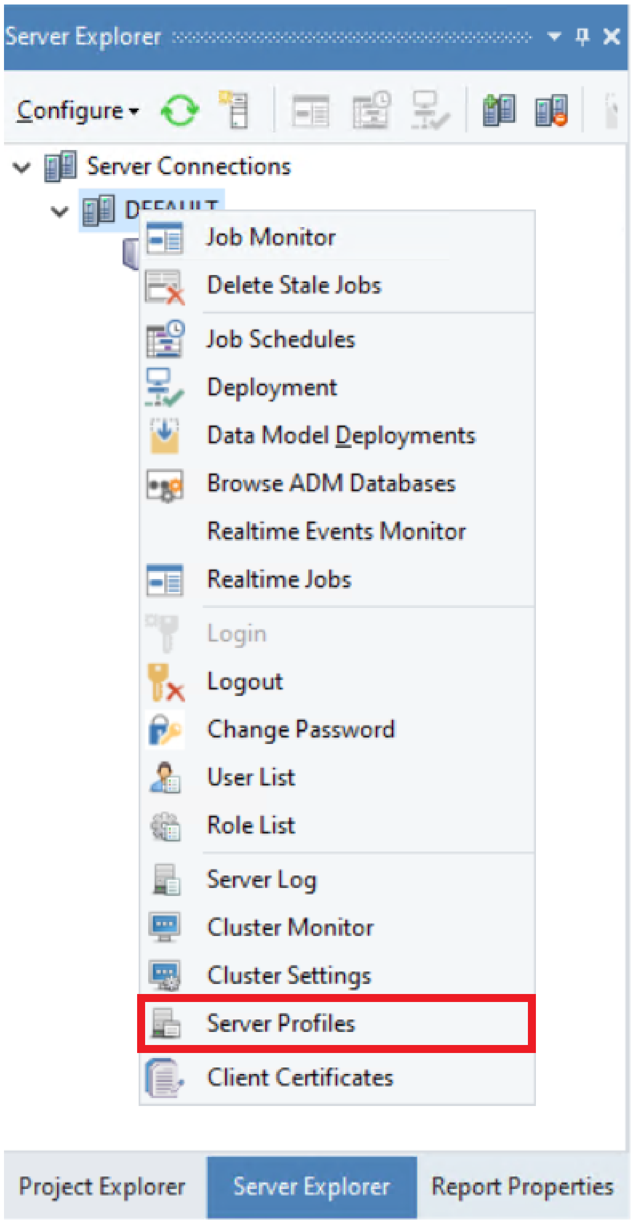
Purge Frequency
Purge Job Info After 7 Days
Purge Server Events After 7 Days
Purge Rest Request Info After 0 Days

This concludes API Monitoring in Astera API Management.

10.15 Logging and Tracing

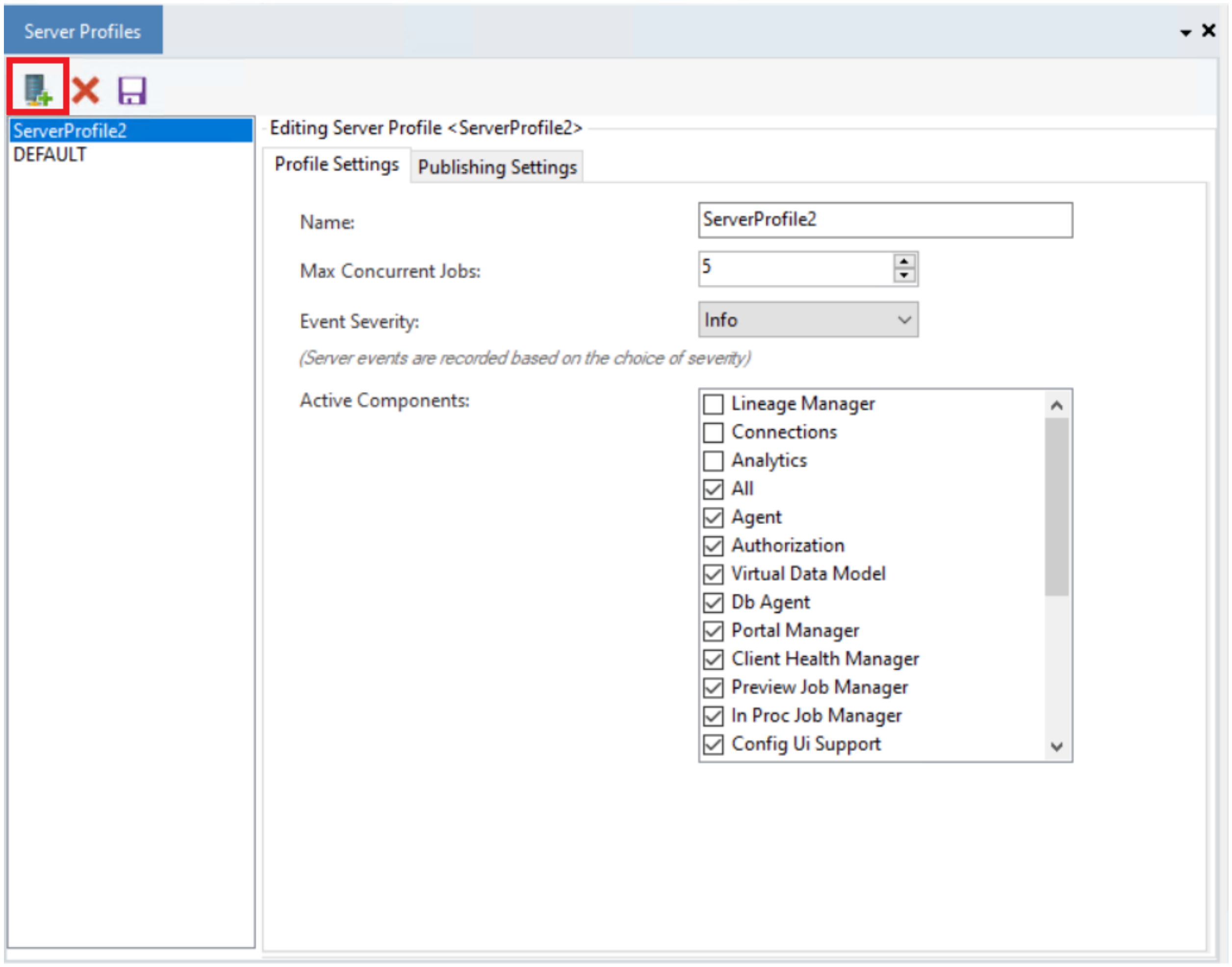
In Astera API Management, users can troubleshoot runtime issues by monitoring live tracing for any APIs deployed on the server.

To configure logging, create a new server profile by right-clicking on the cluster node in the Server Explorer and selecting *Server Profiles* from the context menu.



This will open a new window.

1. Create a new profile by selecting the *Add a new server profile* option.



Name: ServerProfile2

Max Concurrent Jobs: 5

Event Severity: Info

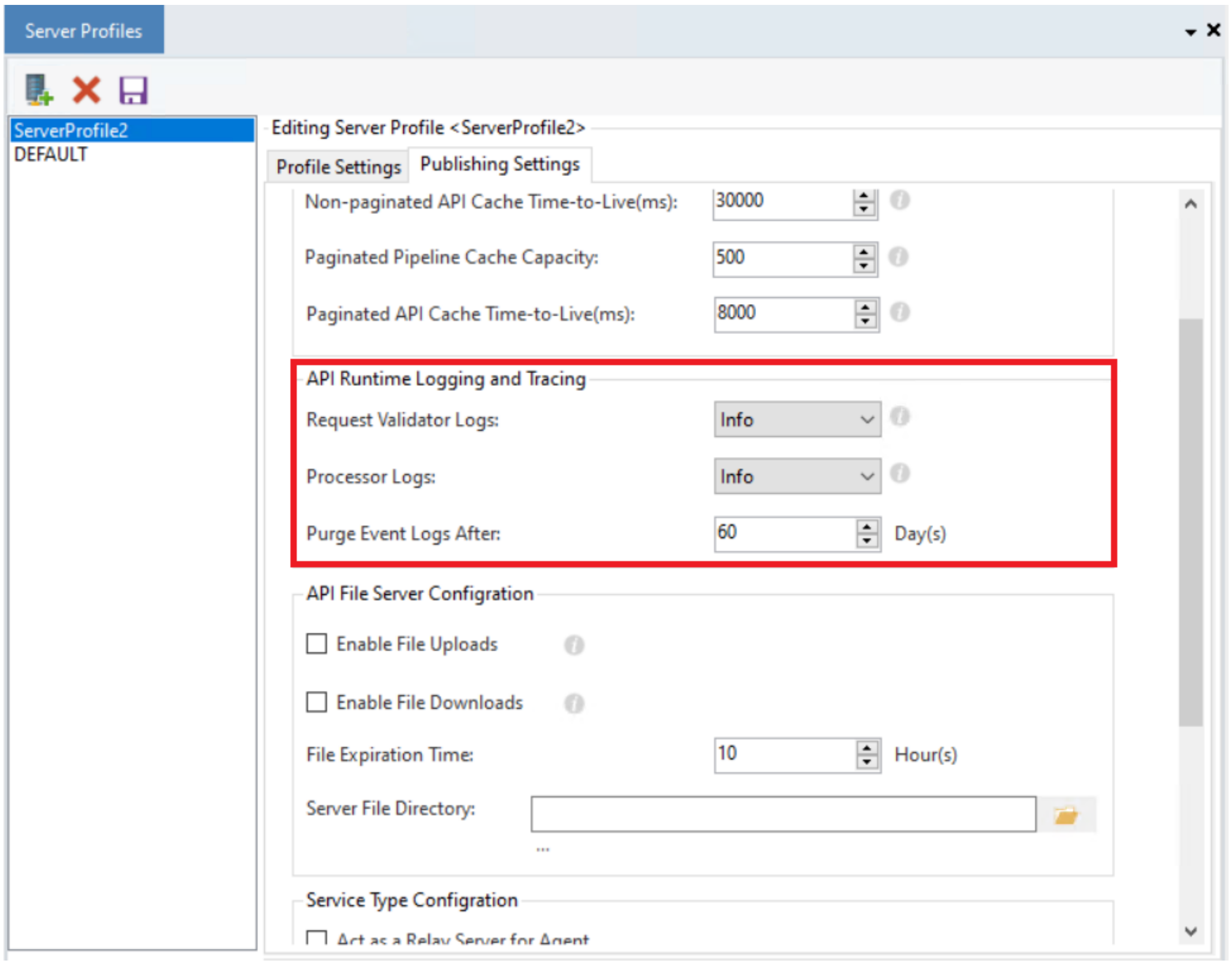
(Server events are recorded based on the choice of severity)

Active Components:

- [ ] Lineage Manager
- [ ] Connections
- [ ] Analytics
- [x] All
- [x] Agent
- [x] Authorization
- [x] Virtual Data Model
- [x] Db Agent
- [x] Portal Manager
- [x] Client Health Manager
- [x] Preview Job Manager
- [x] In Proc Job Manager
- [x] Config Ui Support

As you can see, we already have a server profile created.

2. Select the *Publishing Settings* tab and scroll down to the *API Runtime logging and tracing* section.



Server Profiles

ServerProfile2
DEFAULT

Editing Server Profile <ServerProfile2>

Profile Settings | Publishing Settings

Non-paginated API Cache Time-to-Live(ms): 30000

Paginated Pipeline Cache Capacity: 500

Paginated API Cache Time-to-Live(ms): 8000

API Runtime Logging and Tracing

Request Validator Logs: Info

Processor Logs: Info

Purge Event Logs After: 60 Day(s)

API File Server Configration

☐ Enable File Uploads

☐ Enable File Downloads

File Expiration Time: 10 Hour(s)

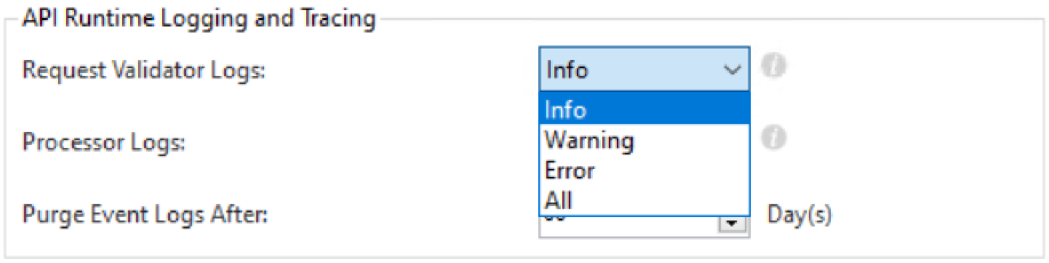
Server File Directory:

Service Type Configration

☐ Act as a Relay Server for Agent

Here, the user can select the level of logs to be traced, including information, warnings, errors, or all-inclusive. The logging stages include,

Request Validator Logs: It includes pre-validating the request context before sending it through the runtime processor by validating the server availability, and deployment activity, and inspecting if the request has the supported formats.



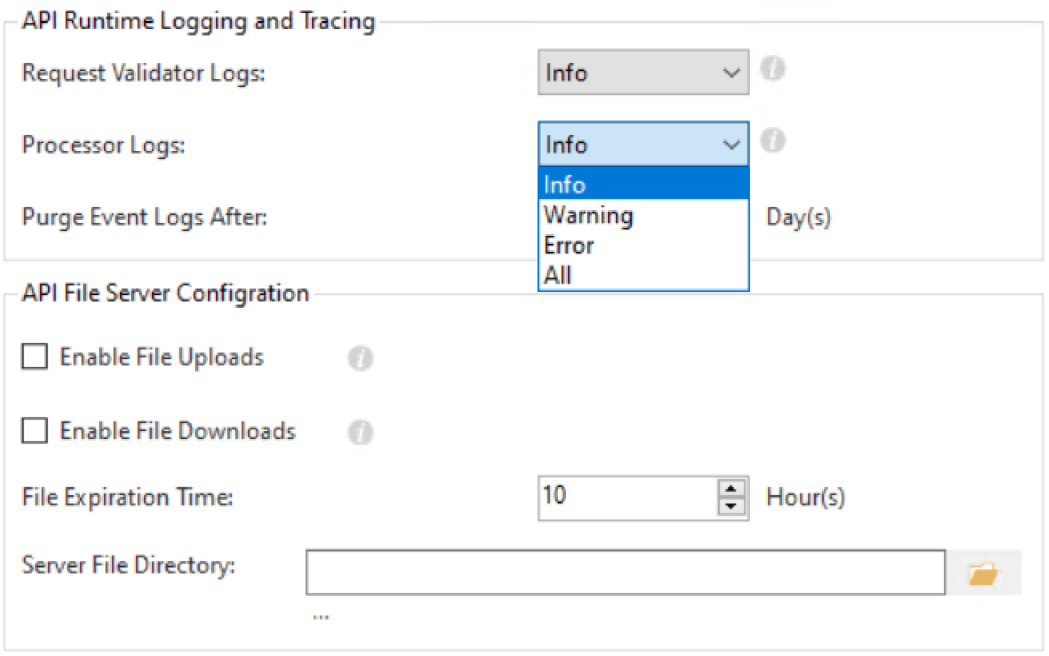
API Runtime Logging and Tracing

Request Validator Logs: Info

Processor Logs:

Purge Event Logs After: Day(s)

Processor Logs: Processor logs include runtime components of the request, including information about a cached request pipeline, the concurrent pipelines in execution, and runtime capacity.



API Runtime Logging and Tracing

Request Validator Logs: Info

Processor Logs: Info

Purge Event Logs After: Day(s)

API File Server Configration

☐ Enable File Uploads

☐ Enable File Downloads

File Expiration Time: 10 Hour(s)

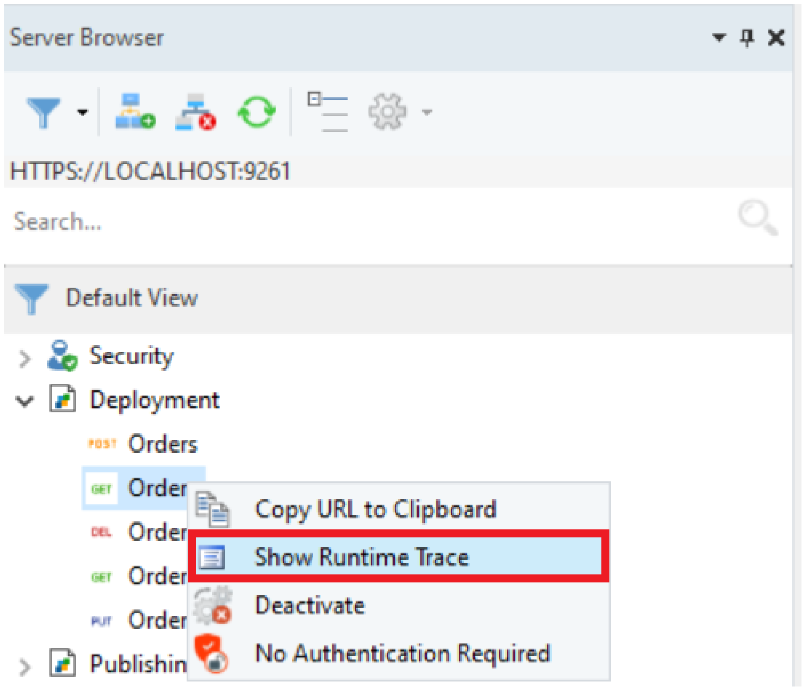
Server File Directory:

Purge Event Logs After: This counter shows the number of days after which the logs will be purged/removed since a lot of them can accumulate at runtime.

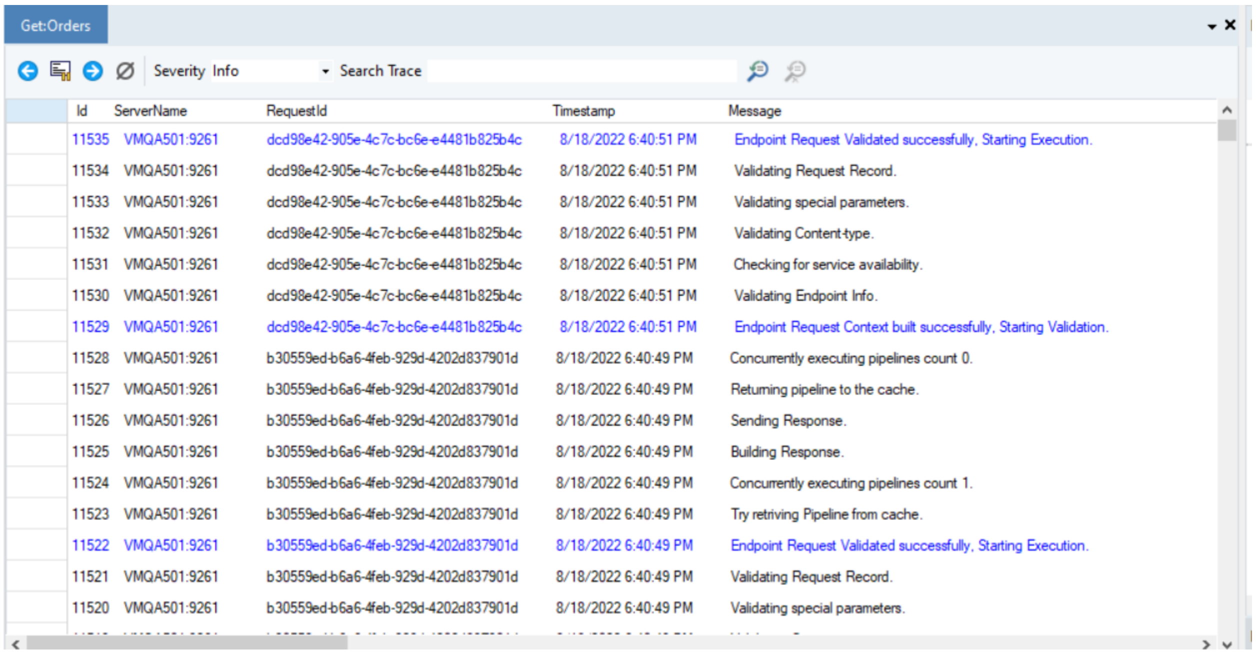
Once the server profile is configured and saved, the next step is to select this profile in the *Server Properties*.

After logging and tracing have been configured, users can now view the live runtime traces generated for all deployed APIs.

Next, go to the Server Browser and open the Deployed Endpoint View. To view the tracing for any deployed API, right-click and select *Show Runtime Trace* for any API deployment listed.



You can now see the trace.



Id	ServerName	RequestId	Timestamp	Message
11535	VMQA501:9261	dcd98e42-905e-4c7c-bc6e-e4481b825b4c	8/18/2022 6:40:51 PM	Endpoint Request Validated successfully, Starting Execution.
11534	VMQA501:9261	dcd98e42-905e-4c7c-bc6e-e4481b825b4c	8/18/2022 6:40:51 PM	Validating Request Record.
11533	VMQA501:9261	dcd98e42-905e-4c7c-bc6e-e4481b825b4c	8/18/2022 6:40:51 PM	Validating special parameters.
11532	VMQA501:9261	dcd98e42-905e-4c7c-bc6e-e4481b825b4c	8/18/2022 6:40:51 PM	Validating Content-type.
11531	VMQA501:9261	dcd98e42-905e-4c7c-bc6e-e4481b825b4c	8/18/2022 6:40:51 PM	Checking for service availability.
11530	VMQA501:9261	dcd98e42-905e-4c7c-bc6e-e4481b825b4c	8/18/2022 6:40:51 PM	Validating Endpoint Info.
11529	VMQA501:9261	dcd98e42-905e-4c7c-bc6e-e4481b825b4c	8/18/2022 6:40:51 PM	Endpoint Request Context built successfully, Starting Validation.
11528	VMQA501:9261	b30559ed-b6a6-4feb-929d-4202d837901d	8/18/2022 6:40:49 PM	Concurrently executing pipelines count 0.
11527	VMQA501:9261	b30559ed-b6a6-4feb-929d-4202d837901d	8/18/2022 6:40:49 PM	Returning pipeline to the cache.
11526	VMQA501:9261	b30559ed-b6a6-4feb-929d-4202d837901d	8/18/2022 6:40:49 PM	Sending Response.
11525	VMQA501:9261	b30559ed-b6a6-4feb-929d-4202d837901d	8/18/2022 6:40:49 PM	Building Response.
11524	VMQA501:9261	b30559ed-b6a6-4feb-929d-4202d837901d	8/18/2022 6:40:49 PM	Concurrently executing pipelines count 1.
11523	VMQA501:9261	b30559ed-b6a6-4feb-929d-4202d837901d	8/18/2022 6:40:49 PM	Try retriving Pipeline from cache.
11522	VMQA501:9261	b30559ed-b6a6-4feb-929d-4202d837901d	8/18/2022 6:40:49 PM	Endpoint Request Validated successfully, Starting Execution.
11521	VMQA501:9261	b30559ed-b6a6-4feb-929d-4202d837901d	8/18/2022 6:40:49 PM	Validating Request Record.
11520	VMQA501:9261	b30559ed-b6a6-4feb-929d-4202d837901d	8/18/2022 6:40:49 PM	Validating special parameters.

This concludes logging and tracing in Astera API Management.

CHAPTER

ELEVEN

API CONSUMPTION

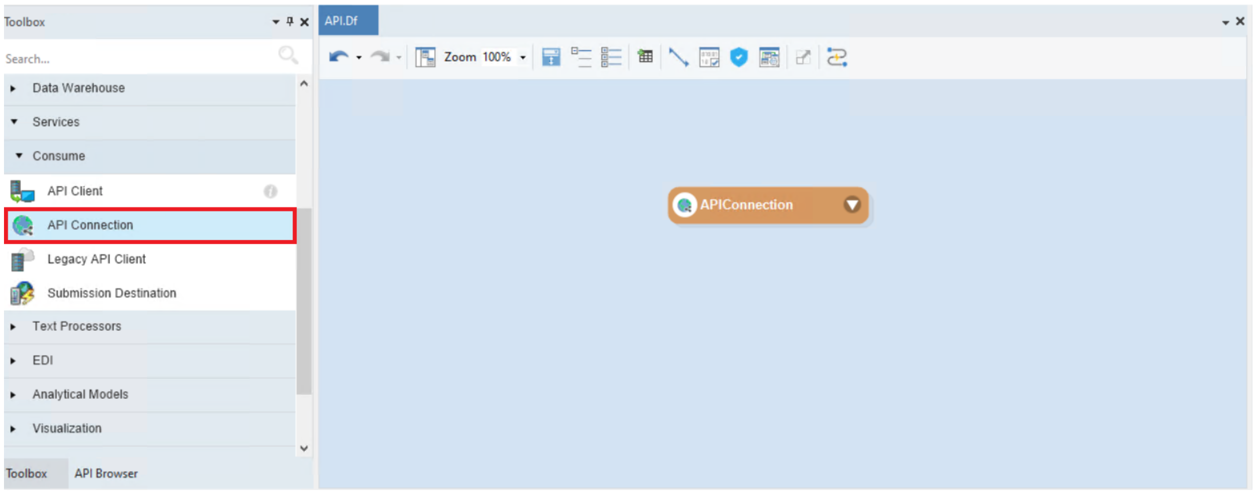
11.1 API Connection

To make an API call, an *API Connection* object needs to be configured first. This object stores all the common information that can be shared across multiple API requests.

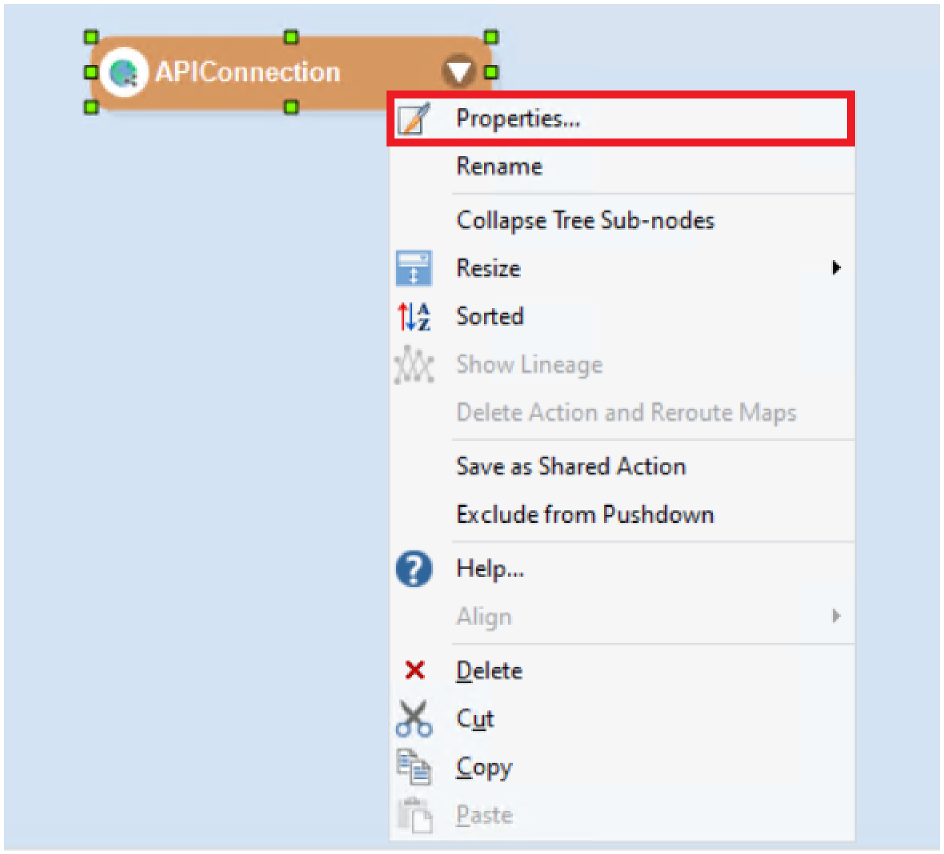
11.1.1 Configuring The API Connection Object

1. Drag-and-drop the *API Connection* object from the Toolbox onto a dataflow.

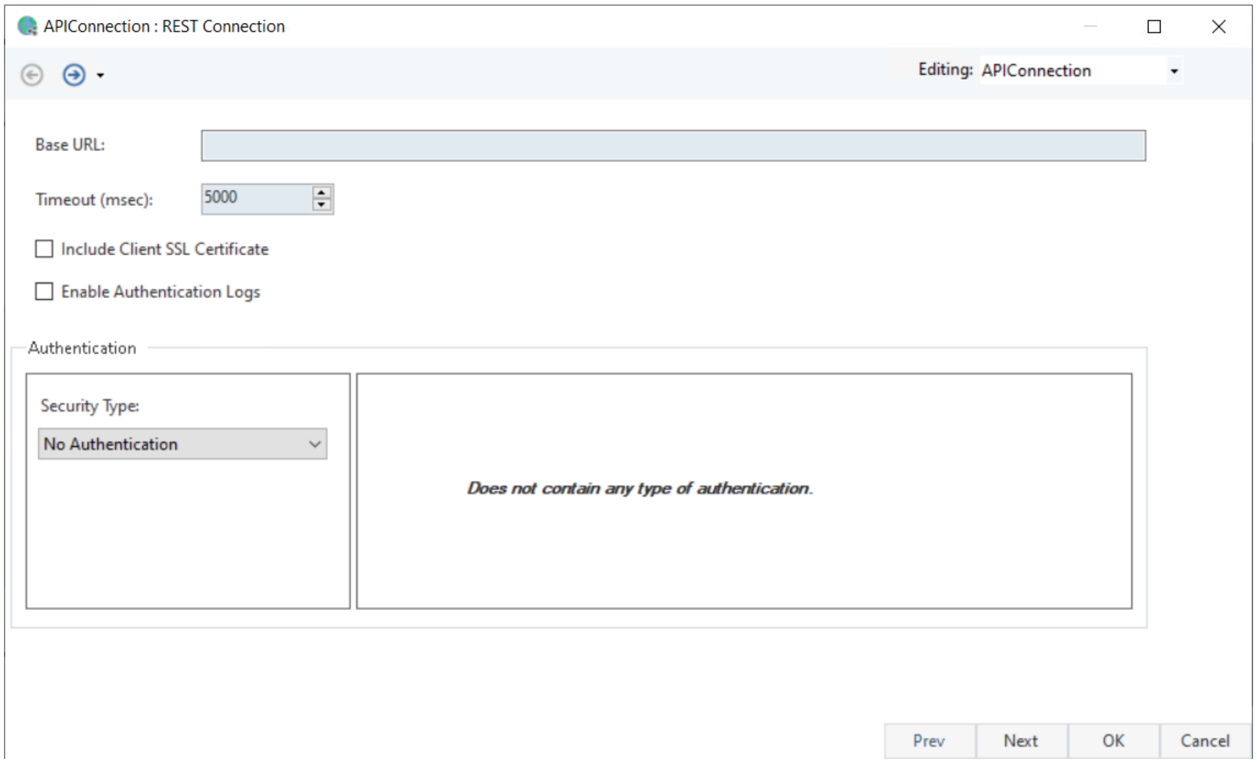
Note: It can also be stored as a shared action file.



2. Right-click on the *API Connection* object and select *Properties* from the context menu.



A configuration window will appear on your screen.



Editing: APIConnection

Base URL:

Timeout (msec): 5000

Include Client SSL Certificate

Enable Authentication Logs

Authentication

Security Type: No Authentication

Does not contain any type of authentication.

Base URL: Here, you can specify the base URL of the API which will prepend as a common path to all API endpoints sharing this connection. A Base URL usually consists of the scheme, hostname, and port of the API web address.

Note: When a user imports an API definition, a shared connection file containing the *Base URL* and authentication type is automatically created within the project. To learn more about importing APIs in Astera Centerprise, click here.

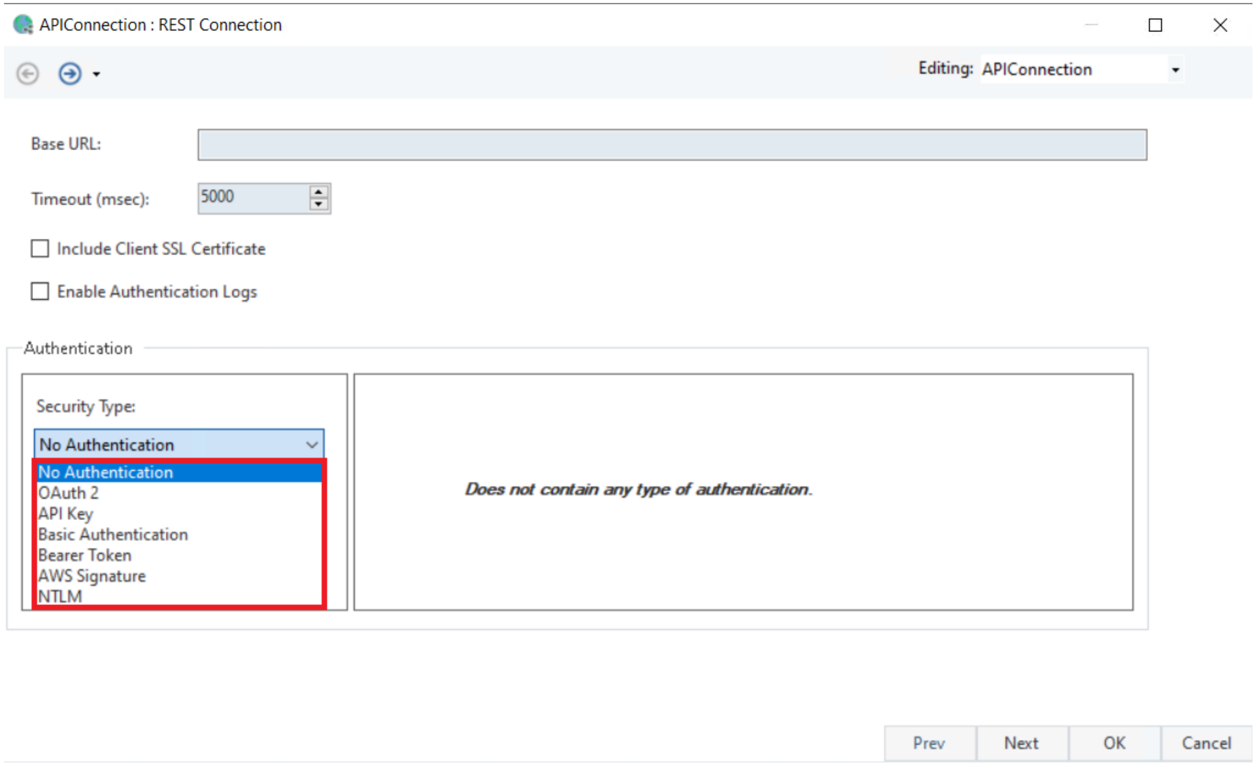
Timeout (msec): Specify the duration, in milliseconds, to wait for the API server to respond before giving a timeout error.

Include Client SSL Certificate: Check this box to include an imported client certificate for the specified base URL. To learn more about importing SSL certificates, click here.

Enable Authentication Logs: Select this checkbox to enable authentication logging for APIs.

Authentication – Security Type: Specify the authentication type for the API.

Astera supports the following authentication types


APIConnection : REST Connection

Editing: APIConnection

Base URL:

Timeout (msec): 5000

☐ Include Client SSL Certificate

☐ Enable Authentication Logs

Authentication

Security Type: No Authentication

- No Authentication
- OAuth 2
- API Key
- Basic Authentication
- Bearer Token
- AWS Signature
- NTLM

Does not contain any type of authentication.

Types Of Authentications:

Identification and verification of a user is an important aspect of authentication. Authentication allows an application to determine whether a user identity is valid/authorized; based on the outcome, a user is provided access control to the application.

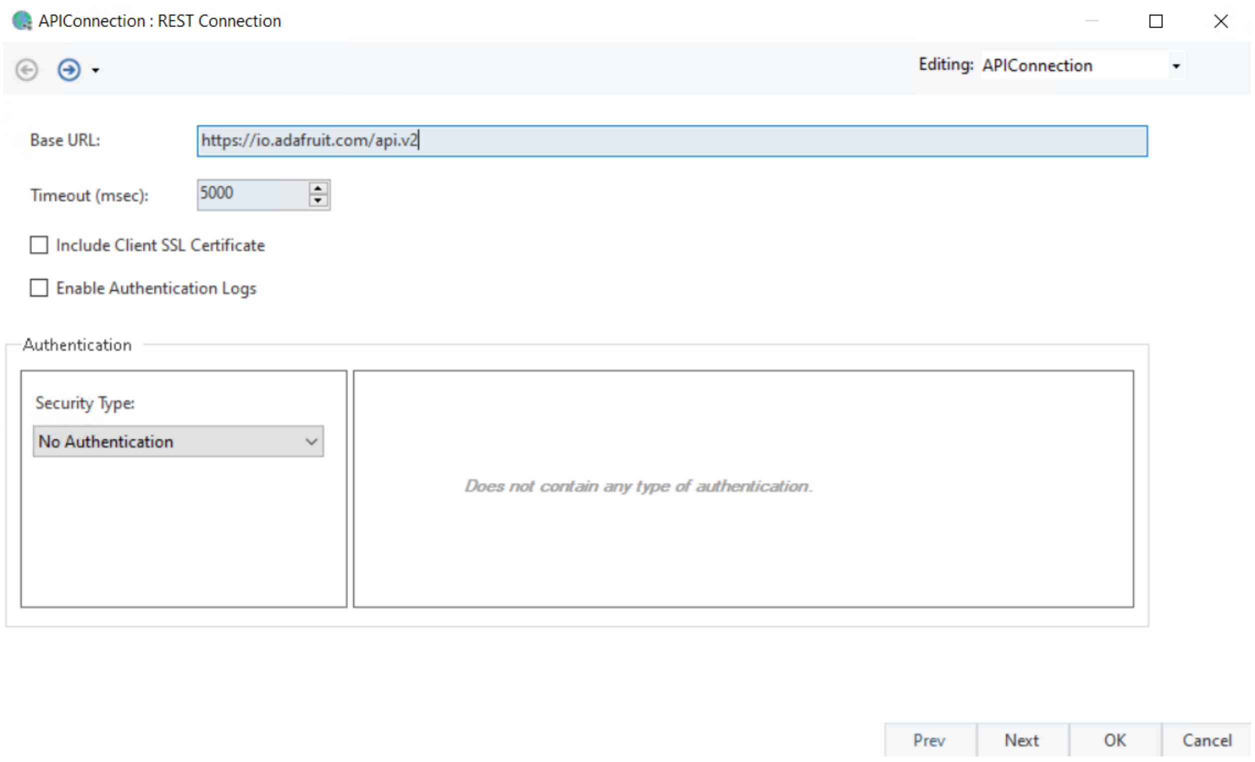
For APIs, authentication plays a key role in authorizing requests to the API platform's resources. The following authentication types are available within the *API Connection* object.

1. No Authentication
2. OAuth 2
3. API Key
4. Basic Authentication
5. Bearer Token

6. AWS Signature
7. NTLM

No Authentication

With this security type, the user can send API requests without including any authentication parameters.



Base URL: https://io.adafruit.com/api.v2

Timeout (msec): 5000

☐ Include Client SSL Certificate

☐ Enable Authentication Logs

Security Type: No Authentication

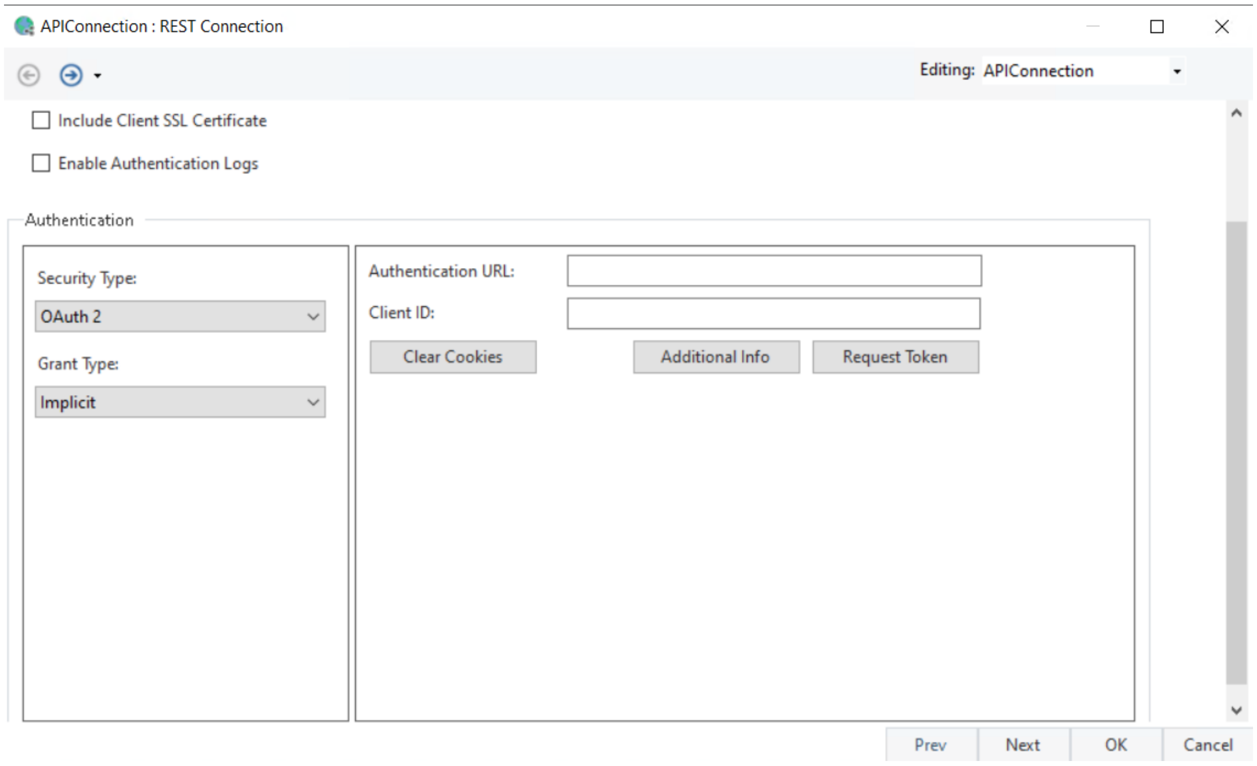
Does not contain any type of authentication.

OAuth 2

This type is used when an unrelated application login is used to acquire permission to access data on your behalf for another application. Instead of giving away your password to the application, *OAuth 2* enables delegated authorization through a third-party Authorization Server.

In response to a valid authorization, the Auth Server issues an Access Token with a restricted scope and validity to authenticate the user with permissions. When the Access Token expires, its Refresh Token is used to obtain another valid Access Token.

Configure an *OAuth 2* request to generate Access and Refresh tokens. The tokens will be implicitly added to the request and auto-refreshed if expired.


APIConnection : REST Connection

Editing: APIConnection

☐ Include Client SSL Certificate

☐ Enable Authentication Logs

Authentication

Security Type: OAuth 2

Grant Type: Implicit

Authentication URL:

Client ID:

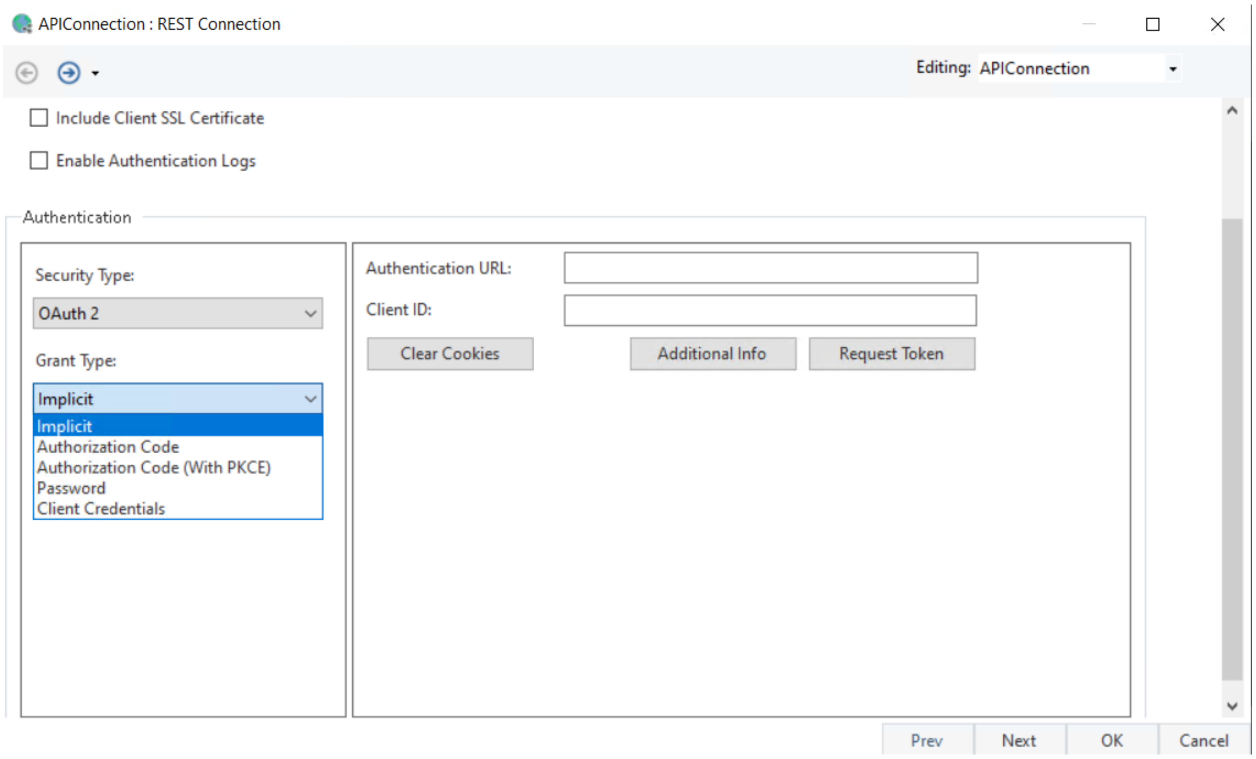
The *OAuth 2* authentication supports different flows for various scenarios. You can select any of the following *Grant Types*:

1. Implicit
2. Authorization Code
3. Authorization Code (with PKCE)
4. Password
5. Client Credentials

Implicit

In this *Grant Type*, you only need to provide an *Authentication URL* and *Client ID* to request a token without an intermediate code exchange. It was built for apps such as native Java script clients, and mobile or browser-based applications where client secrets cannot be exposed.

Hence, this flow promptly gets the token directly exposed in the URL and is considered less secure for web applications.


APIConnection : REST Connection

Editing: APIConnection

☐ Include Client SSL Certificate

☐ Enable Authentication Logs

Authentication

Security Type: OAuth 2

Grant Type: Implicit

- Implicit
- Authorization Code
- Authorization Code (With PKCE)
- Password
- Client Credentials

Authentication URL:

Client ID:

Authentication URL: This is the login page, where the API user authorizes itself to the Authentication Server.

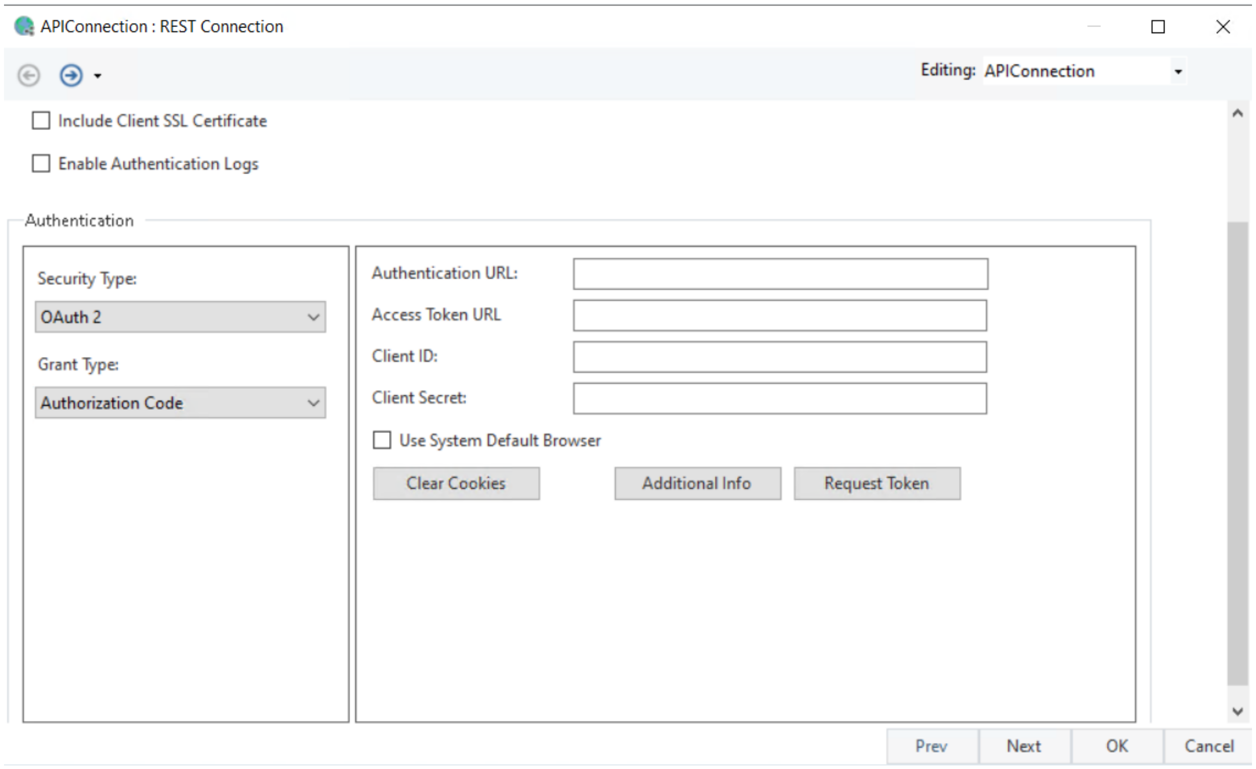
Client ID: This is the public identifier for accessing the registered API Server application.

Authorization Code

This flow type is popular for mobile and web server-side applications.

In this *Grant Type*, you need to provide an *Authentication URL*, *Access Token URL*, *Client ID*, and, optionally, a *Client Secret* to authorize.

The flow first requests a one-time authorization code from the authorization server. The authorized request is redirected to the API Server along with its client secret which then authenticates the user for its resources by exchanging the code for an *Access Token*.



Authentication URL: This is the login page, where the API user authorizes itself to the Auth Server.

Access Token URL: This URL is provided to generate an Access Token for authentication after the user has been authorized successfully.

Client ID: The public identifier for accessing the registered API Server application.

Client Secret: It is provided alongside the *Client ID*, as a secret credential to access the registered application from the Auth Server.

After providing the authentication details, click on the *Request Token* option to sign in and fetch the token(s).

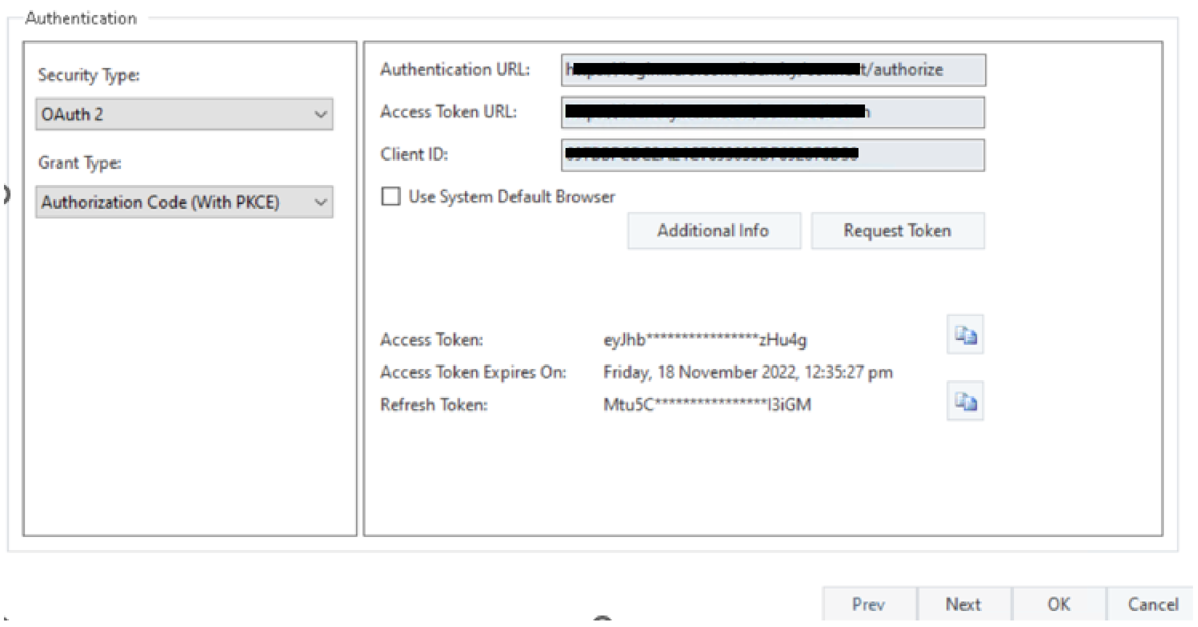
Authorization Code with PKCE

The Proof Key for Code Exchange flow has replaced implicit authentication flow by being more secure to be used in single-page native, mobile, and browser-based apps. As such apps existing on the browser cannot store client secrets, this Authorization Code flow keeps the client secret hidden.

Instead, the client sends a dynamically generated string generated using a code_verifier hashed to a code_challange to the Auth Server. The Auth Server stores this for verifying the client during the OAuth2 exchange.

The Client app then makes an authorization request and receives the Auth Code as a result. It then requests an Access Token by sending the Auth Code together with the code_verifier that is hashed by the Authorization server and compared to its saved copy for verification.

In this Grant Type, you need to provide an Authentication URL, Access Token URL, and the Client ID to authorize.



Authentication

Security Type: OAuth 2

Grant Type: Authorization Code (With PKCE)

Authentication URL: [redacted]t/authorize

Access Token URL: [redacted]

Client ID: [redacted]

☐ Use System Default Browser

Access Token: eyJhb****************zHu4g

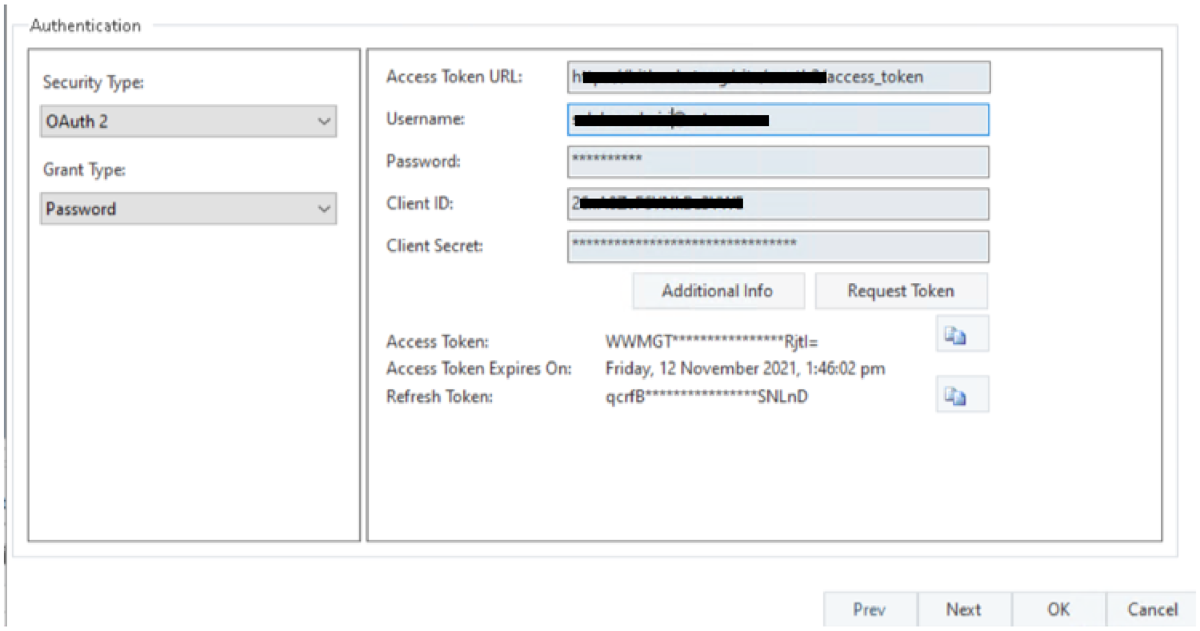
Access Token Expires On: Friday, 18 November 2022, 12:35:27 pm

Refresh Token: Mtu5C****************I3iGM

Password

In this Grant Type, you need an Access Token URL, Username, Password, Client ID, and Client Secret to authorize. It is considered for internal services and not recommended for third-party applications as it authenticates the given credentials in a single step.

Since user credentials are exposed to the client application, this flow type outlaws the OAuth2 principles and is now deprecated.



Authentication

Security Type: OAuth 2

Grant Type: Password

Access Token URL: h[redacted]access_token

Username: [redacted]

Password: **********

Client ID: [redacted]

Client Secret: ********************************

Access Token: WWMGT****************Rjtl=

Access Token Expires On: Friday, 12 November 2021, 1:46:02 pm

Refresh Token: qcrfB****************SNLnD

Access Token URL: The URL through which the Access token is going to be generated for authentication.

Username: The application login name of the user for authentication.

Password: The application user password is provided for authentication.

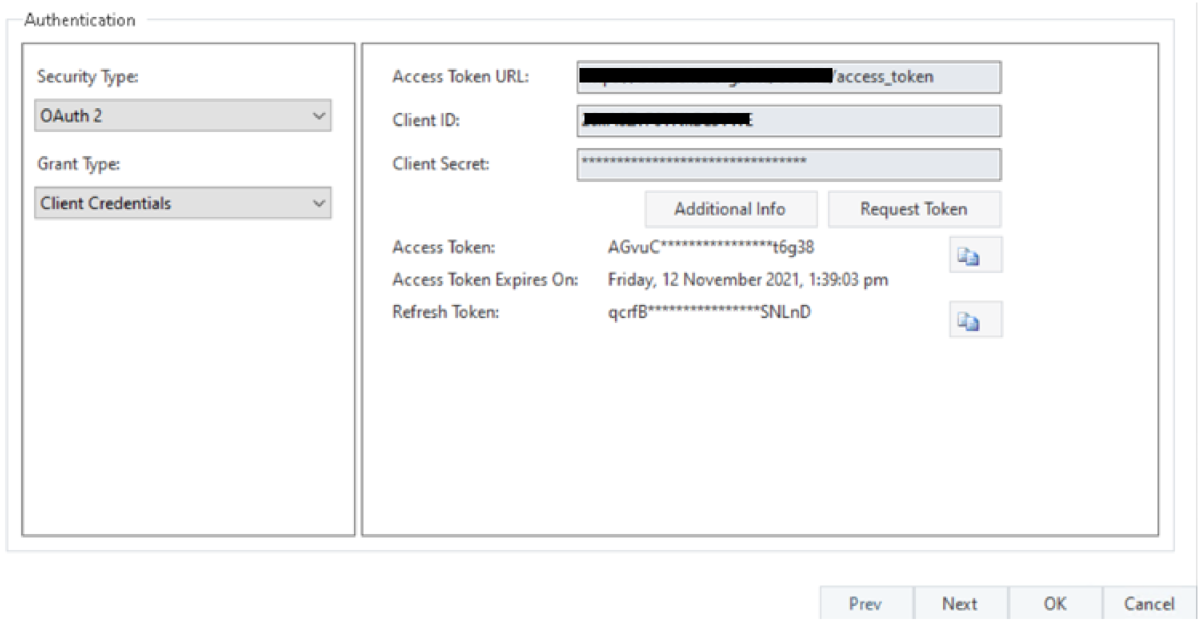
Client ID: The public identifier for accessing the registered API Server Application.

Client Secret: It is provided alongside the Client ID, as a secret credential to access the registered application from the Auth Server.

After providing the authentication details, click on *Request Token* to fetch the token(s).

Client Credentials

In this *Grant Type*, you need the *Access Token URL, Client ID,* and *Client Secret* to authorize. This is used with the client application. It self-authenticates access to its resources without a user context.



Authentication

Security Type: OAuth 2

Grant Type: Client Credentials

Access Token URL: [redacted]/access_token

Client ID: [redacted]

Client Secret: ********************************

Access Token: AGvuC****************t6g38

Access Token Expires On: Friday, 12 November 2021, 1:39:03 pm

Refresh Token: qcrfB*****************SNLnD

Access Token URL: This URL is provided to generate an access token for authentication.

Client ID: The public identifier for accessing the registered API Server application.

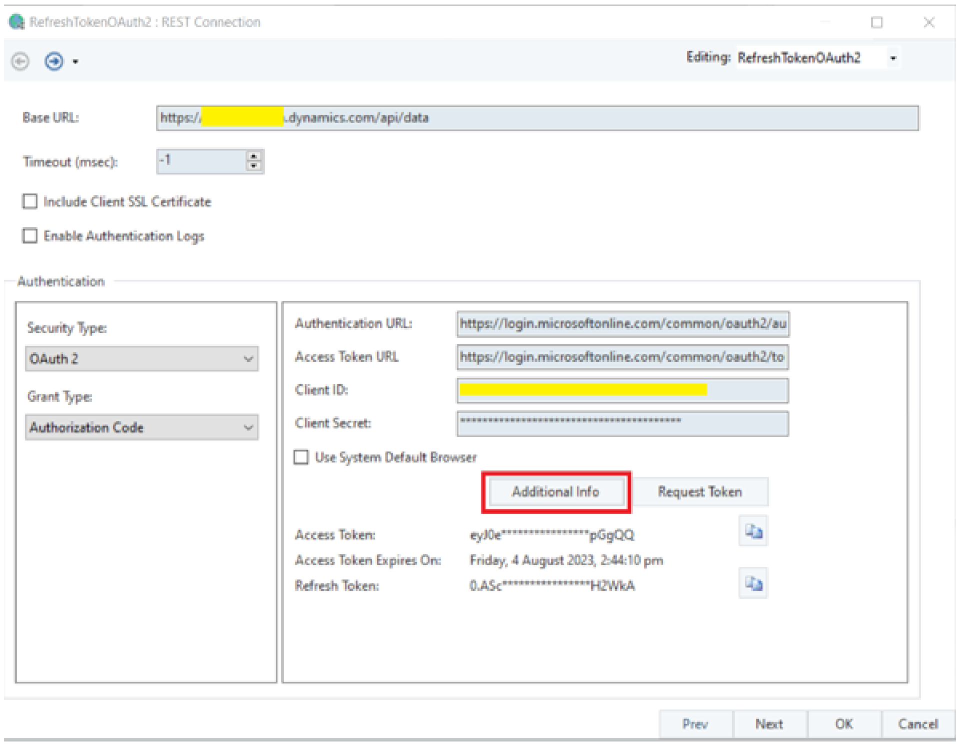
Client Secret: It is provided alongside the Client ID, as a secret credential to access the registered application from the Auth Server.

After providing the authentication details, click on *Reqeust Token* to fetch the token(s).

Additional OAuth 2 Info

An OAuth 2 authentication flow requires some additional parameters to specify resources and scope permissions associated with the given Access Token.

To provide additional information required by an API provider for an OAuth2 request, click on the *Additional Info* button.



RefreshTokenOAuth2 : REST Connection

Editing: RefreshTokenOAuth2

Base URL: https://[redacted].dynamics.com/api/data

Timeout (msec): -1

☐ Include Client SSL Certificate

☐ Enable Authentication Logs

Authentication

Security Type: OAuth 2

Grant Type: Authorization Code

Authentication URL: https://login.microsoftonline.com/common/oauth2/au

Access Token URL: https://login.microsoftonline.com/common/oauth2/to

Client ID: [redacted]

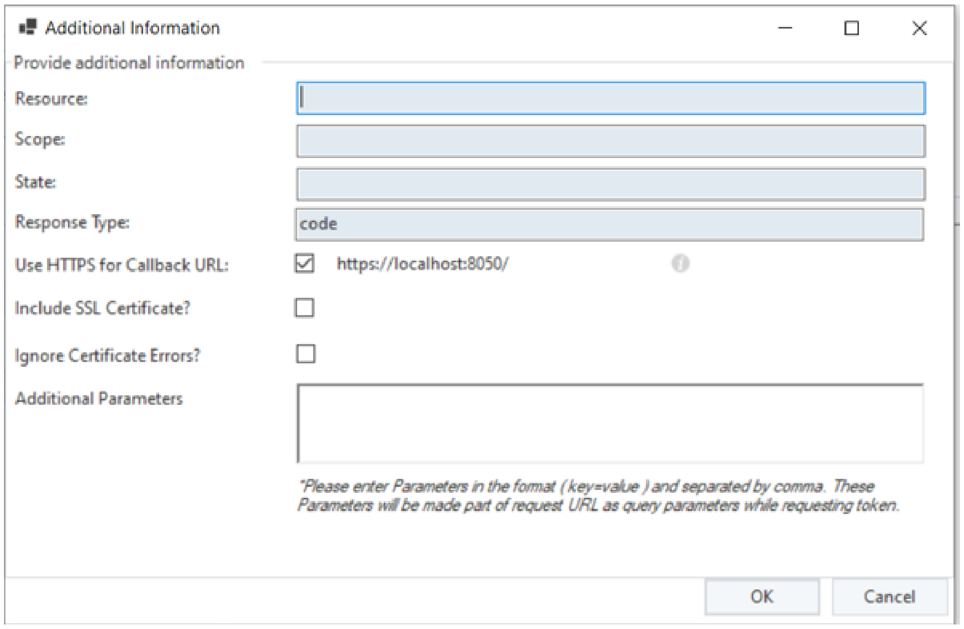
Client Secret: **

☐ Use System Default Browser

Access Token: eyJ0e****************pGgQQ

Access Token Expires On: Friday, 4 August 2023, 2:44:10 pm

Refresh Token: 0.ASc****************H2WkA



Additional Information

Provide additional information

Resource:

Scope:

State:

Response Type: code

Use HTTPS for Callback URL: ☑ https://localhost:8050/

Include SSL Certificate? ☐

Ignore Certificate Errors? ☐

Additional Parameters

*Please enter Parameters in the format (key=value) and separated by comma. These Parameters will be made part of request URL as query parameters while requesting token.

Resource: Use this to identify the URL of the web API intended for user access.

Scope: Use this to specify what the authenticating application can do on behalf of a user by imposing a limit on which resources it can access and with what rights.

State: This parameter is useful to protect against XSRF as the client generates and sends a random string while the Auth Server returns it back again on authenticating as a verification.

Response Type: This parameter is used to specify the expected type to be received from the authorization server on valid authorization. The most common inputs are “code” and “token”. Code is used for the Authorization Code grant type where it is exchanged in the follow-up request for the token. A token is used for implicit grant type where the Access Token is returned directly.

Callback URL: Redirected URL after the authentication request at which the token/code will be returned. For Astera Centerprise, use “http://localhost:8050/” or “https://localhost:8050/”

Include SSL Certificate: To include the client certificate in the OAuth2 token generation request.

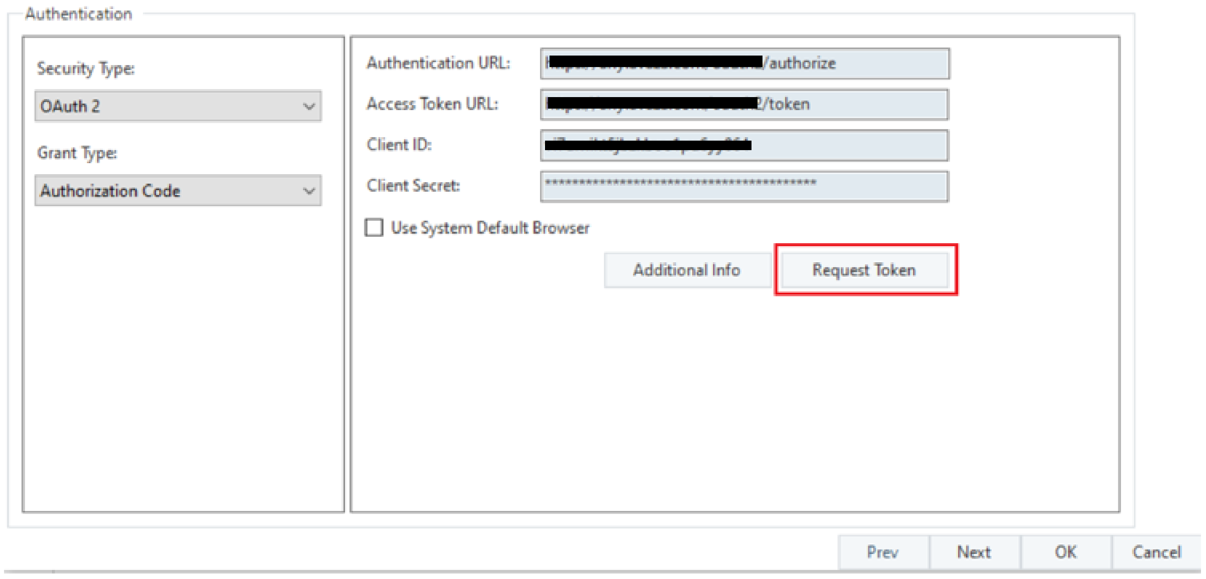
Ignore Certificate Errors: Check to ignore any certificate errors while authenticating.

Additional Parameters: Any additional parameters apart from the above list that are required to be sent in the authentication request can be added here as key-value pairs, separated by a comma.

Token Caching and Auto-Refresh

Following the security policy of authenticating an API call, clients are required to obtain Access/Refresh tokens for authenticating an API request. These tokens may have a defined validity and need to be invoked again to generate a new token.

Once authentication details are fully configured, users need to manually ‘Request Token’ in the API Connection.



Authentication

Security Type: OAuth 2

Grant Type: Authorization Code

Authentication URL: [redacted]/authorize

Access Token URL: [redacted]/token

Client ID: [redacted]

Client Secret: **

☐ Use System Default Browser

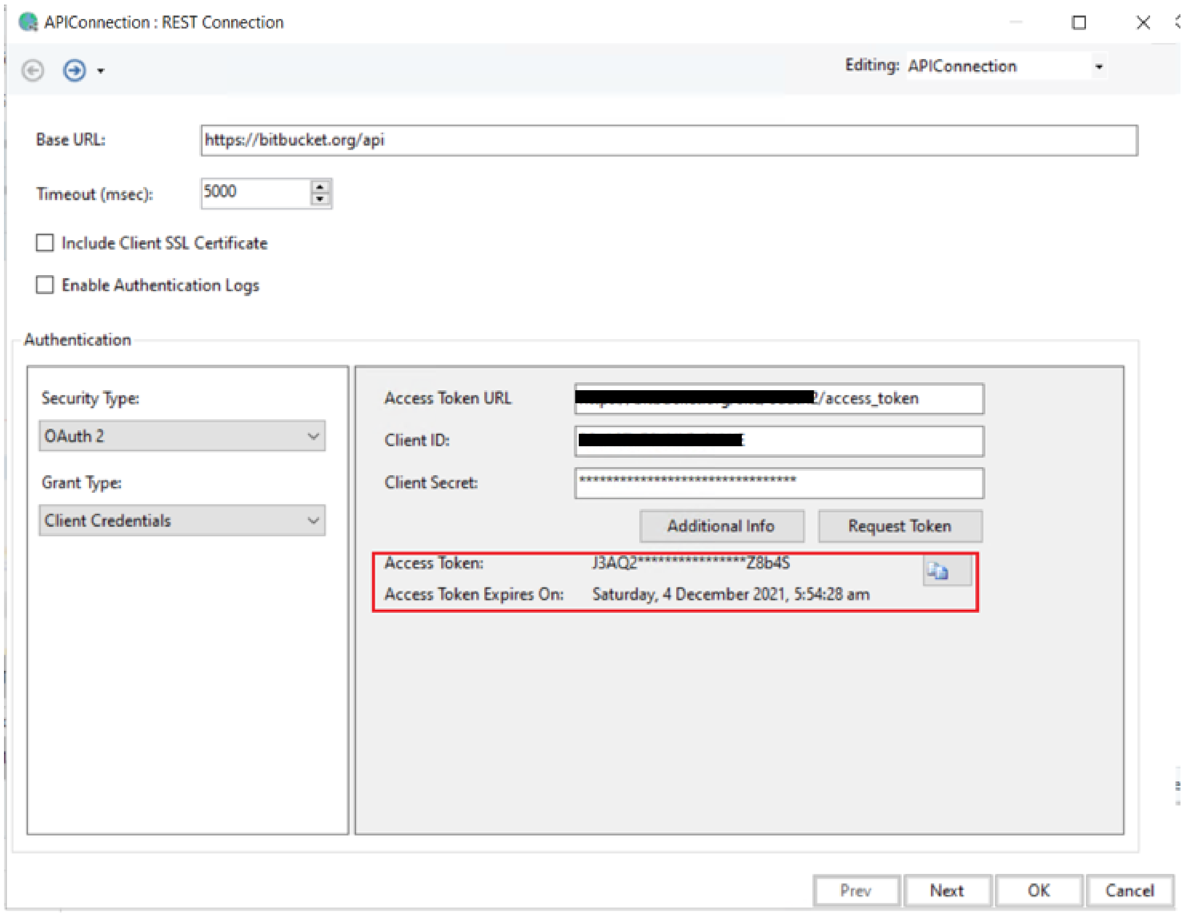
Handling token expiry and Automation

For the OAuth2 grant flow which requires users to authenticate when requesting a token, the refresh token can be used to obtain a new access token. While other grant flows directly make the call to request an access token, Astera Centerprise can automatically obtain a new token in the background so your flows can be automated.

You can make use of the auto-generation and caching of these tokens which enables you to automate API requests ensuring new tokens are generated for use without needing to manually update the tokens each time.

Using 'Client Credentials' or 'Password' OAuth2 Grant Types

These grant flows work by making a single call requesting an Access Token along with the provided client application credentials. Since the flow is not dependent on any user input for authentication, it can be automated for the regeneration of a new token when the existing token expires.



APIConnection : REST Connection

Editing: APIConnection

Base URL: https://bitbucket.org/api

Timeout (msec): 5000

☐ Include Client SSL Certificate

☐ Enable Authentication Logs

Authentication

Security Type: OAuth 2

Grant Type: Client Credentials

Access Token URL: [redacted]/access_token

Client ID: [redacted]

Client Secret: ********************************

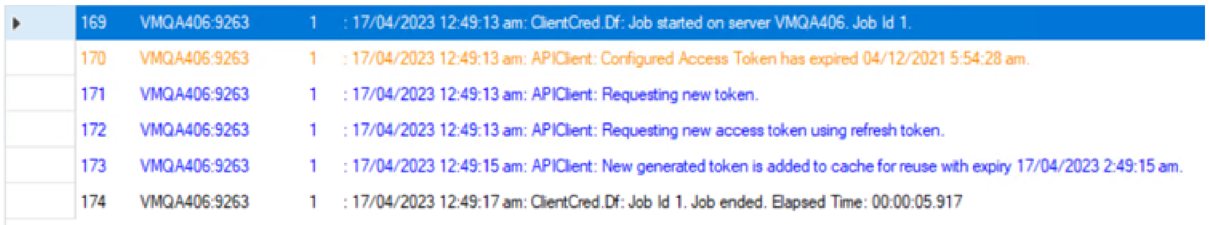
Additional Info | Request Token

Access Token: J3AQ2****************Z8b4S

Access Token Expires On: Saturday, 4 December 2021, 5:54:28 am

Prev | Next | OK | Cancel

Here, I have a pre-configured authentication with an expired token. Let's see what happens when this flow is executed with an expired token.



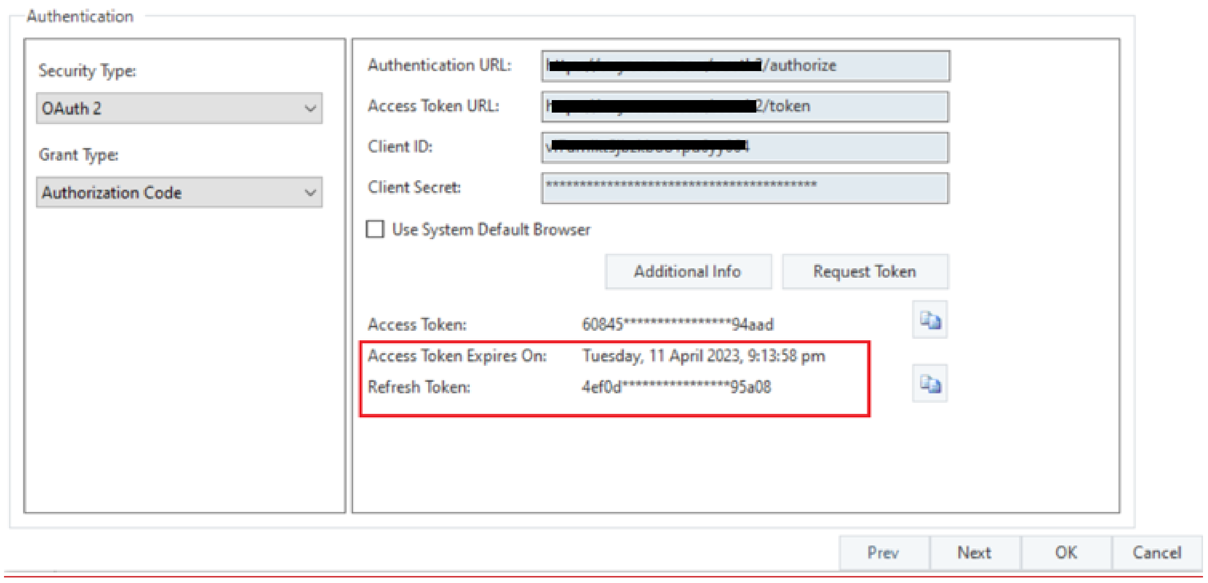
```
169   VMQA406:9263   1   : 17/04/2023 12:49:13 am: ClientCred.Df: Job started on server VMQA406. Job Id 1.
170   VMQA406:9263   1   : 17/04/2023 12:49:13 am: APIClient: Configured Access Token has expired 04/12/2021 5:54:28 am.
171   VMQA406:9263   1   : 17/04/2023 12:49:13 am: APIClient: Requesting new token.
172   VMQA406:9263   1   : 17/04/2023 12:49:13 am: APIClient: Requesting new access token using refresh token.
173   VMQA406:9263   1   : 17/04/2023 12:49:15 am: APIClient: New generated token is added to cache for reuse with expiry 17/04/2023 2:49:15 am.
174   VMQA406:9263   1   : 17/04/2023 12:49:17 am: ClientCred.Df: Job Id 1. Job ended. Elapsed Time: 00:00:05.917
```

The job trace shows that an expired token was found, and a new token has been generated for this connection and saved to the server cache for future reuse.

On the next run, the server is bound to check the cache for a valid token before opting to generate a new one. The cache stores a token for each unique connection used across all jobs running on the server.

Use of a Refresh Token

For other OAuth2 grant flows that require the user to authenticate first, the refresh token is used to regenerate the access token automatically.



Authentication

Security Type: OAuth 2

Grant Type: Authorization Code

Authentication URL: [redacted]/authorize

Access Token URL: [redacted]2/token

Client ID: [redacted]

Client Secret: **

☐ Use System Default Browser

Access Token: 60845*****************94aad

Access Token Expires On: Tuesday, 11 April 2023, 9:13:58 pm

Refresh Token: 4ef0d*****************95a08

Using Default User Browser for User Authentication

Some API Providers restrict using an embedded browser for authenticating using the OAuth2 code exchange. An alternate option is to request token through a more secure browser-based OAuth authentication.

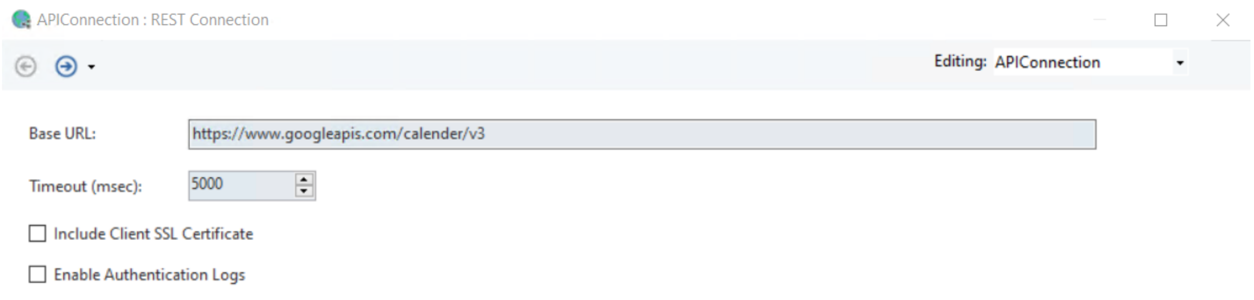
In this article, we'll discuss how to run an OAuth2.0 flow for Google Calendar API using the user's default browser. Users will first need to create an oauth2 application on the Google Developers' account and obtain the client id and secret.

Authenticating the Client Application

For this example, we will be authenticating Google APIs which do not allow the use of an Embedded Browser for an OAuth2 exchange.

1. Open the API Connection to configure authentication information.

As Google Calendar API works with OAuth2.0 security with Authorization Code grant type, we can select and configure it accordingly.



APIConnection : REST Connection

Editing: APIConnection

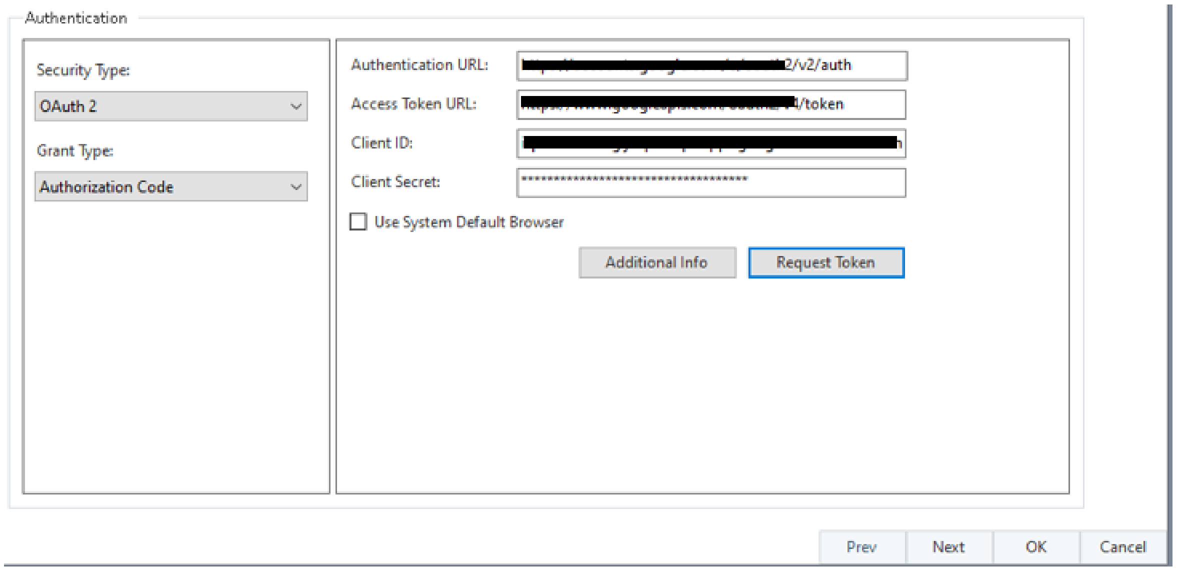
Base URL: https://www.googleapis.com/calender/v3

Timeout (msec): 5000

☐ Include Client SSL Certificate

☐ Enable Authentication Logs

We must enter parameters such as Authentication URL, Access Token URL, Client ID, Client Secret, and Additional Information according to the authentication and authorization information provided by Google. Now, let's click on the Request Token button to generate the access and refresh tokens.


Authentication

Security Type: OAuth 2

Grant Type: Authorization Code

Authentication URL: [redacted]2/v2/auth

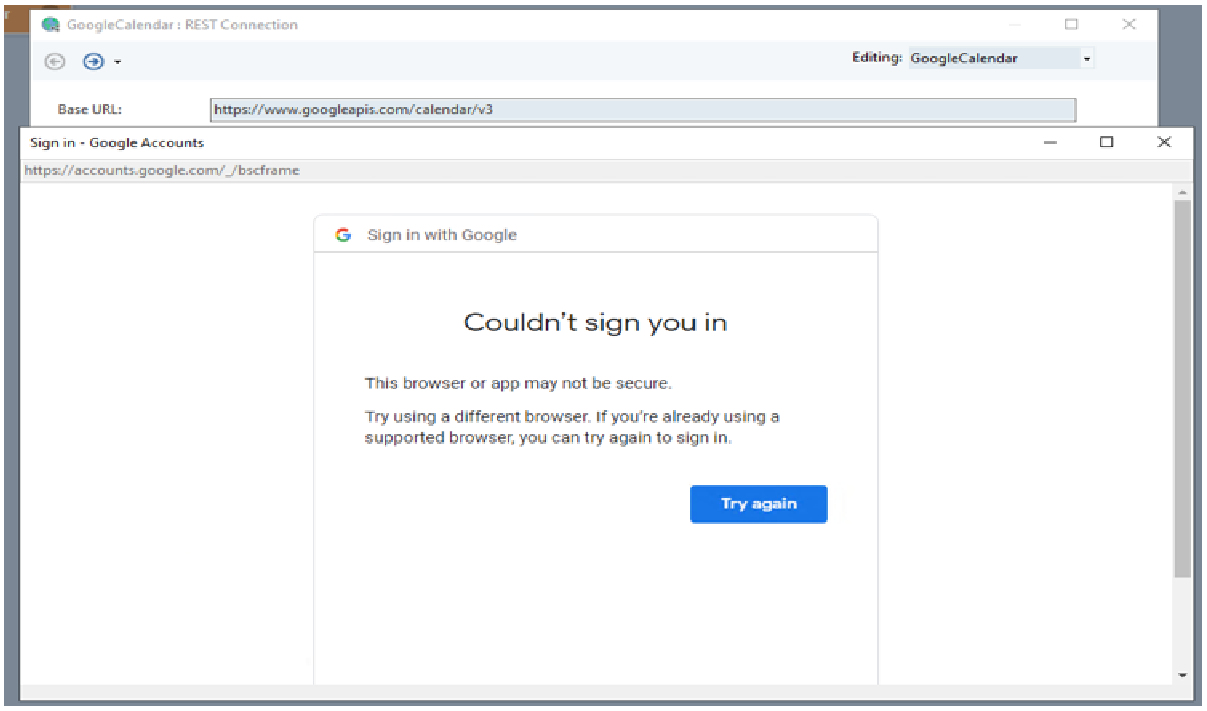
Access Token URL: [redacted]/token

Client ID: [redacted]

Client Secret: ************************************

☐ Use System Default Browser

This opens the Embedded Browser of the Astera Client which will result in an error as Google does not allow authentication via an embedded browser. For such platforms, it is necessary to use a more secure user-default browser for OAuth2 authentication exchange.


GoogleCalendar : REST Connection

Editing: GoogleCalendar

Base URL: https://www.googleapis.com/calendar/v3

Sign in - Google Accounts

https://accounts.google.com/_/bscframe

Sign in with Google

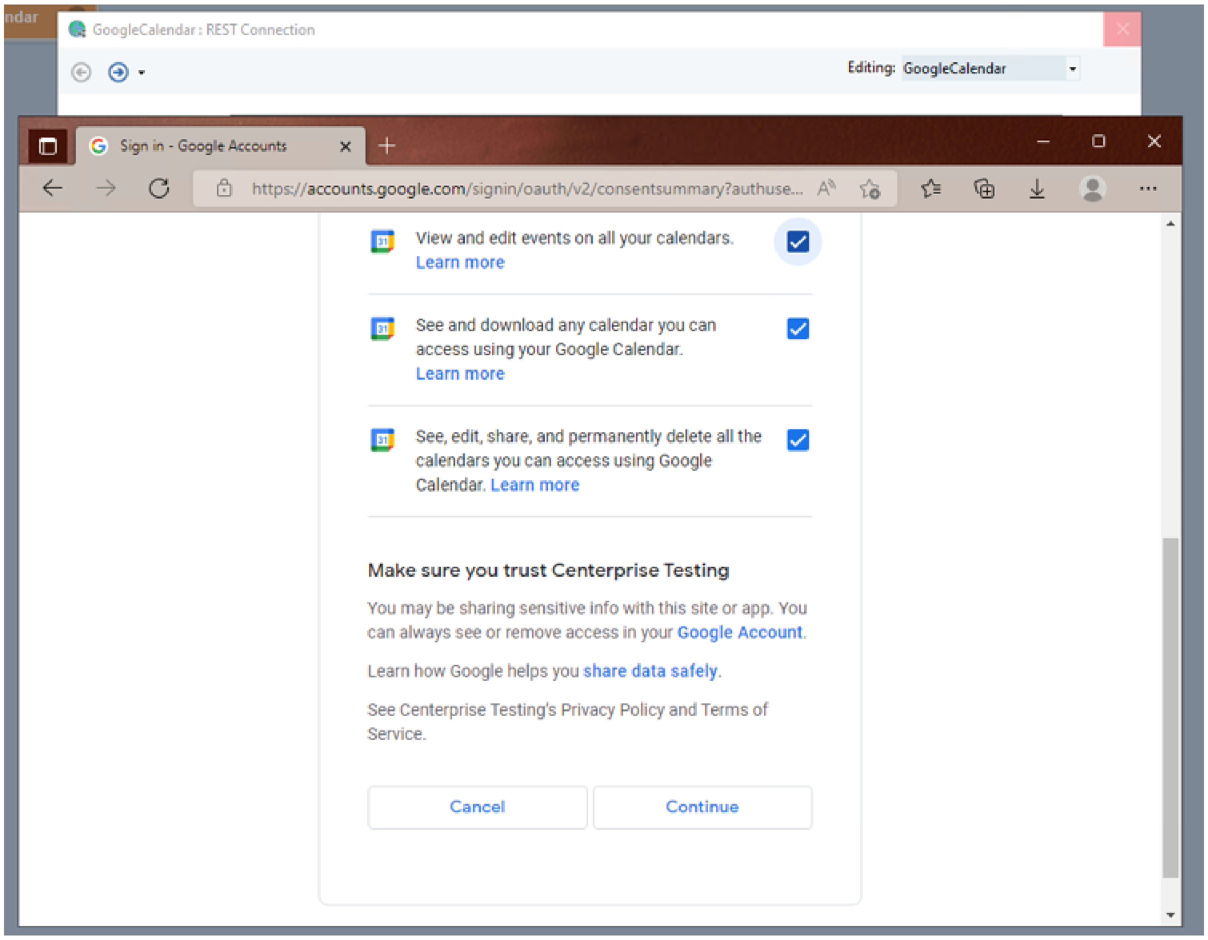
Couldn't sign you in

This browser or app may not be secure.

Try using a different browser. If you're already using a supported browser, you can try again to sign in.

Close the embedded browser window. Now, check the option to *Use System's Default Browser* and click on the Request Token button again.

This opens the user system's default browser for authentication, and this allows us to successfully retrieve the access token on logging in. In our case, the default Microsoft Edge web browser has opened.


GoogleCalendar : REST Connection

Editing: GoogleCalendar

Sign in - Google Accounts

https://accounts.google.com/signin/oauth/v2/consentsummary?authuse...

View and edit events on all your calendars. Learn more ✓

See and download any calendar you can access using your Google Calendar. Learn more ✓

See, edit, share, and permanently delete all the calendars you can access using Google Calendar. Learn more ✓

Make sure you trust Centerprise Testing

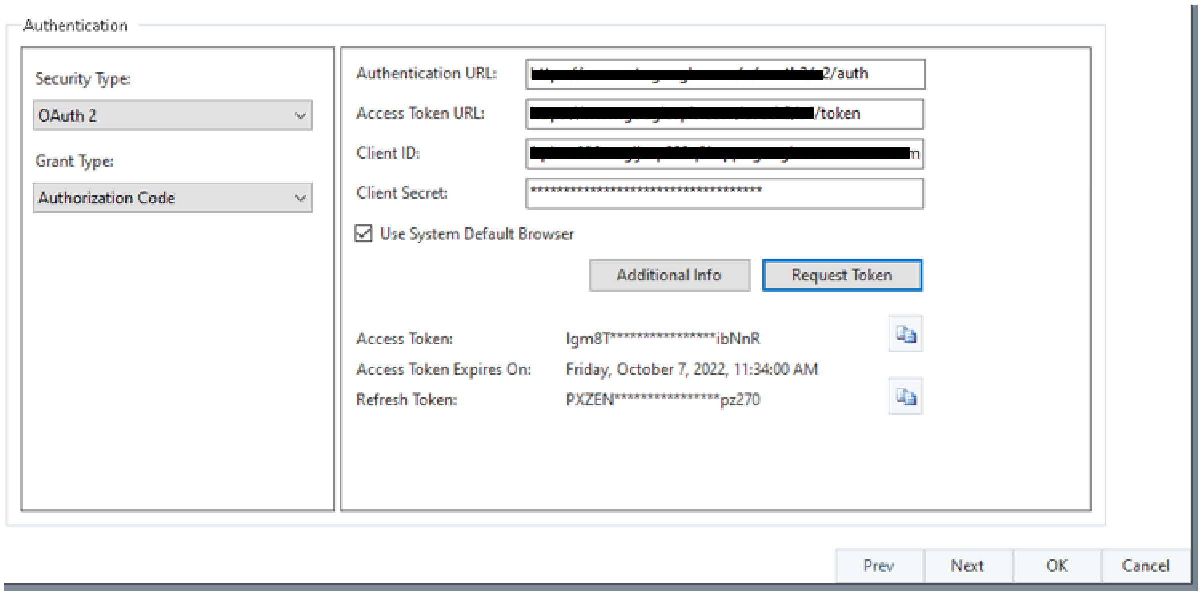
You may be sharing sensitive info with this site or app. You can always see or remove access in your Google Account.

Learn how Google helps you share data safely.

See Centerprise Testing's Privacy Policy and Terms of Service.

Note: Whether the embedded or secure browsers are allowed for authentication strictly depends upon the API provider.

Click on Continue.


Authentication

Security Type: OAuth 2

Grant Type: Authorization Code

Authentication URL: [redacted]2/auth

Access Token URL: [redacted]/token

Client ID: [redacted]m

Client Secret: ************************************

✓ Use System Default Browser

Access Token: Igm8T****************ibNnR

Access Token Expires On: Friday, October 7, 2022, 11:34:00 AM

Refresh Token: PXZEN****************pz270

The generated Access Token along with the Refresh Token (if supported by the API provider) are displayed on the REST Connection window with their respective expiry date and time.

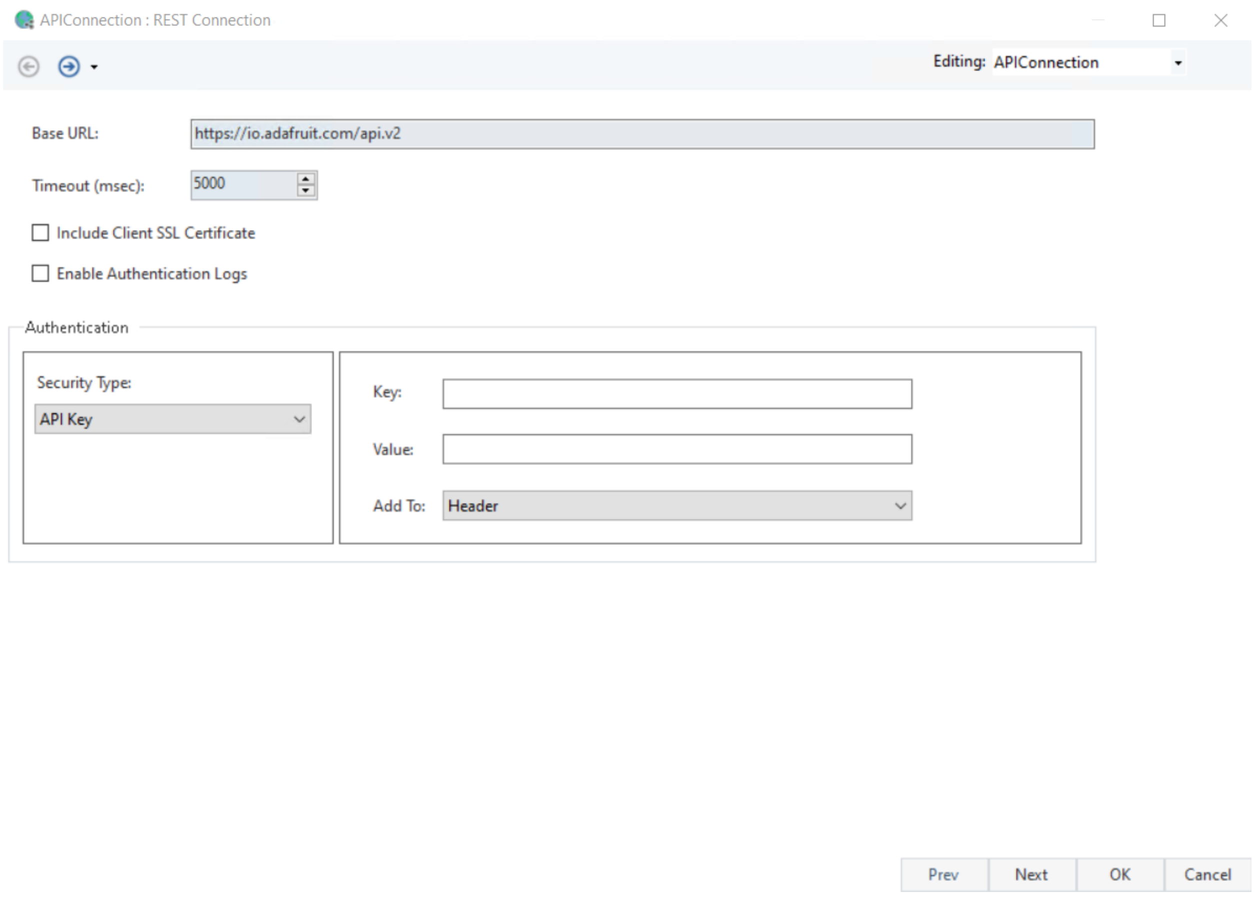
Tested System Browsers

The following browsers have been successfully tested for the Astera Client,

- Google Chrome
- Microsoft Edge
- Firebox

API Key

An *API Key* is a key-value pair that a client provides when it makes an API request. They can be sent in the *Query* string or as a request *Header*.



Editing: APIConnection

Base URL: https://io.adafruit.com/api.v2

Timeout (msec): 5000

☐ Include Client SSL Certificate

☐ Enable Authentication Logs

Authentication

Security Type: API Key

Key:

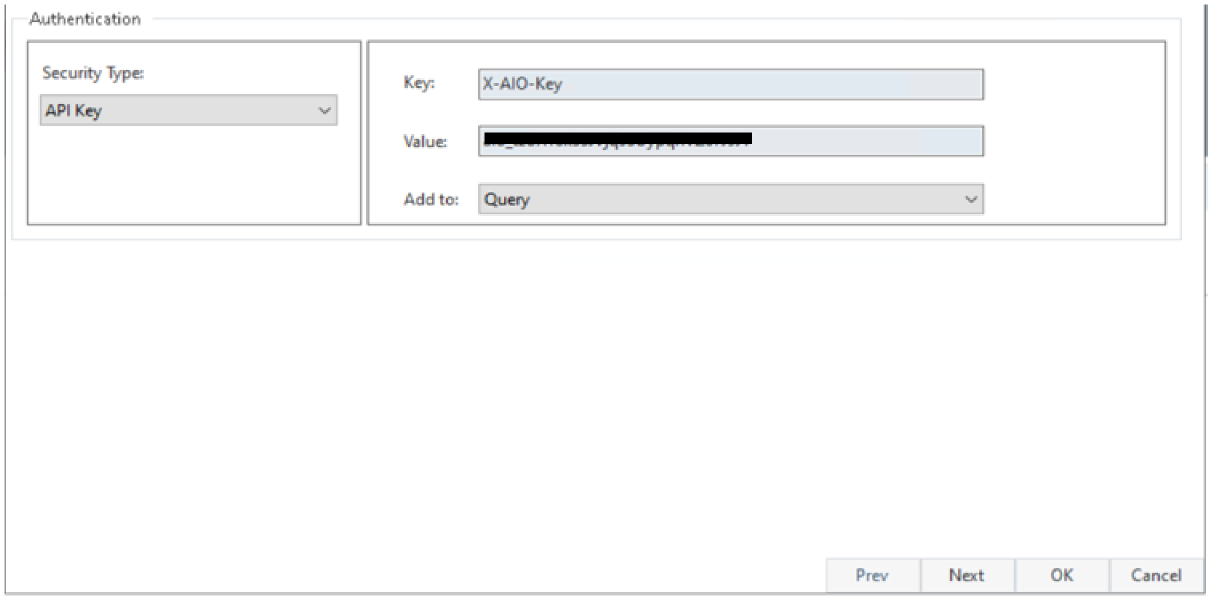
Value:

Add To: Header

It requires two parameters for authentication:

1. Key
2. Value

API Key as a Query


Authentication

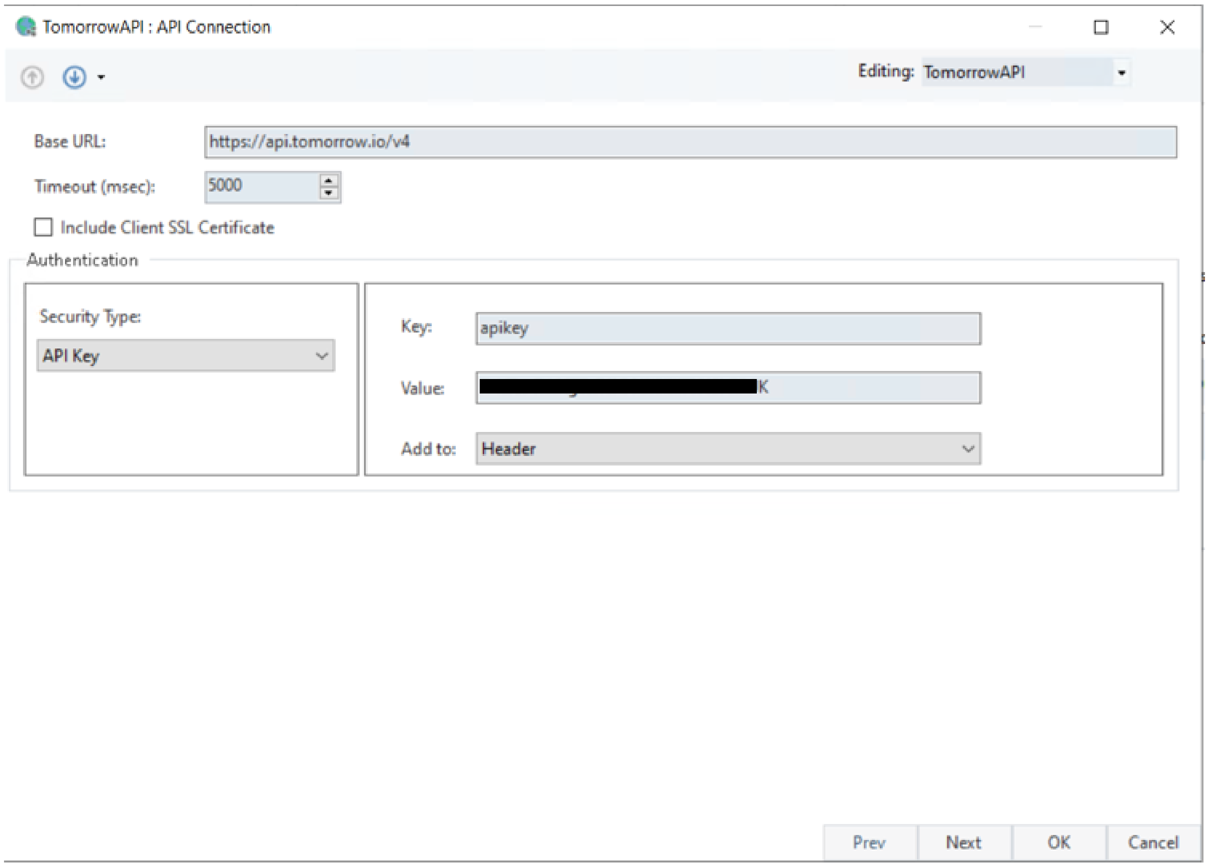
Security Type: API Key

Key: X-AIO-Key

Value: [redacted]

Add to: Query

API Key as a Header


TomorrowAPI : API Connection

Editing: TomorrowAPI

Base URL: https://api.tomorrow.io/v4

Timeout (msec): 5000

☐ Include Client SSL Certificate

Authentication

Security Type: API Key

Key: apikey

Value: [redacted]K

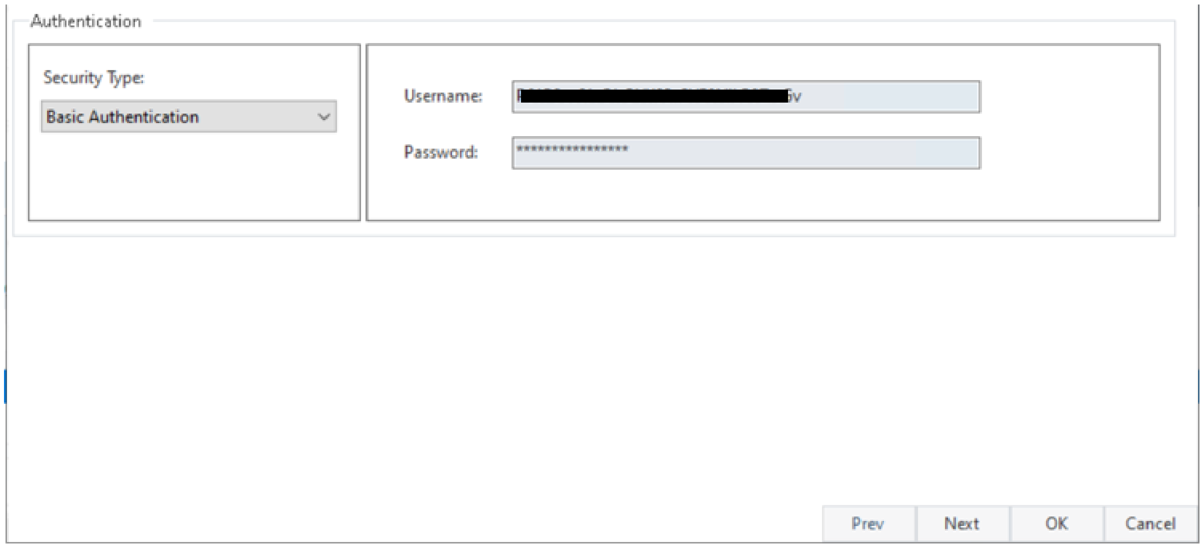
Add to: Header

Note: API Key is sent in as a key-value pair in the header such as "apikey: cZRcTZt7R3gnTt9l2C9YHXke0SNDAPJK"

Basic Authentication

Basic Authentication is structured according to the HTTP protocol to provide a *Username* and *Password* when making an API request.

In basic HTTP authentication, a request header parameter is included in the form of "Authentication: Basic", where the encoded string is the Base64 encoded.



Authentication

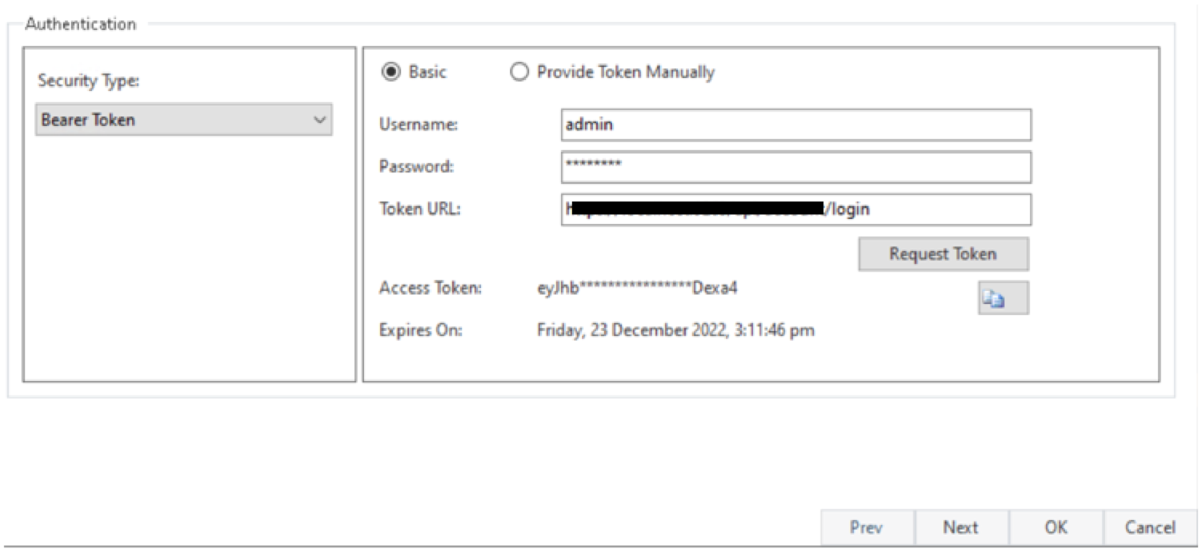
Security Type: Basic Authentication

Username: [redacted]

Password: ****************

Bearer Token

Bearer Token is an HTTP-based authentication. The access token generated by the server in response to a login request is in turn included in the request header.



Authentication

Security Type: Bearer Token

● Basic ○ Provide Token Manually

Username: admin

Password: ********

Token URL: [redacted]/login

Access Token: eyJhb****************Dexa4

Expires On: Friday, 23 December 2022, 3:11:46 pm

To generate a *Bearer Token*, you need:

1. User Name

2. Password

3. Token URL

Note: This Authentication type is needed to access Astera APIs, and the request is sent as “application/JSON”.

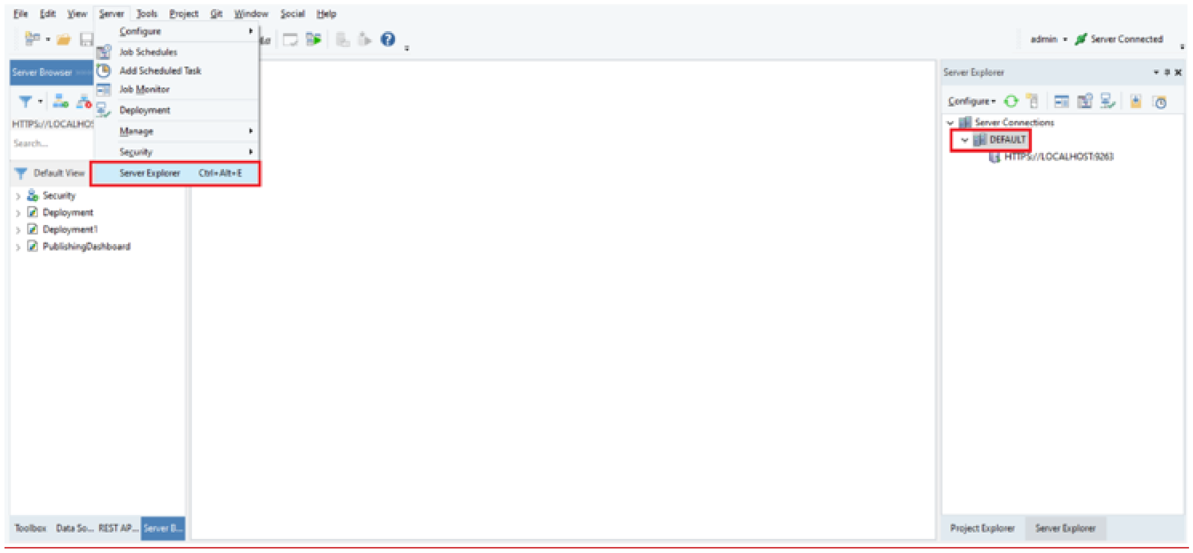
SSL Certificate Authentication

API clients can enable the use of a private signed certificate to authenticate themselves when accessing APIs through mutual TLS. You can configure APIs to use a .pem or a .pkt certificate paired with a certificate key or password.

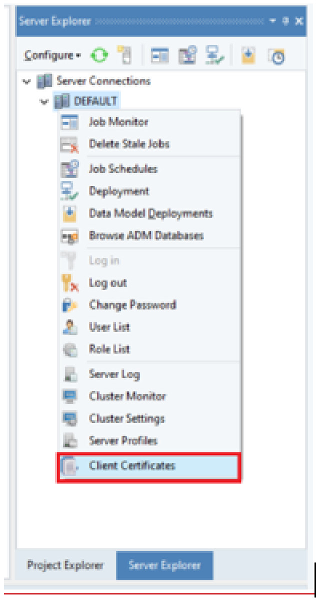
A Client certificate contains information used to identify the client including a digital signature and it is imported for a specific domain. All HTTPS - SSL-enabled requests matching the domain URL will authenticate using the installed client certificate.

All certificates used in authenticating API requests from the client will be imported to Astera’s Server and are included as authentication when an API request is sent. To import a client certificate for authenticating API requests,

1. Navigate to the Server tab on the main menu bar.



2. Right-click on the cluster node and select Client Certificates.

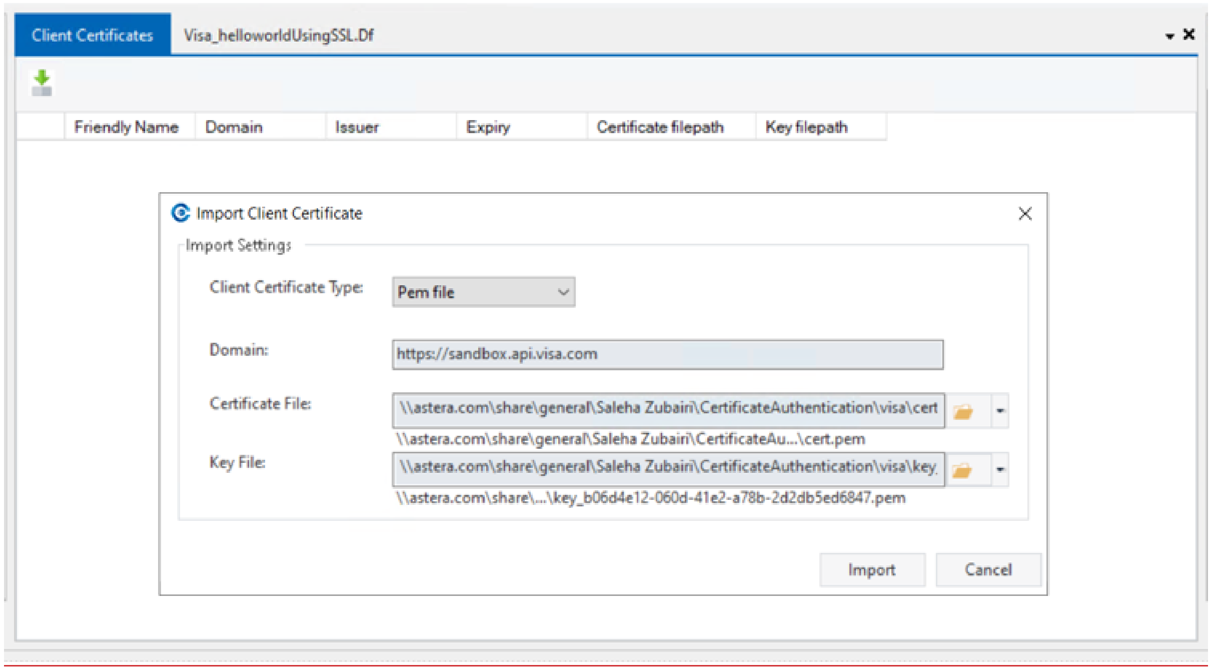


This opens the wizard to manage client SSL certificates.

3. Click on the import icon at the top left to add a certificate authenticating to a domain.

Importing a .pem certificate

- Define the requested domain which will include this certificate.
- Browse the .pem client certificate file obtained as a counterpart to the authenticating server certificate present on the API provider.
- Provide the matching key file for the given client certificate.



Client Certificate Type: Pem file

Domain: https://sandbox.api.visa.com

Certificate File: \\astera.com\share\general\Saleha Zubairi\CertificateAuthentication\visa\cert

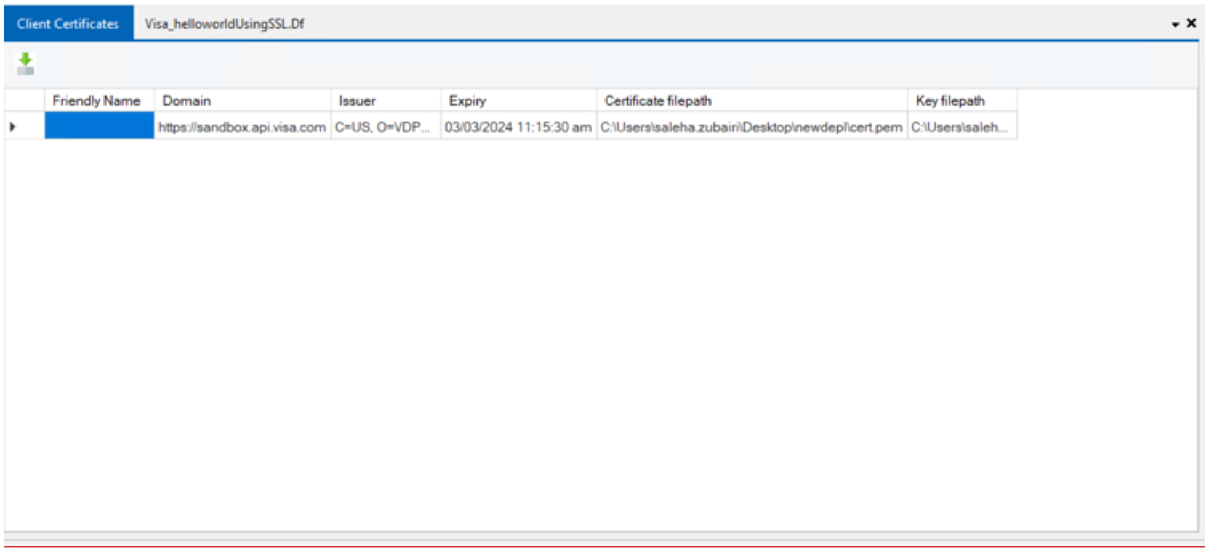
\\astera.com\share\general\Saleha Zubairi\CertificateAu...\cert.pem

Key File: \\astera.com\share\general\Saleha Zubairi\CertificateAuthentication\visa\key

\\astera.com\share\...\key_b06d4e12-060d-41e2-a78b-2d2db5ed6847.pem

Click import.

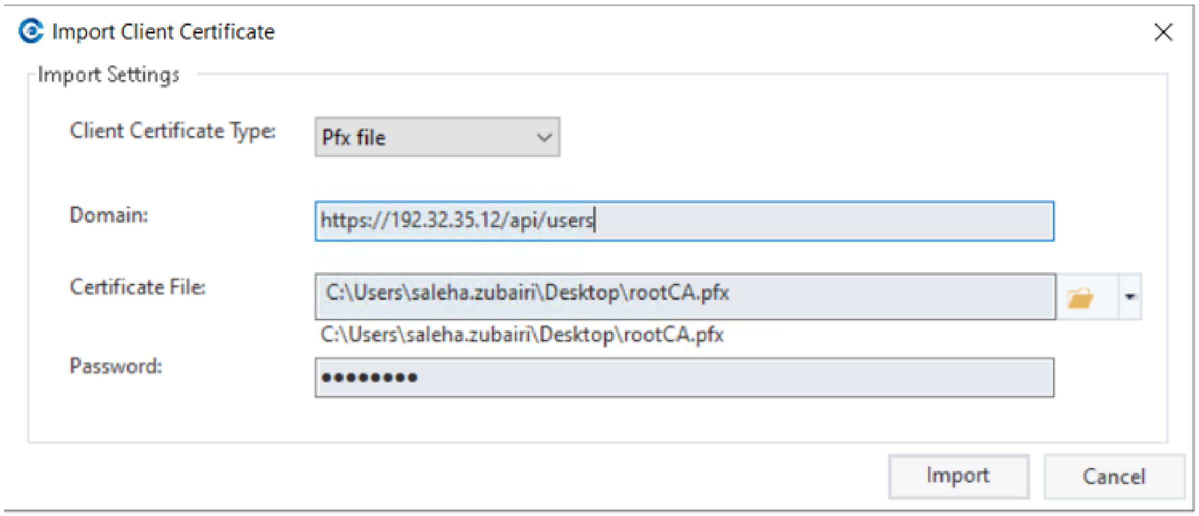
Now this certificate can be used with SSL-enabled authentication for API requests sent to the given domain.



Friendly Name	Domain	Issuer	Expiry	Certificate filepath	Key filepath
	https://sandbox.api.visa.com	C=US, O=VDP...	03/03/2024 11:15:30 am	C:\Users\saleha.zubairi\Desktop\newdep\cert.pem	C:\Users\saleh...

Importing a .pfx certificate

- Define the requested domain which will include this certificate.
- Browser the .pfx client certificate file obtained as a counterpart to the authenticating server certificate present on the API provider.
- Enter the password for the certificate.


Import Client Certificate

Import Settings

Client Certificate Type: Pfx file

Domain: https://192.32.35.12/api/users

Certificate File: C:\Users\saleha.zubairi\Desktop\rootCA.pfx

C:\Users\saleha.zubairi\Desktop\rootCA.pfx

Password: ••••••••

Click Import.

Now this certificate can be used with SSL-enabled authentication for API requests sent to the given domain.

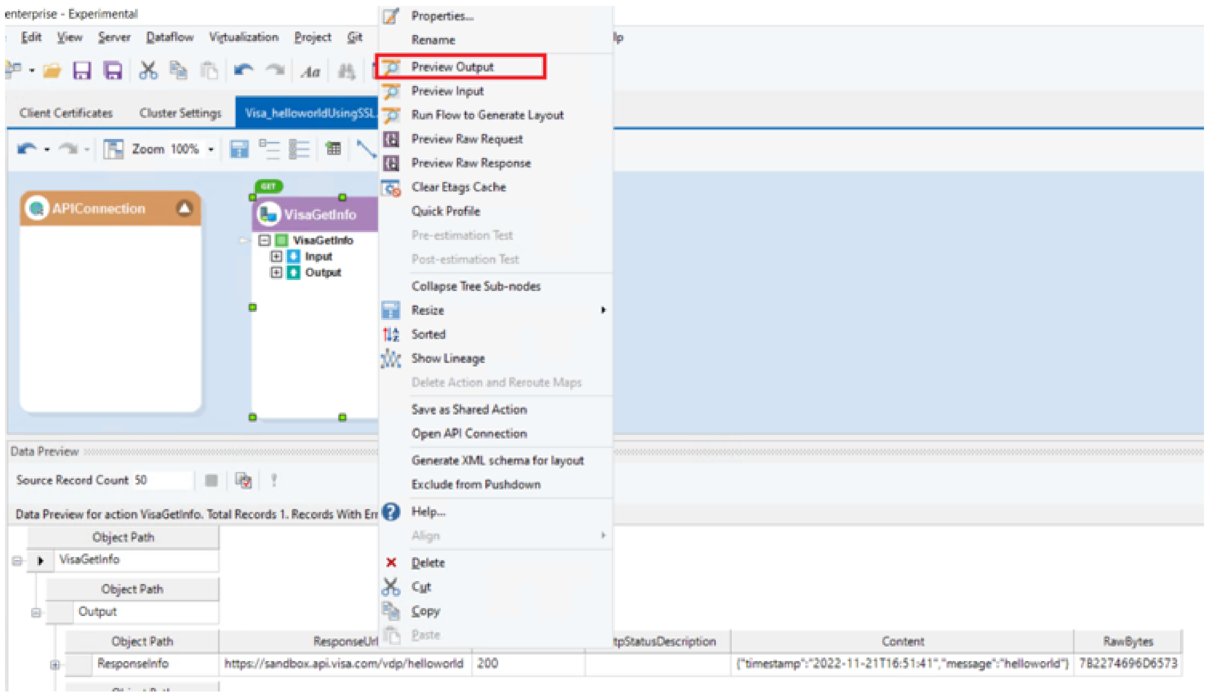
Enabling SSL Certificate Authentication

Once the certificate has been imported for the respective domain, let's see how to make an API request with SSL enabled.

You need to enable SSL verification to include the certificate when making an API call. To enable SSL, open the API Connection object which has the Base URL domain, and the authentication configured. To include the SSL certificate, check the option to "Include Client SSL Certificate".

Click Ok and preview the API Client to send a request.

This request now includes the certificate to validate the client on the mutual TLS authentication.



Data Preview

Source Record Count 50

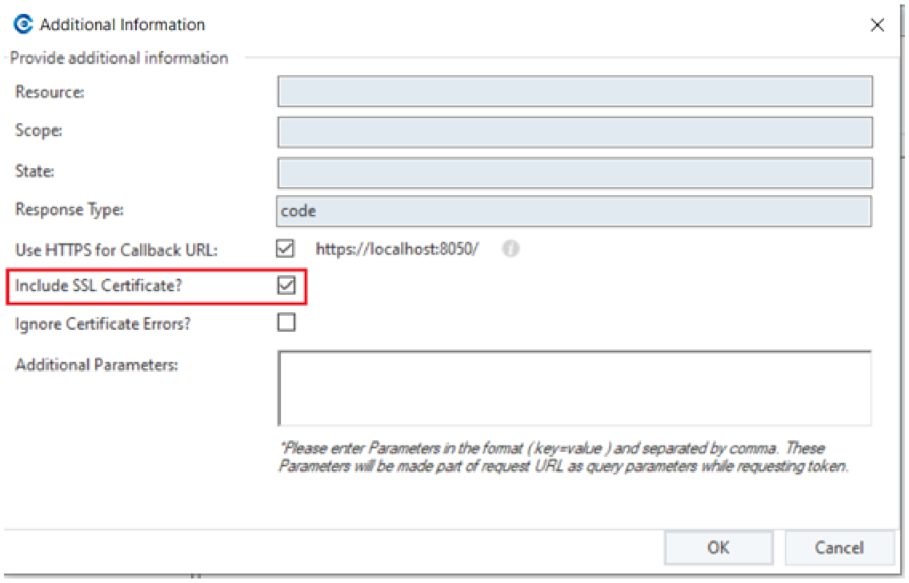
Data Preview for action VisaGetInfo. Total Records 1. Records With Err

Object Path
VisaGetInfo

Object Path
Output

Object Path	ResponseUrl		tpStatusDescription	Content	RawBytes
ResponseInfo	https://sandbox.api.visa.com/vdp/helloworld	200		{"timestamp":"2022-11-21T16:51:41","message":"helloworld"}	7B22746996D6573

Note: To include the client certificate in the Oauth2 request from the API Connection, check this option from Additional Info.


Additional Information

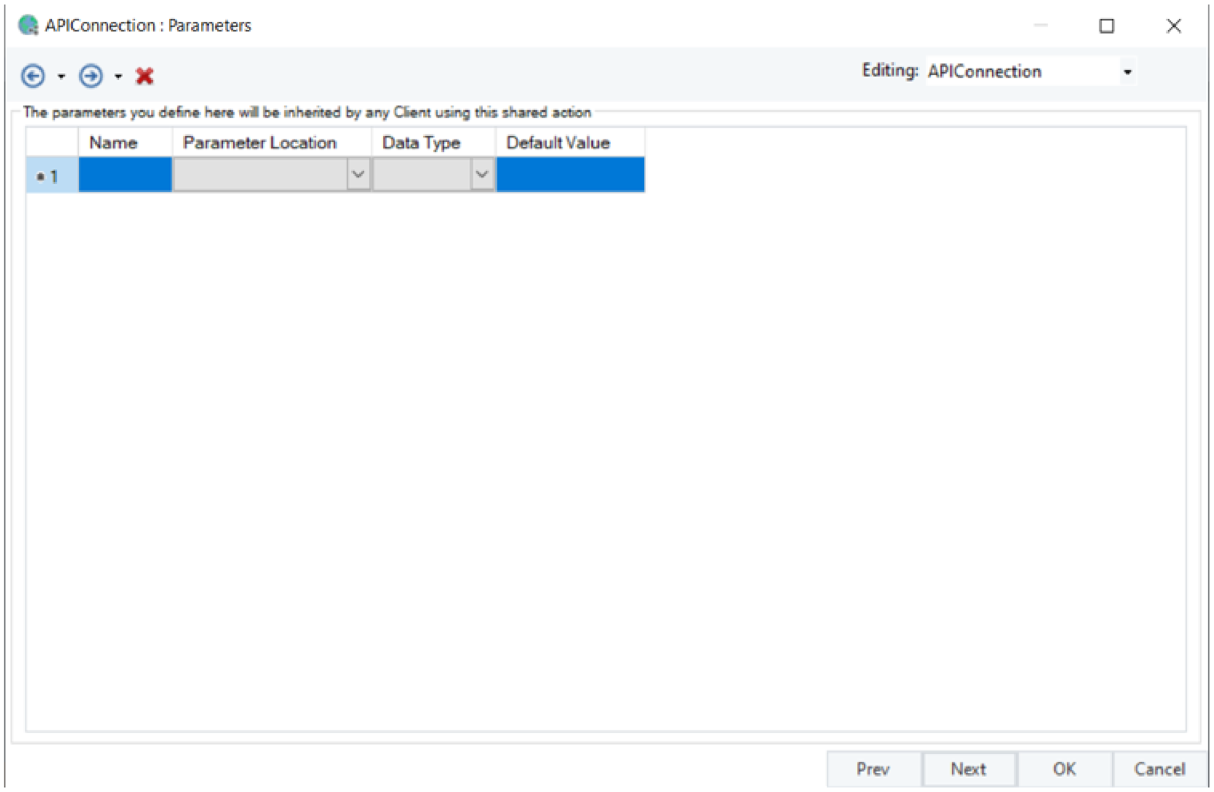
Provide additional information

Resource:	
Scope:	
State:	
Response Type:	code
Use HTTPS for Callback URL:	☑ https://localhost:8050/
Include SSL Certificate?	☑
Ignore Certificate Errors?	☐
Additional Parameters:	

*Please enter Parameters in the format (key=value) and separated by comma. These Parameters will be made part of request URL as query parameters while requesting token.

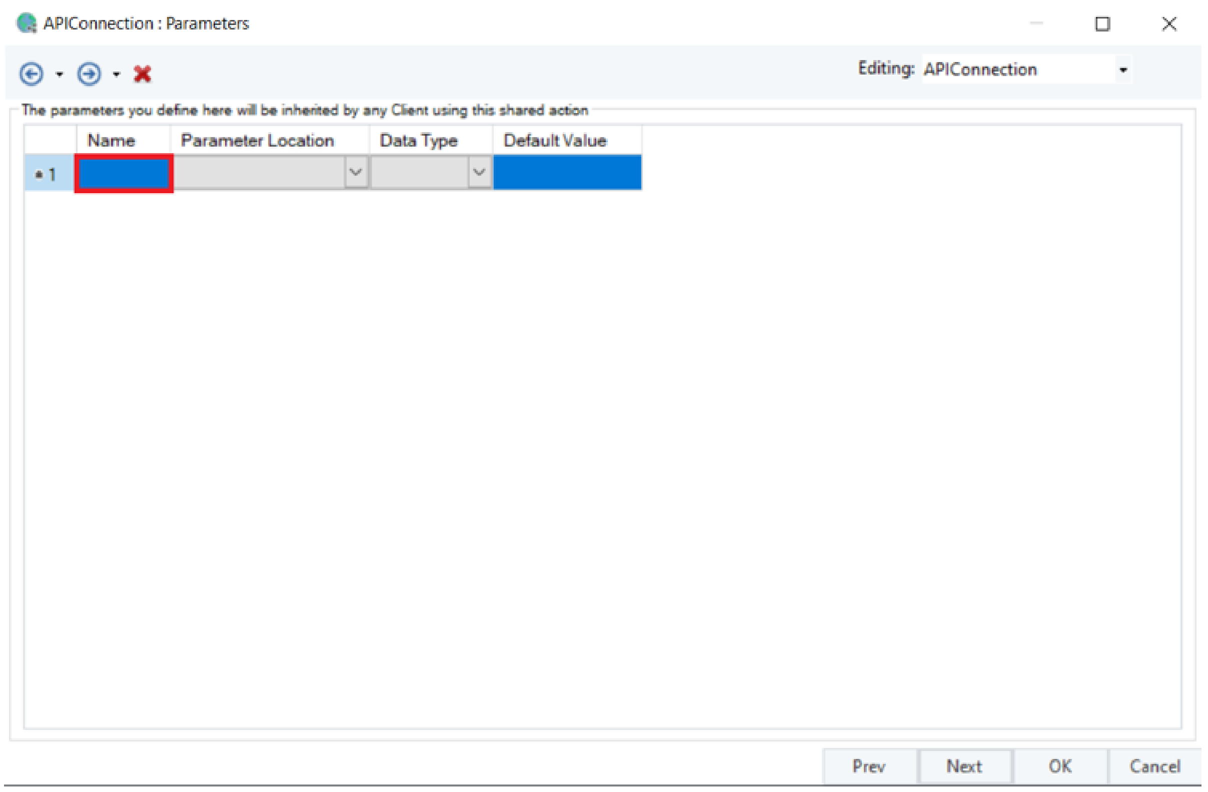
Shared Parameters

This is where you can define query or header parameters to be shared across all clients using the same connection.



The parameters you define here will be inherited by any Client using this shared action

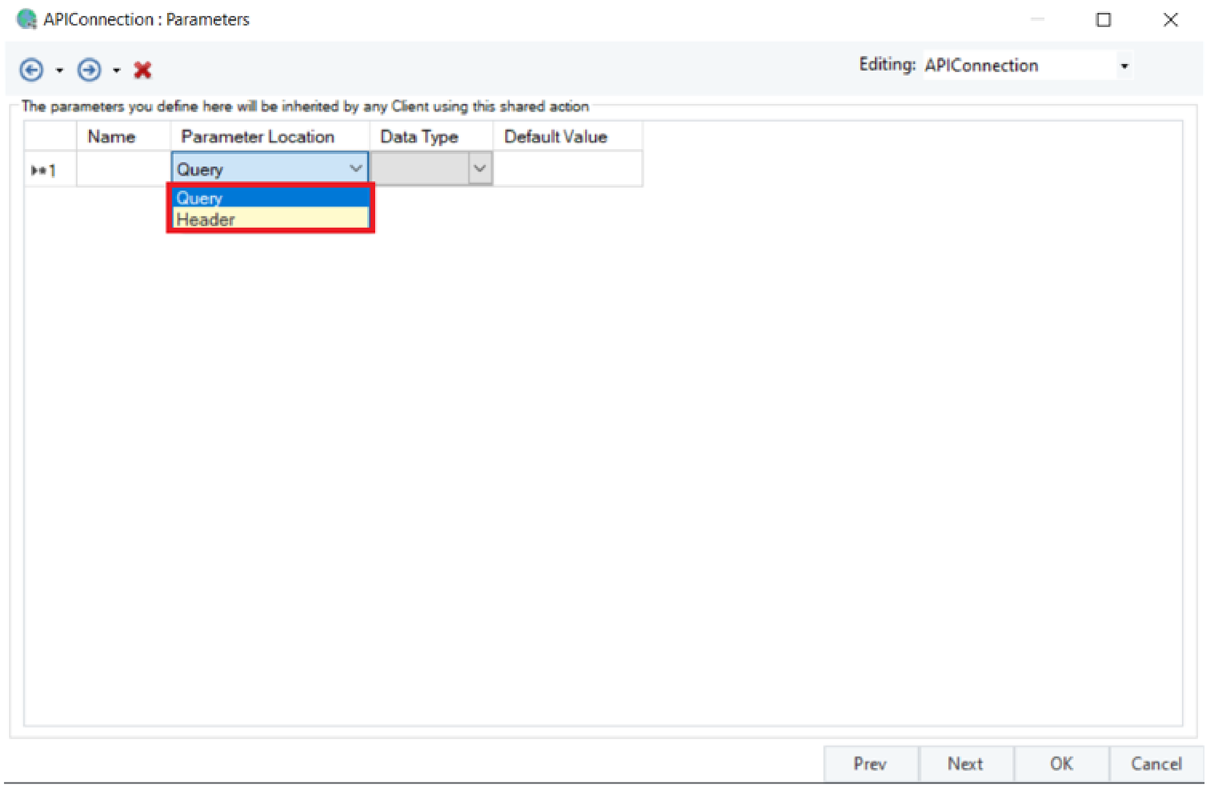
	Name	Parameter Location	Data Type	Default Value
* 1				



The parameters you define here will be inherited by any Client using this shared action

	Name	Parameter Location	Data Type	Default Value
* 1				

Name: The name of a *Query* or *Header* parameter can be defined here.



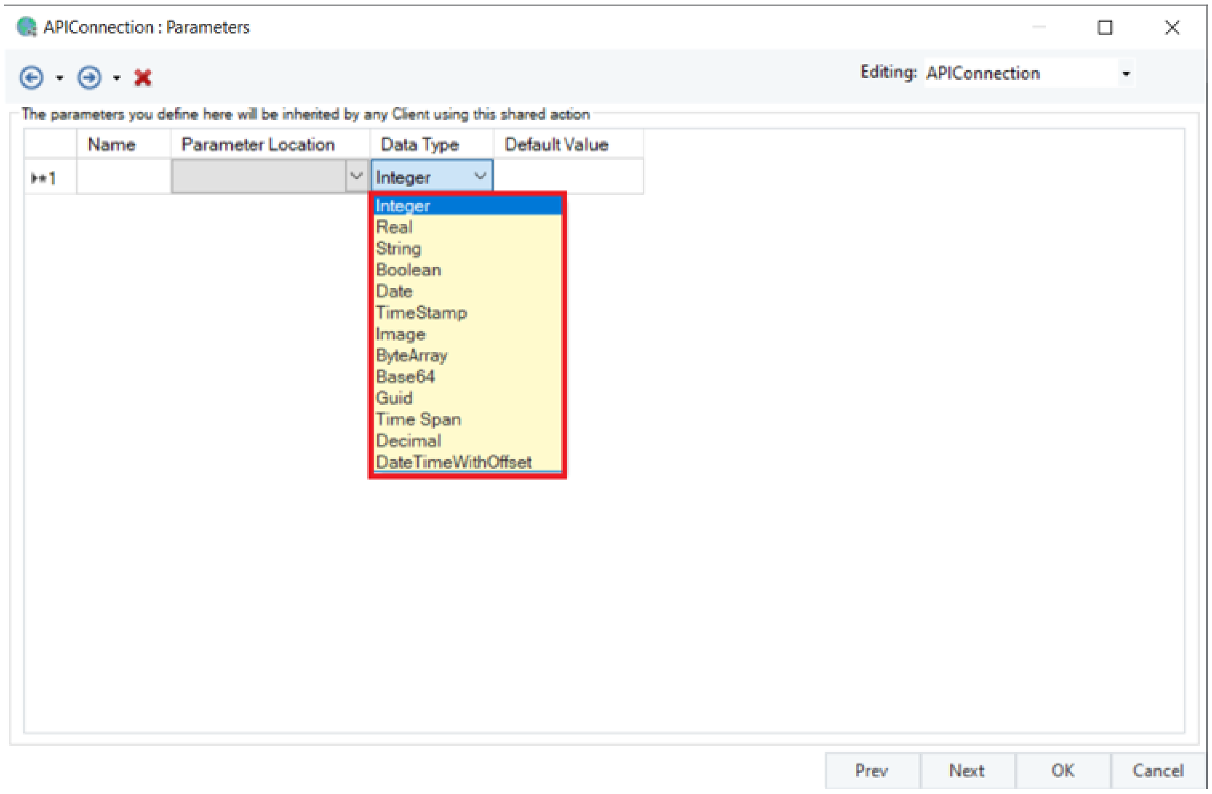
APIConnection : Parameters

Editing: APIConnection

The parameters you define here will be inherited by any Client using this shared action

	Name	Parameter Location	Data Type	Default Value
▸*1		Query		

Parameter Location: This option defines whether the parameter has a *Query* location or a *Header* location.



APIConnection : Parameters

Editing: APIConnection

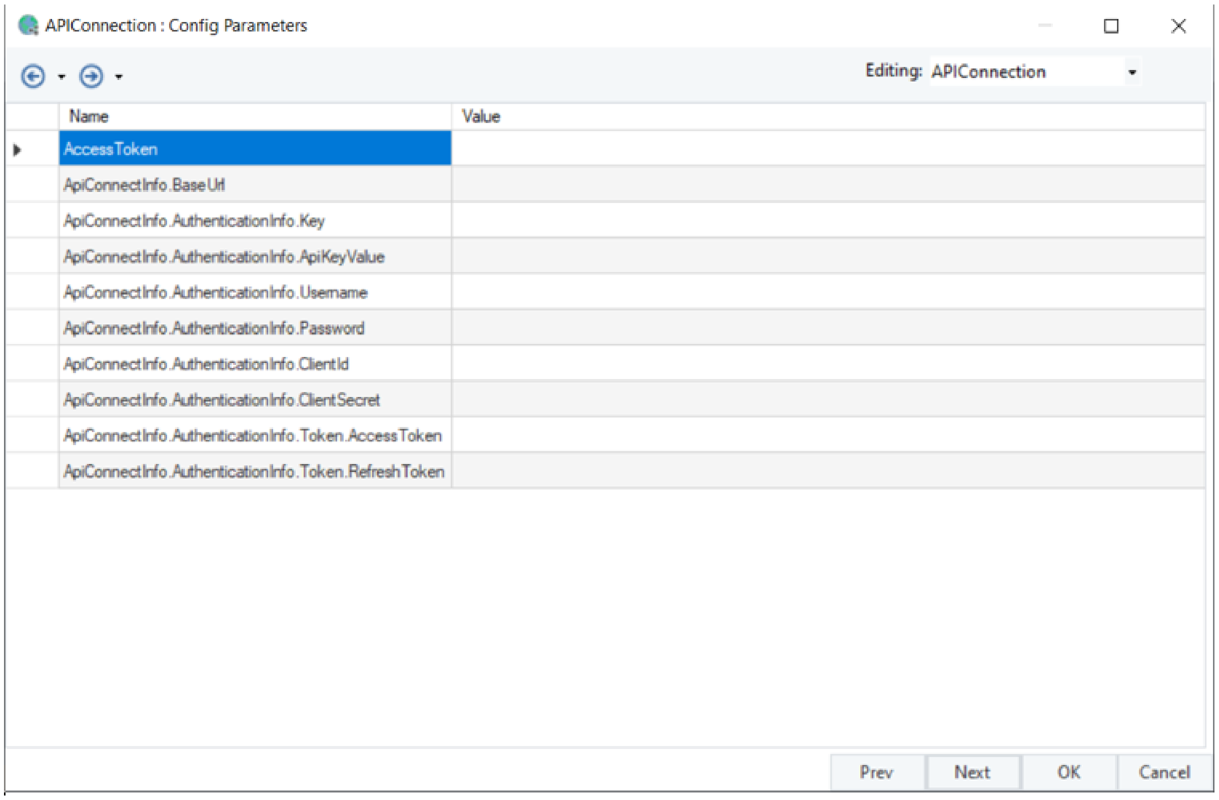
The parameters you define here will be inherited by any Client using this shared action

	Name	Parameter Location	Data Type	Default Value
▸*1			Integer	

Data Type: This option defines the data type of the parameter from a list of options.

The parameter values defined here will be inherited by all API clients using this connection unless overridden individually.

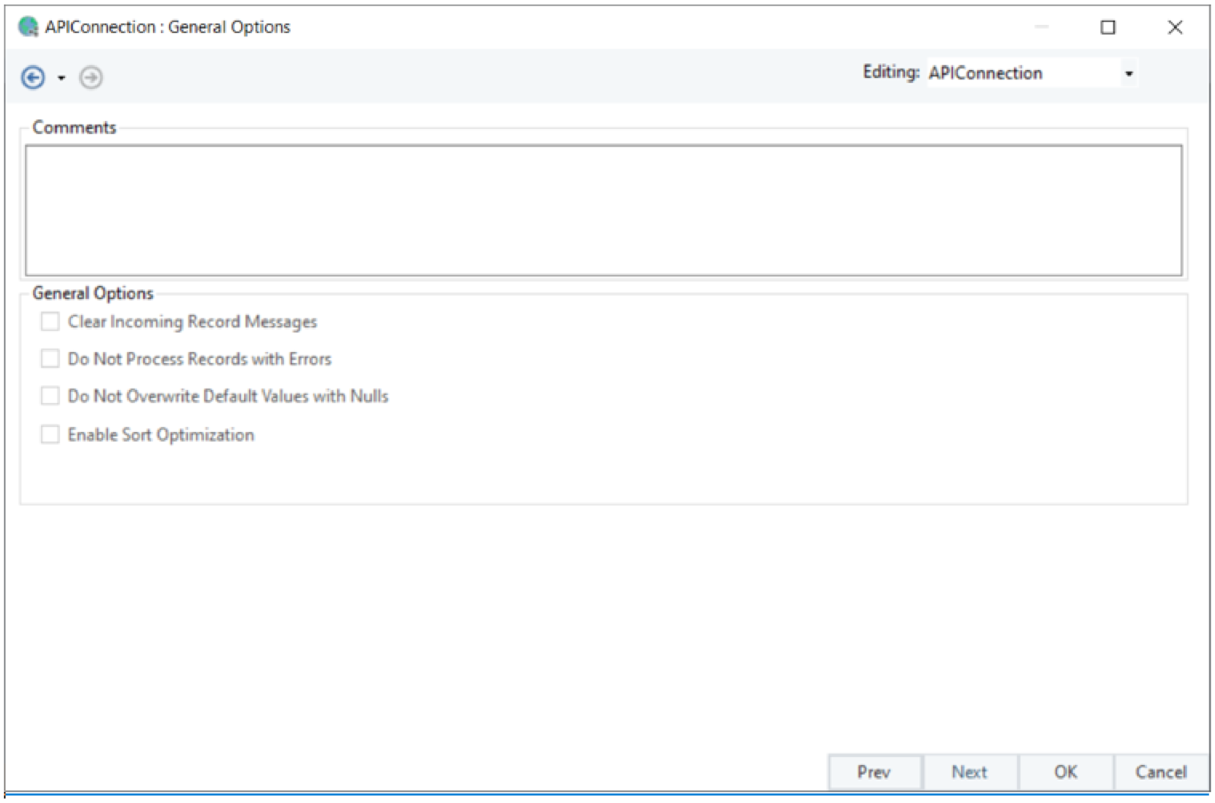
4. Once done, click *Next* and you will be led to the *Config Parameters* screen.



Name	Value
AccessToken	
ApiConnectInfo.BaseUrl	
ApiConnectInfo.AuthenticationInfo.Key	
ApiConnectInfo.AuthenticationInfo.ApiKeyValue	
ApiConnectInfo.AuthenticationInfo.Username	
ApiConnectInfo.AuthenticationInfo.Password	
ApiConnectInfo.AuthenticationInfo.ClientId	
ApiConnectInfo.AuthenticationInfo.ClientSecret	
ApiConnectInfo.AuthenticationInfo.Token.AccessToken	
ApiConnectInfo.AuthenticationInfo.Token.RefreshToken	

Here, config parameter values can be changed according to your application. Parameters not changed will use their default values.

5. Click *Next,* and you will be led to the *General Options* screen.


APIConnection : General Options

Editing: APIConnection

Comments

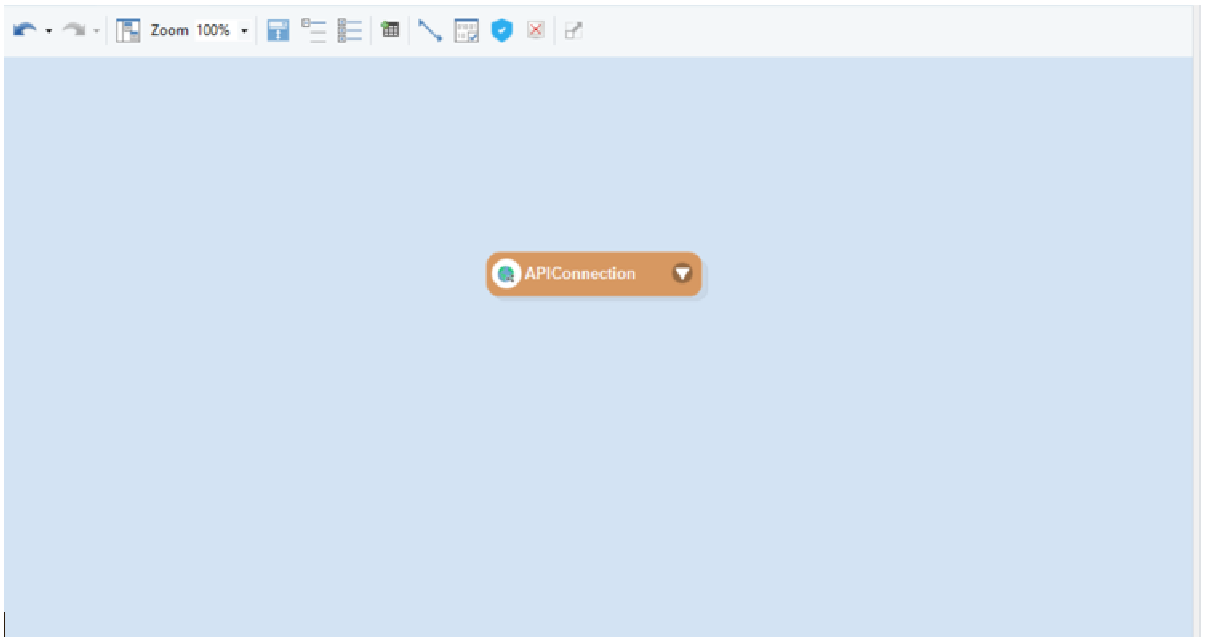
General Options

- [ ] Clear Incoming Record Messages
- [ ] Do Not Process Records with Errors
- [ ] Do Not Overwrite Default Values with Nulls
- [ ] Enable Sort Optimization

Here, you can add any *Comments* that you wish to add. The rest of the options for this object have been disabled.

6. Click *OK* to close the window.

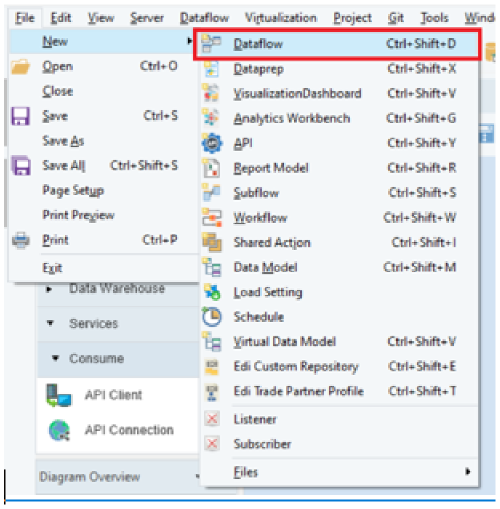
You have successfully configured the *API Connection* object.



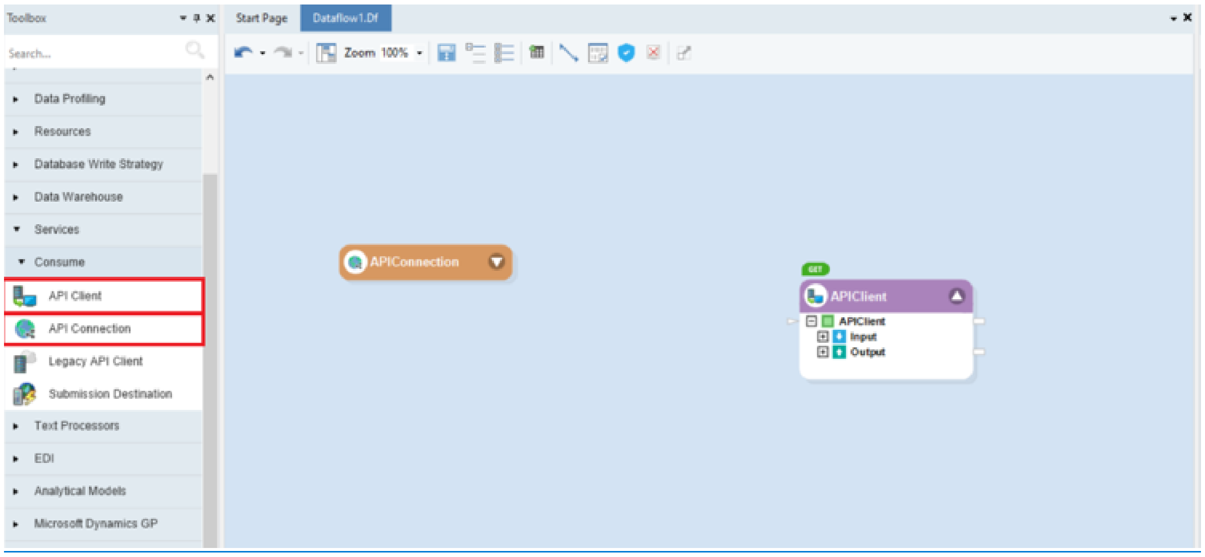
11.1.2 Using the API Connection Object

In a Dataflow

1. Click on *File* in the main toolbar, hover over *New*, and select *Dataflow* from the drop-down menu.



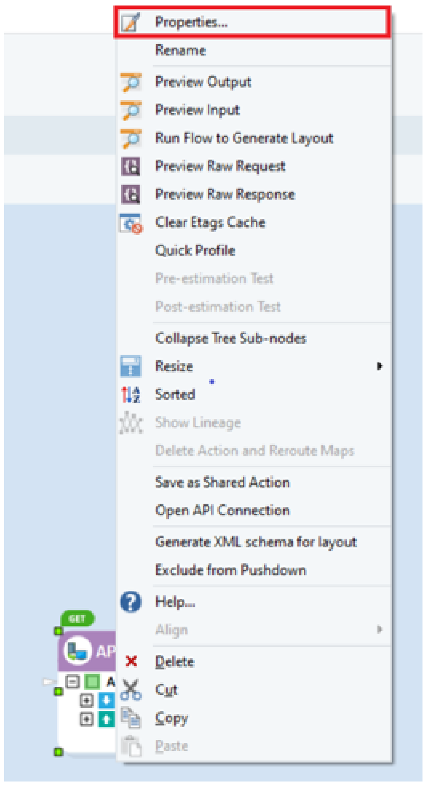
2. Once the dataflow is open, drag-and-drop the *API Connection* and *API Client* objects from the Toolbox onto the dataflow.



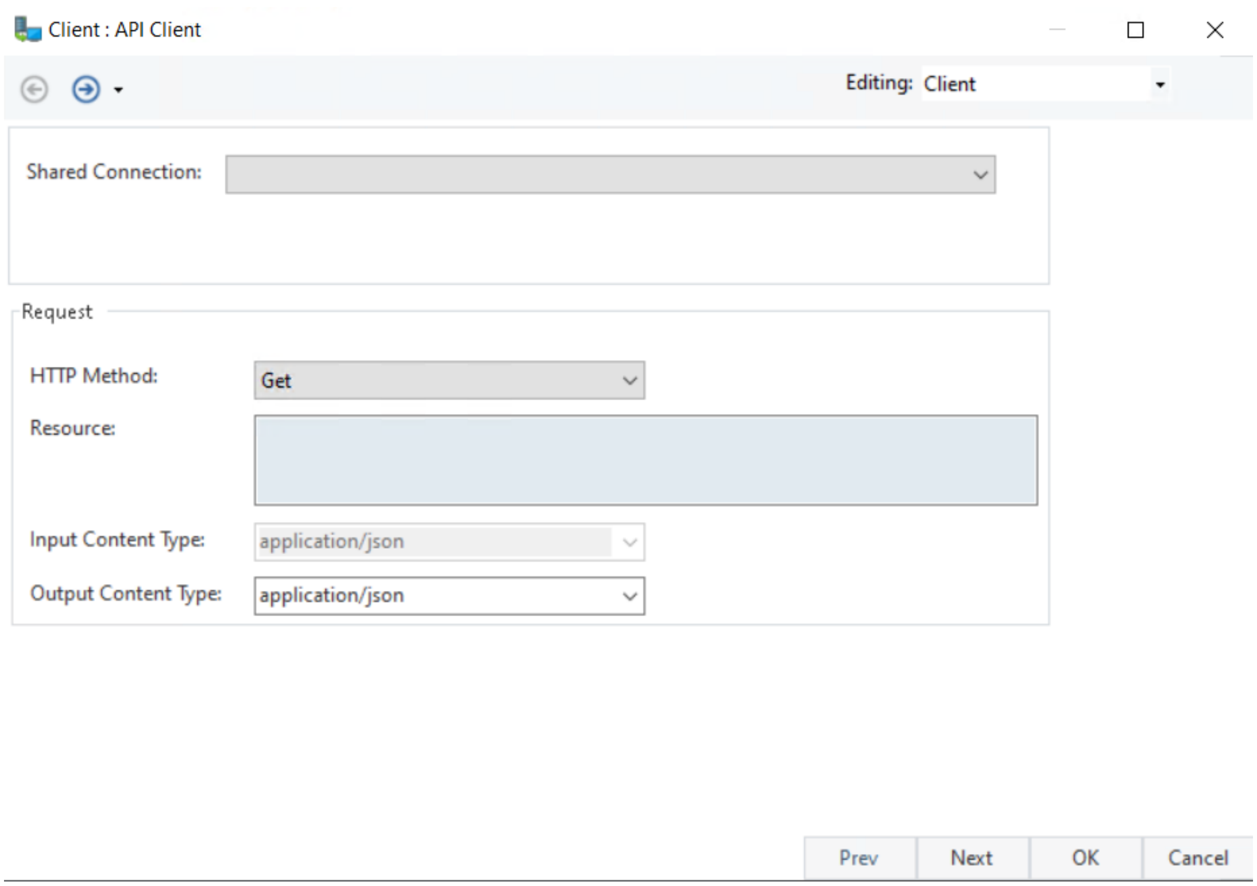
Note: The *API Connection* here can only be accessed within the scope of this dataflow.

3. Configure the *API* Connection object for the Base URL, Authentication.

Right-click on the *API Client* object and select *Properties* from the context menu.



A new *API Client* Properties window will open.



Shared Connection:

Request

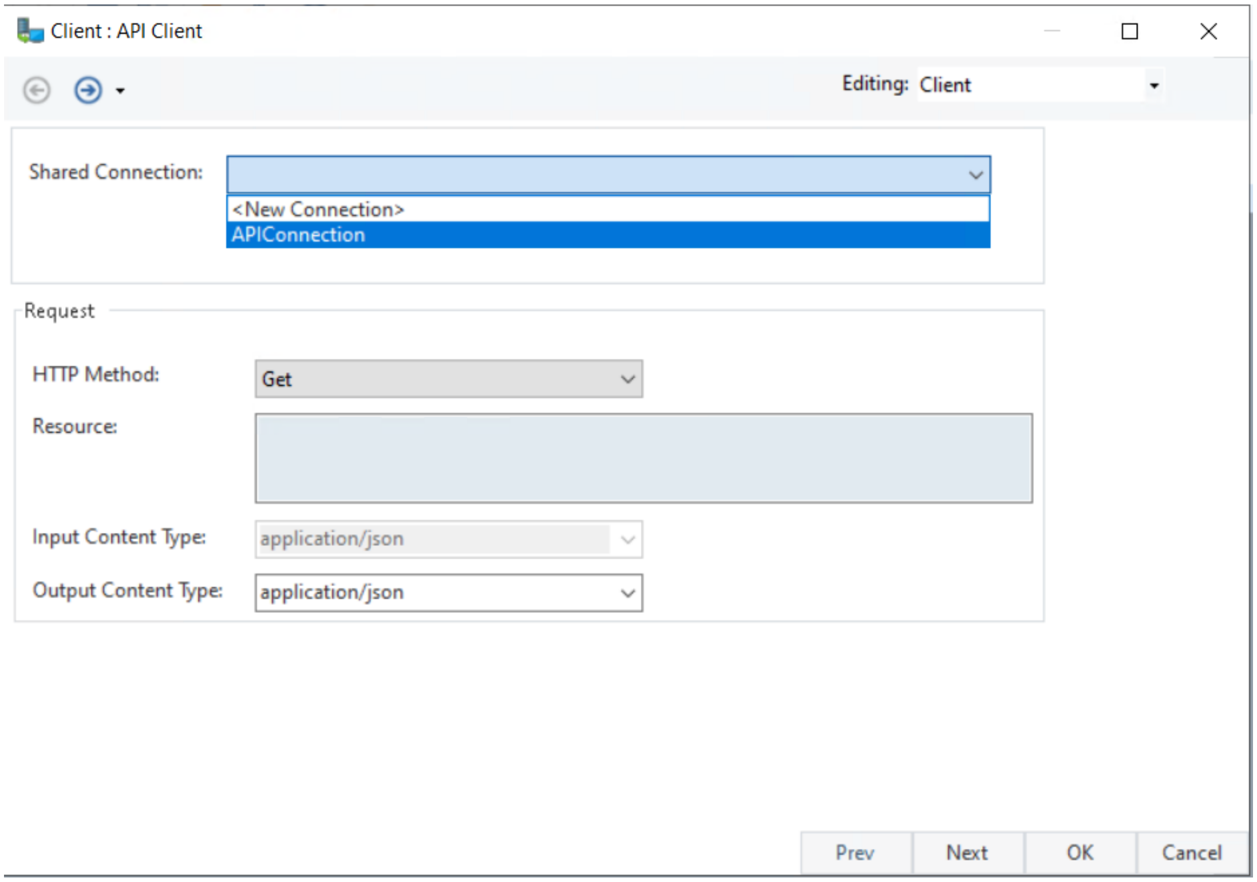
HTTP Method: Get

Resource:

Input Content Type: application/json

Output Content Type: application/json

The *Shared Connection* dropdown list shows us the *API Connection* object present in the same dataflow.



Shared Connection:
<New Connection>
APIConnection

Request

HTTP Method: Get

Resource:

Input Content Type: application/json

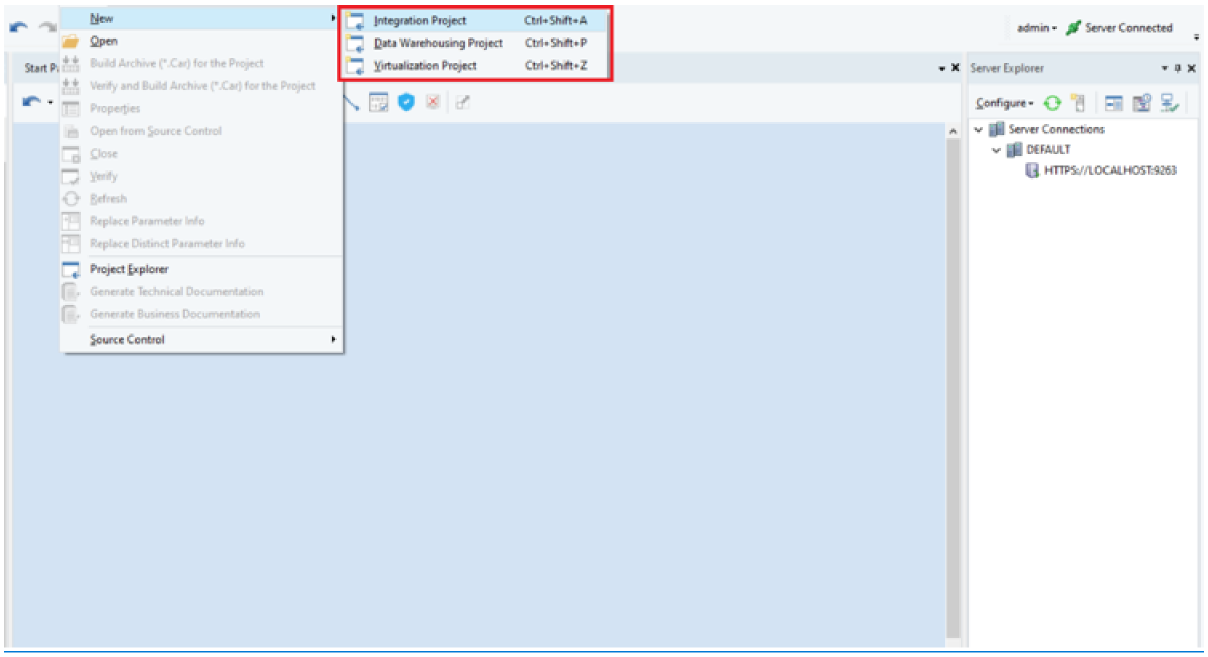
Output Content Type: application/json

You can now use this *API Client* object to make API calls within Astera API Management.

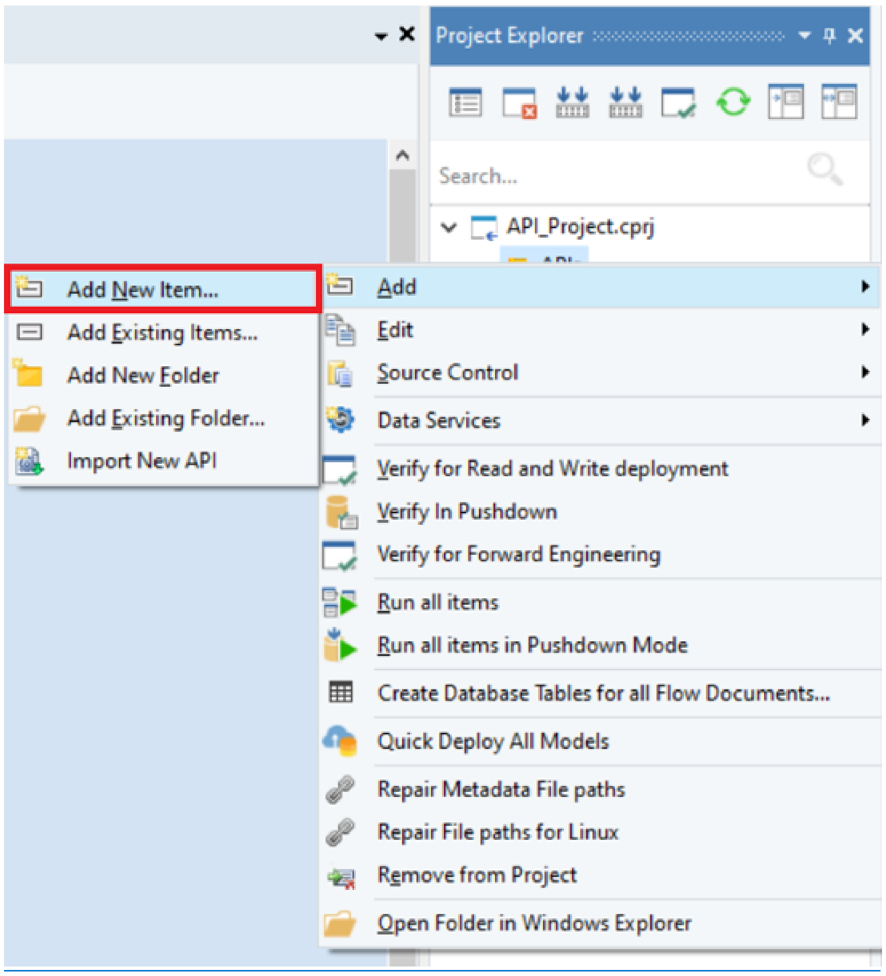
In a Project

1. Navigate to the main toolbar, click *Project*, hover over *New*, and select a project type. Please click here for more information on creating projects.

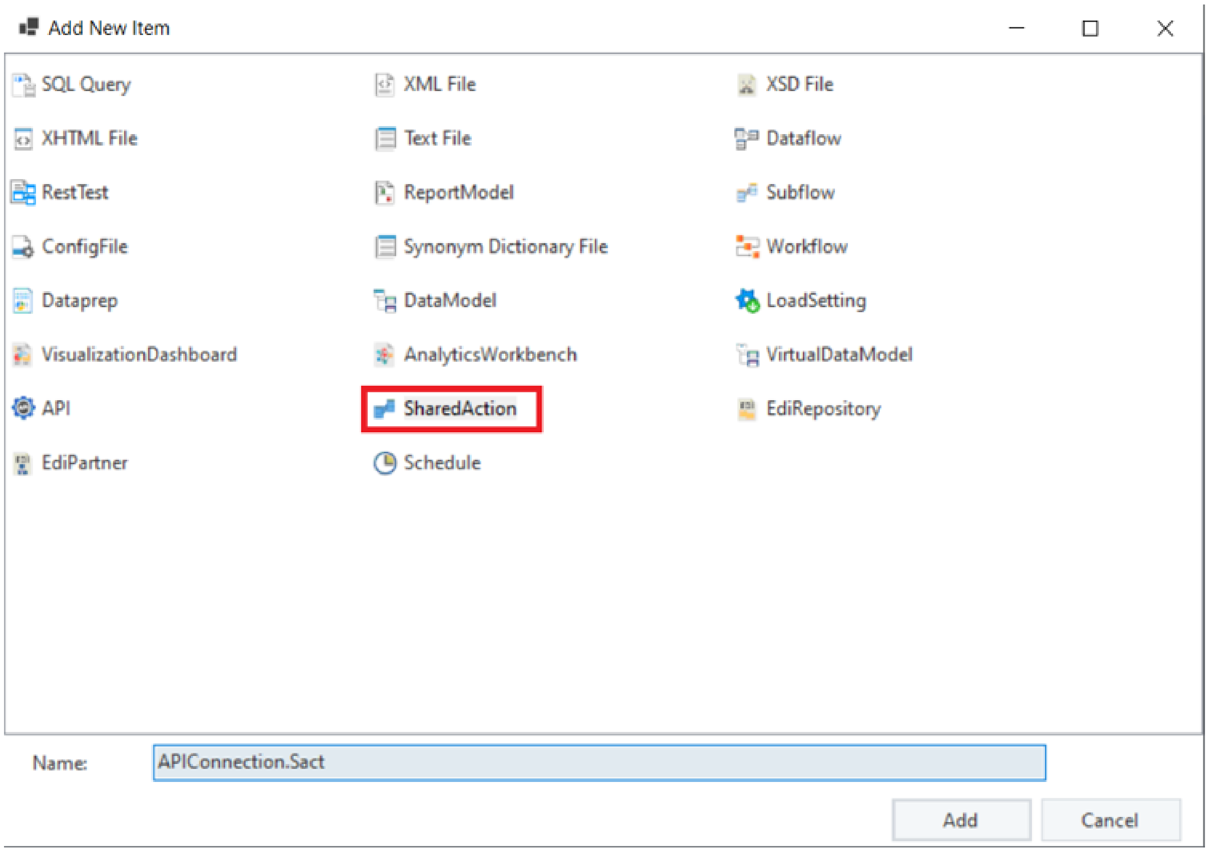
Note: You can also open a previously existing project.


New
Open
Build Archive (*.Car) for the Project
Verify and Build Archive (*.Car) for the Project
Properties
Open from Source Control
Close
Verify
Refresh
Replace Parameter Info
Replace Distinct Parameter Info
Project Explorer
Generate Technical Documentation
Generate Business Documentation
Source Control
Integration Project Ctrl+Shift+A
Data Warehousing Project Ctrl+Shift+P
Virtualization Project Ctrl+Shift+Z
admin
Server Connected
Server Explorer
Configure
Server Connections
DEFAULT
HTTPS://LOCALHOST:9263

2. Locate the Project Explorer on the right, right-click on the project or one of its folders and select *Add New Item* from the context menu.



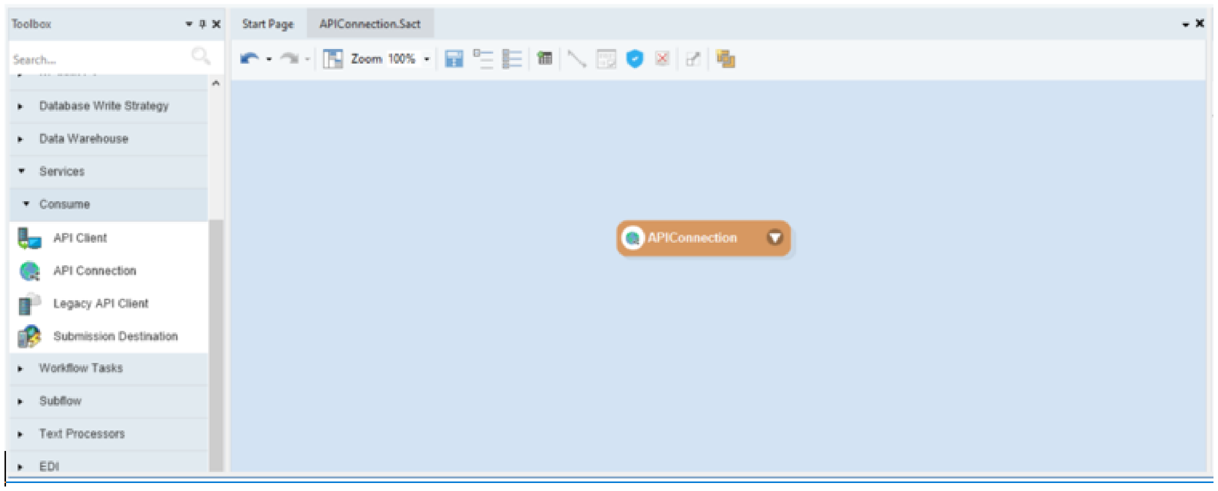
This will open a new window where a new *SharedAction* can be added to the project.



Name: APIConnection.Sact

3. Within the *SharedAction* file, drag-and-drop the *API Connection* object from the Toolbox.

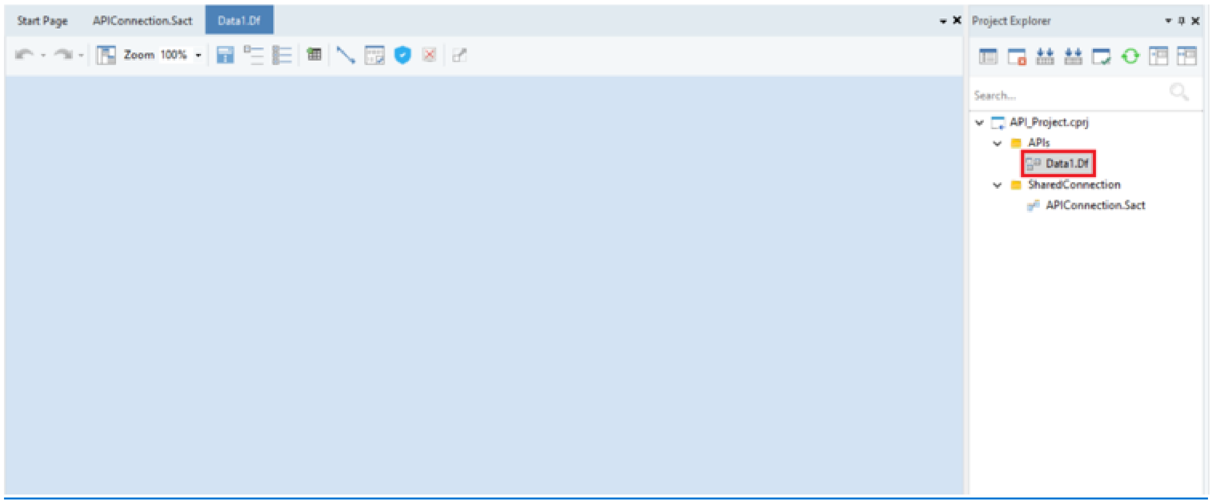
Note: The *SharedAction* file should only contain a single *API Connection* object.



4. Configure the *API Connection* object with *Base URL, Authentication, and Shared Parameters* and save the *SharedAction* file

This *API Connection* can be used in any flow document contained in the same project.

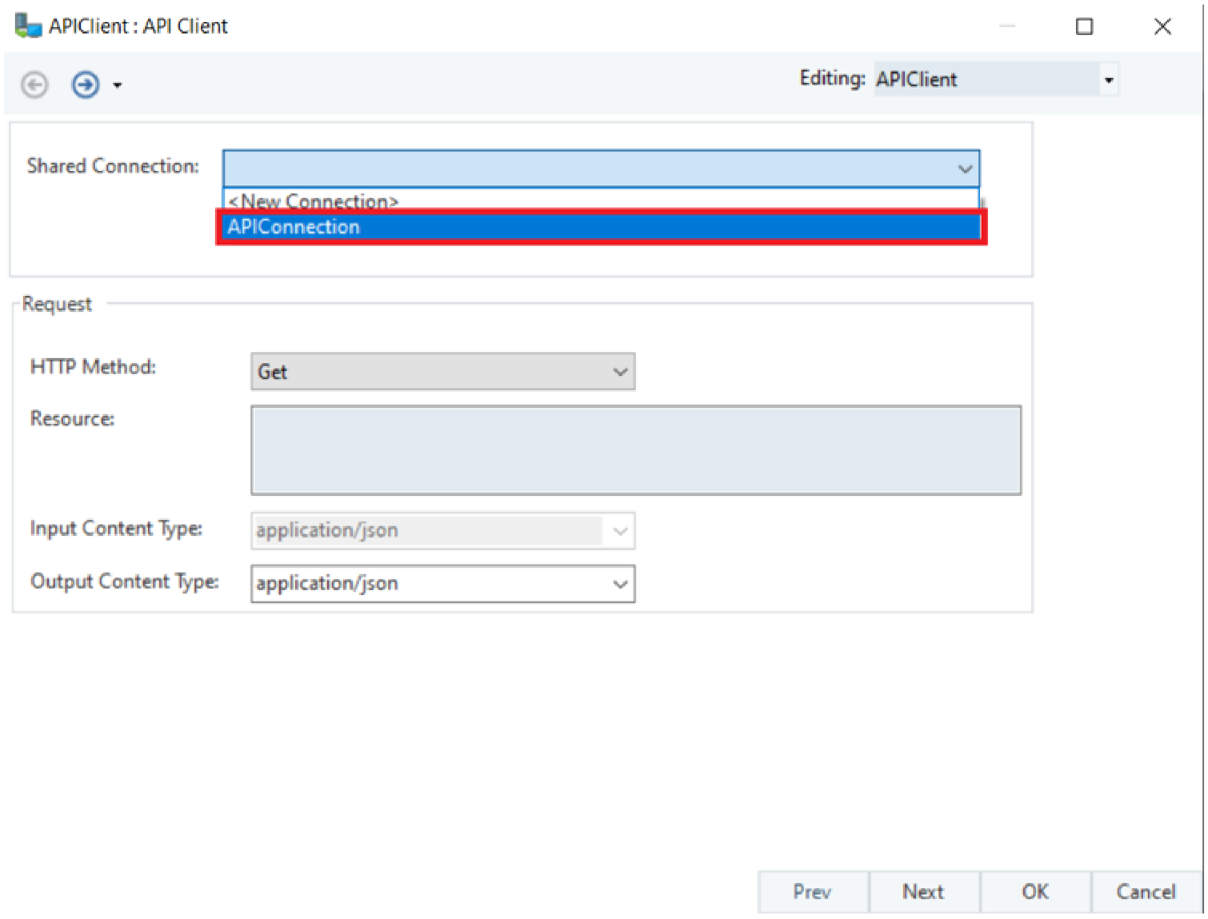
5. Next, open a new dataflow within the project.



6. Drag-and-drop the *API Client* object onto the dataflow, right-click on it, and select *Properties* from the context menu.

A new window will open.

Here, you can see the name of the *Shared Connection* within the drop-down menu of the *Properties* option.


APIClient : API Client

Editing: APIClient

Shared Connection:

<New Connection>

APIConnection

Request

HTTP Method: Get

Resource:

Input Content Type: application/json

Output Content Type: application/json

Note: Within the project, the shared *API Connection* can be accessed within any flow.

- If shared connections with duplicate names exist in the project, only one will be shown and used.

- If duplicate connections exist in the flow and the project, the flow connection will be given preference.

This concludes our discussion on the configuration and use of the *API Connection* object in Astera API Management.

11.2 Making API Calls with the API Client Object in Astera API Management

To make an API call in Astera API Management, an *API Client* object, along with its *API Connection*, needs to be configured.

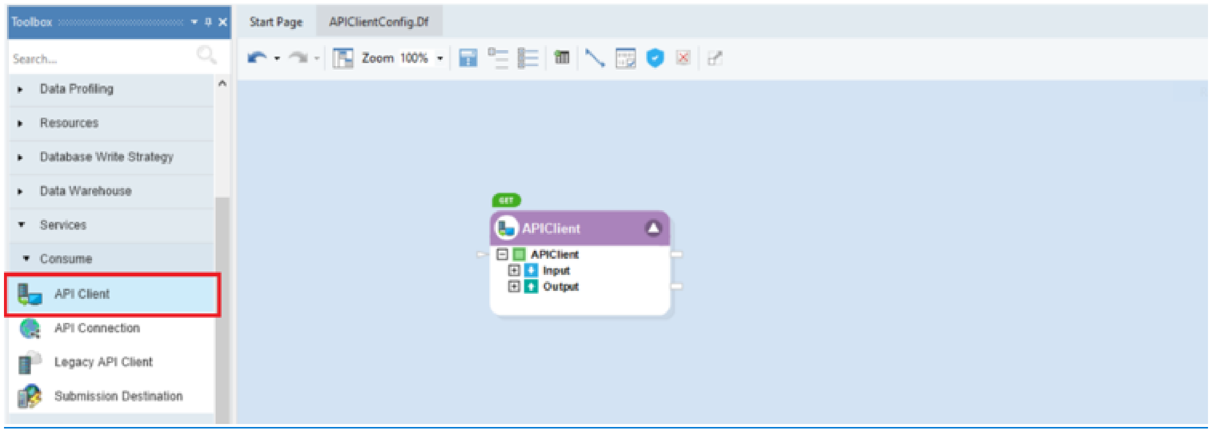
First, drag and drop an *API Connection* object from the Toolbox and configure it in the dataflow. Alternatively, you can use an *API Connection* object in a shared action file within the scope of the project you are working with.

The *API Connection* object contains the Base URL, authentication details, and shared parameters for the API endpoint.

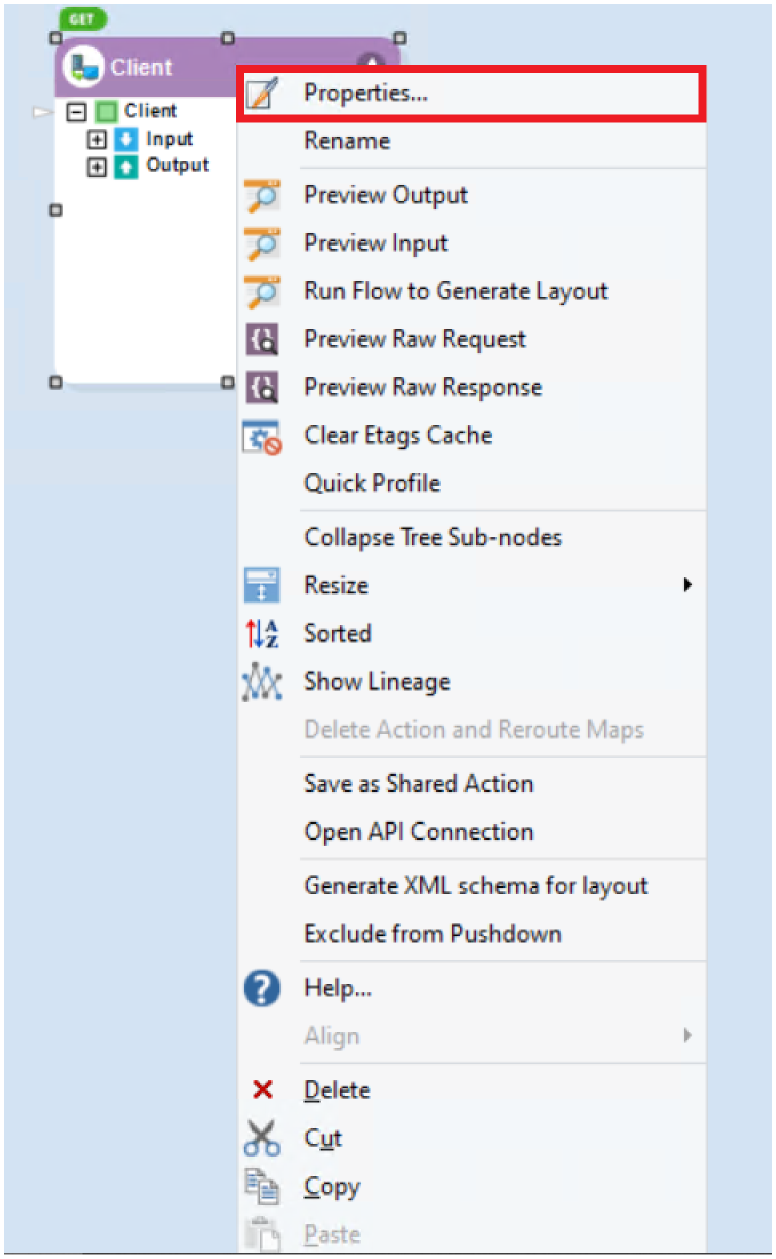
You can learn all about the configuration and usage of the *API Connection* object here.

Next, let's configure the *API Client* object.

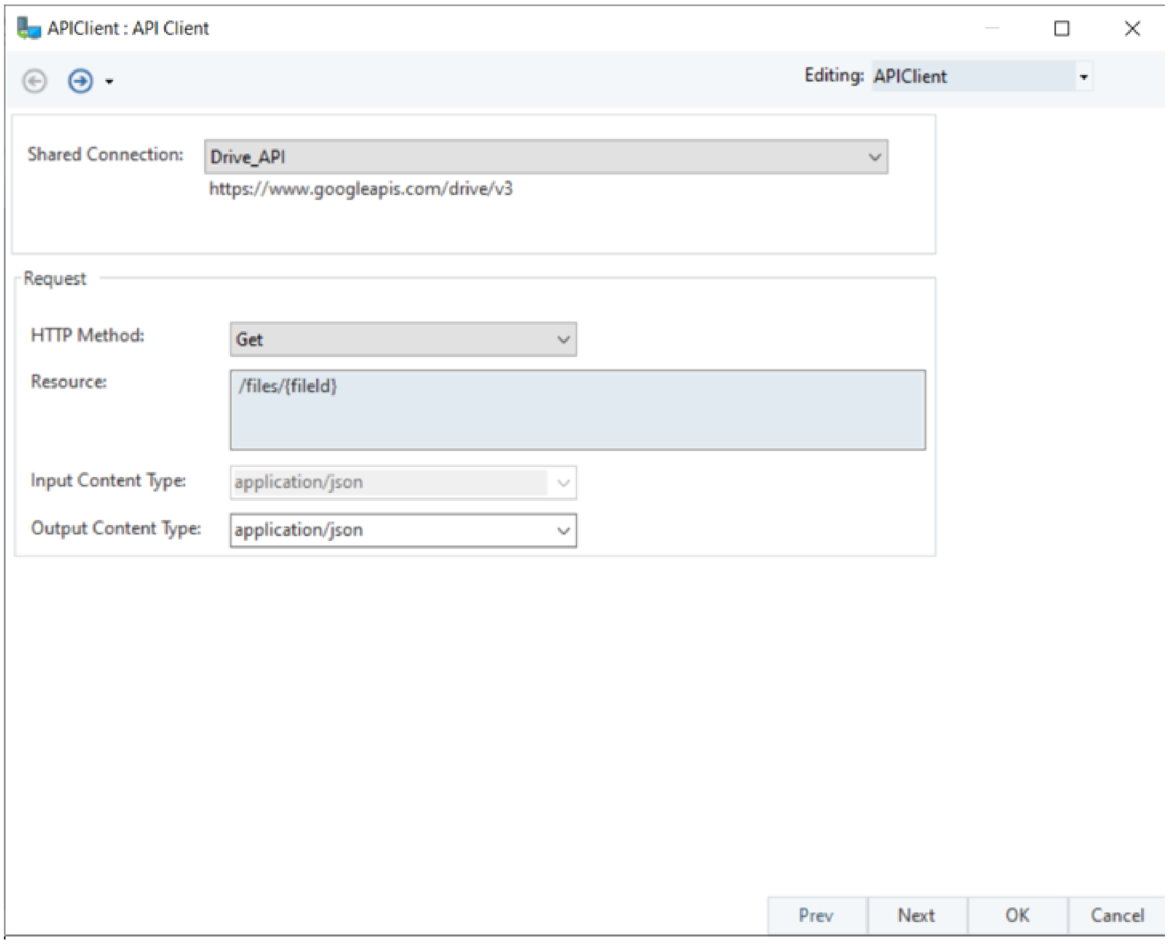
1. First, drag and drop an *API Client* object from the Toolbox onto the dataflow.



2. Right-click on the *API Client* object's header and select *Properties*.



The *API Client* screen will now open. Here you will have to specify the following,



Shared Connection: Drive_API
https://www.googleapis.com/drive/v3

Request

HTTP Method: Get
Resource: /files/{fileId}
Input Content Type: application/json
Output Content Type: application/json

Shared Connection: Establish your *API Client's* connection from this drop-down that lists all shared connections from within the flow as well as from the project.

HTTP Method: The HTTP request verb defines the operation you want to make on the API resource.

Resource: the resource of the API from which you want to make a request. This will be appended after the Base URL from the selected shared connection to form the complete endpoint. Any URI or path parameters must be included in the resource text enclosed in curly brackets, { }.

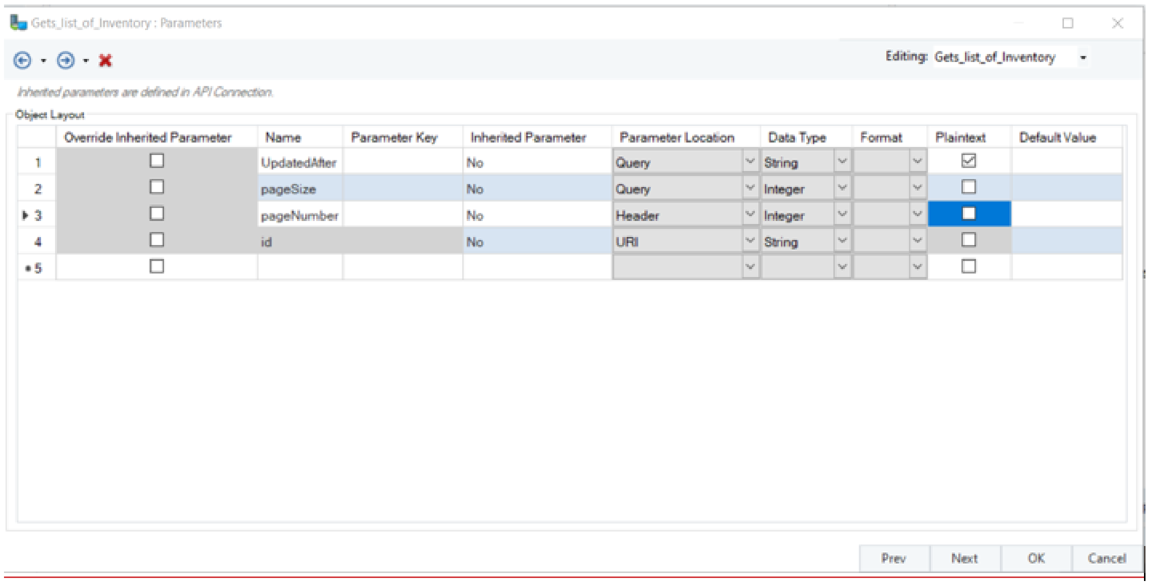
Input Content Type: This is the content-type header for the request payload which is default to application/JSON type. The actual request payload layout can be defined in the input layout screen.

Output Content Type: This is the content type of the response payload which is default to application/JSON type. The actual response payload layout can be defined in the output layout screen.

Note: For an unsupported type, a relevant pop-up notification will appear on-screen.

3. Click *Next*. A *Parameters* screen will appear.

Here you will have to specify the following,



Gets_list_of_Inventory : Parameters

Editing: Gets_list_of_Inventory

Inherited parameters are defined in API Connection.

Object Layout

	Override Inherited Parameter	Name	Parameter Key	Inherited Parameter	Parameter Location	Data Type	Format	Plaintext	Default Value
1	☐	UpdatedAfter		No	Query	String		☑	
2	☐	pageSize		No	Query	Integer		☐	
3	☐	pageNumber		No	Header	Integer		☐	
4	☐	id		No	URI	String		☐	
*5	☐							☐	

Override Inherited Parameter: Check this to override any parameters previously defined and inherited from the shared connection.

Name: The name of your parameter.

Parameter Key: Since the Name column does not allow any special characters, the parameter key can be used to define an alternate name including any special characters to replace the name in the API request.

Parameter Location: The parameter type such as Query, URI, and Header.

Data Type: Specify the data type of your parameter.

Format: Define the datatype format of the parameter's value sent in the API request.

Plaintext: Check this box to disable URL encoding the parameters when the request is sent. The parameters will be sent in plaintext format, or you could optionally encode your parameter values manually using the URLEncode function from the toolbox.

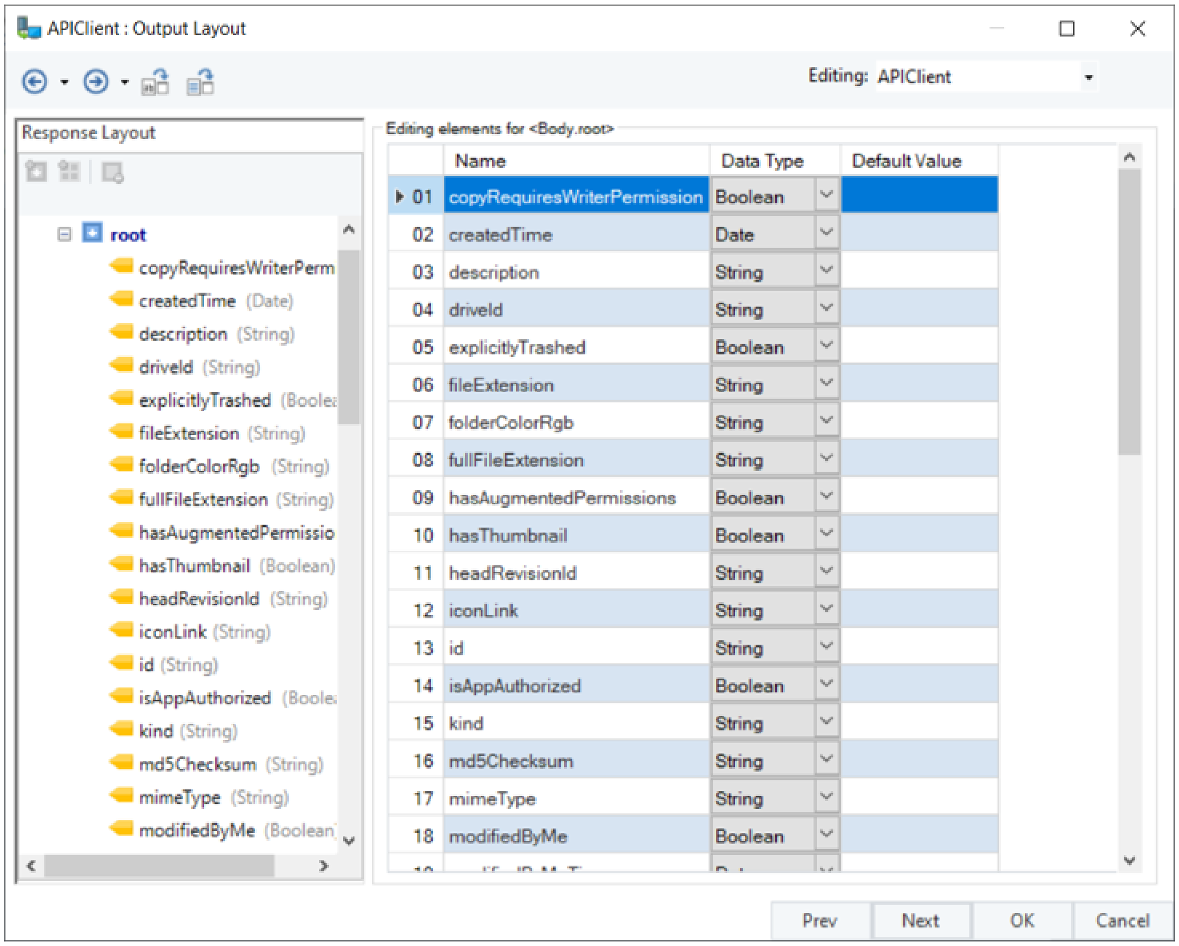
Default Value: The parameter's value for which you want to make a request.

Note: Any values mapped to the input node of the object will take preference.

4. Click *Next*. An *API Client Output Layout* screen will now open.

Here, we will select the *Generate Layout by Running Request* to build the response layout. Alternatively, you can build the layout manually or use a sample text.

Next, click *OK*.



APIClient : Output Layout

Editing: APIClient

Response Layout

- root
  - copyRequiresWriterPerm
  - createdTime (Date)
  - description (String)
  - driveId (String)
  - explicitlyTrashed (Boolea
  - fileExtension (String)
  - folderColorRgb (String)
  - fullFileExtension (String)
  - hasAugmentedPermissio
  - hasThumbnail (Boolean)
  - headRevisionId (String)
  - iconLink (String)
  - id (String)
  - isAppAuthorized (Boolea
  - kind (String)
  - md5Checksum (String)
  - mimeType (String)
  - modifiedByMe (Boolean

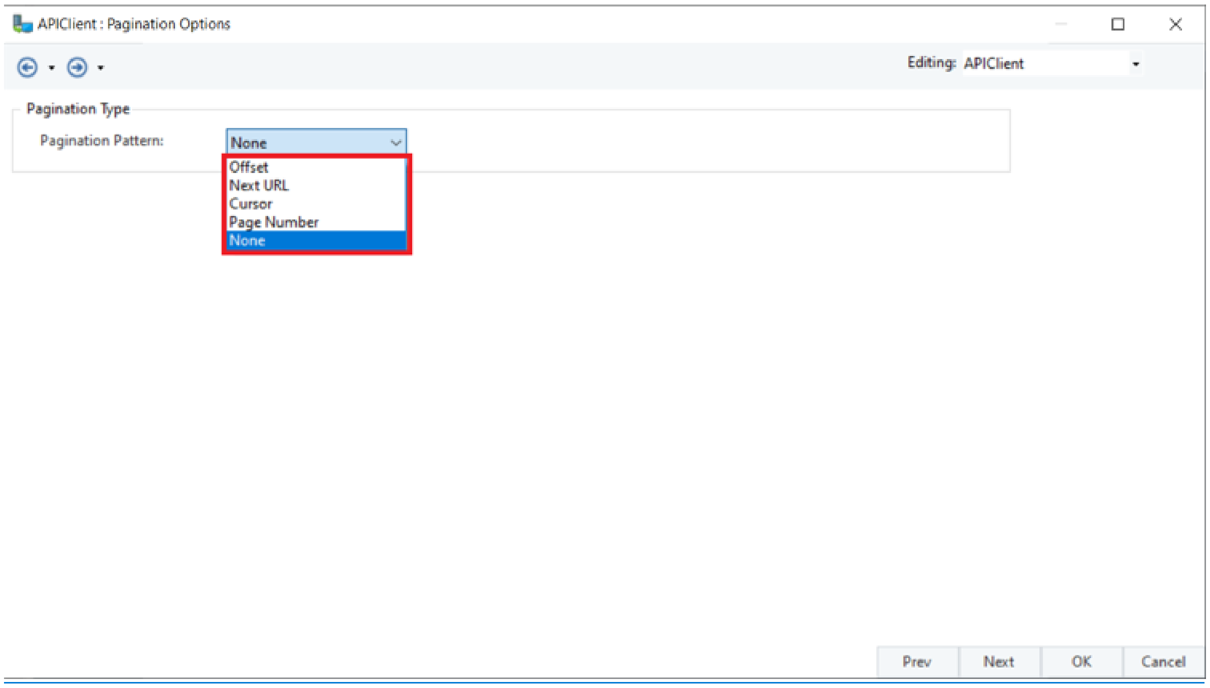
Editing elements for <Body.root>

	Name	Data Type	Default Value
01	copyRequiresWriterPermission	Boolean	
02	createdTime	Date	
03	description	String	
04	driveId	String	
05	explicitlyTrashed	Boolean	
06	fileExtension	String	
07	folderColorRgb	String	
08	fullFileExtension	String	
09	hasAugmentedPermissions	Boolean	
10	hasThumbnail	Boolean	
11	headRevisionId	String	
12	iconLink	String	
13	id	String	
14	isAppAuthorized	Boolean	
15	kind	String	
16	md5Checksum	String	
17	mimeType	String	
18	modifiedByMe	Boolean	

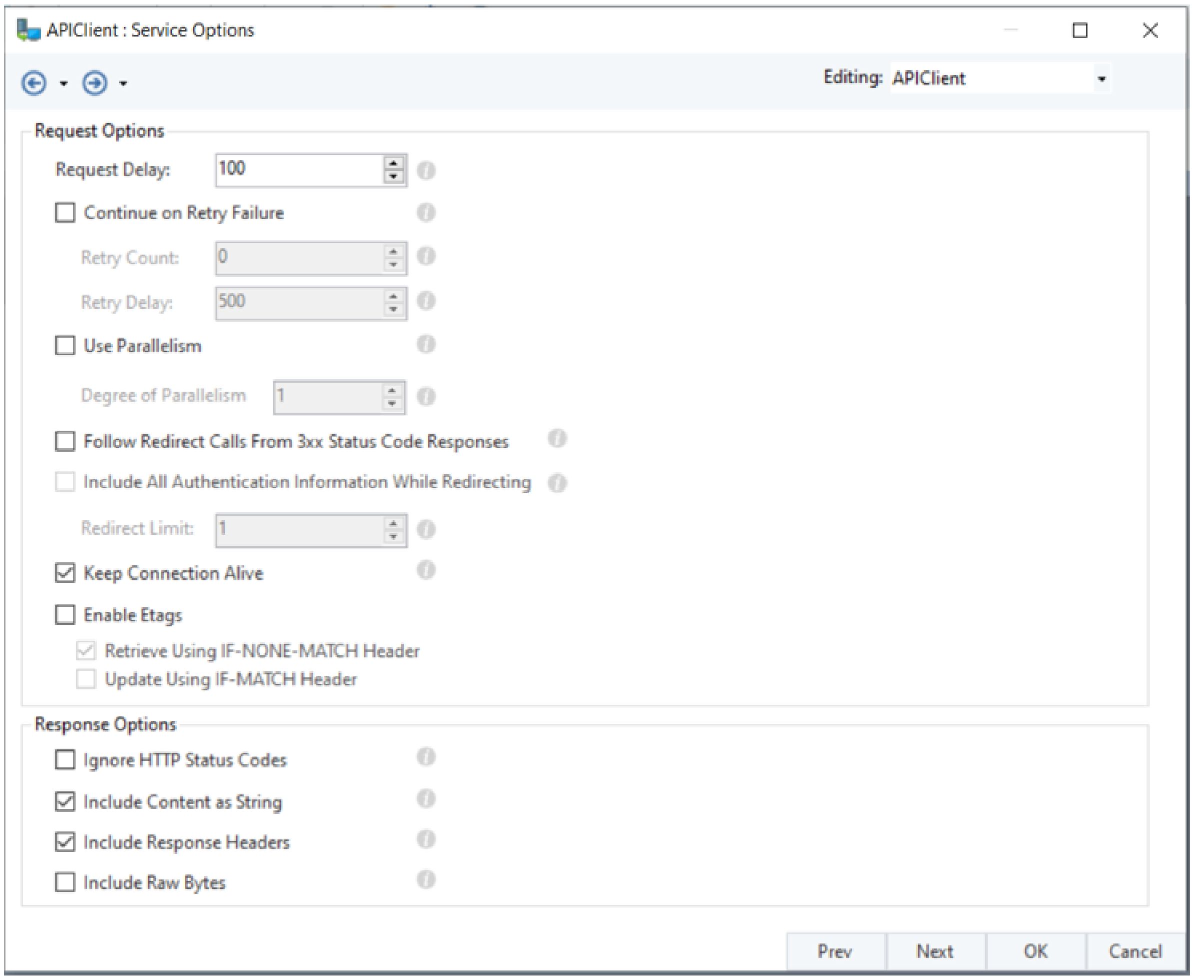
Note: Prior to this screen, there will be an additional screen to configure an *API Client* input layout for the following methods: POST, PUT, and PATCH.

5. Once done, click *Next*, and you will be led to the *Pagination Options* screen.

Here, you can select the type of pagination that has been specified by the API providers. Astera API Management offers the following pagination types.



6. When done, click *Next*, and you will be taken to the *Service Options* screen.


APIClient : Service Options

Editing: APIClient

Request Options

Request Delay: 100

☐ Continue on Retry Failure

Retry Count: 0

Retry Delay: 500

☐ Use Parallelism

Degree of Parallelism 1

☐ Follow Redirect Calls From 3xx Status Code Responses

☐ Include All Authentication Information While Redirecting

Redirect Limit: 1

☑ Keep Connection Alive

☐ Enable Etags

☑ Retrieve Using IF-NONE-MATCH Header

☐ Update Using IF-MATCH Header

Response Options

☐ Ignore HTTP Status Codes

☑ Include Content as String

☑ Include Response Headers

☐ Include Raw Bytes

Request Options -

Request Delay: Delay time (in milliseconds) before sending a request.

Retry Count: Number of retry attempts to be made in case of a time-out error.

Retry Delay: The duration (in milliseconds) between each consecutive retry attempt.

Continue on Retry Failure: Check to succeed the flow even after all retries have failed.

Use Parallelism: Check this option to send requests in parallel. Check this to send requests in parallel. Number of requests to be sent in parallel (max limit of 10).

Follow Redirect: Check to allow forwarding a 3xx response to the redirected URL.

Include Authentication: Check to include authentication in the redirected API call.

Redirect Limit: Number of allowed redirect calls from a request. -1 indicates no limit.

Keep Connection Alive: Check to keep the TCP connection open to reuse for all subsequent requests to the same server.

Enable E-Tags: To learn about E-Tags, click here.

Retrieval: Check this to enable e-tags to request caching for GET requests.

Updates: Check this to enable request concurrency control using etags for PUT, PATCH or DELETE requests.

Response Options -

Ignore HTTP Status Codes: Selecting this option will show and allow processing responses other than 2xx in the flow, which are otherwise considered an error.

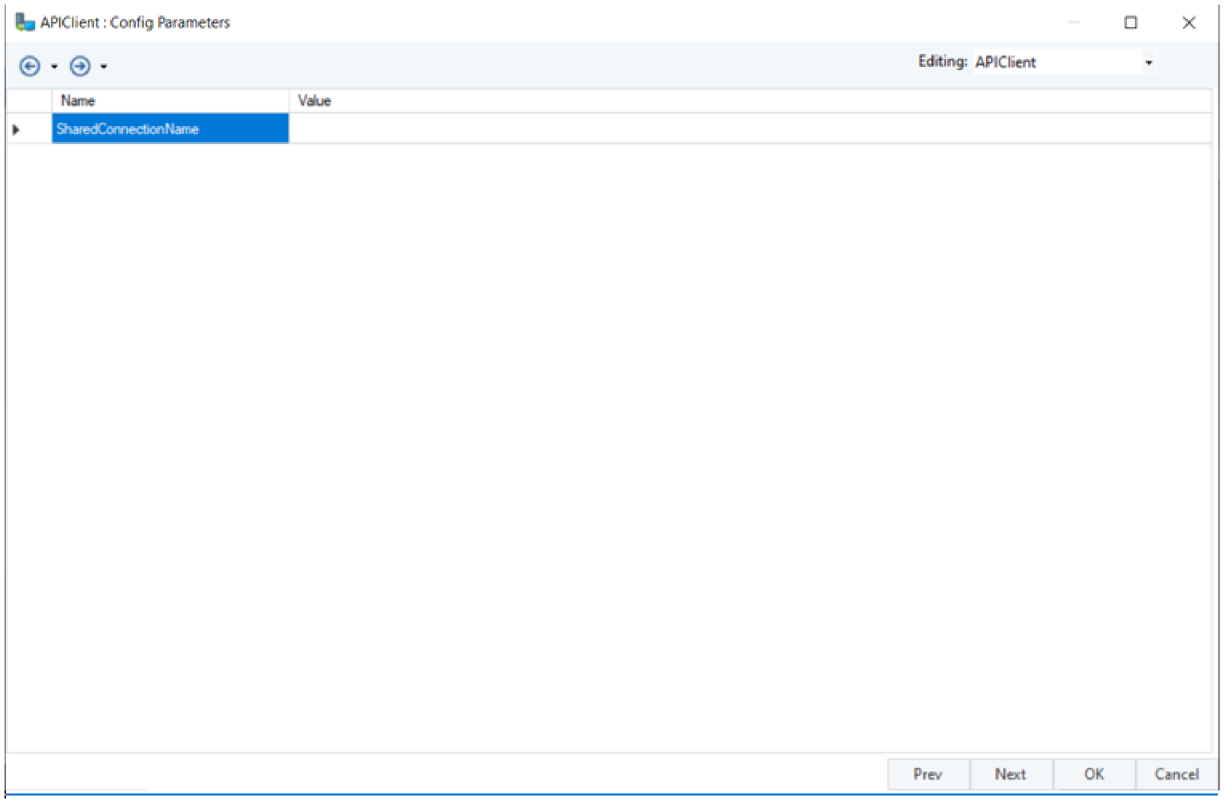
Include Content as String: Adds a field for serialized response content string in the *Response-Info* output node.

Include Response Headers: Adds all response headers as a collection in the *Response-Info* output node.

Include Raw Bytes: Adds a field for response content in the form of raw bytes in the *Response-Info* output node.

7. Click *Next*, and the *Config Parameters* screen will appear.

Config Parameters can enable the deployment of flows by eliminating hardcoded values and provide a dynamic way of changing multiple configurations with a simple value change.

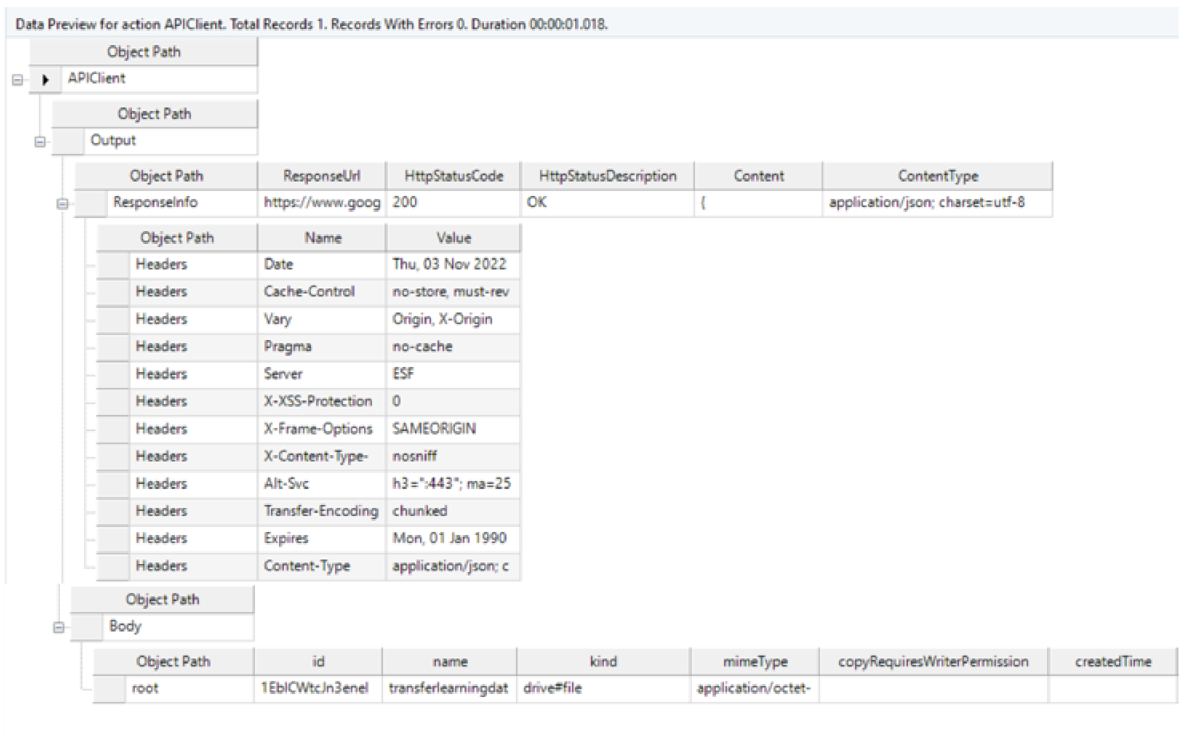


Name	Value
SharedConnectionName	

8. Click *OK*, and the *API Client* object will be configured.

Now, right-click on the *API Client* object's header, and select *Preview Output*.

Your request has been executed successfully, as you can see that the HTTP status code is 200 which means that the *API Client* has successfully carried out the GET request for the provided status.



Data Preview for action APIClient. Total Records 1. Records With Errors 0. Duration 00:00:01.018.

Object Path
APIClient

Object Path
Output

Object Path	ResponseUrl	HttpStatusCode	HttpStatusDescription	Content	ContentType
ResponseInfo	https://www.goog	200	OK	{	application/json; charset=utf-8

Object Path	Name	Value
Headers	Date	Thu, 03 Nov 2022
Headers	Cache-Control	no-store, must-rev
Headers	Vary	Origin, X-Origin
Headers	Pragma	no-cache
Headers	Server	ESF
Headers	X-XSS-Protection	0
Headers	X-Frame-Options	SAMEORIGIN
Headers	X-Content-Type-	nosniff
Headers	Alt-Svc	h3=":443"; ma=25
Headers	Transfer-Encoding	chunked
Headers	Expires	Mon, 01 Jan 1990
Headers	Content-Type	application/json; c

Object Path
Body

Object Path	id	name	kind	mimeType	copyRequiresWriterPermission	createdTime
root	1EbICWtcJn3enel	transferlearningdat	drive#file	application/octet-		

This concludes our discussion on making API calls with the *API Client* object in Astera API Management.

11.3 API Browser

11.3.1 What is an API?

API (Application Programming Interface) is defined as an interface or medium through which one software communicates with another. In other words, it is a set of contracts that allows different software systems to share information with each other. The greatest advantage of an API is that different programs and devices can communicate with each other in a secure manner, without interference.

APIs are messengers that conform to the technical contract between two parties. They are language and platform-independent, which means C# can talk to Java, and Unix can communicate with Mac without any difficulty. An API is not the same as a remote server. In fact, it is part of a remote server that receives requests and sends responses. More precisely, an API is a structured request and response.

11.3.2 API Browser in Astera API Management

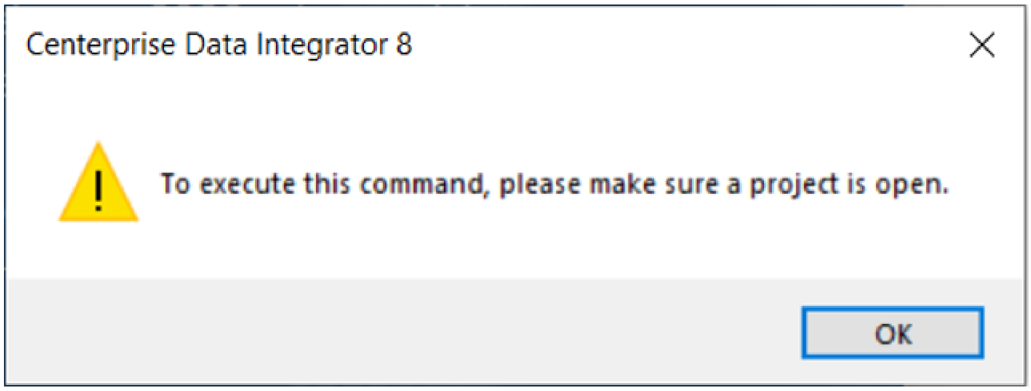
The API Browser in Astera API Management has narrowed down the steps to make HTTP calls using just one-step authentication. Once you have imported an API in API Management, all endpoint operations in that API are populated at once. API definition describes what requests are available and what the responses will look like.

So, once you load an API definition, all supported methods are populated in the *API Browser* unlike Legacy, where all supported methods must be configured separately in each object.

There are two methods of configuring APIs in Astera Centerprise. For open APIs, you only need to provide the *API Import Source* and *File Path* or *Base URL* to configure the connection with a specific API. Once this standardized information is provided, any API that you have imported will populate in API Management's *API Browser*, along with their methods, for example, GET, PUT, POST, PATCH, and DELETE, and they will remain accessible until their authentication period expires. From the *API Browser* in Astera API Management, you can simply drag and drop operations, and use them in your flows.

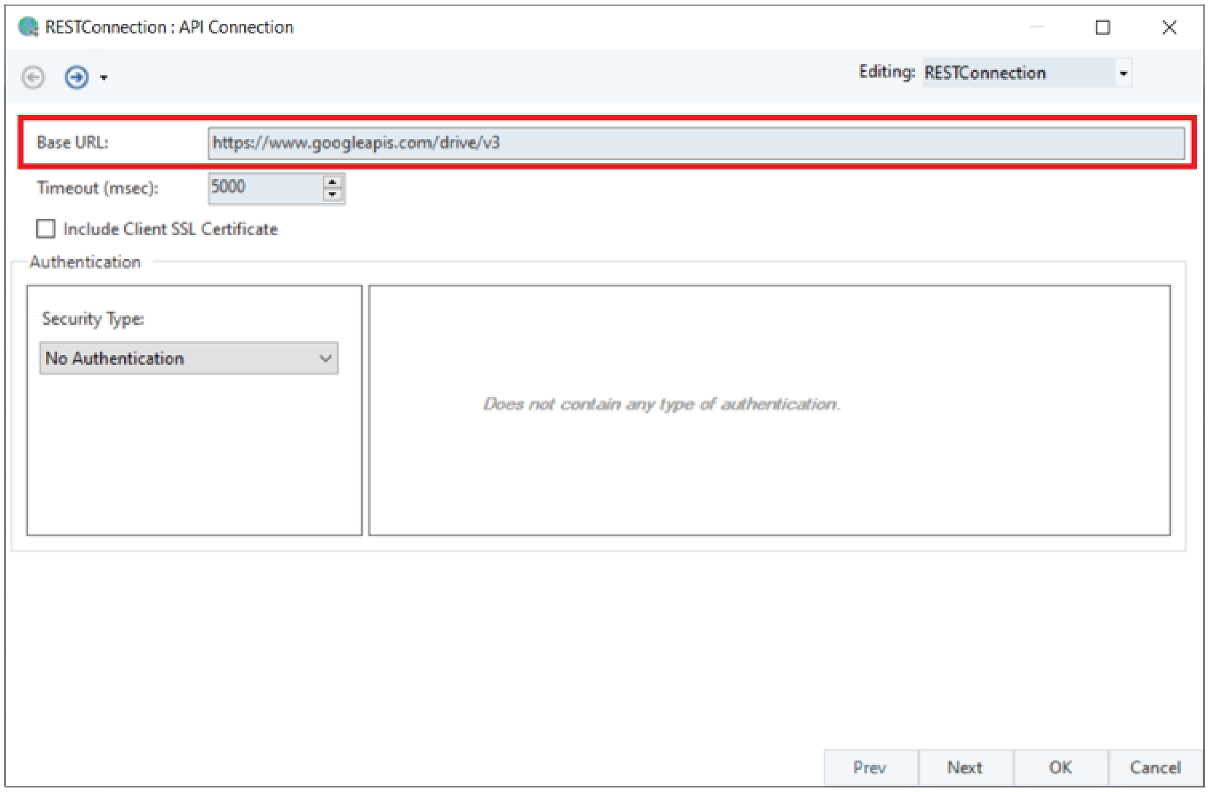
It is important to note that a project must be created before importing APIs to work with the *API Browser*. However, you can access the API without a project when it's an *API Connection* contained in the flow.

The *API Browser*, along with all its features and functionalities, works only within the scope of a project. Otherwise, it will give you the following error,


Centerprise Data Integrator 8

To execute this command, please make sure a project is open.

When a user imports an API, a shared connection file is created within the project automatically. The shared action file contains the Base URL of the imported API.


RESTConnection : API Connection

Editing: RESTConnection

Base URL: https://www.googleapis.com/drive/v3

Timeout (msec): 5000

☐ Include Client SSL Certificate

Authentication

Security Type: No Authentication

Does not contain any type of authentication.

HTTP Request Methods

Astera Centerprise supports the following HTTP request methods:

1. PUT: To update data to a specified resource to be processed on an API.
2. GET: To retrieve data from a specified resource on an API.
3. POST: To create or update an existing record on an API.
4. DELETE: To delete a specified resource on an API.
5. PATCH: To apply partial modifications to an existing resource.

11.3.3 Creating a Project for API Browser

To work with the *API Browser* in Astera API Management, you must first create an *API Client* Project.

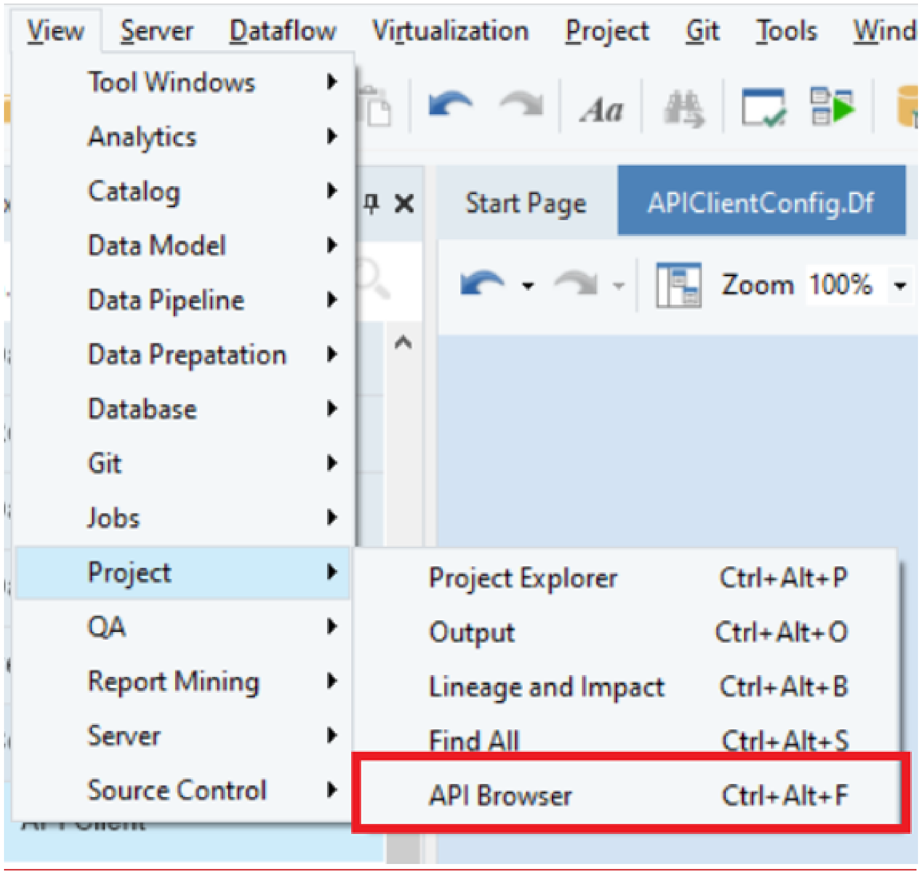
Follow the steps below to create an *API Client* Project in Astera API Management,

1. Go to Menu Bar > Project > New > API Client Project.

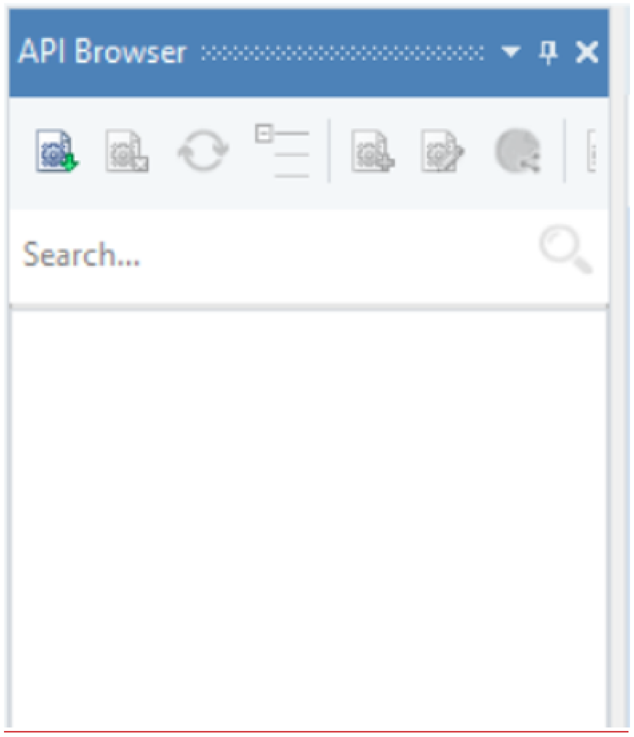
Provide a name to the *API Client* Project and point the path to the location and directory where you want to save it.

Note: It is best practice to always create a new project in a new folder to avoid any errors.

2. Now, open the *API Browser* panel on your API Management client from *Menu Bar > View > API Browser.*



3. Once selected, an API Browser panel will open on the left side of your API Management client window.



Here, you can see three icons in the toolbar of the *API Browser*,

Import API: By clicking this option, you can import different APIs with various available options.

Remove API from Browser: This option removes the selected API from the API Browser.

Refresh API Tree: This option allows you to redraw the browser tree after you have deleted some operations.

Expand/Collapse all: These options show/hide all the requests in the CAPI file.

Add Request: This option allows you to add a new HTTP request to the CAPI file by specifying the request name, resource, and HTTP method.

Edit Properties: You can use this option to change the shared connection or the API name of the CAPI.

Open API Connection: This option allows you to directly open the shared API Connection from the project for the API opened in the API Browser.

Save CAPI file: Any changes made to the CAPI file are saved when you click on this option.

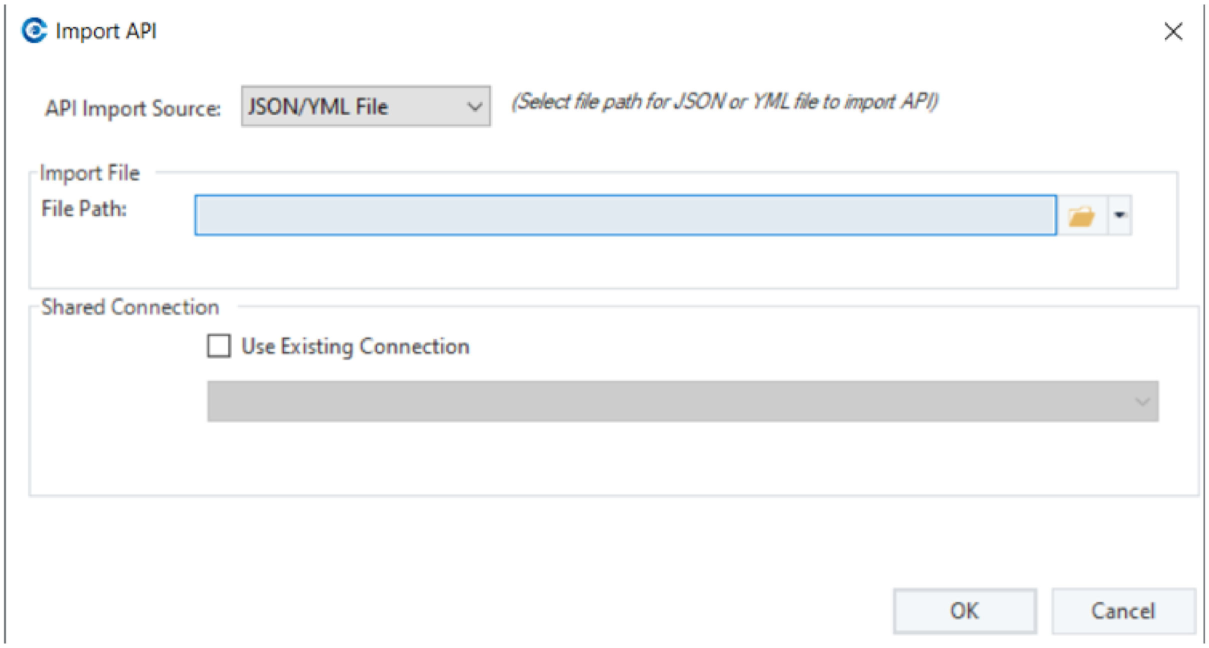
How to Import APIs in API Management

To import an API in Astera API Management, click the *Import API* icon. An *Import API* screen will open.

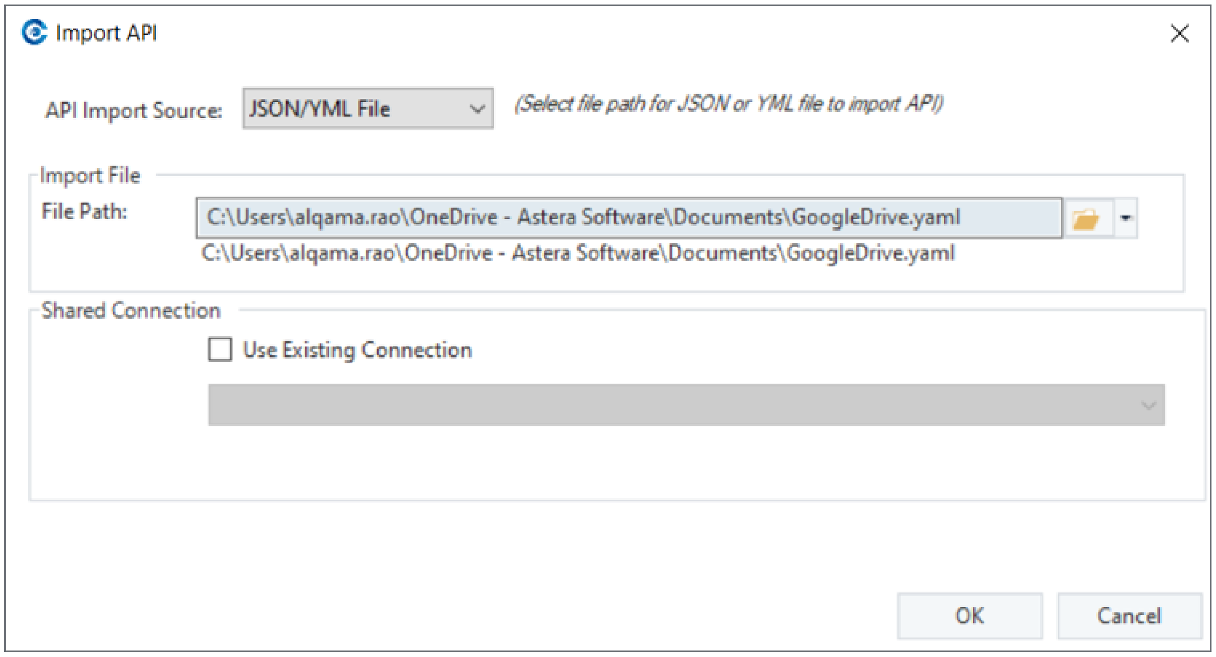
Here, first, you need to select the *API Import Source* type from the drop-down menu. Astera API Management offers three ways to import APIs.

Type 1 – JSON/YML File

JSON/YML File – For this type of API source you only need to provide the Open API Specification *File Path* in JSON or YML file formats.


Import API
API Import Source: JSON/YML File
(Select file path for JSON or YML file to import API)
Import File
File Path:
Shared Connection
Use Existing Connection
OK
Cancel

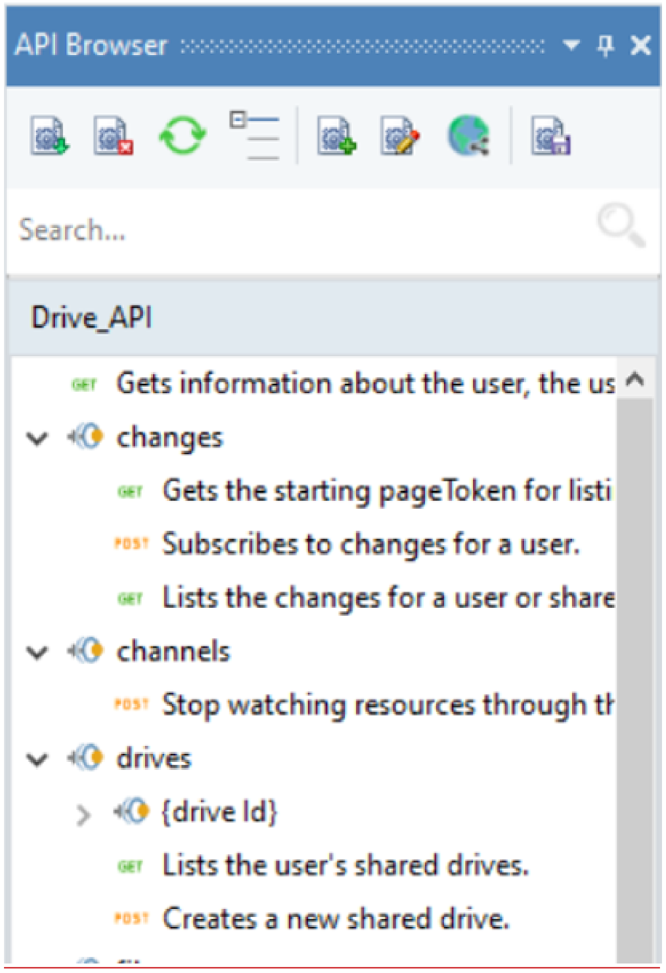
2. Specify the *File Path* and click *OK*.


Import API
API Import Source: JSON/YML File
(Select file path for JSON or YML file to import API)
Import File
Shared Connection
Use Existing Connection
OK
Cancel

File Path: C:\Users\alqama.rao\OneDrive - Astera Software\Documents\GoogleDrive.yaml

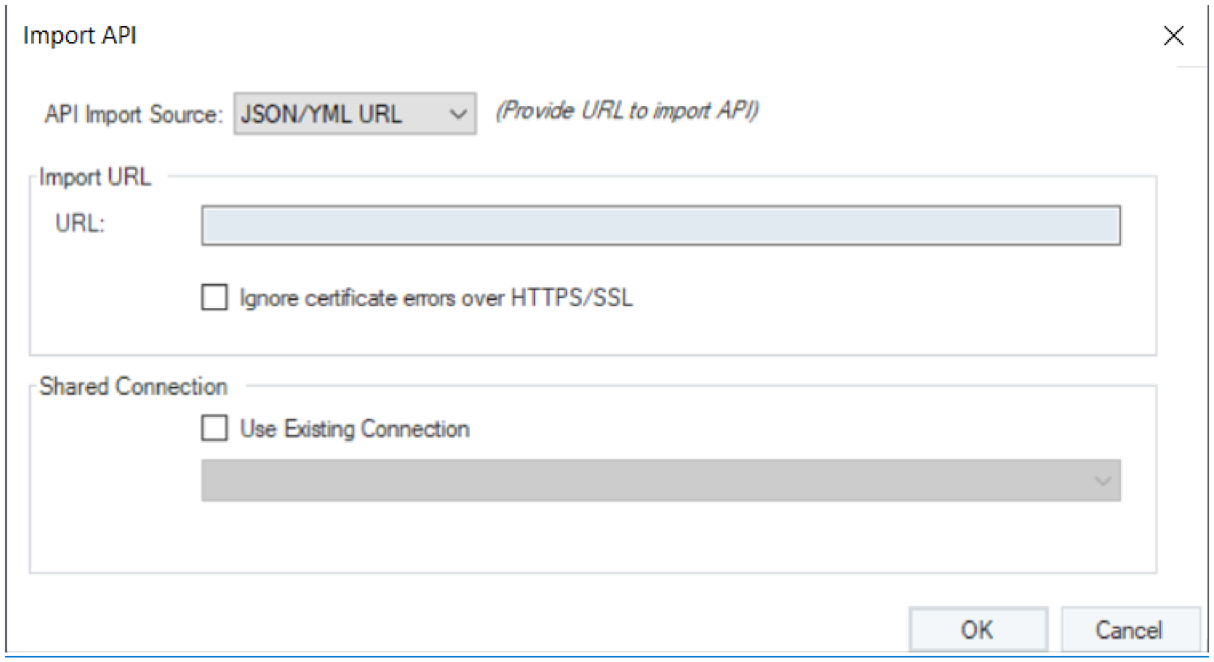
C:\Users\alqama.rao\OneDrive - Astera Software\Documents\GoogleDrive.yaml

This API will be populated in the API Browser panel from where you can simply expand the nodes and drag-and-drop methods onto your designer window.

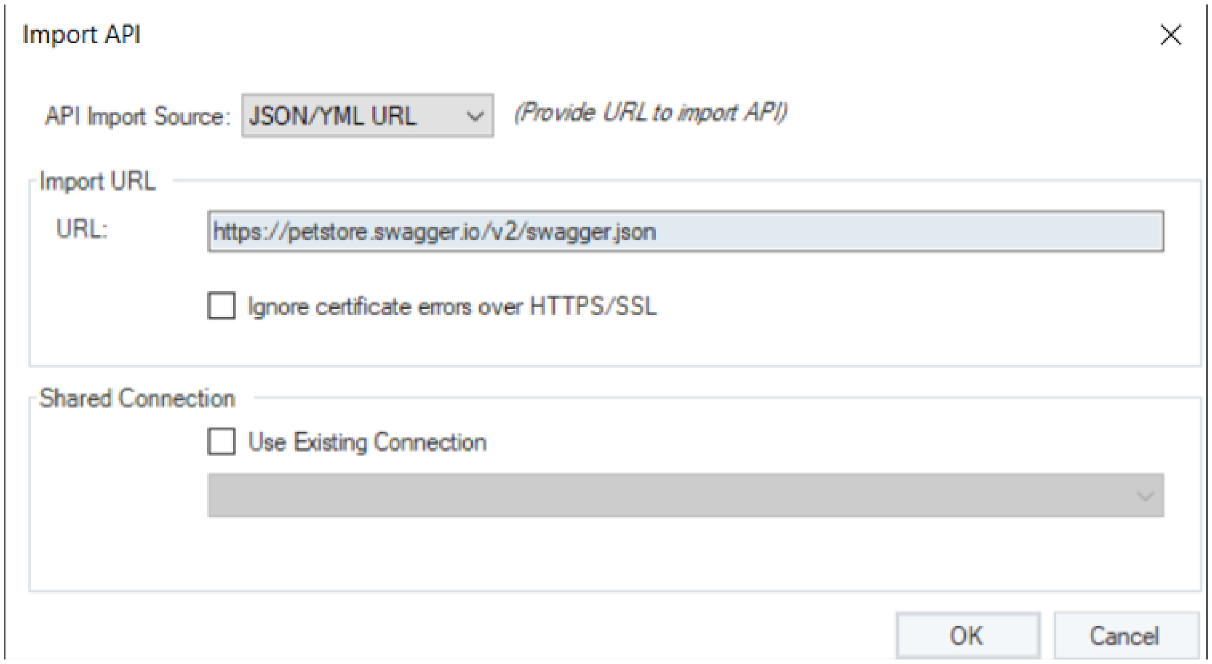

API Browser
Search...
Drive_API
GET Gets information about the user, the us
changes
GET Gets the starting pageToken for listi
POST Subscribes to changes for a user.
GET Lists the changes for a user or share
channels
POST Stop watching resources through th
drives
{drive Id}
GET Lists the user's shared drives.
POST Creates a new shared drive.

Type 2 – JSON/YML URL

JSON/YML URL – For this type of API source, you will need to provide the URL in JSON or YML format.

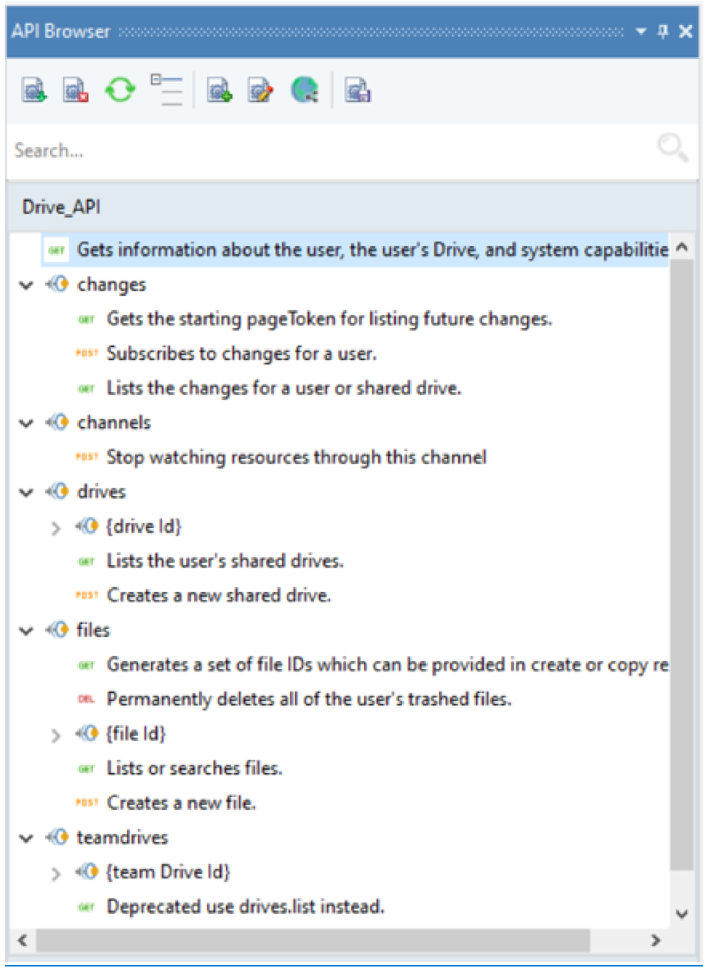

Import API
API Import Source: JSON/YML URL
(Provide URL to import API)
Import URL
URL:
Ignore certificate errors over HTTPS/SSL
Shared Connection
Use Existing Connection
OK
Cancel

2. Specify the *URL* and click on *OK*.


Import API
API Import Source: JSON/YML URL
(Provide URL to import API)
Import URL
Ignore certificate errors over HTTPS/SSL
Shared Connection
Use Existing Connection
OK
Cancel

URL: https://petstore.swagger.io/v2/swagger.json

This API will be populated in the API Browser panel.



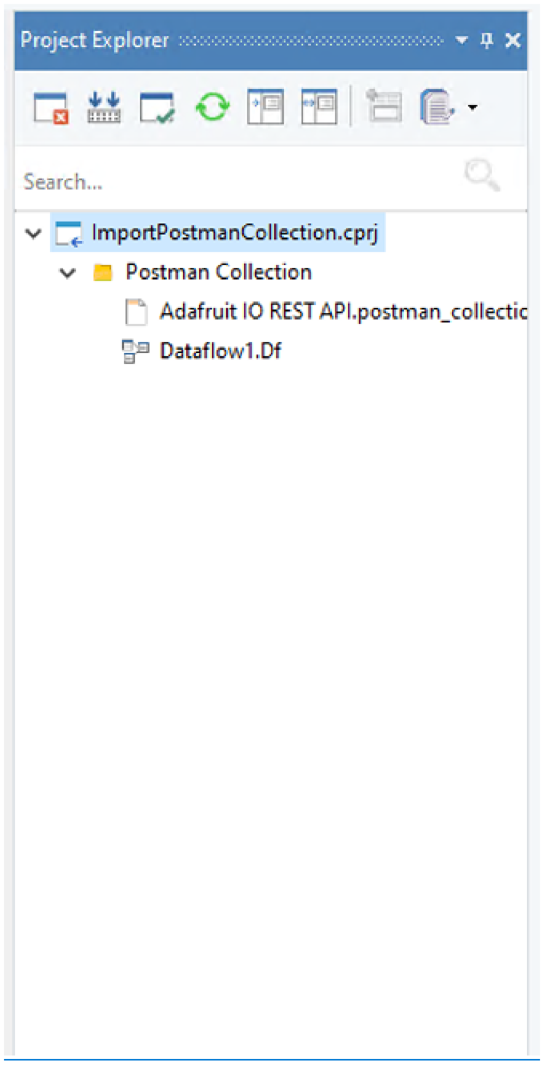
Drive_API

GET Gets information about the user, the user's Drive, and system capabilitie
changes
GET Gets the starting pageToken for listing future changes.
POST Subscribes to changes for a user.
GET Lists the changes for a user or shared drive.
channels
POST Stop watching resources through this channel
drives
{drive Id}
GET Lists the user's shared drives.
POST Creates a new shared drive.
files
GET Generates a set of file IDs which can be provided in create or copy re
DEL Permanently deletes all of the user's trashed files.
{file Id}
GET Lists or searches files.
POST Creates a new file.
teamdrives
{team Drive Id}
GET Deprecated use drives.list instead.

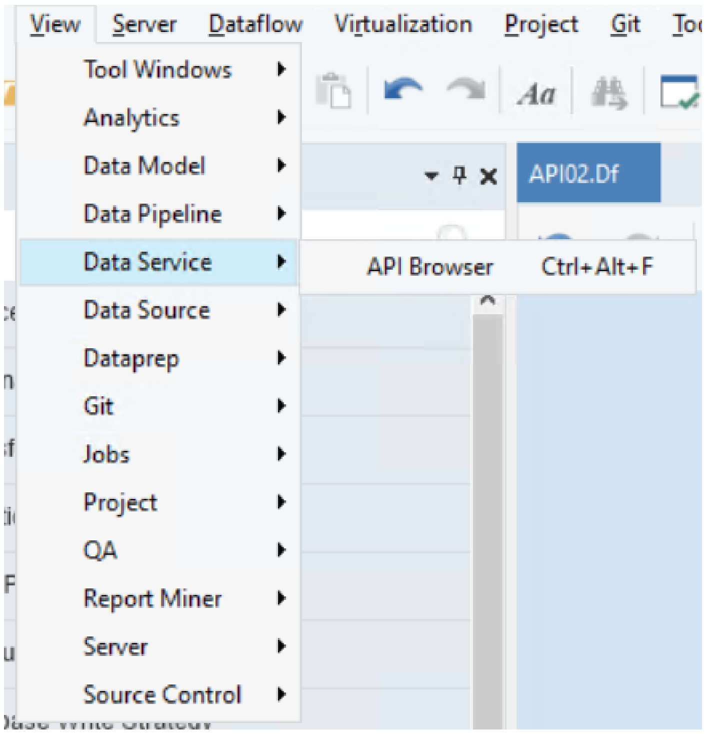
Type 3 – Import Postman API Collections

Let's see what steps are required to import a Postman Collection to the *API Browser*.

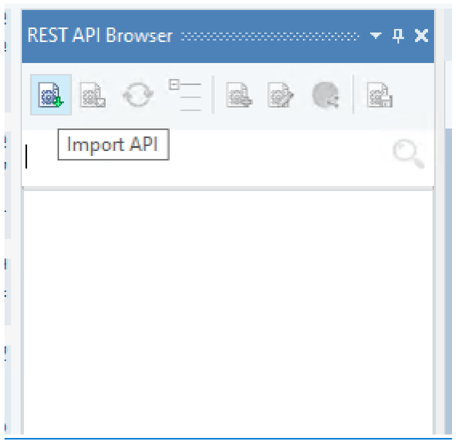
Open an Integration Project.


Project Explorer
Search...
ImportPostmanCollection.cprj
Postman Collection
Adafruit IO REST API.postman_collectic
Dataflow1.Df

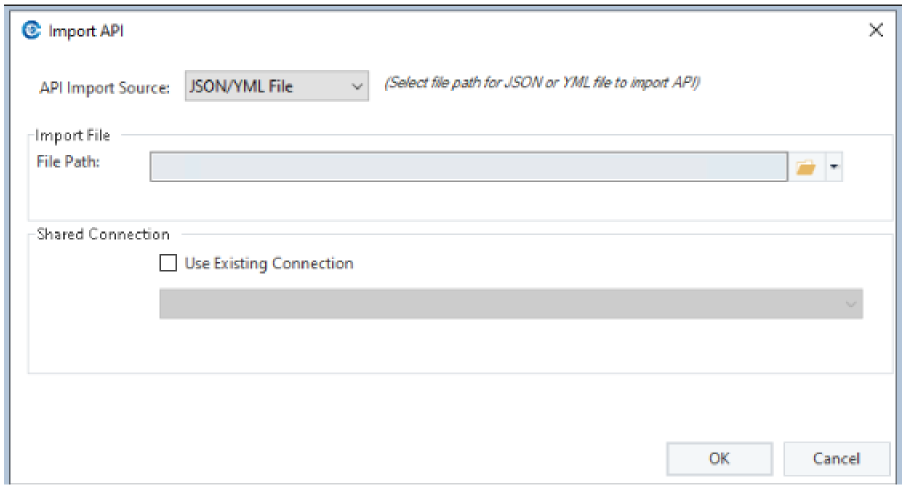
Open the API Browser through *View* > *Data Service* > *API Browser*.



Click on the *Import API* option on the *API Browser*.



This will open the Import API window.



Import API

API Import Source: JSON/YML File (Select file path for JSON or YML file to import API)

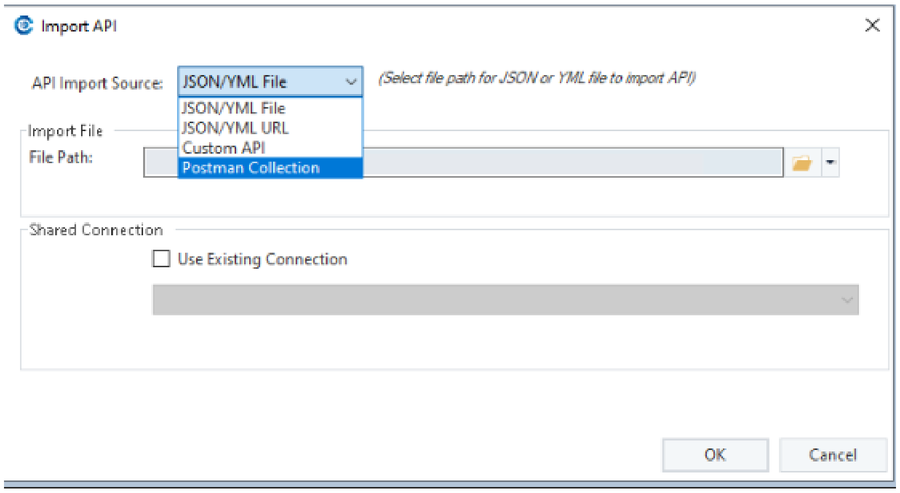
Import File

File Path:

Shared Connection

☐ Use Existing Connection

Select Postman collection from the drop-down of the API Import Source.



Import API

API Import Source: JSON/YML File (Select file path for JSON or YML file to import API)

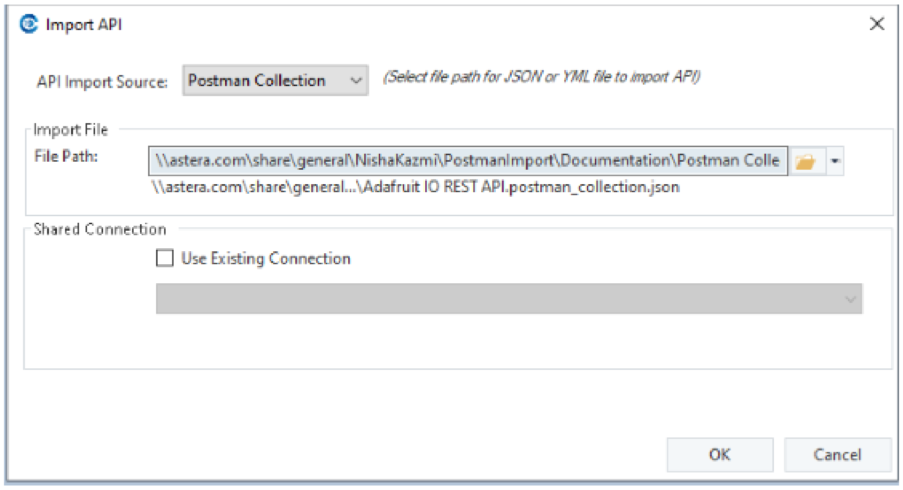
Import File

File Path:

Shared Connection

☐ Use Existing Connection

Browse and provide the path to the Postman Collection and click OK.



Import API

API Import Source: Postman Collection (Select file path for JSON or YML file to import API)

Import File

File Path: \\astera.com\share\general\NishaKazmi\PostmanImport\Documentation\Postman Colle

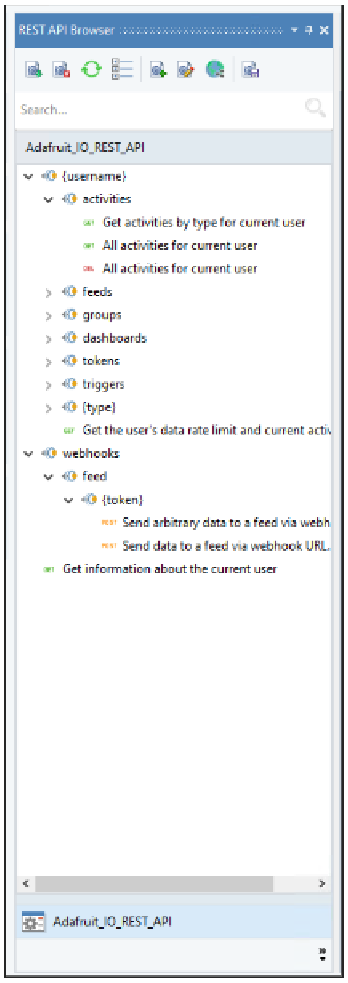
\\astera.com\share\general...\Adafruit IO REST API.postman_collection.json

Shared Connection

☐ Use Existing Connection

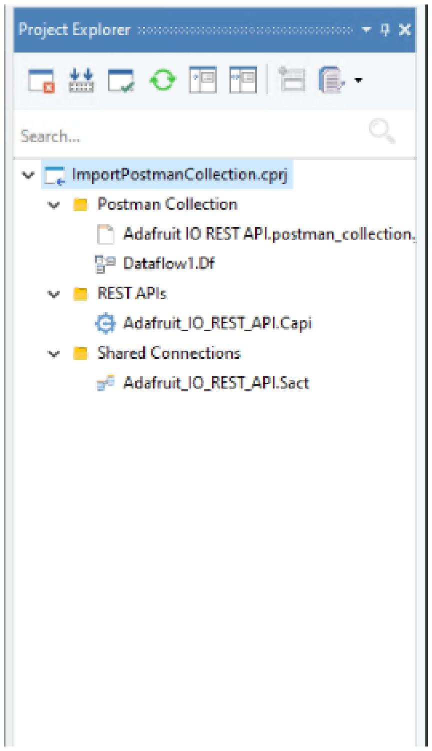
If there is already a Shared Connection available, then we can re-utilize it, instead of auto-generating a new one, by clicking on the Use Existing Connection check box.

Once the Postman Collection is successfully imported, it will populate the API Browser with the available endpoints.

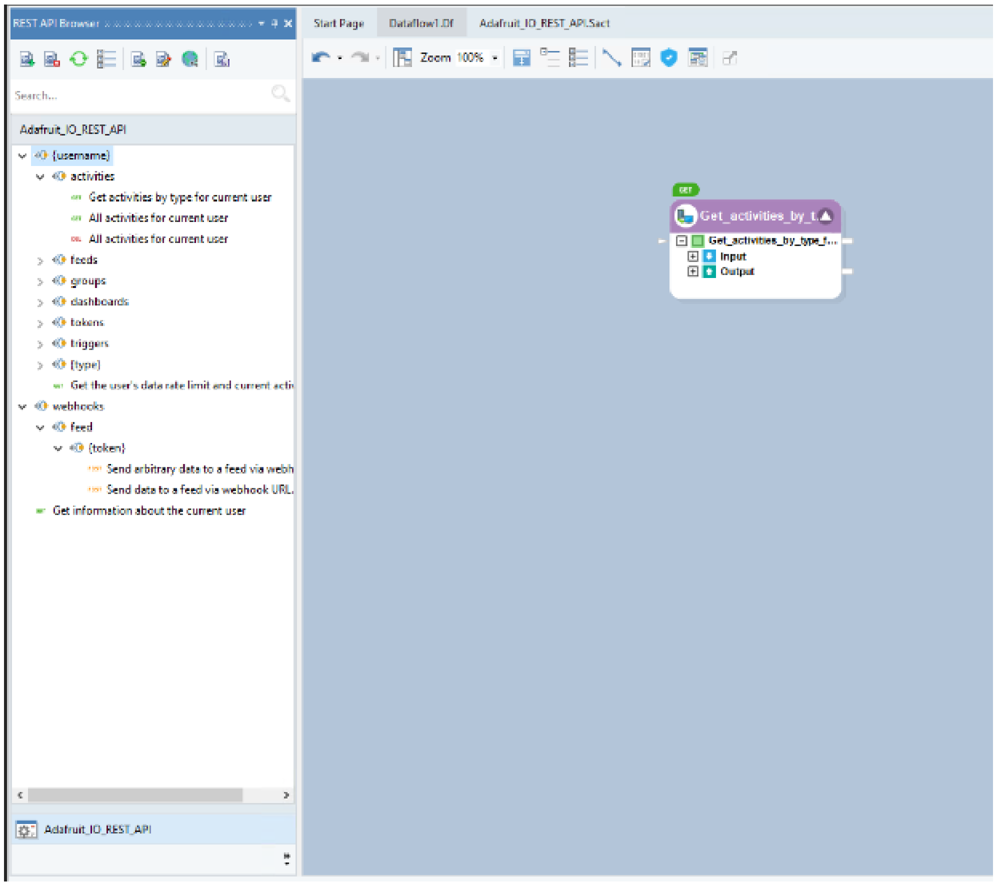

REST API Browser
Search...
Adafruit_IO_REST_API
{username}
activities
Get activities by type for current user
All activities for current user
All activities for current user
feeds
groups
dashboards
tokens
triggers
{type}
Get the user's data rate limit and current activ
webhooks
feed
{token}
Send arbitrary data to a feed via webh
Send data to a feed via webhook URL.
Get information about the current user
Adafruit_IO_REST_API

Note: It is recommended by Postman to export the collections in v2.1 format files. Therefore, Centerprise restricts the user to import only a v2.1 Postman Collection.

The Centerprise API file (.capi) and Shared Connection files will automatically generate and be saved in their respective folders.


Project Explorer
Search...
ImportPostmanCollection.cprj
Postman Collection
Adafruit IO REST API.postman_collection.
Dataflow1.Df
REST APIs
Adafruit_IO_REST_API.Capi
Shared Connections
Adafruit_IO_REST_API.Sact

Now, drag and drop any endpoint onto a logic designing artefact i.e., a dataflow to consume.

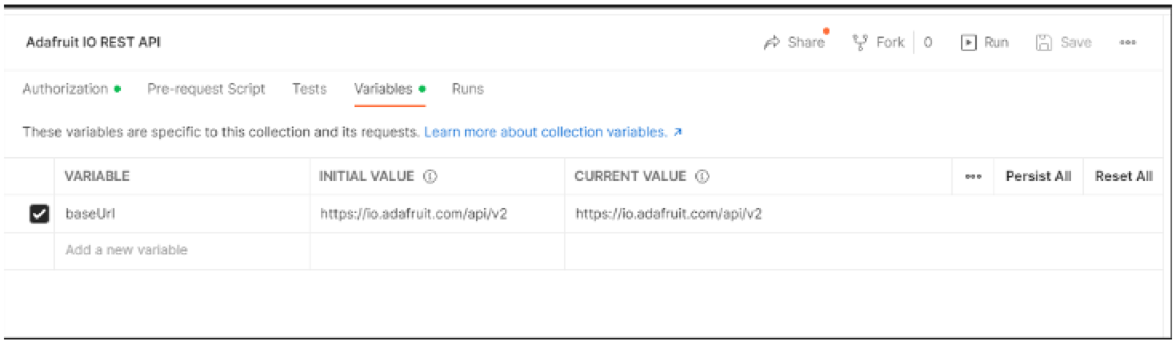

REST API Browser
Search...
Adafruit_IO_REST_API
{username}
activities
Get activities by type for current user
All activities for current user
All activities for current user
feeds
groups
dashboards
tokens
triggers
{type}
Get the user's data rate limit and current acti
webhooks
feed
{token}
Send arbitrary data to a feed via webh
Send data to a feed via webhook URL.
Get information about the current user
Adafruit_IO_REST_API
Start Page
Dataflow1.Df
Adafruit_IO_REST_API.Sact
Zoom 100%
GET
Get_activities_by_t
Get_activities_by_type_f...
Input
Output

11.3.4 Postman Collection Format

Variables Convention

To import a Postman Collection to the API Browser successfully, we must follow certain conventions:

The Postman collection must include a variable namely baseUrl. (This variable is case insensitive)



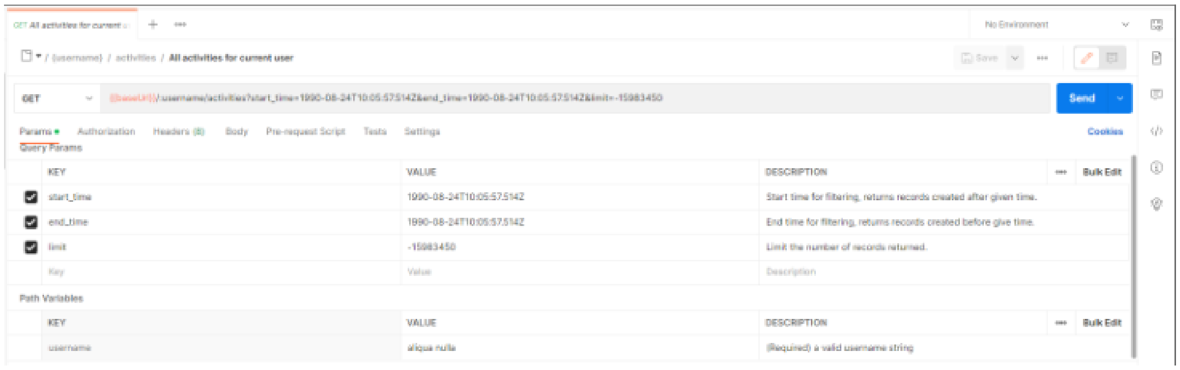
These variables are specific to this collection and its requests. Learn more about collection variables. ↗

	VARIABLE	INITIAL VALUE	CURRENT VALUE
✓	baseUrl	https://io.adafruit.com/api/v2	https://io.adafruit.com/api/v2
	Add a new variable		

Note: A collection in which the baseUrl variable contains a special character(s) will not be imported.

All other variables, except for the baseUrl, will be discarded.

During the import, the baseUrl variable defined in all the endpoints will be replaced with the Base Url text box value in the Shared Connection.



GET {{baseUrl}}/:username/activities?start_time=1990-08-24T10:05:57.514Z&end_time=1990-08-24T10:05:57.514Z&limit=-15983450

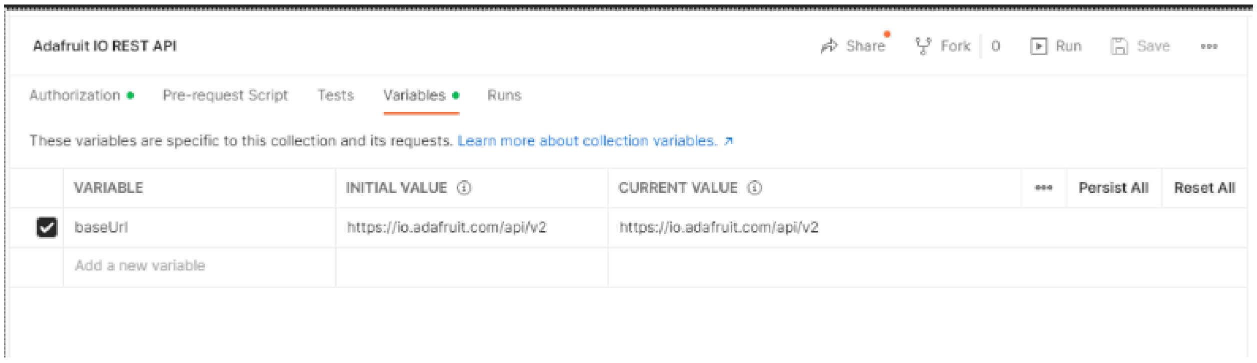
Query Params

	KEY	VALUE	DESCRIPTION
✓	start_time	1990-08-24T10:05:57.514Z	Start time for filtering, returns records created after given time.
✓	end_time	1990-08-24T10:05:57.514Z	End time for filtering, returns records created before give time.
✓	limit	-15983450	Limit the number of records returned.
	Key	Value	Description

Path Variables

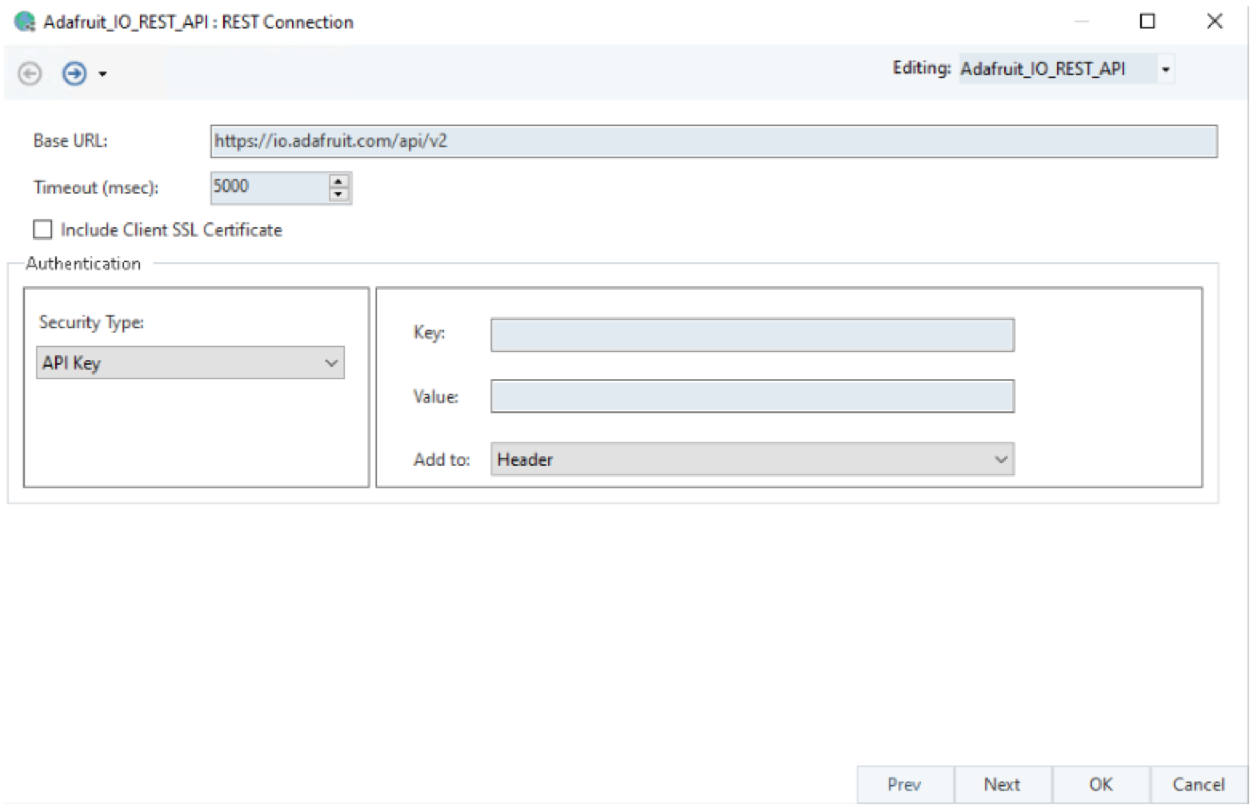
KEY	VALUE	DESCRIPTION
username	aliqua nulla	(Required) a valid username string

This means that the Shared Connection's Base Url will be populated with the baseUrl variable's Current Value that is defined under the Variable section in the collection.



These variables are specific to this collection and its requests. Learn more about collection variables. ↗

	VARIABLE	INITIAL VALUE	CURRENT VALUE
✓	baseUrl	https://io.adafruit.com/api/v2	https://io.adafruit.com/api/v2
	Add a new variable		


Adafruit_IO_REST_API : REST Connection

Editing: Adafruit_IO_REST_API

Base URL: https://io.adafruit.com/api/v2

Timeout (msec): 5000

☐ Include Client SSL Certificate

Authentication

Security Type: API Key

Key:

Value:

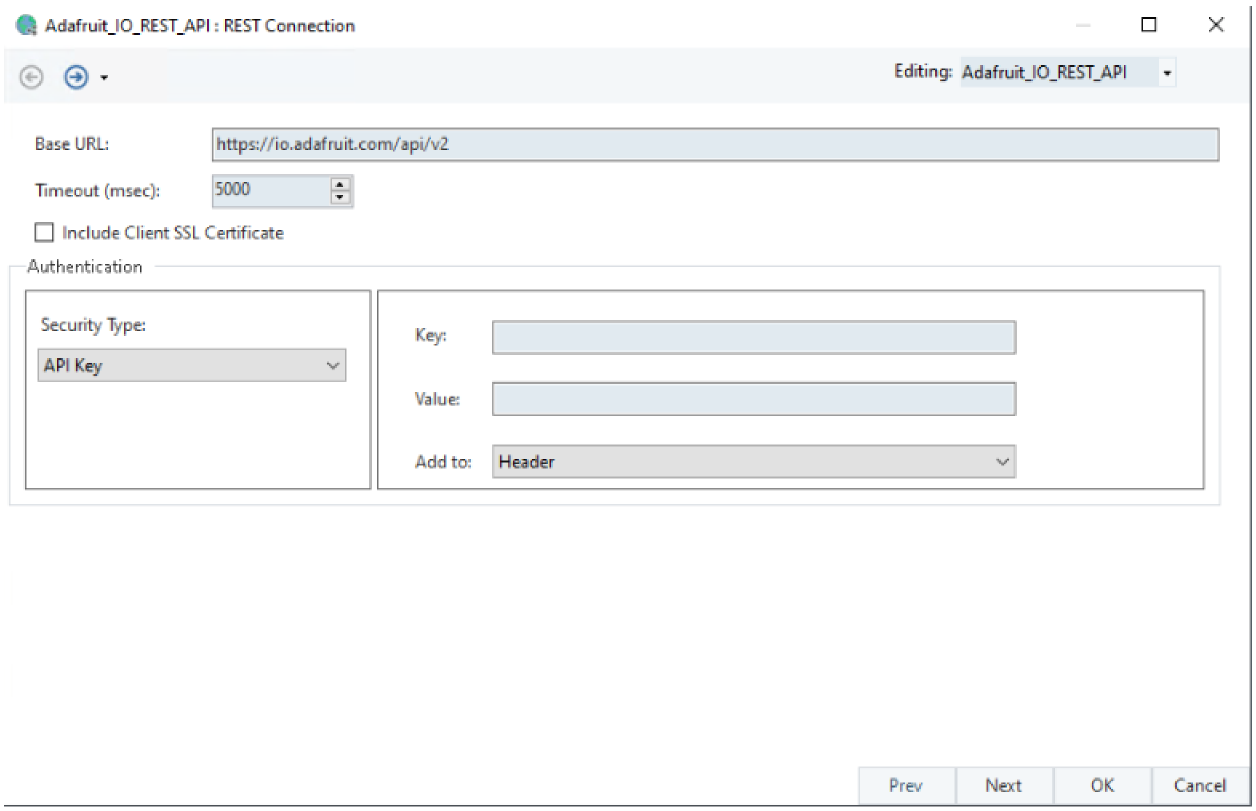
Add to: Header

Preservation of Authentication Information

All valid Postman Collections will be imported with pre-configured Shared Connections. These Shared Connections will have the same Authentication Type selected as in the collections i.e., API Key, Auth Code, Client Credentials, etc.

Note: Confidential data such as credentials are imported for security and protection.

Example of an API Key Security Type



Adafruit_IO_REST_API : REST Connection

Editing: Adafruit_IO_REST_API

Base URL: https://io.adafruit.com/api/v2

Timeout (msec): 5000

☐ Include Client SSL Certificate

Authentication

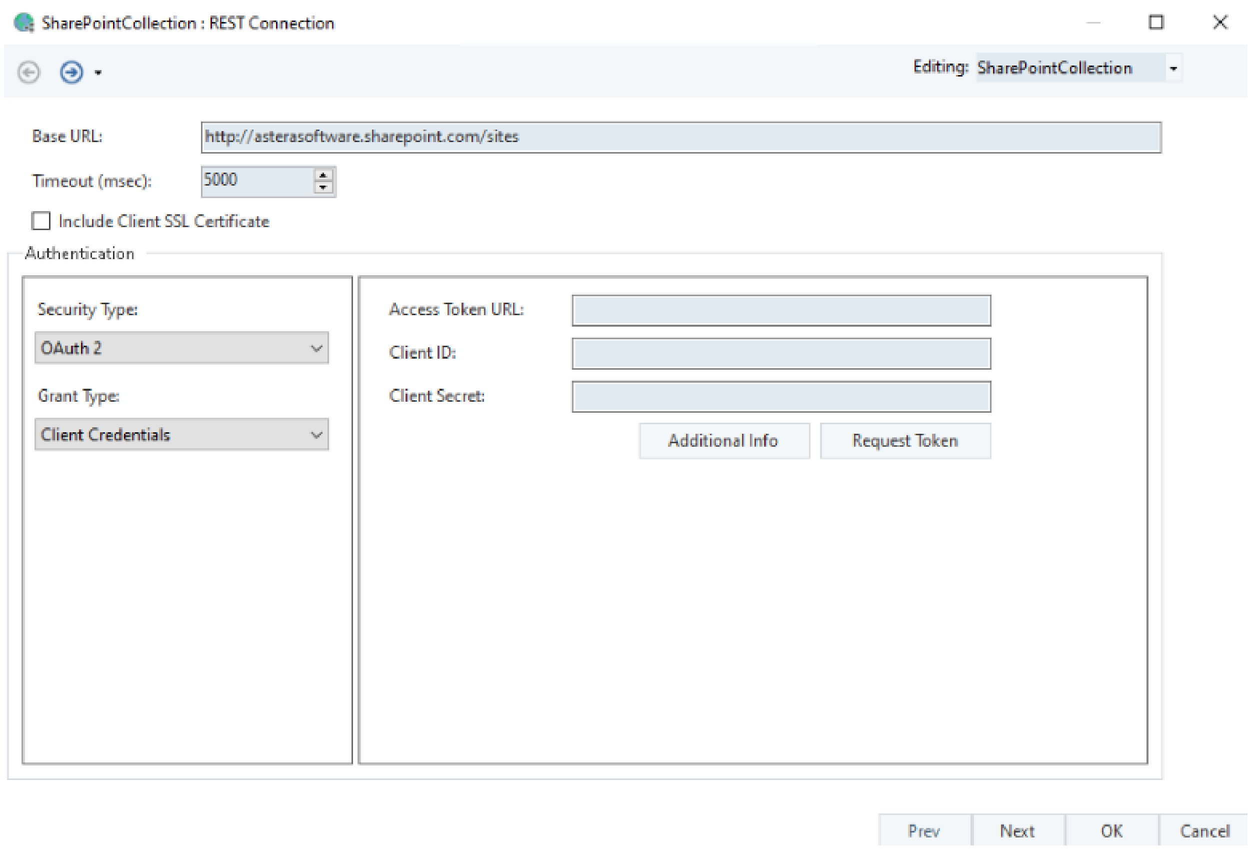
Security Type: API Key

Key:

Value:

Add to: Header

Example of an OAuth 2.0 Security Type



SharePointCollection : REST Connection

Editing: SharePointCollection

Base URL: http://asterasoftware.sharepoint.com/sites

Timeout (msec): 5000

☐ Include Client SSL Certificate

Authentication

Security Type: OAuth 2

Grant Type: Client Credentials

Access Token URL:

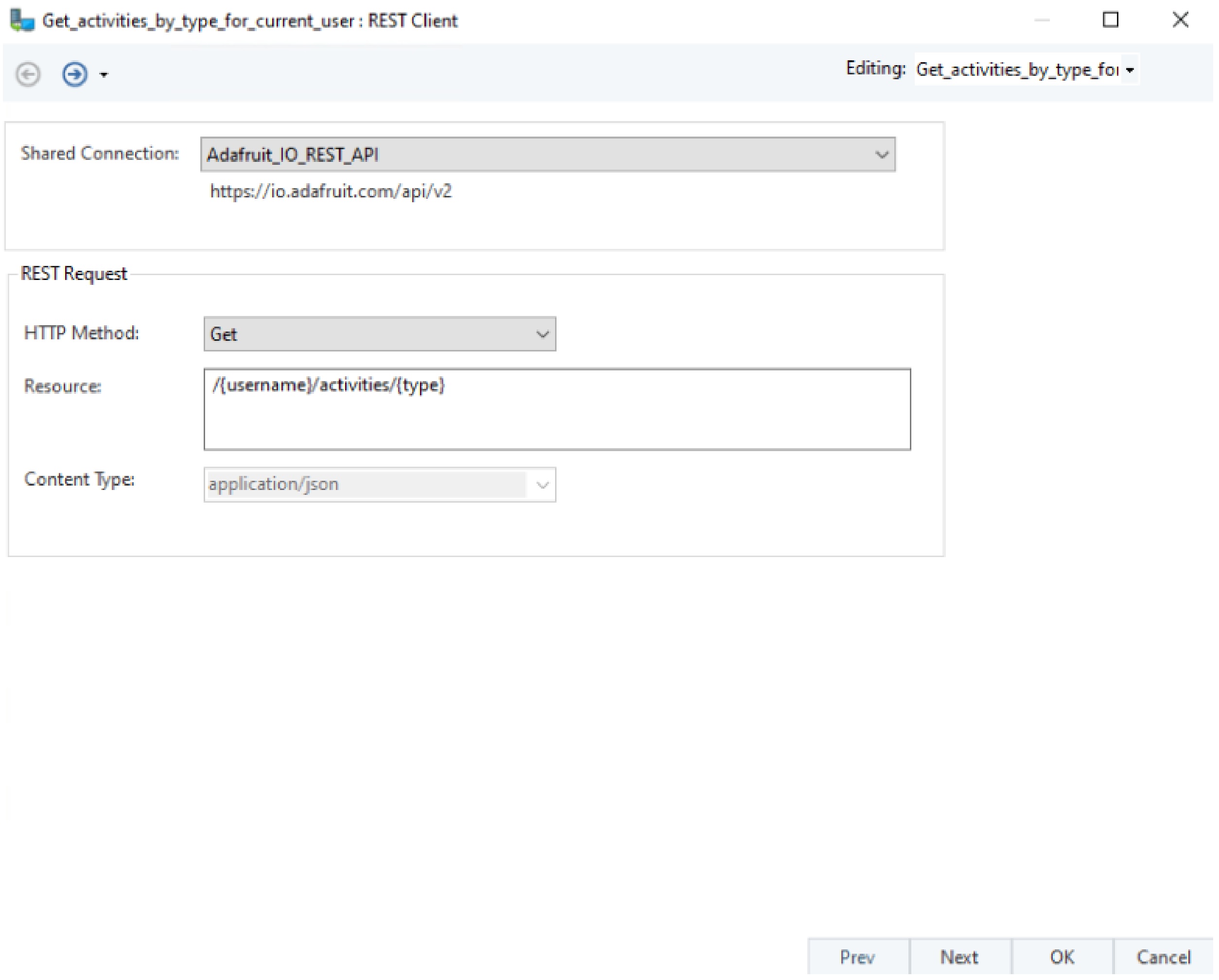
Client ID:

Client Secret:

Preservation of Endpoint's Configuration

On importing a Postman Collection, each endpoint's configuration i.e., methods, resources, parameters, and request/response payloads will also be preserved.

HTTP Method and Resource


Get_activities_by_type_for_current_user : REST Client
Editing: Get_activities_by_type_foı
Prev
Next
OK
Cancel

Shared Connection: Adafruit_IO_REST_API
https://io.adafruit.com/api/v2

REST Request

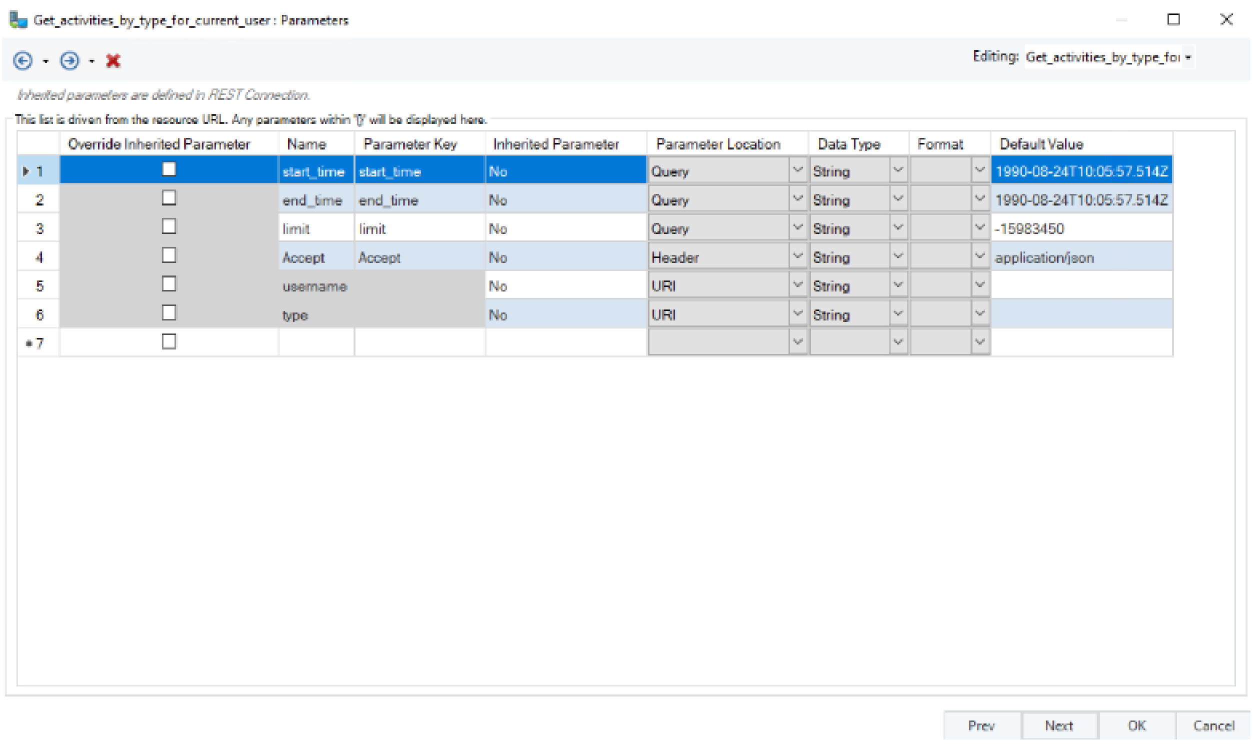
HTTP Method: Get

Resource: /{username}/activities/{type}

Content Type: application/json

Parameter

All parameters with their respective default values are populated in the API Client's Parameter window.



Get_activities_by_type_for_current_user : Parameters

Editing: Get_activities_by_type_foi

Inherited parameters are defined in REST Connection.

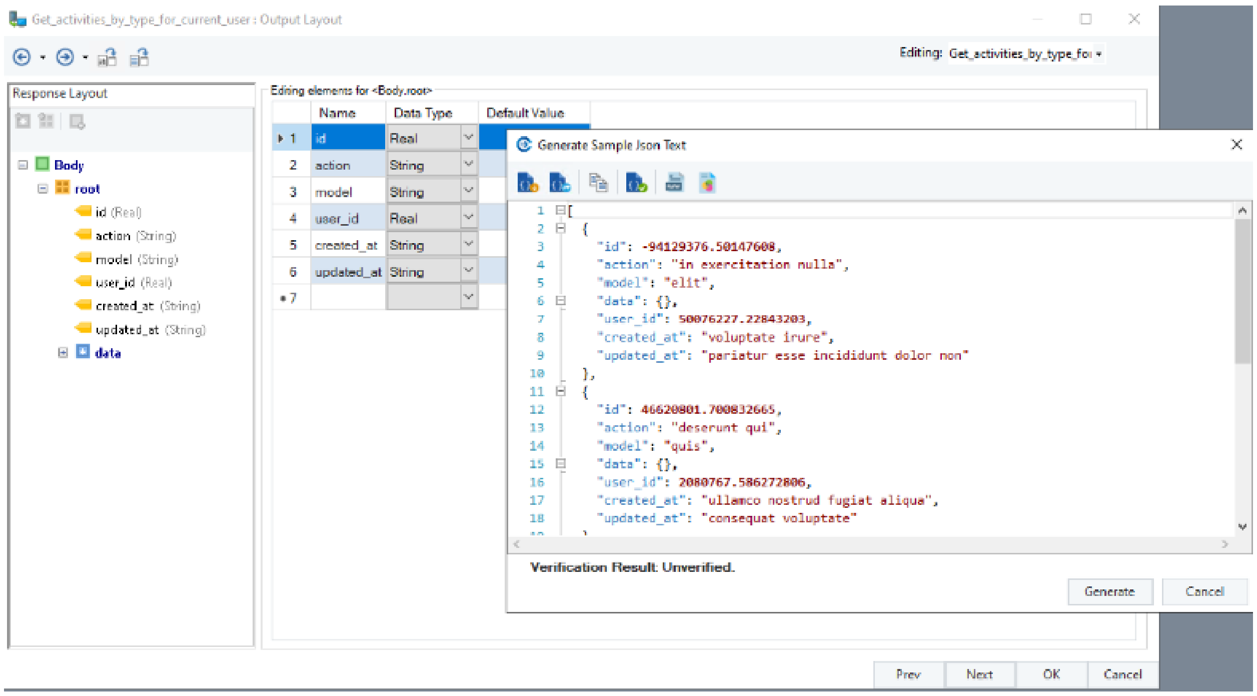
This list is driven from the resource URL. Any parameters within '{}' will be displayed here.

	Override Inherited Parameter	Name	Parameter Key	Inherited Parameter	Parameter Location	Data Type	Format	Default Value
1	☐	start_time	start_time	No	Query	String		1990-08-24T10:05:57.514Z
2	☐	end_time	end_time	No	Query	String		1990-08-24T10:05:57.514Z
3	☐	limit	limit	No	Query	String		-15983450
4	☐	Accept	Accept	No	Header	String		application/json
5	☐	username		No	URI	String		
6	☐	type		No	URI	String		
7	☐							

Note: Sensitive data such as the URI parameter value is not preserved for security.

Payload

The input and output layouts/payload are structured in the respective Input and Output Layout windows. Additionally, the sample text bodies used to generate the layouts are preserved in the Sample JSON Text window.



Get_activities_by_type_for_current_user : Output Layout

Editing: Get_activities_by_type_foi

Response Layout

```
Body
  root
    id (Real)
    action (String)
    model (String)
    user_id (Real)
    created_at (String)
    updated_at (String)
    data
```

Editing elements for <Body.root>

	Name	Data Type	Default Value
1	id	Real	
2	action	String	
3	model	String	
4	user_id	Real	
5	created_at	String	
6	updated_at	String	
7			

Generate Sample Json Text

```
[
  {
    "id": -94129376.50147608,
    "action": "in exercitation nulla",
    "model": "elit",
    "data": {},
    "user_id": 50076227.22843203,
    "created_at": "voluptate irure",
    "updated_at": "pariatur esse incididunt dolor non"
  },
  {
    "id": 46620801.700832665,
    "action": "deserunt qui",
    "model": "quis",
    "data": {},
    "user_id": 2080767.586272806,
    "created_at": "ullamco nostrud fugiat aliqua",
    "updated_at": "consequat voluptate"
```

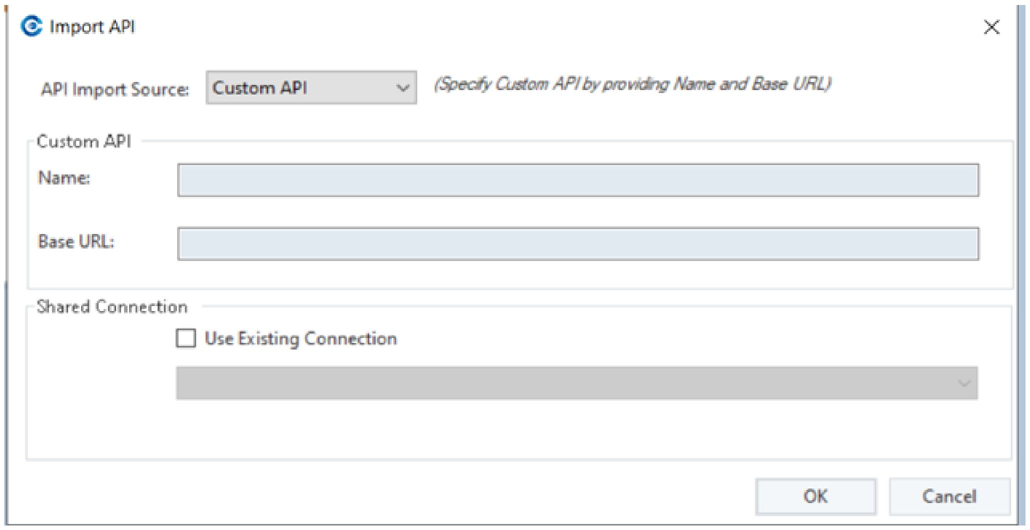
Verification Result: Unverified.

Type 4 - Create or customize API collection:

Users can create and maintain custom API collections in case the API provider does not offer existing documentation for its APIs.

1. From the API Browser, open the import wizard and select Custom API as the API Import Source.

2. Next, provide a name for your custom API and the base URL of the API provider. On import, a new API shared connection (.sact) and a Custom-API (.capi) file will be created in the project.

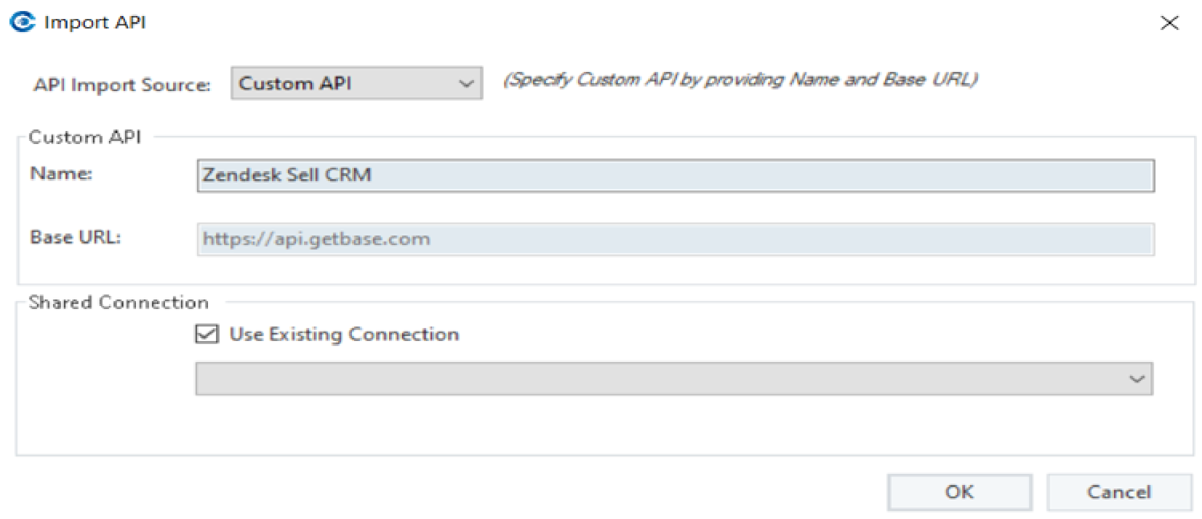


API Import Source: Custom API (Specify Custom API by providing Name and Base URL)

Custom API
Name:
Base URL:

Shared Connection
☐ Use Existing Connection

Alternatively, the custom API can also point to an existing pre-configured connection from the project.

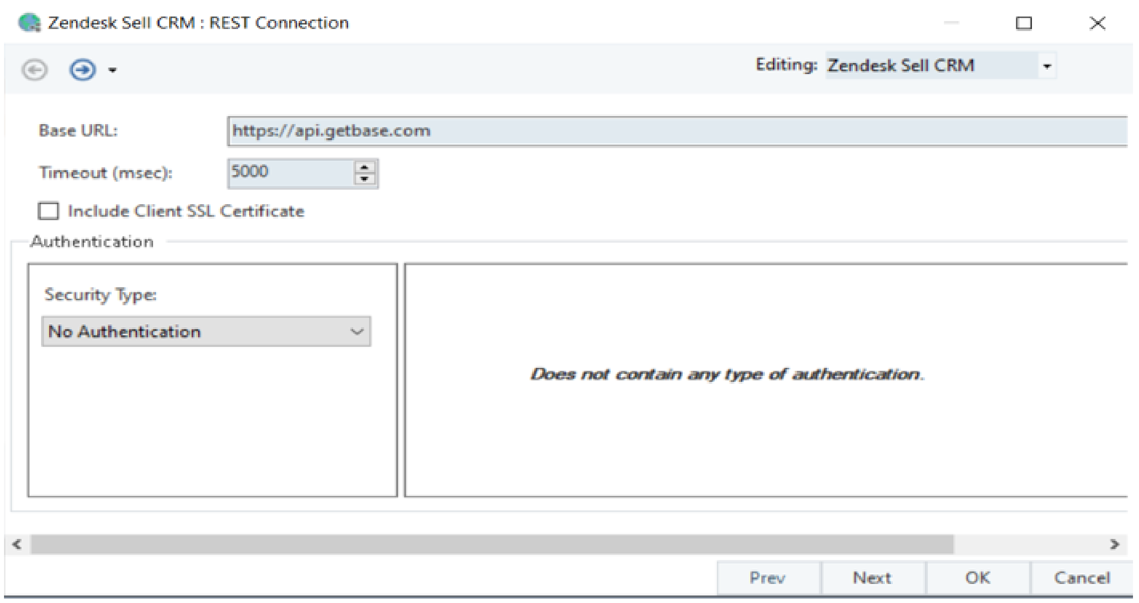


API Import Source: Custom API (Specify Custom API by providing Name and Base URL)

Custom API
Name: Zendesk Sell CRM
Base URL: https://api.getbase.com

Shared Connection
☑ Use Existing Connection

You can configure the API connection object in the shared connection file by providing valid authentication and defining parameters if need be.


Zendesk Sell CRM : REST Connection

Editing: Zendesk Sell CRM

Base URL: https://api.getbase.com

Timeout (msec): 5000

☐ Include Client SSL Certificate

Authentication

Security Type: No Authentication

Does not contain any type of authentication.

Once you are done configuring the connection object, the CAPI file will open in the API browser.

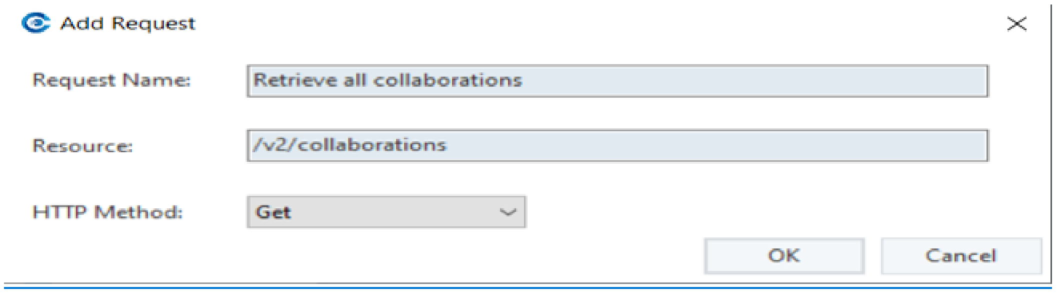
To add API requests to your custom CAPI file, click on the *Add Request* icon from the top toolbar menu of the API browser.

Here, define the request properties,

Request Name: This is used as the request name and description.

Resources: The unique request resource path including the URI or path parameters which appends after the Server Base URL.

HTTP Method: Select the standard HTTP method to be used for this request.


Add Request

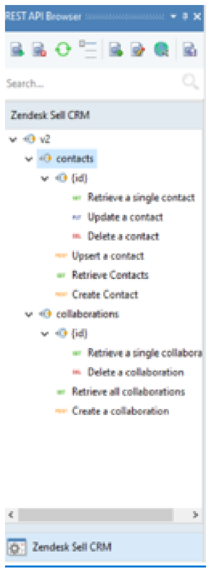
Request Name: Retrieve all collaborations

Resource: /v2/collaborations

HTTP Method: Get

The request will be added to the CAPI file in the API Browser. Repeat this process to add all the required requests in your CAPI file.

Once you have populated the requests in your CAPI file, it may look something like this in the API browser.


REST API Browser

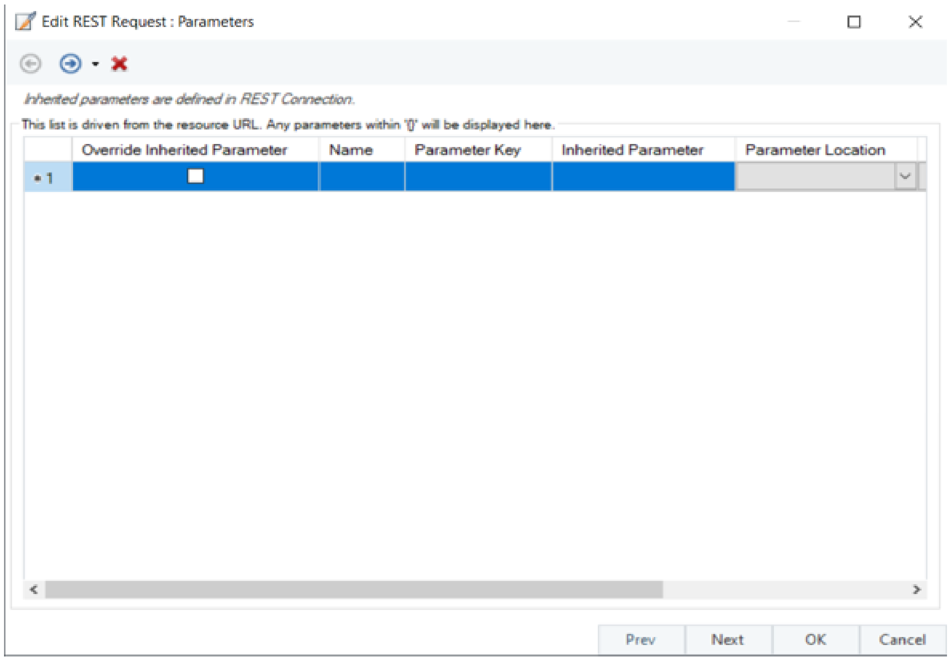
Search...

Zendesk Sell CRM
- v2
  - contacts
    - {id}
      - Retrieve a single contact
      - Update a contact
      - Delete a contact
    - Upsert a contact
    - Retrieve Contacts
    - Create Contact
  - collaborations
    - {id}
      - Retrieve a single collabora
      - Delete a collaboration
    - Retrieve all collaborations
    - Create a collaboration

Zendesk Sell CRM

Note: You may have to include a URI parameter in the resource for some requests. Some API documentations display the URI parameter after a (:) symbol. However, you will have to replace the colon (:)with curly brackets ({ }) for the parameter to be considered as URI.

To configure the parameters, input/output layout, or pagination options for any request, right-click on it and choose the *Edit Request* option.


Edit REST Request : Parameters

Inherited parameters are defined in REST Connection.

This list is driven from the resource URL. Any parameters within '{}' will be displayed here.

	Override Inherited Parameter	Name	Parameter Key	Inherited Parameter	Parameter Location
1	☐				

You can also configure and save the request properties by dragging and dropping.

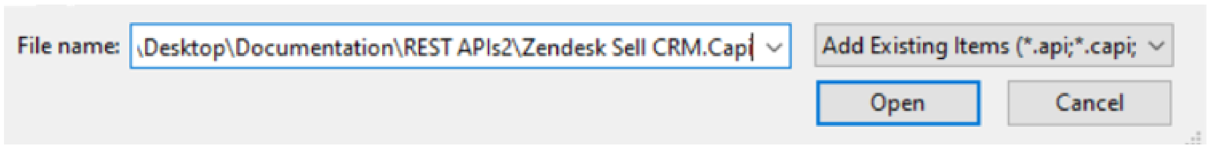
- Drag the request from the API browser to a flow designer.
- Right-click on the API Client and select Properties. Make changes to the properties of the API client object.
- To save the changes, just drag and drop the client object back to the API Browser from the flow designer.

Once you are done populating your Capi file by configuring all request properties and authentication, click on the Save Capi file icon on the top of the API browser to save your changes.

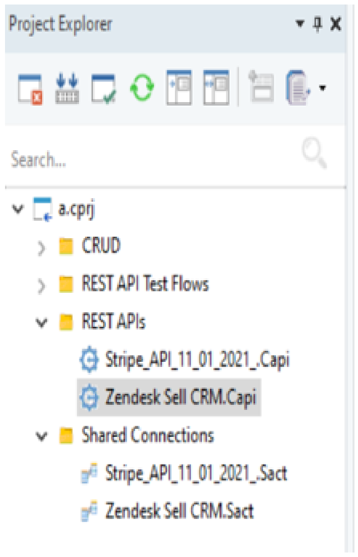
This will save all the configurations you have made including parameters, input/output body, and pagination settings to the request.

Sharing and adding the Capi file to a new project

- Fully configured CAPI files act as a connector for your API provider. If you want to add the Capi file to another project, right-click on the CAPI file from the project explorer and click on *copy full path*.
- Then open the other project, right-click on the folder you want to add the CAPI file to and click on *Add Existing Items*.
- A box will open. Paste the file path in the box next to *File name* and click on Open.


File name: \Desktop\Documentation\REST APIs2\Zendesk Sell CRM.Capi

The CAPI file will be added to the project along with its corresponding Sact file.



This concludes the basic concepts of working with the *API Browser* in Astera API Management.

11.4 Request Service Options - eTags

11.4.1 What is eTag?

An eTag also called an entity tag is an HTTP response header field that includes an identifier for the specific version or the state of the resource at the time the request was sent. This identifier helps to differentiate between the different versions of the resource and to check if the caches at the client side hold the updated representation of the state of the resource.

11.4.2 How do they work?

Let's try to understand what is meant by eTags and how these options work.

If the client wants to check if the caches of a resource are usable or fresh, it can send the eTag in the If-None-Match header field in the request to the server. The server will match the client's eTag with the one that it has for the current version of the resource. If the ETags match, the server will not send any representation of the state of the resource in the response implying that the client's caches are fresh and usable.

11.4.3 Two eTag Use-Cases

There are two major uses of eTags in API requests:

- Data Caching
- Concurrency Control.

Let's investigate these uses one by one. For now, let's see how this Data Caching works with a use case.

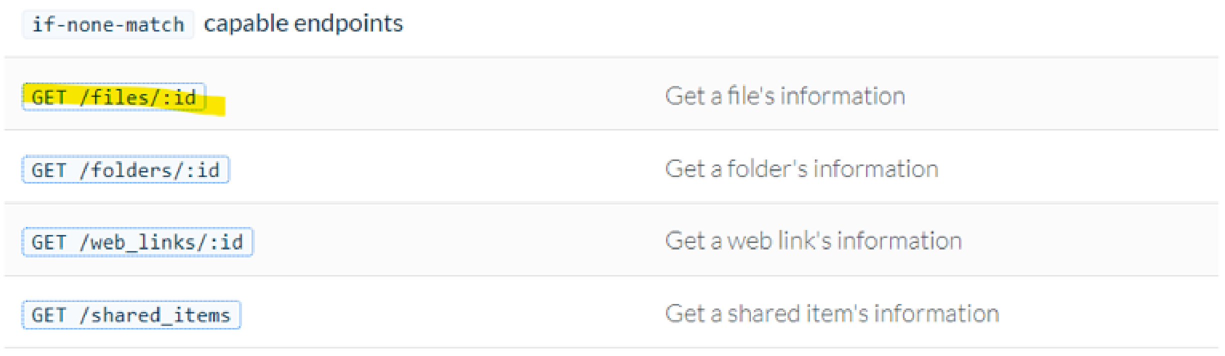
If None Match eTag

So, we will make an API call to one of the endpoint operations of Box APIs. Here, we have a dataflow in which we are making an API call to fetch file details from one of the files on our Box account.

We will send a GET request to the */file/{fileid}* resource with the help of the API client and API connection object. We have configured the API connection object in a shared action file. From the API documentation site, we can see that Box supports OAuth 2 authentication and the grant type of authorization code. Hence, we have already generated our access code after providing the client credentials from our Box app.

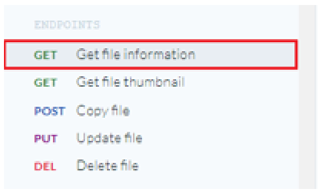
Coming back to our flow, let's open the properties of the API Client object. Here, we are using the shared action API connection object that is providing the base URL. We have specified the HTTP method of GET and provided the name of the resource. Here, the curly brackets specify that the path parameter of file id will be passed along the request to fetch the information on the file related to that file id. In our flow, we are providing the file id through the constant object.

From the API documentation of Box APIs, we can see that the if none match header is supported for the endpoint at which we are making a call in our dataflow.



`if-none-match` capable endpoints

`GET /files/:id`	Get a file's information
`GET /folders/:id`	Get a folder's information
`GET /web_links/:id`	Get a web link's information
`GET /shared_items`	Get a shared item's information


ENDPOINTS

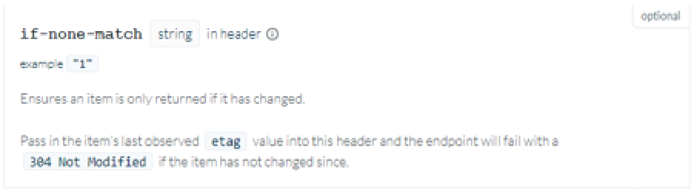
GET Get file information

GET Get file thumbnail

POST Copy file

PUT Update file

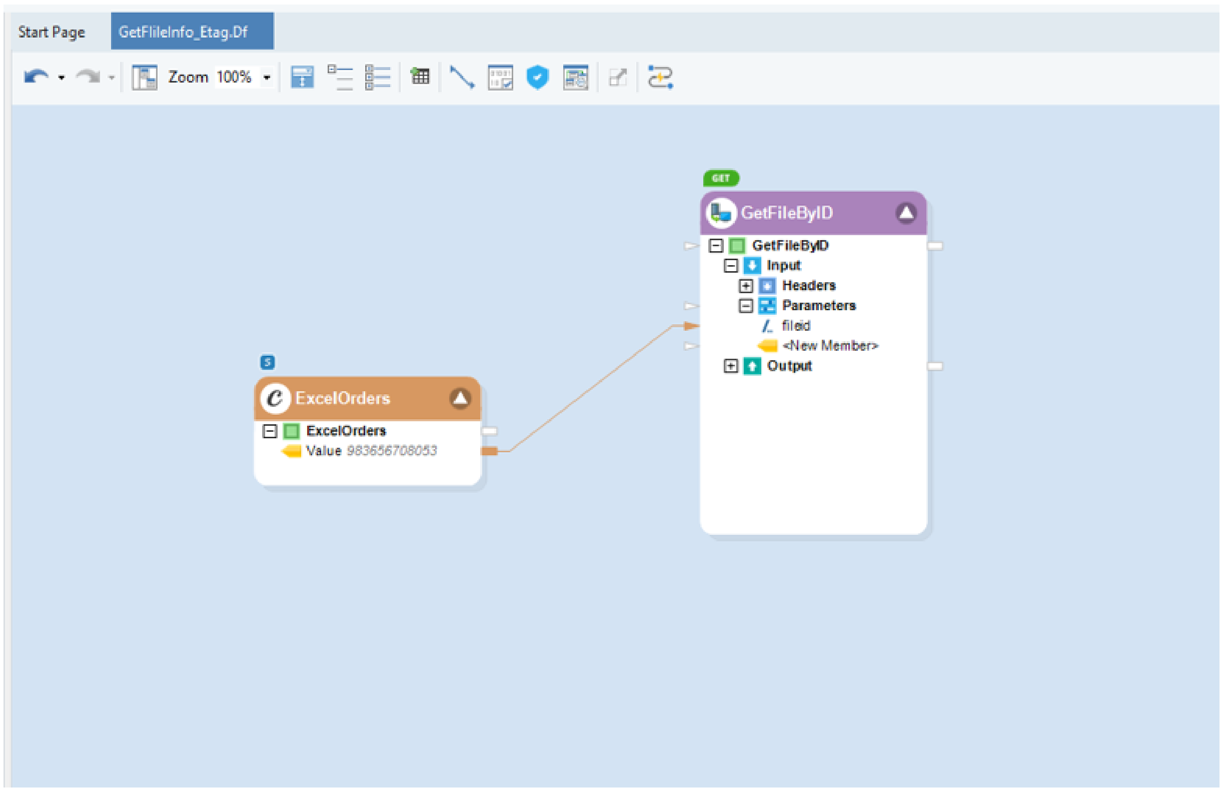
DEL Delete file


`if-none-match` string in header optional

example "1"

Ensures an item is only returned if it has changed.

Pass in the item's last observed `etag` value into this header and the endpoint will fail with a `304 Not Modified` if the item has not changed since.



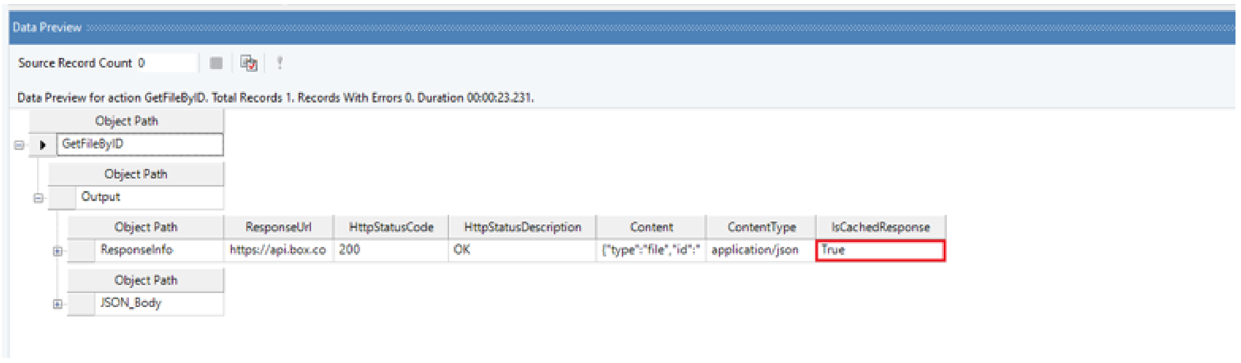
Now, if we go to the Service options screen of the *API Client* object, we can see that we have a checkbox to enable eTags which further gives us two checkboxes of,

- Retrieve if None Match Header
- Update Using If-Match Header.

We need to enable the eTag and the If-None-Match header checkbox.

When the request is sent to the server to fetch the file information for the first time via the GET API request, the response will be returned with an eTag value. This eTag value along with the response will be stored in the response caches at the client side.

- The field of "Is cached response" in the response info node will be returned True because we are making the call for the first time and receiving the response from the server that will be cached.
- In the future, if the client makes an API call again to fetch the file information, the eTag in response caches will be first compared with the latest eTag from the Server. So, for the consecutive API calls having the same cached eTag, we will see the "Is Cached Response" field as true.
- In case, the file has not changed or updated, and the caches are reusable. The server will send a No modification response as we can see in our job trace. This means that the server does not have to send the requested information again in the response instead the client can use its response caches.


Data Preview

Source Record Count 0

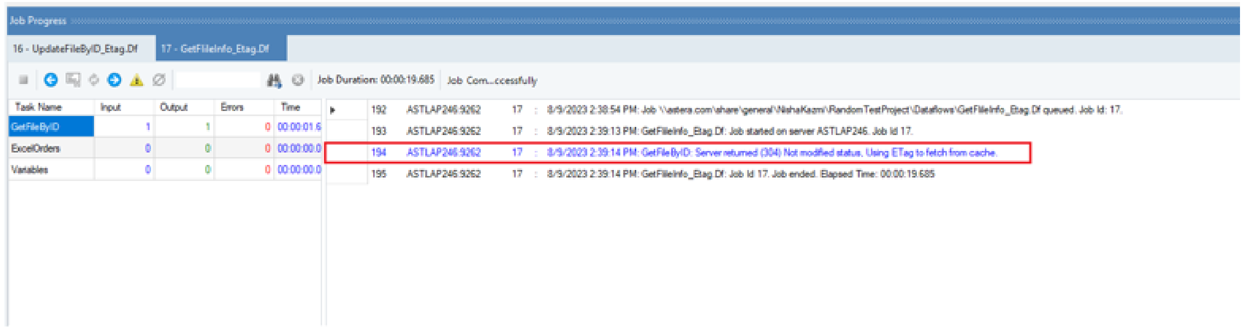
Data Preview for action GetFileByID. Total Records 1. Records With Errors 0. Duration 00:00:23.231.

Object Path
GetFileByID

Object Path
Output

Object Path	ResponseUrl	HttpStatusCode	HttpStatusDescription	Content	ContentType	IsCachedResponse
ResponseInfo	https://api.box.co	200	OK	{"type":"file","id":"	application/json	True

Object Path
JSON_Body


Job Progress

16 - UpdateFileByID_Etag.Df | 17 - GetFileInfo_Etag.Df

Job Duration: 00:00:19.685 Job Com...ccessfully

Task Name	Input	Output	Errors	Time
GetFileByID	1	1	0	00:00:01.6
ExcelOrders	0	0	0	00:00:00.0
Variables	0	0	0	00:00:00.0

```
192  ASTLAP246:9262  17 : 8/9/2023 2:38:54 PM: Job \\astera.com\share\general\NishaKazmi\RandomTestProject\Dataflows\GetFileInfo_Etag.Df queued. Job Id: 17.
193  ASTLAP246:9262  17 : 8/9/2023 2:39:13 PM: GetFileInfo_Etag.Df: Job started on server ASTLAP246. Job Id 17.
194  ASTLAP246:9262  17 : 8/9/2023 2:39:14 PM: GetFileByID: Server returned (304) Not modified status. Using ETag to fetch from cache.
195  ASTLAP246:9262  17 : 8/9/2023 2:39:14 PM: GetFileInfo_Etag.Df: Job Id 17. Job ended. Elapsed Time: 00:00:19.685
```

So, this is how eTags help to prevent unnecessary download and retrievable of information in turn saving the server's bandwidth and request processing time.

If-Match eTag

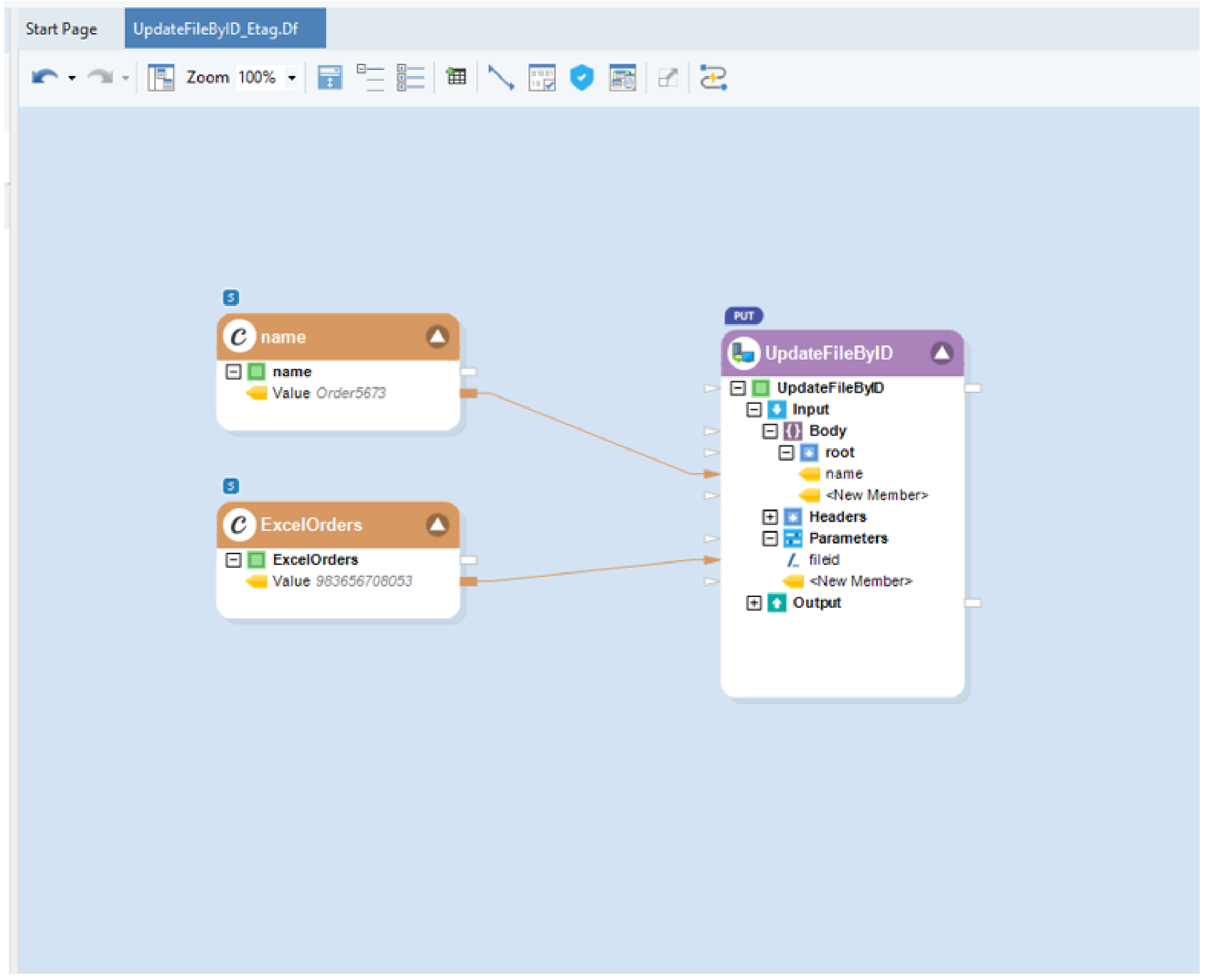
Let's look at another use case of eTag related to concurrency control.

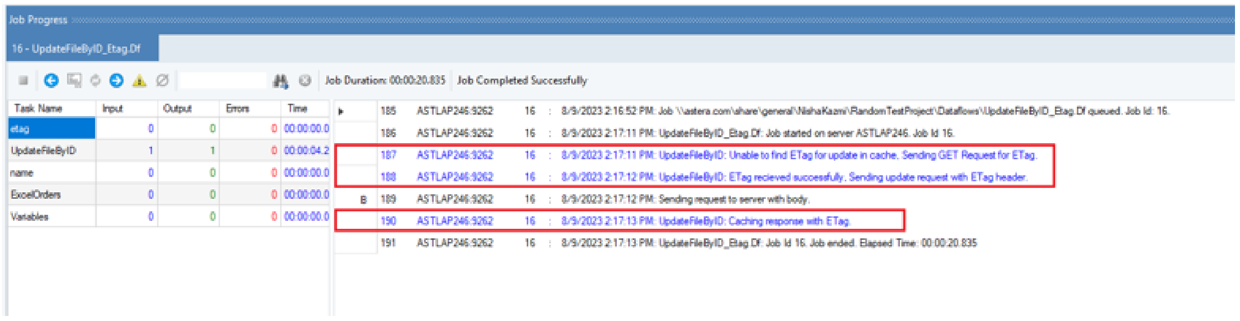
It is possible that more than one client is sending requests to update the resources of the server. Then, to prevent loss of changes and to detect simultaneous updates, the client can send an eTag in the If-Match header field in the request to the server, if another client updates the resource in between or the file is modified, the server can compare the client's eTag with its own current one and if they don't match, the server can prevent clients from overwriting the changes or it will ensure that the latest version of the resource gets updated.

Let's try to understand it with the help of a use case.

- We will make two update API calls to change the name of a file uploaded at our Box account. But, in between the two calls we will make some changes to the file at the server side and see how the eTags play their part in ensuring consistency.
- Here, we have a dataflow in which we are making a PUT request to update the name of the file associated with the file ID of "983656708053" on our Box account. From the API documentation, we can see that the if-match header field is supported in the PUT */file/{fileid}* endpoint of Box APIs so in the API client service options, we will enable the If-Match Header checkbox.
- The API client first checks if an eTag and response corresponding to this endpoint URL already exists in the cache. In the job trace, we can see that there is no eTag in the response caches because we are making the update request for this file for the first time.
- Behind the scenes, the client first makes a Get call to the same endpoint URL and stores the eTag and response in its cache. Next, a PUT request is sent to the same endpoint URL including the eTag received earlier as the value of the If-Match header.

- The server processes the update request and the eTag and the response returned is cached as the eTag received matches to the most recent version of the resource at the server.





Job Duration: 00:00:20.835 Job Completed Successfully

Task Name	Input	Output	Errors	Time
etag	0	0	0	00:00:00.0
UpdateFileByID	1	1	0	00:00:04.2
name	0	0	0	00:00:00.0
ExcelOrders	0	0	0	00:00:00.0
Variables	0	0	0	00:00:00.0

```
185  ASTLAP246:9262  16  :  8/9/2023 2:16:52 PM: Job \\astera.com\share\general\NishaKazmi\RandomTestProject\Dataflows\UpdateFileByID_Etag.Df queued. Job Id: 16.
186  ASTLAP246:9262  16  :  8/9/2023 2:17:11 PM: UpdateFileByID_Etag.Df: Job started on server ASTLAP246. Job Id 16.
187  ASTLAP246:9262  16  :  8/9/2023 2:17:11 PM: UpdateFileByID: Unable to find ETag for update in cache. Sending GET Request for ETag.
188  ASTLAP246:9262  16  :  8/9/2023 2:17:12 PM: UpdateFileByID: ETag recieved succesfully. Sending update request with ETag header.
189  ASTLAP246:9262  16  :  8/9/2023 2:17:12 PM: Sending request to server with body.
190  ASTLAP246:9262  16  :  8/9/2023 2:17:13 PM: UpdateFileByID: Caching response with ETag.
191  ASTLAP246:9262  16  :  8/9/2023 2:17:13 PM: UpdateFileByID_Etag.Df: Job Id 16. Job ended. Elapsed Time: 00:00:20.835
```

This concludes our discussion on the eTag request service options and how they help with response caching and maintaining concurrency control in Astera API Management.

11.5 HTTP Redirect Calls

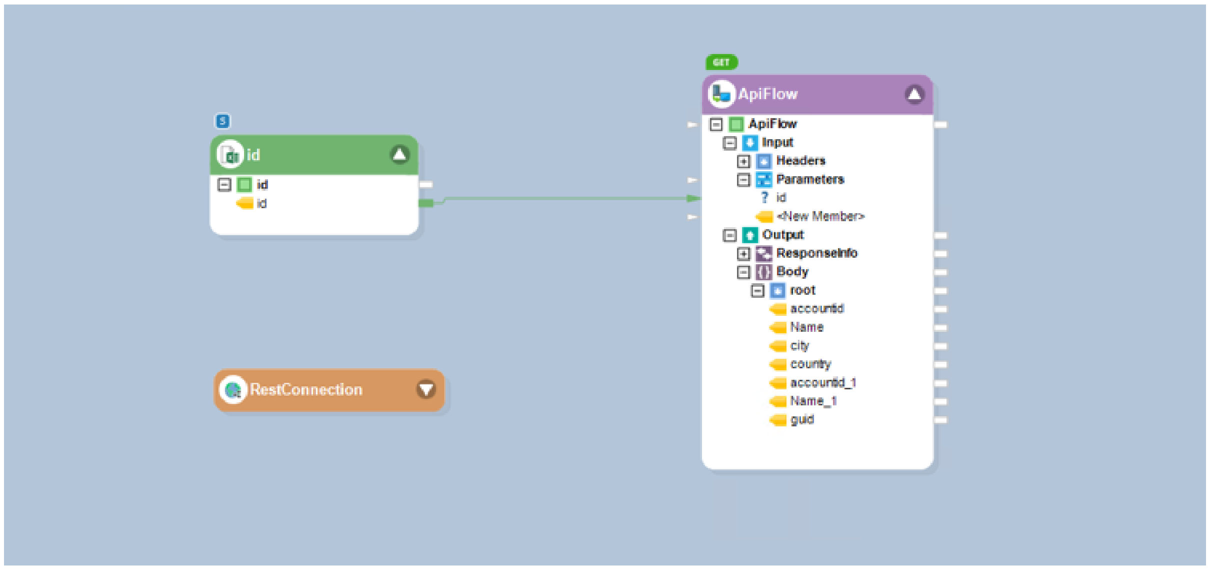
11.5.1 What is an HTTP redirection?

HTTP redirection, also known as URL forwarding, allows an API to provide more than one URL location to the resource in the response. HTTP redirects usually happen due to temporal or permanent unavailability of the application, website, or pages. For example, unavailability due to server maintenance or re-organization of the URL links.

Redirect responses from the server have a 3xx series HTTP status code along with a *Location* header parameter that provides the URL to the resource's new address.

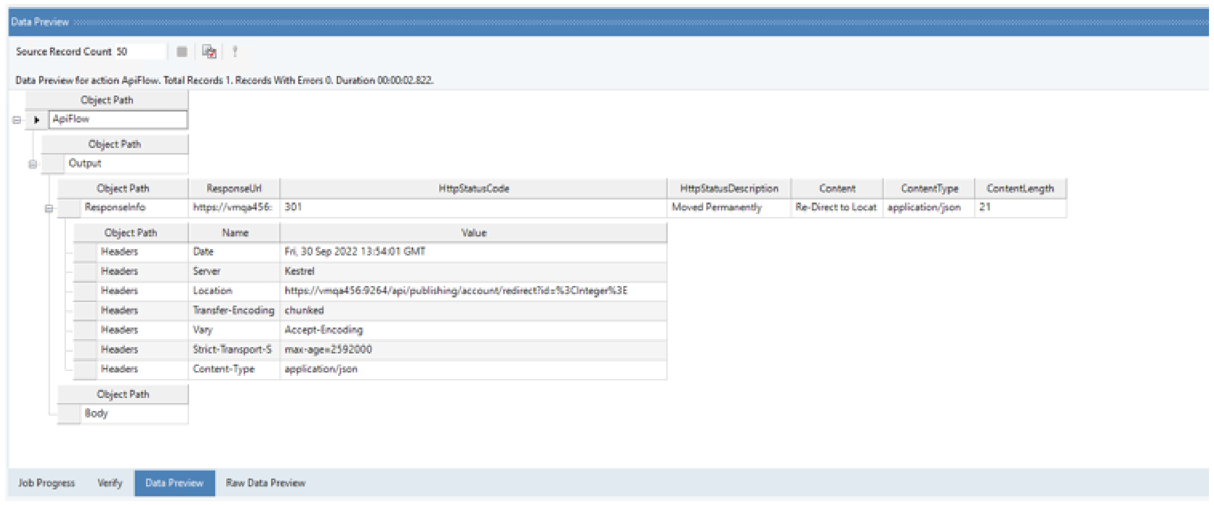
11.5.2 Use Case

In this use case, we have a GET API resource *account* that returns account details based on the provided ID Query parameter.



On previewing the *API Client,* a request is sent to fetch the *account* for the given Id. In the response returned the API request is redirected returning a *302 Found* status code response indicating that the resource is temporarily unavailable.

It may be due to server maintenance or any other unforeseeable reason. We can see that the *Location* header parameter is received with the response too. The *Value* of this header is the address to the alternative resource that must be accessed to retrieve the required account details.



Data Preview

Source Record Count 50

Data Preview for action ApiFlow. Total Records 1. Records With Errors 0. Duration 00:00:02.822.

Object Path
ApiFlow

Object Path
Output

Object Path	ResponseUrl	HttpStatusCode	HttpStatusDescription	Content	ContentType	ContentLength
ResponseInfo	https://vmqa456:	301	Moved Permanently	Re-Direct to Locat	application/json	21

Object Path	Name	Value
Headers	Date	Fri, 30 Sep 2022 13:54:01 GMT
Headers	Server	Kestrel
Headers	Location	https://vmqa456:9264/api/publishing/account/redirect?id=%3Cinteger%3E
Headers	Transfer-Encoding	chunked
Headers	Vary	Accept-Encoding
Headers	Strict-Transport-S	max-age=2592000
Headers	Content-Type	application/json

Object Path
Body

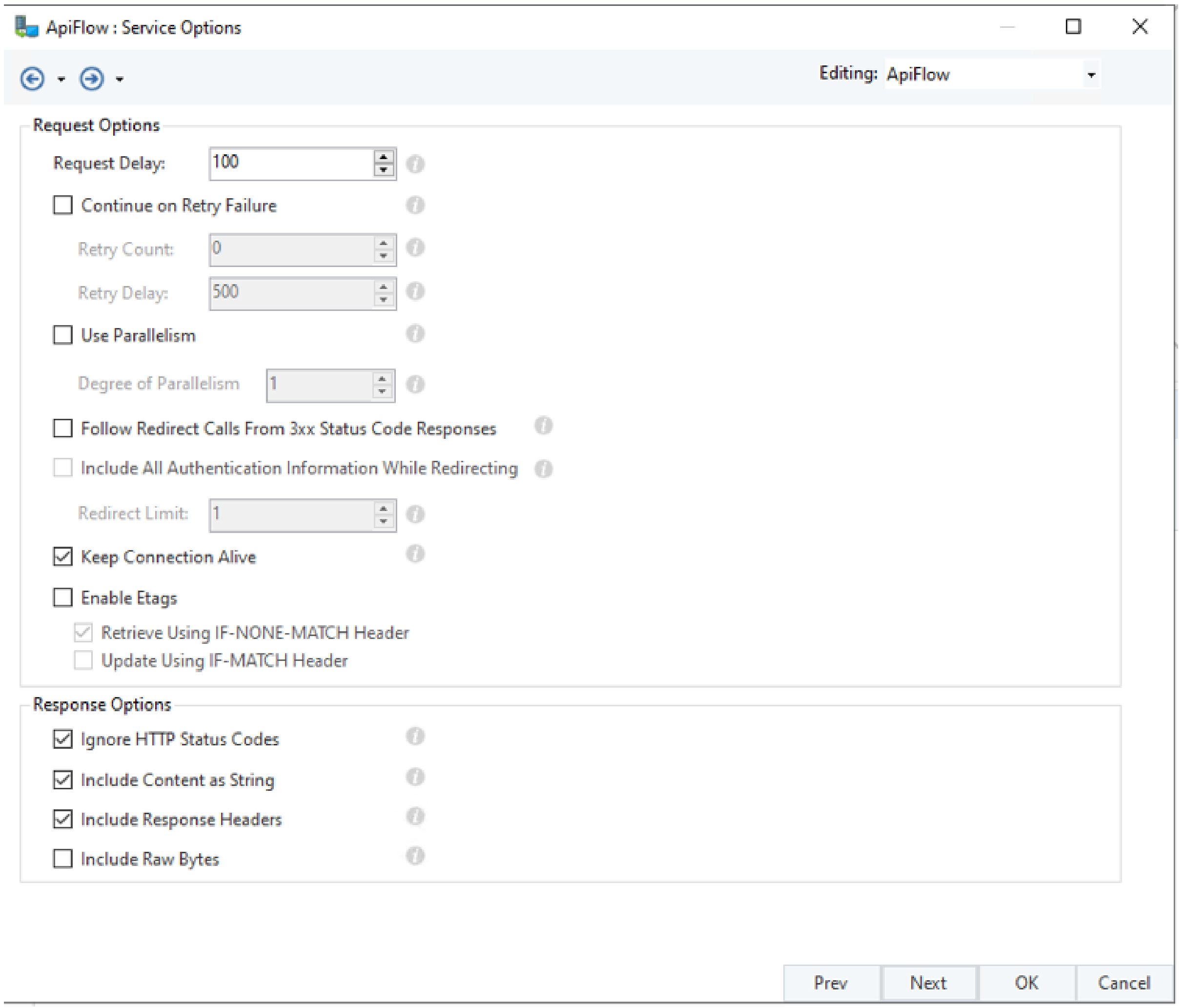
Job Progress · Verify · Data Preview · Raw Data Preview

Let's see how we can configure the API client properties to automatically follow any redirect responses to the new URL Location.

Enable Auto-Redirect Calls

Right-click on the API Client and select properties. Next, navigate to the Service Options window. Here*,* there are multiple options available to configure the redirect call(s).

- *Follow Redirect Calls From 3xx Code Responses* – This option allows auto-redirecting a 3xx HTTP response to the redirected location URL.
- *Redirect Authentication Information* – This option allows forwarding all the authentication details along with the redirected call.
- *Redirect Limit* – This option allows us to specify a limit to the number of redirected calls followed.


ApiFlow : Service Options

Editing: ApiFlow

Request Options

Request Delay: 100

☐ Continue on Retry Failure

Retry Count: 0

Retry Delay: 500

☐ Use Parallelism

Degree of Parallelism 1

☐ Follow Redirect Calls From 3xx Status Code Responses

☐ Include All Authentication Information While Redirecting

Redirect Limit: 1

☑ Keep Connection Alive

☐ Enable Etags

☑ Retrieve Using IF-NONE-MATCH Header

☐ Update Using IF-MATCH Header

Response Options

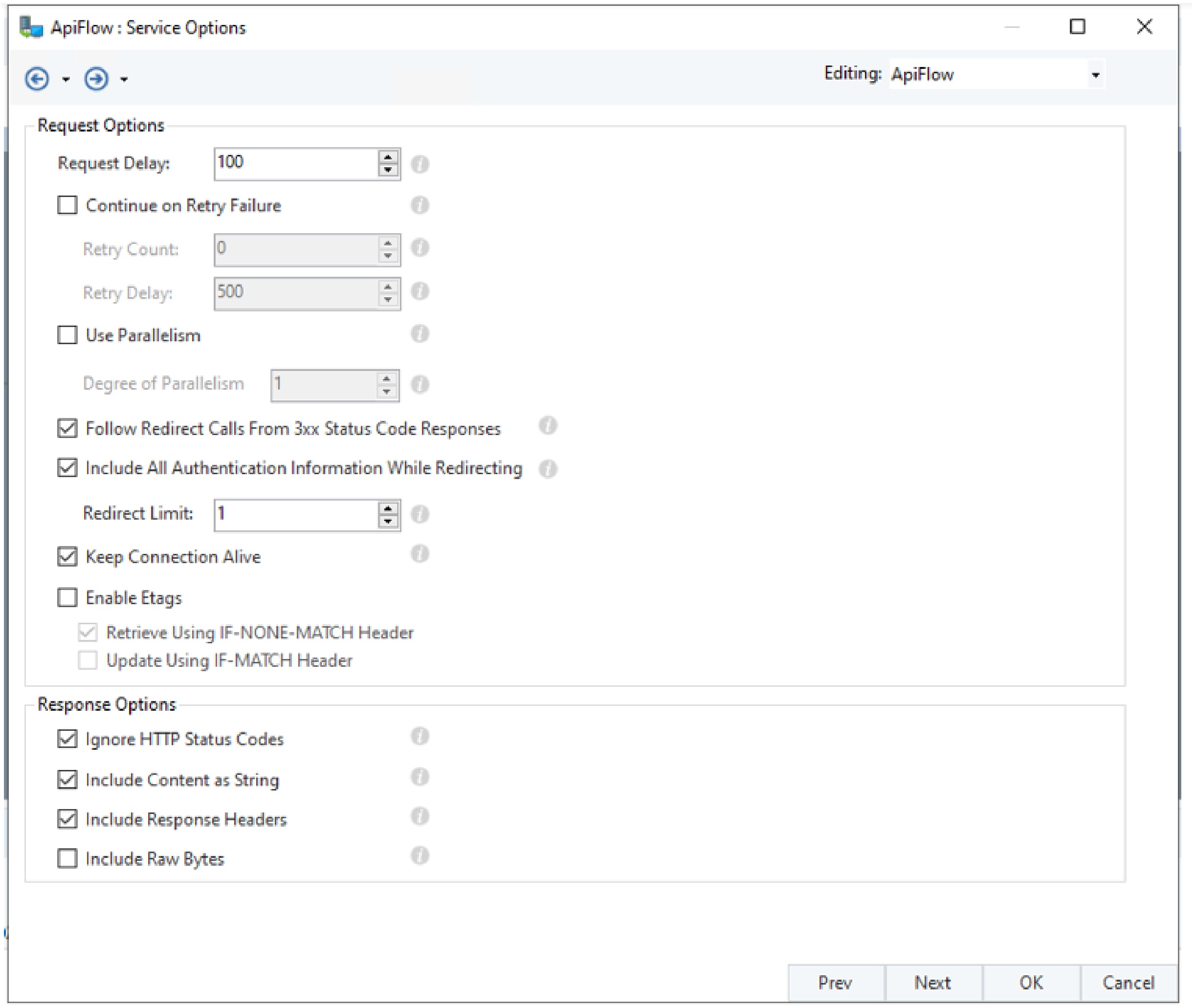
☑ Ignore HTTP Status Codes

☑ Include Content as String

☑ Include Response Headers

☐ Include Raw Bytes

Let’s enable the redirect and authentication options while keeping the redirect limit as 1.



Request Options

Request Delay: 100

☐ Continue on Retry Failure

Retry Count: 0

Retry Delay: 500

☐ Use Parallelism

Degree of Parallelism 1

☑ Follow Redirect Calls From 3xx Status Code Responses

☑ Include All Authentication Information While Redirecting

Redirect Limit: 1

☑ Keep Connection Alive

☐ Enable Etags

☑ Retrieve Using IF-NONE-MATCH Header

☐ Update Using IF-MATCH Header

Response Options

☑ Ignore HTTP Status Codes

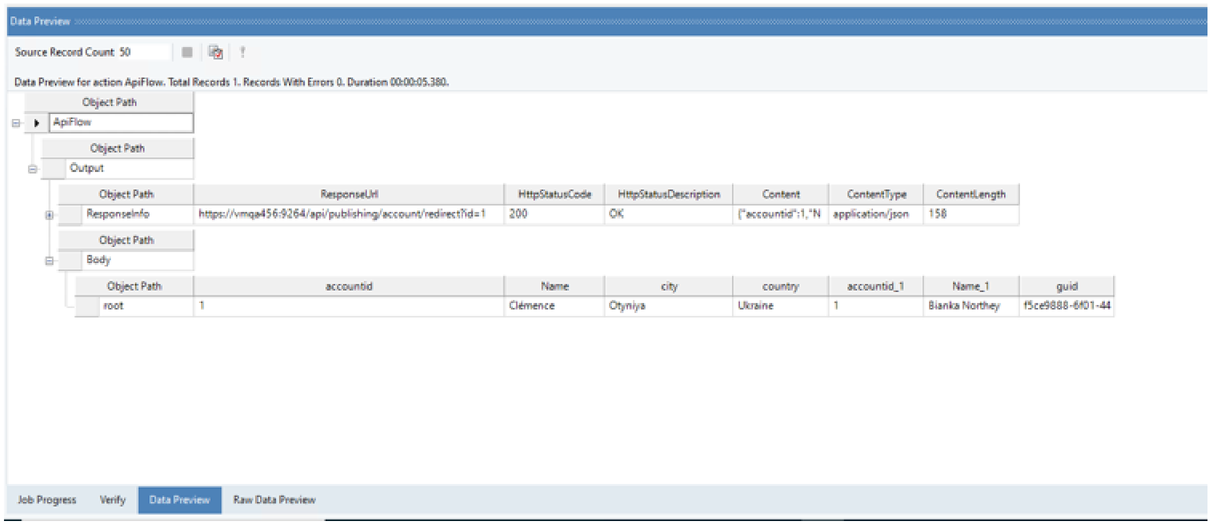
☑ Include Content as String

☑ Include Response Headers

☐ Include Raw Bytes

Note: By default, the *Redirect Authentication Information* and *Redirect Limit* options are disabled. Only on checking the *Follow Redirect Calls From 3xx Code Responses* option are they enabled for configuration.

Now, on previewing the output we can see that a 200 OK status response is received instead of a 302 Found. The request URL field shows that the request was successfully auto-redirected to the redirecting URL.



Data Preview

Source Record Count: 50

Data Preview for action ApiFlow. Total Records 1. Records With Errors 0. Duration 00:00:05.380.

Object Path
ApiFlow

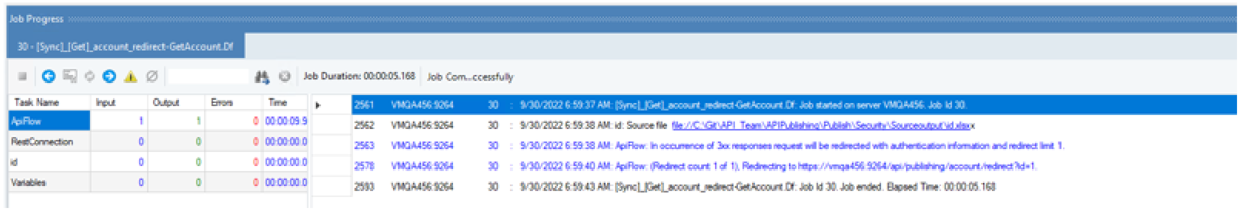
Object Path
Output

Object Path	ResponseUrl	HttpStatusCode	HttpStatusDescription	Content	ContentType	ContentLength
ResponseInfo	https://vmqa456:9264/api/publishing/account/redirect?id=1	200	OK	{"accountid":1,"N	application/json	158

Object Path
Body

Object Path	accountid	Name	city	country	accountid_1	Name_1	guid
root	1	Clémence	Otyniya	Ukraine	1	Bianka Northey	f5ce9888-6f01-44

Now, let's execute the data flow.



Job Progress

30 - [Sync]_[Get]_account_redirect-GetAccount.Df

Job Duration: 00:00:05.168 Job Com...ccessfully

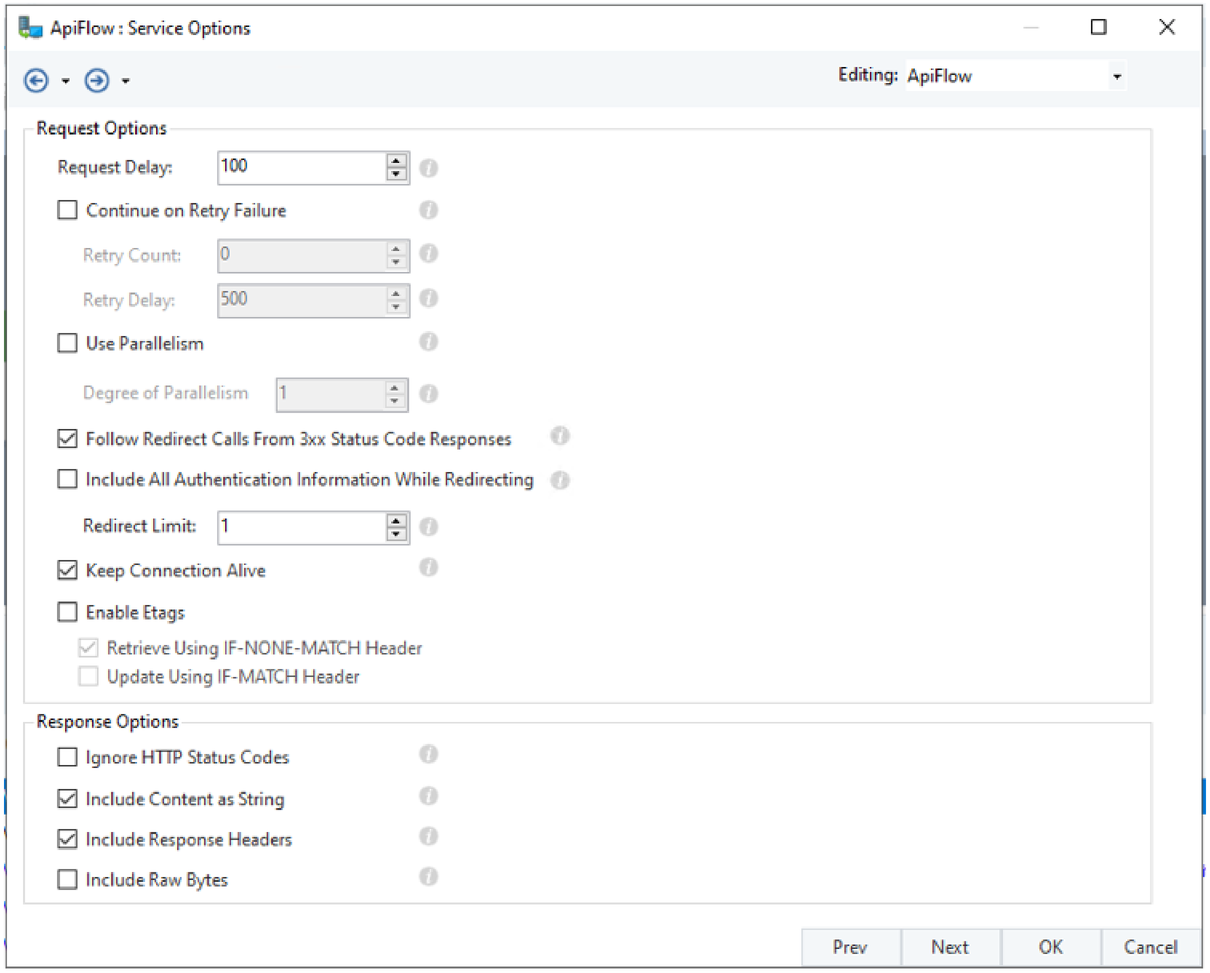
Task Name	Input	Output	Errors	Time
ApiFlow	1	1	0	00:00:09.9
RestConnection	0	0	0	00:00:00.0
id	0	0	0	00:00:00.0
Variables	0	0	0	00:00:00.0

```
2561  VMQA456:9264  30 : 9/30/2022 6:59:37 AM: [Sync]_[Get]_account_redirect-GetAccount.Df: Job started on server VMQA456. Job Id 30.
2562  VMQA456:9264  30 : 9/30/2022 6:59:38 AM: id: Source file file://C:\Git\API_Team\APIPublishing\Publish\Security\Sourceoutput\id.xlsx
2563  VMQA456:9264  30 : 9/30/2022 6:59:38 AM: ApiFlow: In occurrence of 3xx responses request will be redirected with authentication information and redirect limit 1.
2578  VMQA456:9264  30 : 9/30/2022 6:59:40 AM: ApiFlow: (Redirect count: 1 of 1), Redirecting to https://vmqa456:9264/api/publishing/account/redirect?id=1.
2593  VMQA456:9264  30 : 9/30/2022 6:59:43 AM: [Sync]_[Get]_account_redirect-GetAccount.Df: Job Id 30. Job ended. Elapsed Time: 00:00:05.168
```

Here, we can see the job traces show all steps of the redirected calls including how the authentication information was forwarded along with the request, what was the redirect limit, where the request was redirected to, and if the job executed successfully.

Scenario 1 - No Authentication Information Redirected

Let's consider a scenario where the redirected API requires authentication, and we don't send the authentication information along with the redirect call by unchecking the *Redirect Authentication Information* option from the Service Options window.



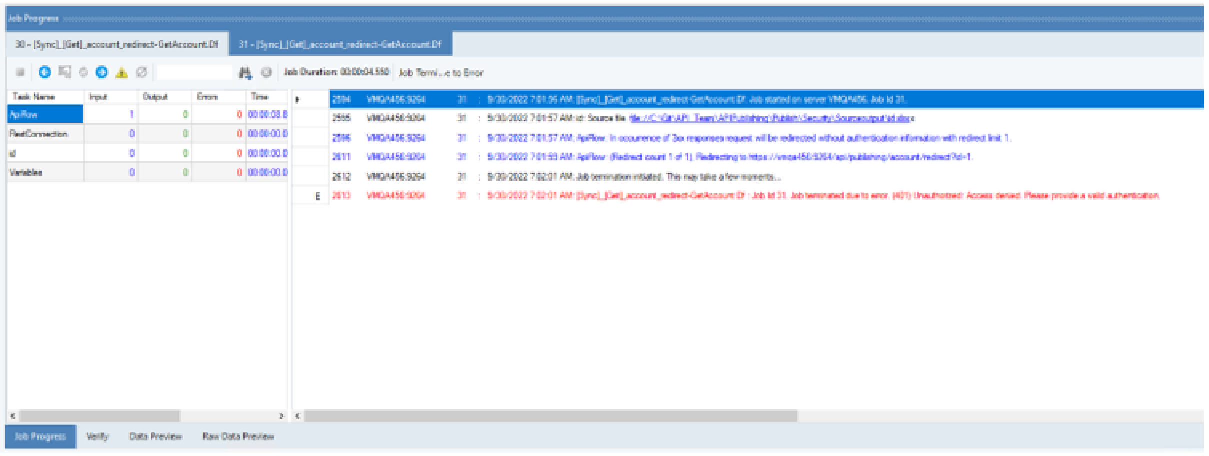
Request Options

Request Delay:	100
☐ Continue on Retry Failure	
Retry Count:	0
Retry Delay:	500
☐ Use Parallelism	
Degree of Parallelism	1
☑ Follow Redirect Calls From 3xx Status Code Responses	
☐ Include All Authentication Information While Redirecting	
Redirect Limit:	1
☑ Keep Connection Alive	
☐ Enable Etags	
☑ Retrieve Using IF-NONE-MATCH Header	
☐ Update Using IF-MATCH Header	

Response Options

☐ Ignore HTTP Status Codes	
☑ Include Content as String	
☑ Include Response Headers	
☐ Include Raw Bytes	

On executing the job, we can see that the request is redirected without the authentication information, and as a result, the server sends back a 401 Unauthorized error response.


Job Progress
30 - [Sync]_[Get]_account_redirect-GetAccount.Df
31 - [Sync]_[Get]_account_redirect-GetAccount.Df
Job Progress
Verify
Data Preview
Raw Data Preview

Job Duration: 00:00:04.550 Job Termi...e to Error

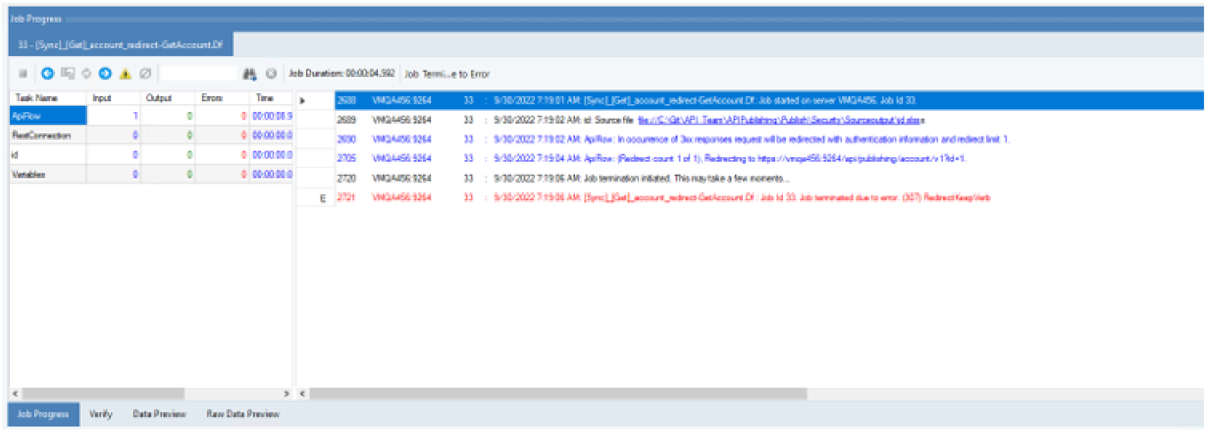
Task Name	Input	Output	Errors	Time
ApiFlow	1	0	0	00:00:08.8
RestConnection	0	0	0	00:00:00.0
id	0	0	0	00:00:00.0
Variables	0	0	0	00:00:00.0

```
2594  VMQA456:9264  31 : 9/30/2022 7:01:56 AM: [Sync]_[Get]_account_redirect-GetAccount.Df: Job started on server VMQA456. Job Id 31.
2595  VMQA456:9264  31 : 9/30/2022 7:01:57 AM: id: Source file: file://C:\Git\API_Team\APIPublishing\Publish\Security\Sourceoutput\id.xlsx
2596  VMQA456:9264  31 : 9/30/2022 7:01:57 AM: ApiFlow: In occurrence of 3xx responses request will be redirected without authentication information with redirect limit 1.
2611  VMQA456:9264  31 : 9/30/2022 7:01:59 AM: ApiFlow: (Redirect count 1 of 1). Redirecting to https://vmqa456:9264/api/publishing/account/redirect?id=1.
2612  VMQA456:9264  31 : 9/30/2022 7:02:01 AM: Job termination initiated. This may take a few moments...
E 2613  VMQA456:9264  31 : 9/30/2022 7:02:01 AM: [Sync]_[Get]_account_redirect-GetAccount.Df : Job Id 31. Job terminated due to error. (401) Unauthorized: Access denied. Please provide a valid authentication.
```

Scenario 2 (Multiple Redirect Calls)

Now, let's consider a scenario where an API request hops through more than one redirected call.

1. The first redirect request returns a 3xx series response. We can see in the Job Trace that on redirecting the request we received a 307-status response indicating that the service is temporarily unavailable. As the redirect count was set to 1 so, only one redirect call was sent by the *API Client.*



Job Progress

33 - [Sync]_[Get]_account_redirect-GetAccount.Df

Job Duration: 00:00:04.592 | Job Termi...e to Error

Task Name	Input	Output	Errors	Time
ApiFlow	1	0	0	00:00:08.9
RestConnection	0	0	0	00:00:00.0
id	0	0	0	00:00:00.0
Variables	0	0	0	00:00:00.0

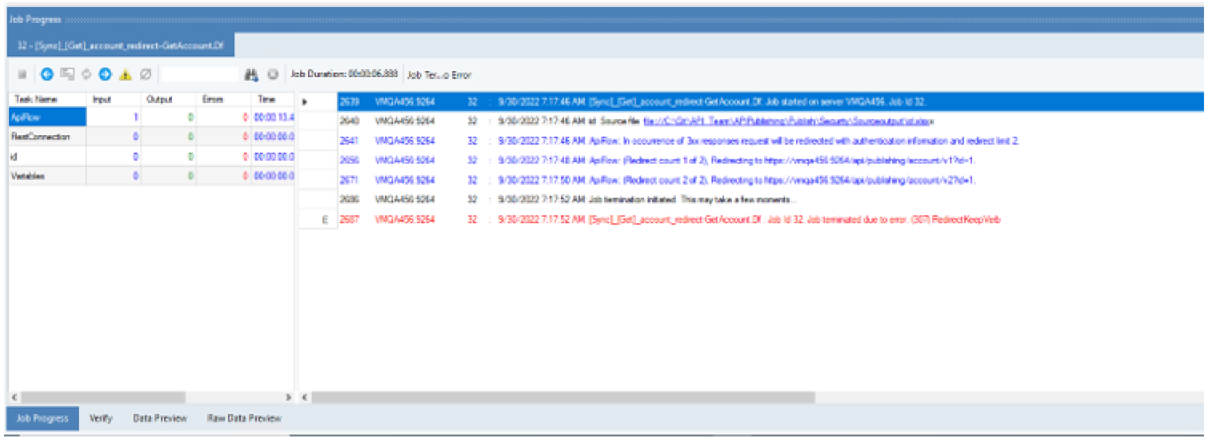
```
   2688  VMQA456:9264  33 : 9/30/2022 7:19:01 AM: [Sync]_[Get]_account_redirect-GetAccount.Df: Job started on server VMQA456. Job Id 33.
   2689  VMQA456:9264  33 : 9/30/2022 7:19:02 AM: id: Source file: file://C:\Git\API_Team\APIPublishing\Publish\Security\Sourceoutput\id.xlsx
   2690  VMQA456:9264  33 : 9/30/2022 7:19:02 AM: ApiFlow: In occurrence of 3xx responses request will be redirected with authentication information and redirect limit 1.
   2705  VMQA456:9264  33 : 9/30/2022 7:19:04 AM: ApiFlow: (Redirect count 1 of 1), Redirecting to https://vmqa456:9264/api/publishing/account/v1?id=1.
   2720  VMQA456:9264  33 : 9/30/2022 7:19:06 AM: Job termination initiated. This may take a few moments...
E  2721  VMQA456:9264  33 : 9/30/2022 7:19:06 AM: [Sync]_[Get]_account_redirect-GetAccount.Df : Job Id 33. Job terminated due to error. (307) RedirectKeepVerb
```

Job Progress | Verify | Data Preview | Raw Data Preview

2. In such a situation, we need to follow all the redirect requests until a 200 OK response is received. For that, we can increase the *Redirect Limit* count.

For example, we will set the limit to 2 and send the request.

In the Job Progress window, we can see that two redirect calls have been exhausted, but we still received a 307-status response.



Job Progress

32 - [Sync]_[Get]_account_redirect-GetAccount.Df

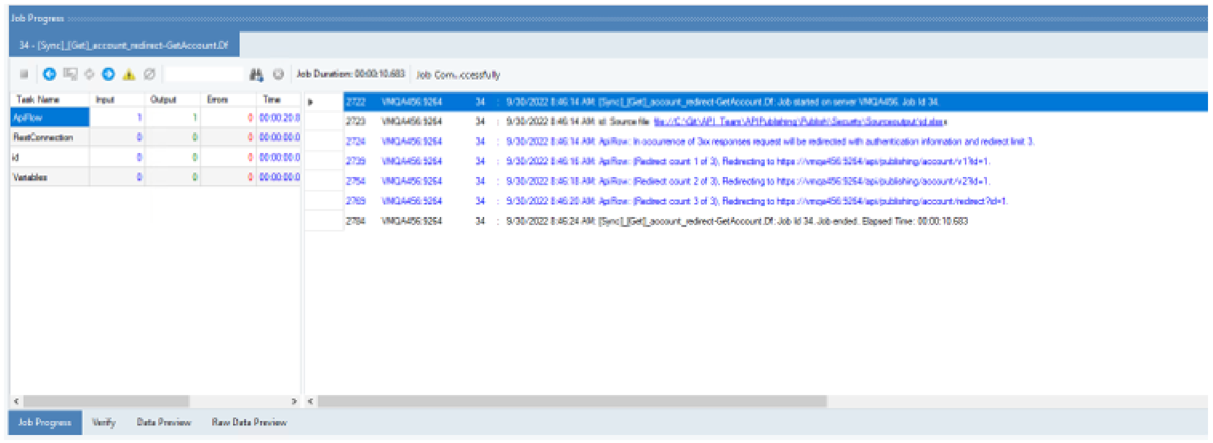
Job Duration: 00:00:06.888 | Job Ter...o Error

Task Name	Input	Output	Errors	Time
ApiFlow	1	0	0	00:00:13.4
RestConnection	0	0	0	00:00:00.0
id	0	0	0	00:00:00.0
Variables	0	0	0	00:00:00.0

```
   2639  VMQA456:9264  32 : 9/30/2022 7:17:46 AM: [Sync]_[Get]_account_redirect-GetAccount.Df: Job started on server VMQA456. Job Id 32.
   2640  VMQA456:9264  32 : 9/30/2022 7:17:46 AM: id: Source file: file://C:\Git\API_Team\APIPublishing\Publish\Security\Sourceoutput\id.xlsx
   2641  VMQA456:9264  32 : 9/30/2022 7:17:46 AM: ApiFlow: In occurrence of 3xx responses request will be redirected with authentication information and redirect limit 2.
   2656  VMQA456:9264  32 : 9/30/2022 7:17:48 AM: ApiFlow: (Redirect count 1 of 2), Redirecting to https://vmqa456:9264/api/publishing/account/v1?id=1.
   2671  VMQA456:9264  32 : 9/30/2022 7:17:50 AM: ApiFlow: (Redirect count 2 of 2), Redirecting to https://vmqa456:9264/api/publishing/account/v2?id=1.
   2686  VMQA456:9264  32 : 9/30/2022 7:17:52 AM: Job termination initiated. This may take a few moments...
E  2687  VMQA456:9264  32 : 9/30/2022 7:17:52 AM: [Sync]_[Get]_account_redirect-GetAccount.Df : Job Id 32. Job terminated due to error. (307) RedirectKeepVerb
```

Job Progress | Verify | Data Preview | Raw Data Preview

Let's increase the limit to 3 and send the request.



Job Progress

34 - [Sync]_[Get]_account_redirect-GetAccount.Df

Job Duration: 00:00:10.683 Job Com...ccessfully

Task Name	Input	Output	Errors	Time
ApiFlow	1	1	0	00:00:20.0
RestConnection	0	0	0	00:00:00.0
id	0	0	0	00:00:00.0
Variables	0	0	0	00:00:00.0

```
2722   VMQA456:9264   34  :  9/30/2022 8:46:14 AM: [Sync]_[Get]_account_redirect-GetAccount.Df: Job started on server VMQA456. Job Id 34.
2723   VMQA456:9264   34  :  9/30/2022 8:46:14 AM: at Source file: file://C:\Git\API_Team\APIPublishing\Publish\Security\Sourceoutput\id.xlsx
2724   VMQA456:9264   34  :  9/30/2022 8:46:14 AM: ApiFlow: In occurrence of 3xx responses request will be redirected with authentication information and redirect limit 3.
2739   VMQA456:9264   34  :  9/30/2022 8:46:16 AM: ApiFlow: (Redirect count 1 of 3), Redirecting to https://vmqa456:9264/api/publishing/account/v1?id=1.
2754   VMQA456:9264   34  :  9/30/2022 8:46:18 AM: ApiFlow: (Redirect count 2 of 3), Redirecting to https://vmqa456:9264/api/publishing/account/v2?id=1.
2769   VMQA456:9264   34  :  9/30/2022 8:46:20 AM: ApiFlow: (Redirect count 3 of 3), Redirecting to https://vmqa456:9264/api/publishing/account/redirect?id=1.
2784   VMQA456:9264   34  :  9/30/2022 8:46:24 AM: [Sync]_[Get]_account_redirect-GetAccount.Df: Job Id 34. Job ended. Elapsed Time: 00:00:10.683
```

Finally, a 200 OK response is received on the third redirect call.

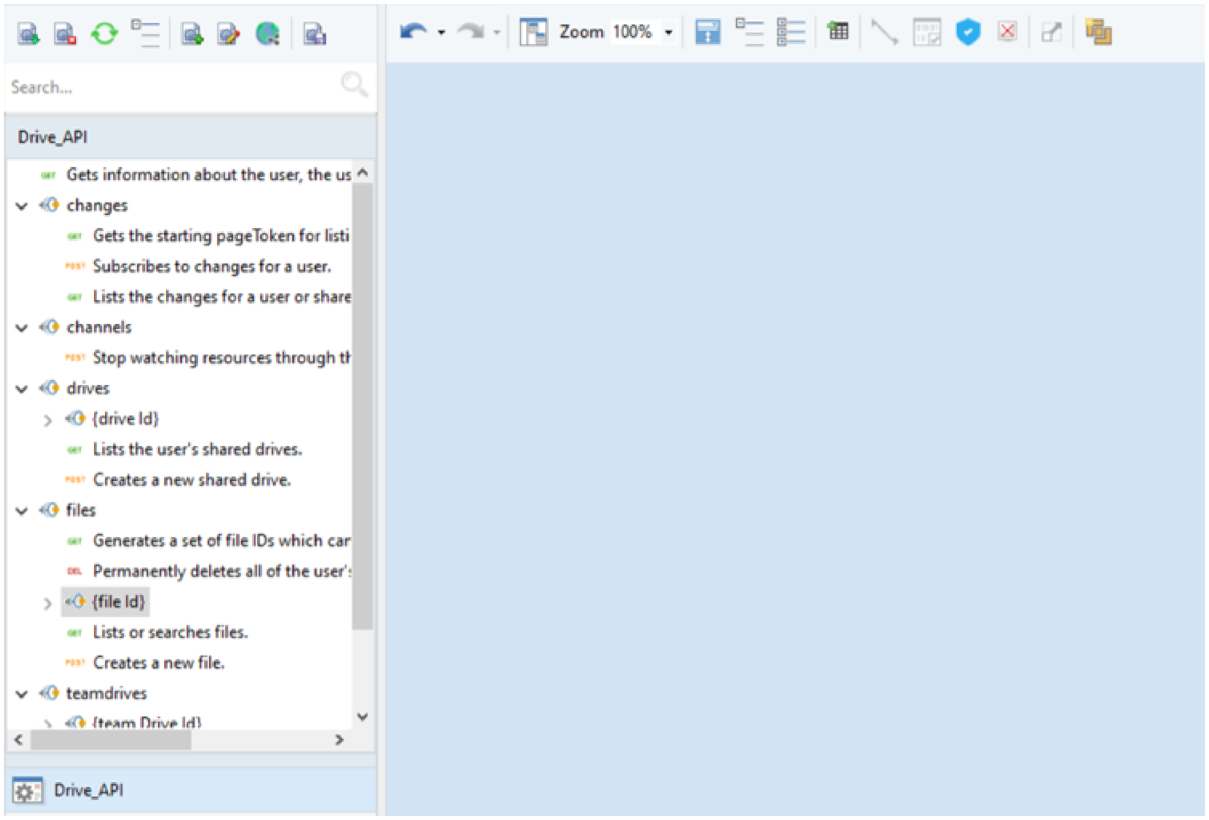
This concludes the article on how HTTP redirect calls are automated by the API Client in Astera API Management.

11.6 Method Operations

In this article, we will be discussing various HTTP methods. We will see how HTTP requests can be made through the *API Client* object in Astera API Management.

For our use cases, we have made use of the *Petstore* Open-API definition. We can import the API to the *API Browser* using its import URL.

Once done, it automatically establishes various pre-defined endpoints as *API Client* objects. They can then be dragged and dropped onto a dataflow for further configurations and transformations.

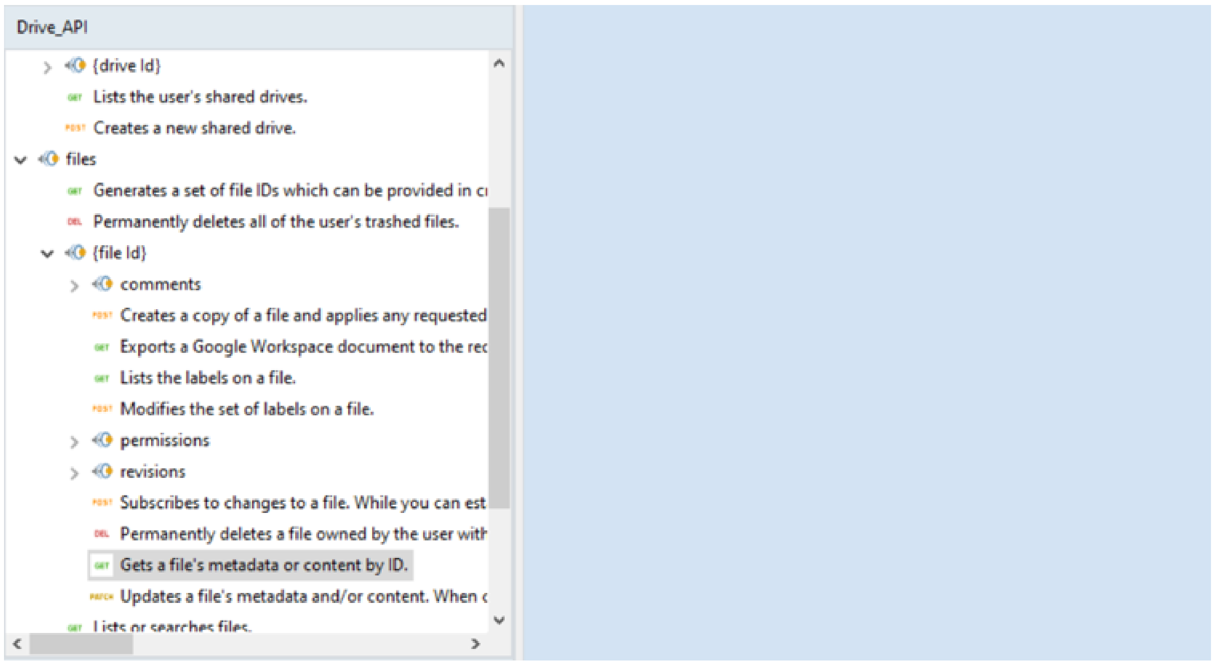

Zoom 100%
Search...
Drive_API
GET Gets information about the user, the us
changes
GET Gets the starting pageToken for listi
POST Subscribes to changes for a user.
GET Lists the changes for a user or share
channels
POST Stop watching resources through th
drives
{drive Id}
GET Lists the user's shared drives.
POST Creates a new shared drive.
files
GET Generates a set of file IDs which car
DEL Permanently deletes all of the user's
{file Id}
GET Lists or searches files.
POST Creates a new file.
teamdrives
{team Drive Id}
Drive_API

Note: When imported, a shared connection object will also be created containing the base URL and authentication details.

To learn more about importing a URL to the REST API Browser, click here.

11.6.1 Making a GET Request

1. First, drag and drop the *Get a file's metadata or Content by ID* endpoint from the browser onto the dataflow.

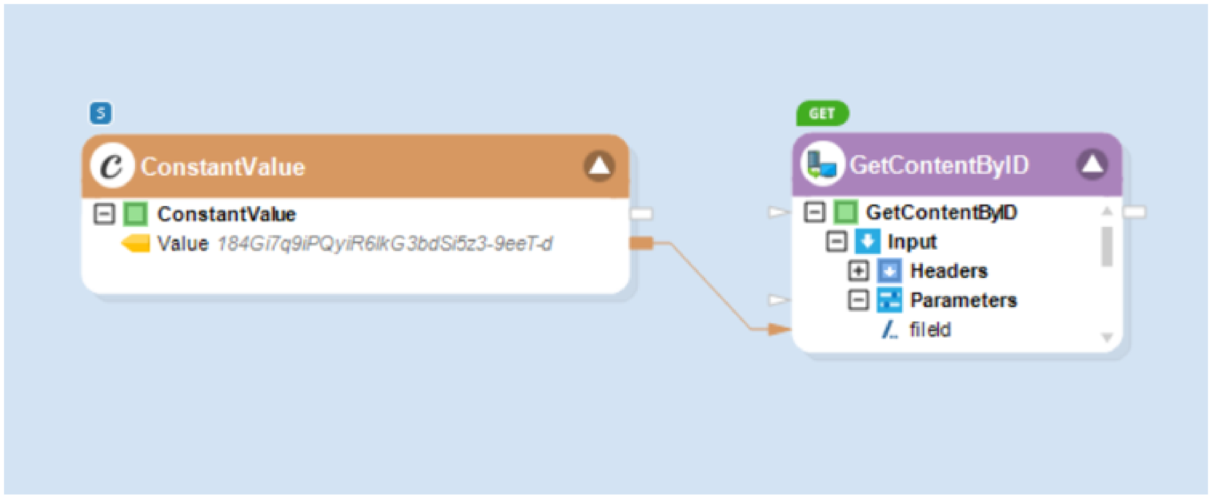


Drive_API

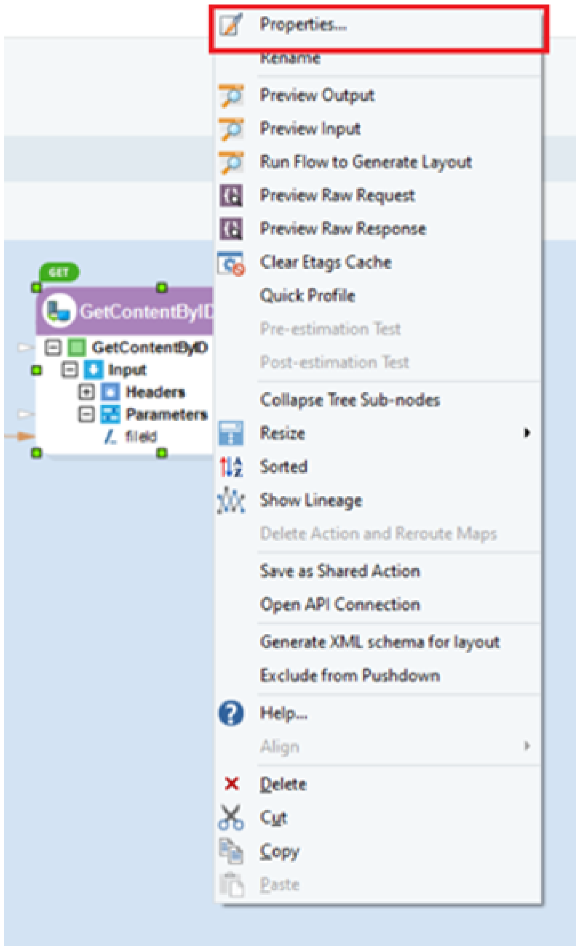
- {drive Id}
  - GET Lists the user's shared drives.
  - POST Creates a new shared drive.
- files
  - GET Generates a set of file IDs which can be provided in c
  - DEL Permanently deletes all of the user's trashed files.
  - {file Id}
    - comments
    - POST Creates a copy of a file and applies any requested
    - GET Exports a Google Workspace document to the rec
    - GET Lists the labels on a file.
    - POST Modifies the set of labels on a file.
    - permissions
    - revisions
    - POST Subscribes to changes to a file. While you can est
    - DEL Permanently deletes a file owned by the user with
    - GET Gets a file's metadata or content by ID.
    - PATCH Updates a file's metadata and/or content. When c
  - GET Lists or searches files.

In this scenario, we want to get metadata for a file with *file**ID*, "184Gi7q9iPQyiR6lkG3bdSi5z3-9eeT-d".

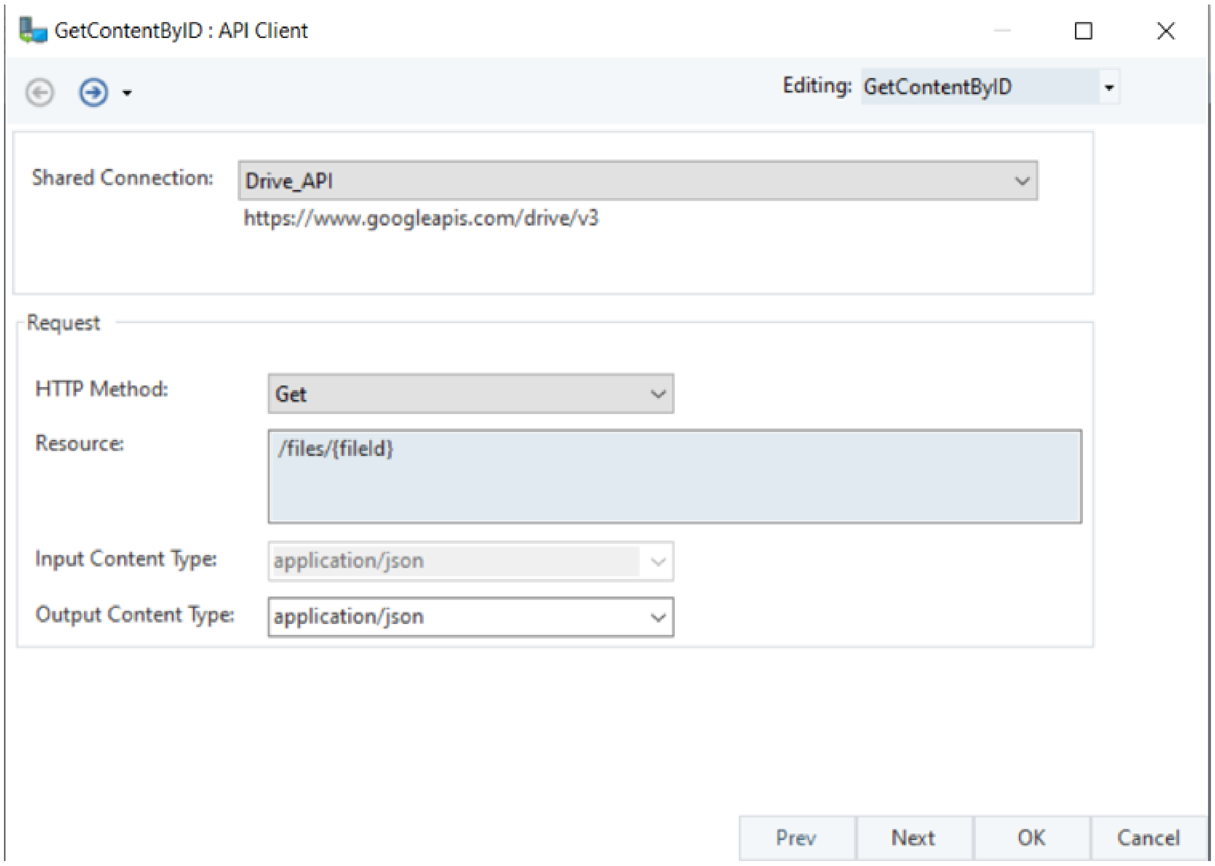
For this, we will pass the relevant *fileID* using a *ConstantValue* object.



2. To explore the *API Client* object for this method, right-click on the object's header and select *Properties.*



This will open the *API Client* screen where the connection info of your API is defined.


GetContentByID : API Client

Editing: GetContentByID

Shared Connection: Drive_API
https://www.googleapis.com/drive/v3

Request

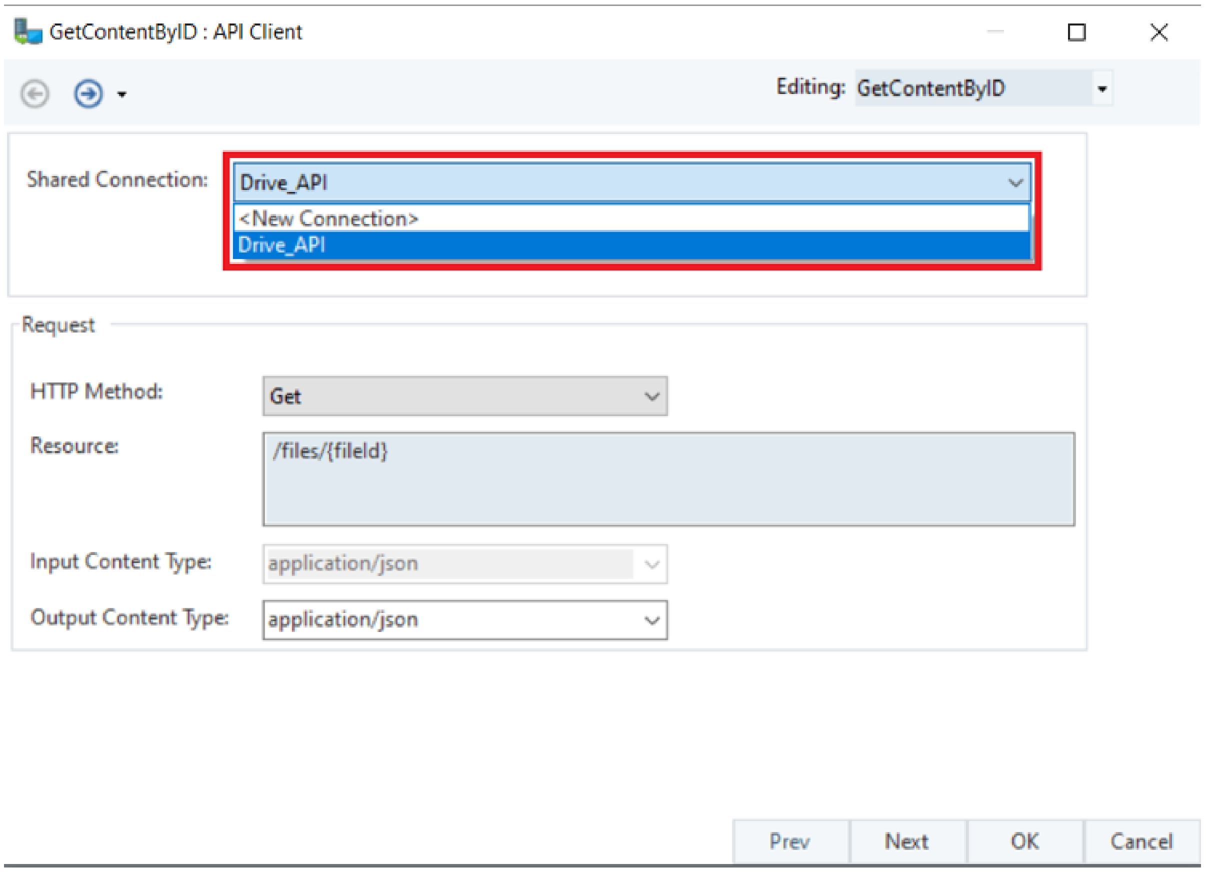
HTTP Method: Get

Resource: /files/{fileId}

Input Content Type: application/json

Output Content Type: application/json

The *Shared Connection*, *Method*, and *Resource* here are already configured. Notice that *Resource* consists of ‘files’ along with the ‘fileid’ URL parameter.



Editing: GetContentByID

Shared Connection: Drive_API
<New Connection>
Drive_API

Request

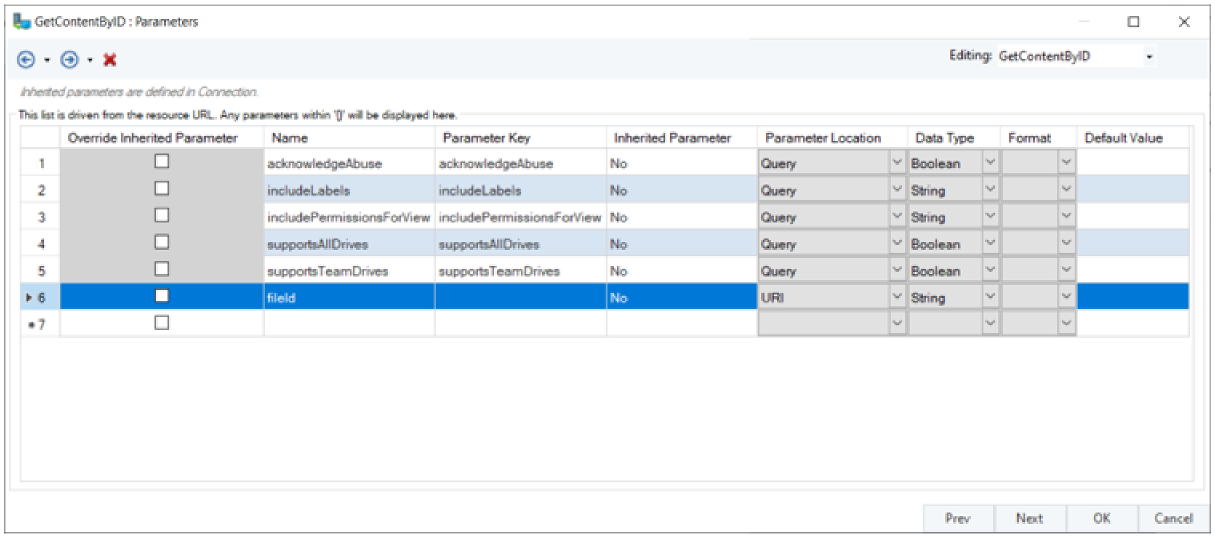
HTTP Method: Get

Resource: /files/{fileId}

Input Content Type: application/json

Output Content Type: application/json

3. Click *Next*.



Editing: GetContentByID

Inherited parameters are defined in Connection.

This list is driven from the resource URL. Any parameters within '{}' will be displayed here.

	Override Inherited Parameter	Name	Parameter Key	Inherited Parameter	Parameter Location	Data Type	Format	Default Value
1	☐	acknowledgeAbuse	acknowledgeAbuse	No	Query	Boolean		
2	☐	includeLabels	includeLabels	No	Query	String		
3	☐	includePermissionsForView	includePermissionsForView	No	Query	String		
4	☐	supportsAllDrives	supportsAllDrives	No	Query	Boolean		
5	☐	supportsTeamDrives	supportsTeamDrives	No	Query	Boolean		
6	☐	fileId		No	URI	String		
7	☐							

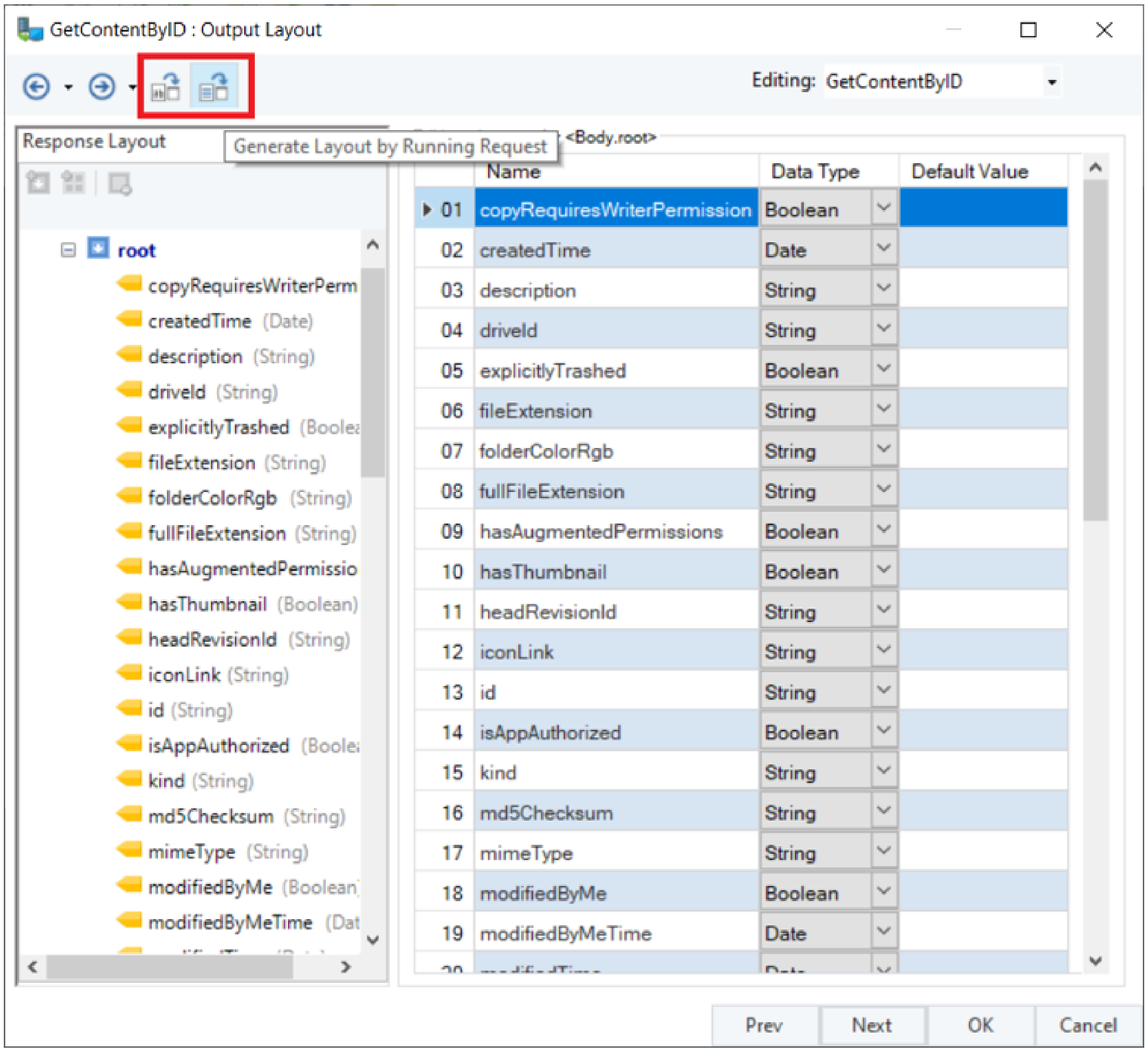
Here, the ‘fileId’ URL parameter follows from the defined resource.

For our use case, we will use this parameter to get details for a pet.

Click *Next* to proceed to the *Output Layout* screen, where you can view the *Response Layout* of your API. There are two ways in which you can generate the output layout if required.

- The first one is by providing sample text by clicking the *Generate Layout by providing Sample Text* option.

- The other way to do this is by running a request by clicking the *Generate Layout by running Request* option.



GetContentByID : Output Layout

Editing: GetContentByID

Response Layout

Generate Layout by Running Request

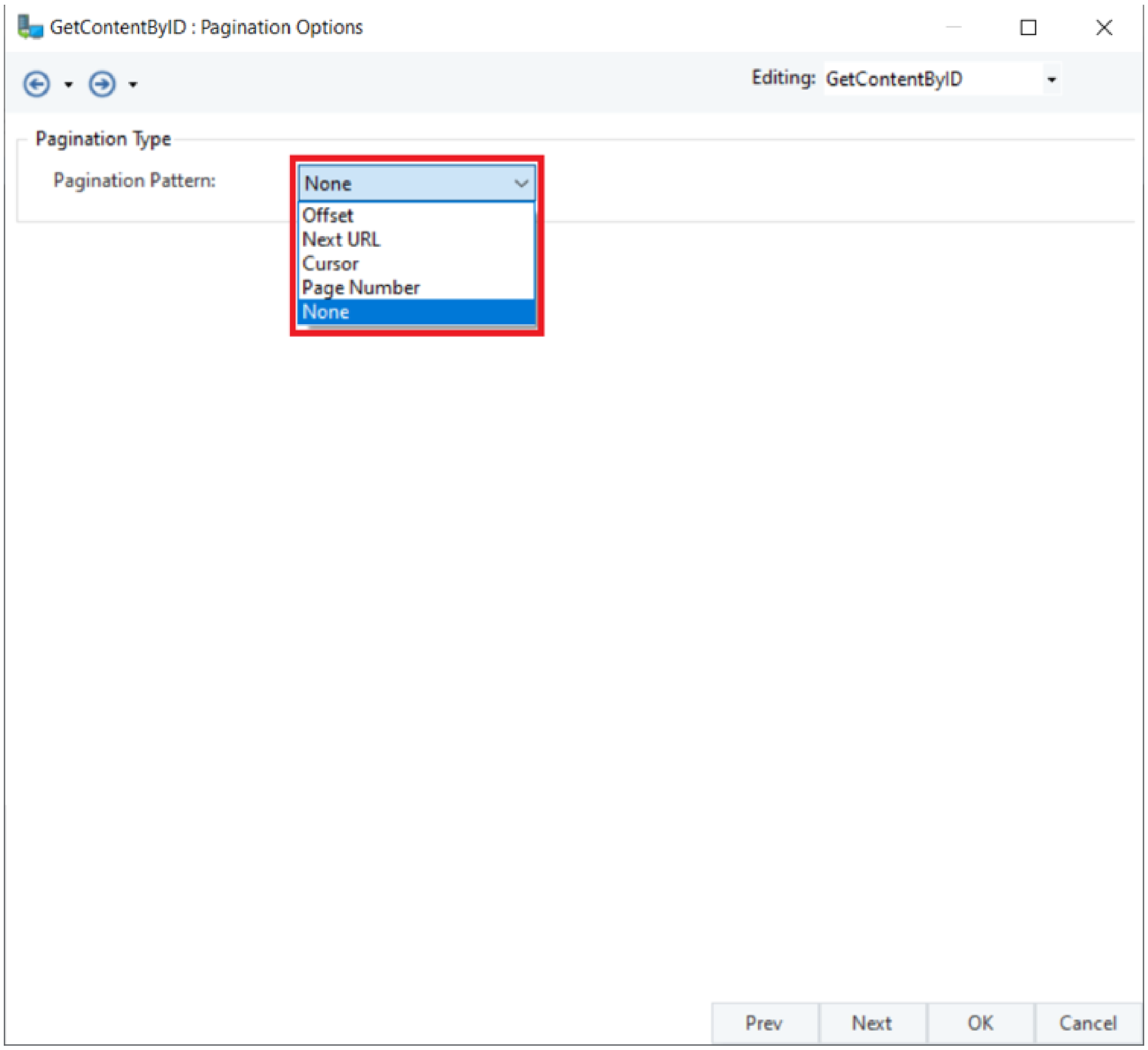
<Body.root>

root
- copyRequiresWriterPerm
- createdTime (Date)
- description (String)
- driveId (String)
- explicitlyTrashed (Boole
- fileExtension (String)
- folderColorRgb (String)
- fullFileExtension (String)
- hasAugmentedPermissio
- hasThumbnail (Boolean)
- headRevisionId (String)
- iconLink (String)
- id (String)
- isAppAuthorized (Boole
- kind (String)
- md5Checksum (String)
- mimeType (String)
- modifiedByMe (Boolean
- modifiedByMeTime (Dat

	Name	Data Type	Default Value
01	copyRequiresWriterPermission	Boolean	
02	createdTime	Date	
03	description	String	
04	driveId	String	
05	explicitlyTrashed	Boolean	
06	fileExtension	String	
07	folderColorRgb	String	
08	fullFileExtension	String	
09	hasAugmentedPermissions	Boolean	
10	hasThumbnail	Boolean	
11	headRevisionId	String	
12	iconLink	String	
13	id	String	
14	isAppAuthorized	Boolean	
15	kind	String	
16	md5Checksum	String	
17	mimeType	String	
18	modifiedByMe	Boolean	
19	modifiedByMeTime	Date	

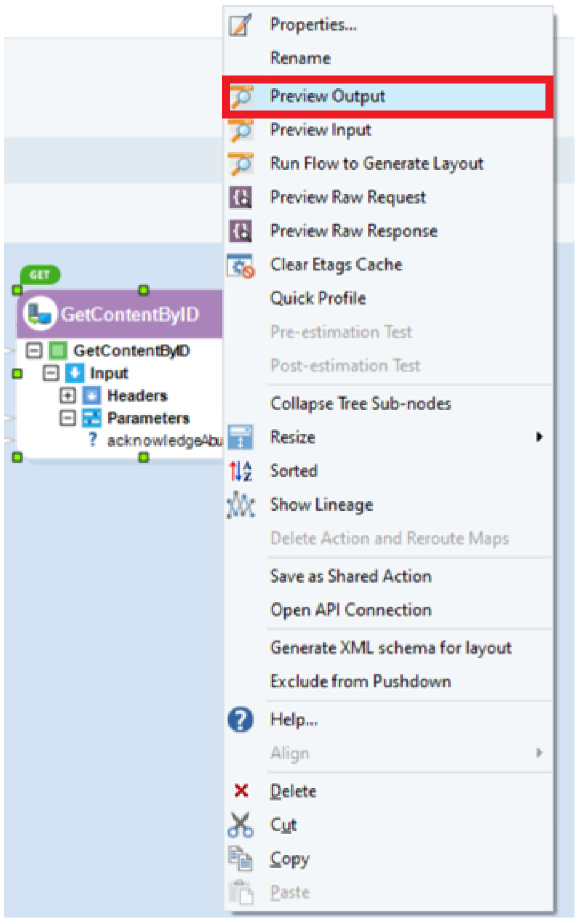
4. Click *Next* to proceed to the *Pagination Options* screen.

For our use case, we have selected *None*.

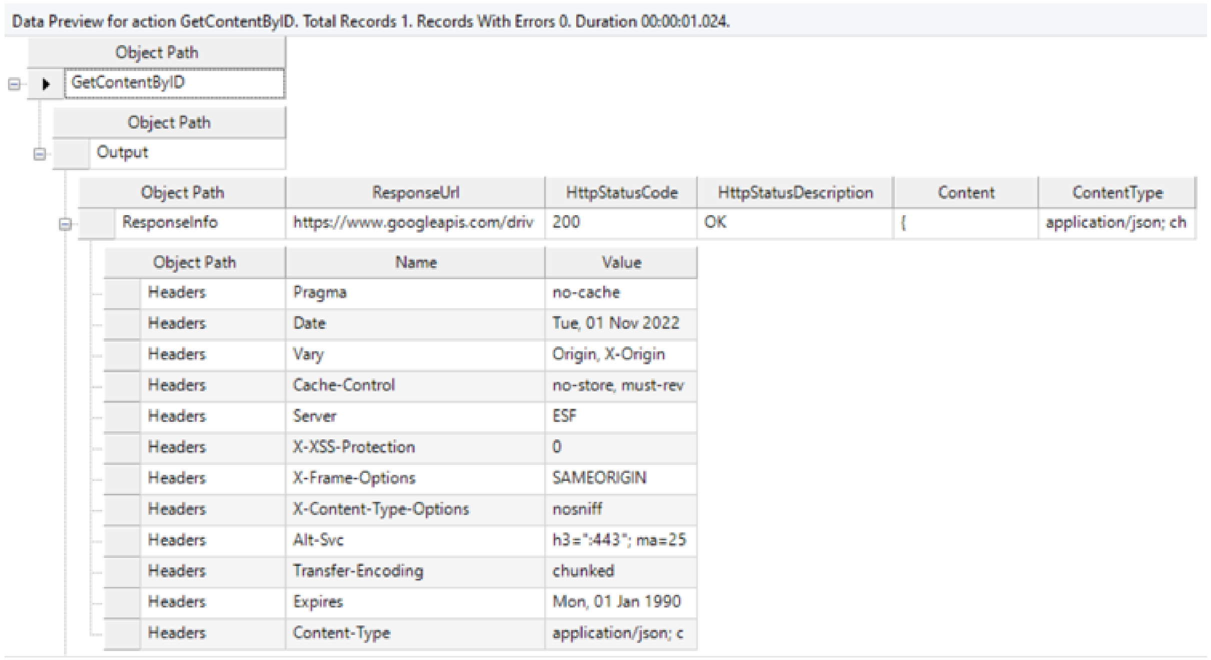


5. Click *OK*.

6. You can preview the data by right-clicking on the object and selecting *Preview Output* from the context menu.



As seen below, the GET request that was made, has fetched data according to the user application.


Data Preview for action GetContentByID. Total Records 1. Records With Errors 0. Duration 00:00:01.024.

Object Path
GetContentByID

Object Path
Output

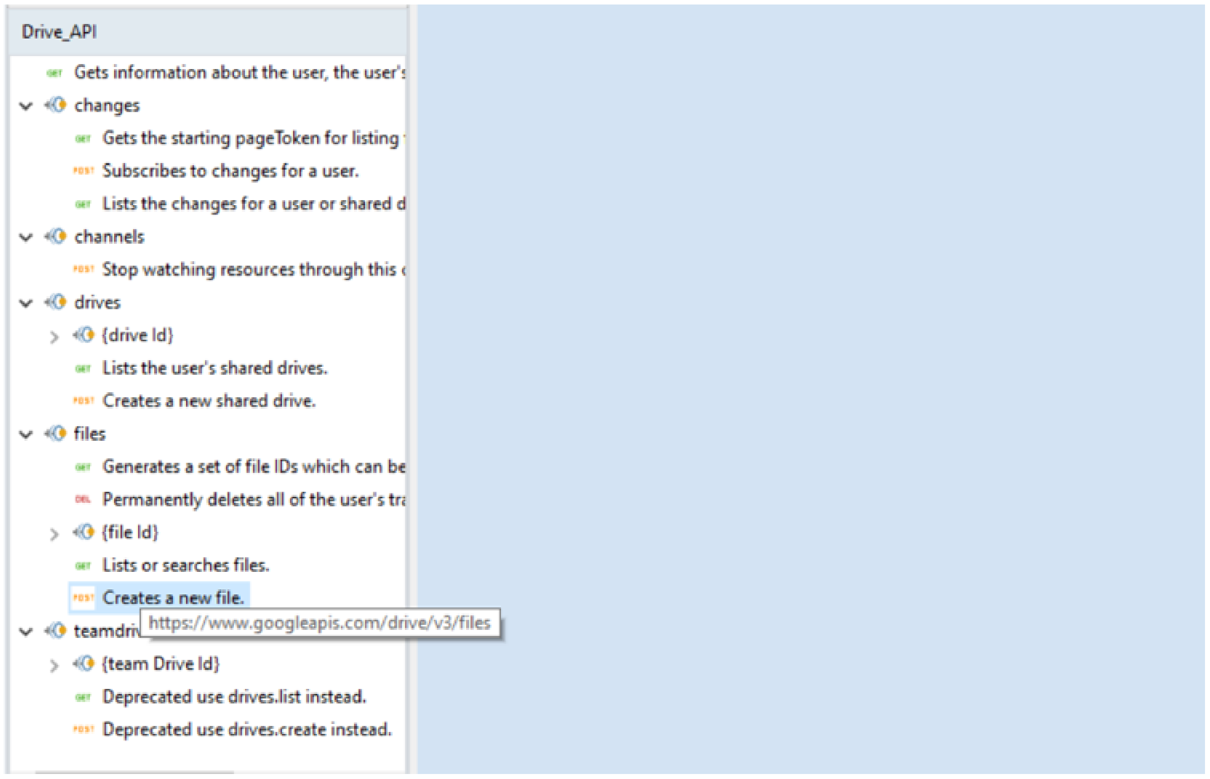
Object Path	ResponseUrl	HttpStatusCode	HttpStatusDescription	Content	ContentType
ResponseInfo	https://www.googleapis.com/driv	200	OK	{	application/json; ch

Object Path	Name	Value
Headers	Pragma	no-cache
Headers	Date	Tue, 01 Nov 2022
Headers	Vary	Origin, X-Origin
Headers	Cache-Control	no-store, must-rev
Headers	Server	ESF
Headers	X-XSS-Protection	0
Headers	X-Frame-Options	SAMEORIGIN
Headers	X-Content-Type-Options	nosniff
Headers	Alt-Svc	h3=":443"; ma=25
Headers	Transfer-Encoding	chunked
Headers	Expires	Mon, 01 Jan 1990
Headers	Content-Type	application/json; c

11.6.2 Making a POST Request

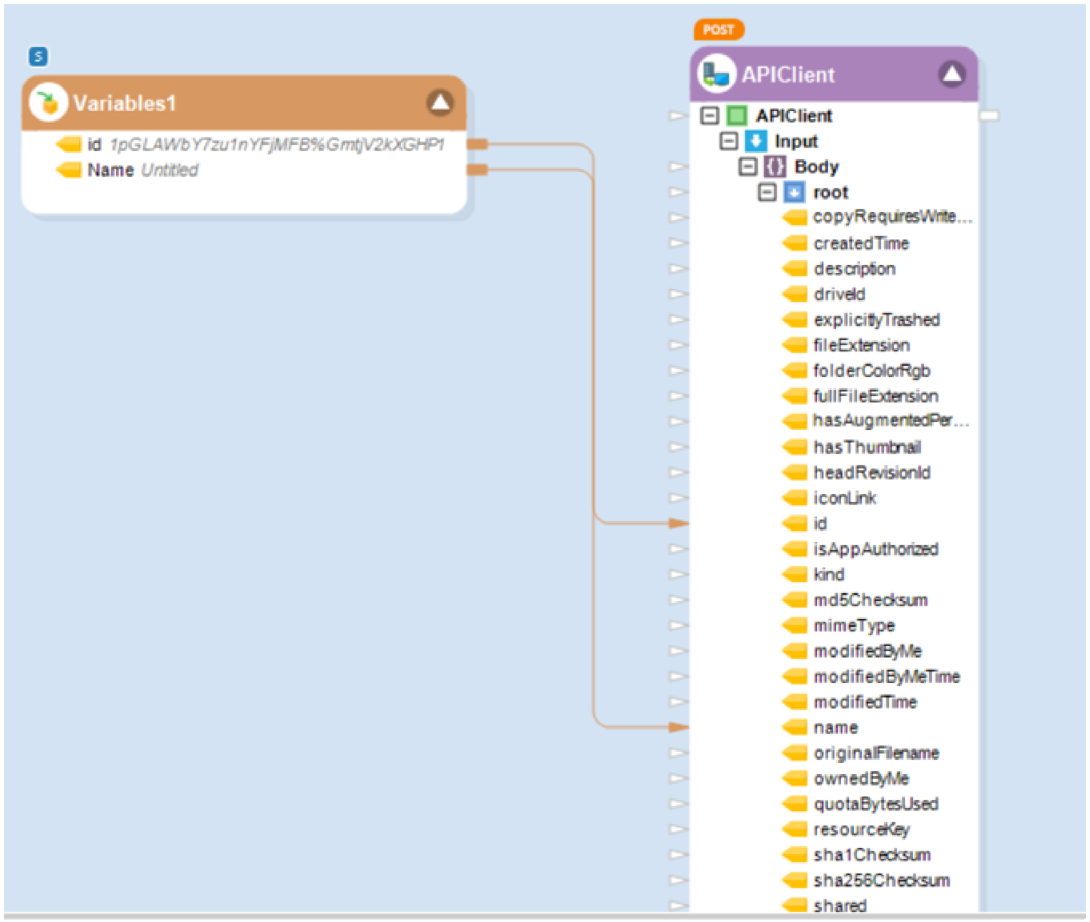
Now, let’s try creating a new file.

1. Drag-and-drop the POST method as an *API Client* object and open its *properties*.

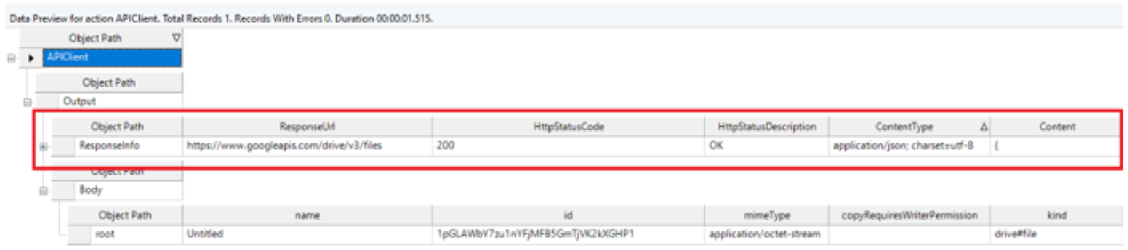


Drive_API
GET Gets information about the user, the user's
changes
GET Gets the starting pageToken for listing
POST Subscribes to changes for a user.
GET Lists the changes for a user or shared d
channels
POST Stop watching resources through this
drives
{drive Id}
GET Lists the user's shared drives.
POST Creates a new shared drive.
files
GET Generates a set of file IDs which can be
DEL Permanently deletes all of the user's tra
{file Id}
GET Lists or searches files.
POST Creates a new file.
https://www.googleapis.com/drive/v3/files
teamdri
{team Drive Id}
GET Deprecated use drives.list instead.
POST Deprecated use drives.create instead.

We will pass the required parameters to the *POST request* object using a *Variables* object.



2. Now, right-click on the *API Client* object and select *Preview Output*.


Data Preview for action APIClient. Total Records 1. Records With Errors 0. Duration 00:00:01.515.

Object Path
APIClient

Object Path
Output

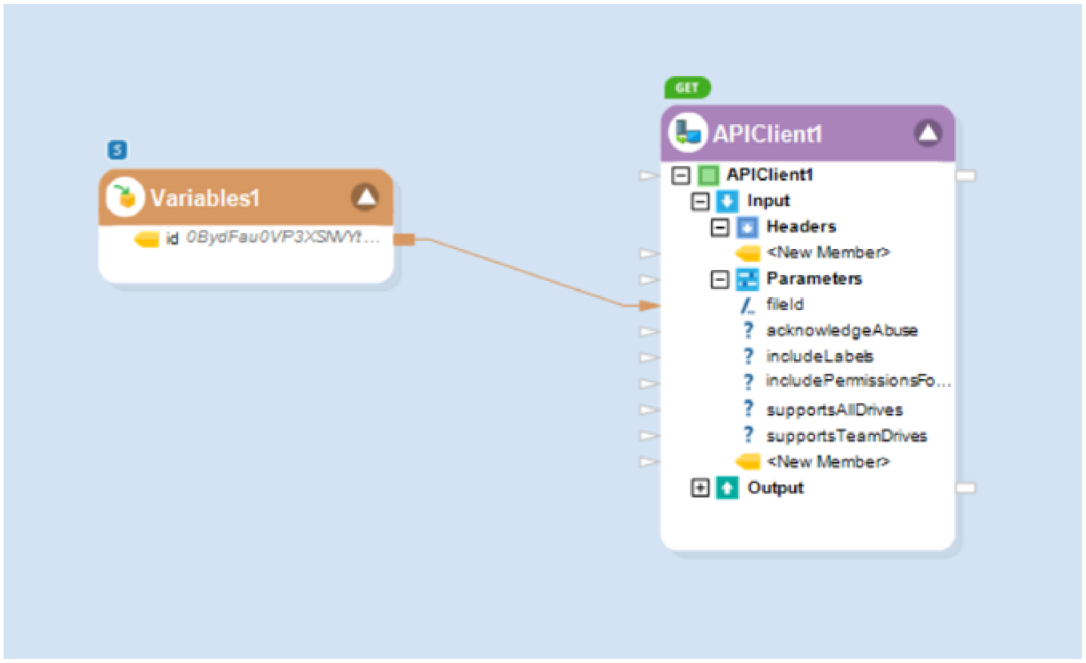
Object Path	ResponseUrl	HttpStatusCode	HttpStatusDescription	ContentType	Content
ResponseInfo	https://www.googleapis.com/drive/v3/files	200	OK	application/json; charset=utf-8	{

Object Path
Body

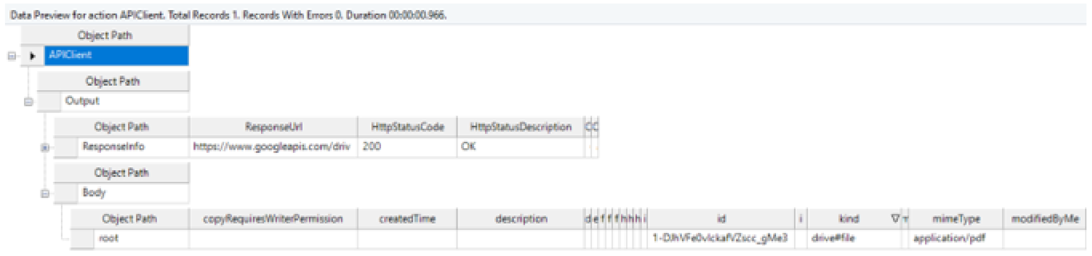
Object Path	name	id	mimeType	copyRequiresWriterPermission	kind
root	Untitled	1pGLAWbY7zu1nYFjMFB5GmTjVK2kXGHP1	application/octet-stream		drive#file

You can see that the *HTTPStatusCode* is “200”, which means that the API has successfully carried out the action requested by the client.

Let’s verify it by making a GET request for the same *FileId* that we had posted earlier.



You can see that a GET request for *FileID* has returned the same information that we had posted.



Data Preview for action APIClient. Total Records 1. Records With Errors 0. Duration 00:00:00.966.

Object Path
APIClient

Object Path
Output

Object Path	ResponseUrl	HttpStatusCode	HttpStatusDescription	CC
ResponseInfo	https://www.googleapis.com/driv	200	OK	

Object Path
Body

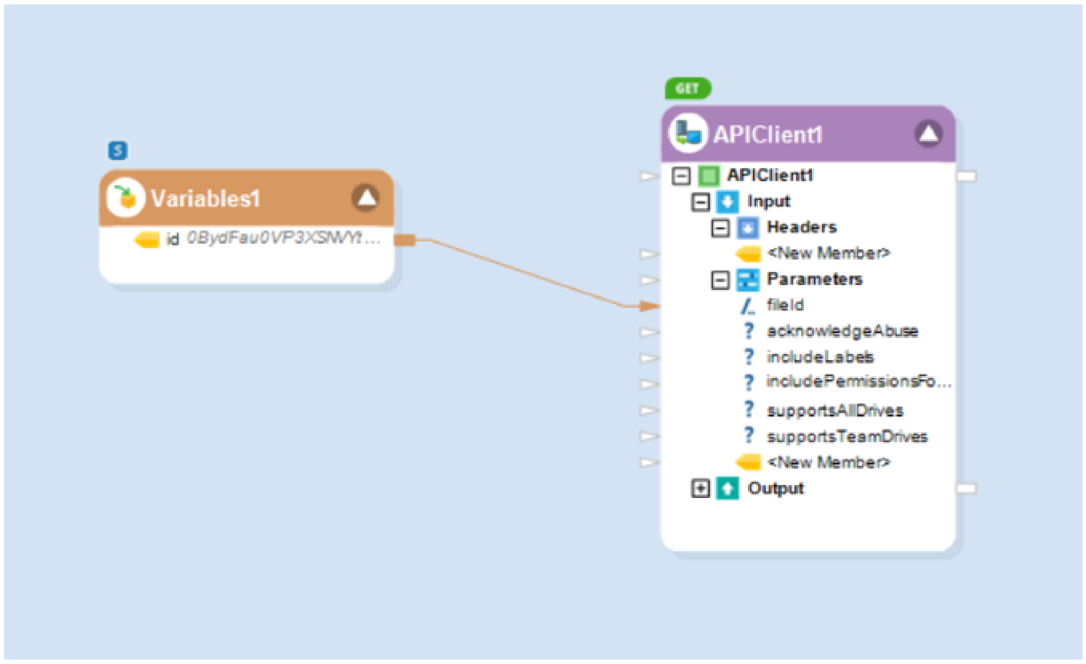
Object Path	copyRequiresWriterPermission	createdTime	description	d	e	f	f	f	h	h	h	i	id	i	kind	∇	mimeType	modifiedByMe
root													1-DJhVFe0vlckafVZscc_gMe3		drive#file		application/pdf	

11.6.3 Making a DELETE Request

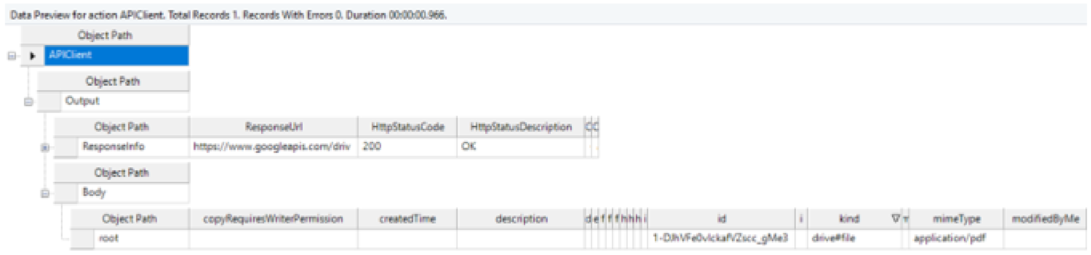
Now, let's try making a DELETE request.

1. For this, we will first make a GET request to check whether that file exists in the records before we try to delete this record.

We will pass a *fileId* using a *ConstantValue* object.



2. Right-click on the *API Client* object and select *Preview Output*.



Data Preview for action APIClient. Total Records 1. Records With Errors 0. Duration 00:00:00.966.

Object Path
APIClient

Object Path
Output

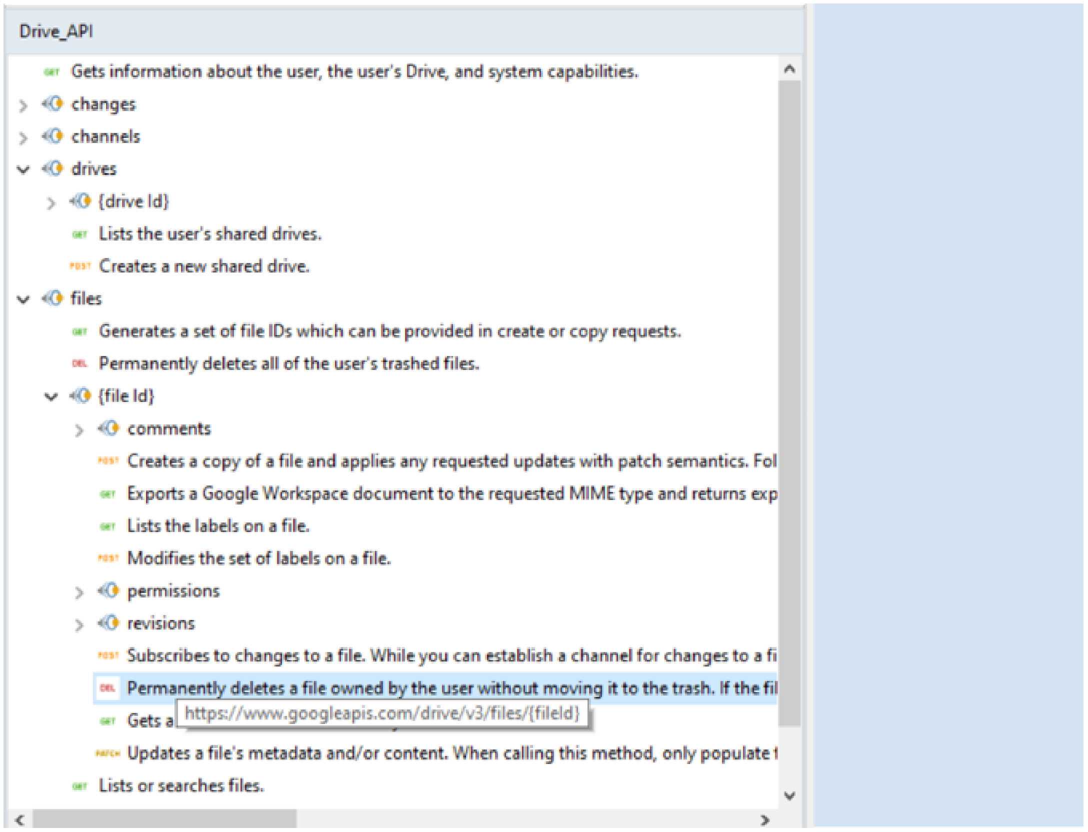
Object Path	ResponseUrl	HttpStatusCode	HttpStatusDescription
ResponseInfo	https://www.googleapis.com/driv	200	OK

Object Path
Body

Object Path	copyRequiresWriterPermission	createdTime	description	id	kind	mimeType	modifiedByMe
root				1-DJhVFe0vIckafVZscc_gMe3	drive#file	application/pdf	

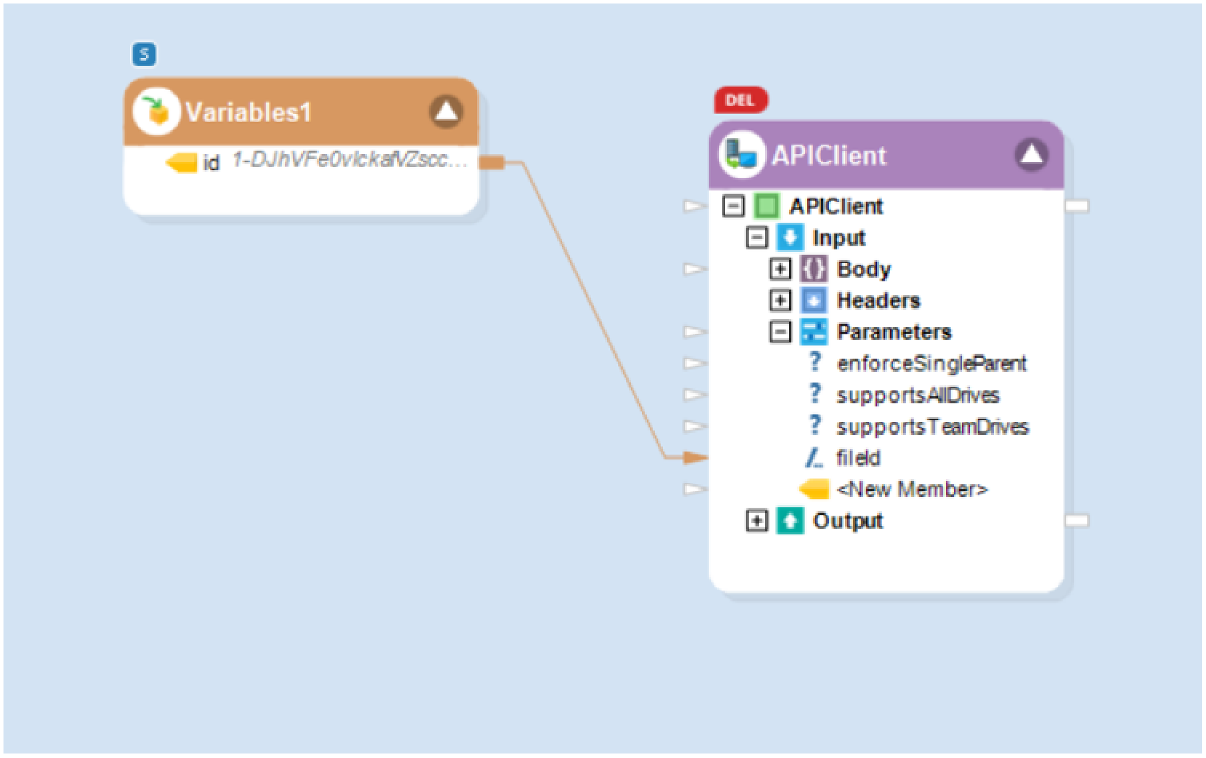
It has fetched the details of the file with the *fileId* and the status shows that the field is available.

To delete this file record, we will drag and drop another DELETE *API Client* object onto the flow and configure its *Properties* according to the DELETE method.

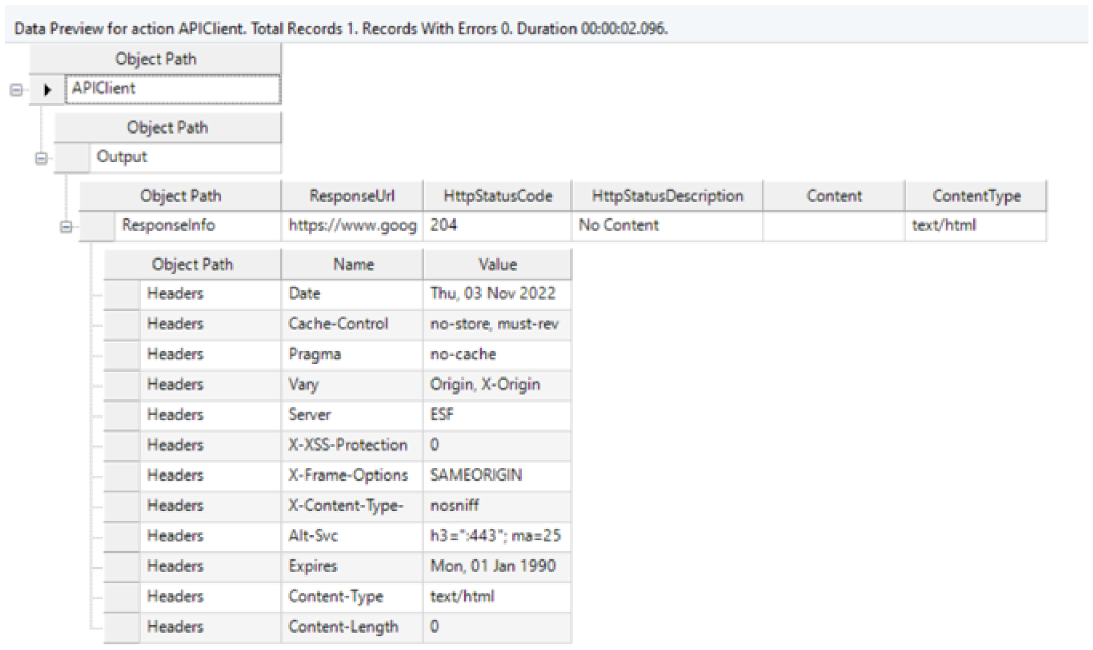


Drive_API
- GET Gets information about the user, the user's Drive, and system capabilities.
- changes
- channels
- drives
  - {drive Id}
  - GET Lists the user's shared drives.
  - POST Creates a new shared drive.
- files
  - GET Generates a set of file IDs which can be provided in create or copy requests.
  - DEL Permanently deletes all of the user's trashed files.
  - {file Id}
    - comments
    - POST Creates a copy of a file and applies any requested updates with patch semantics. Fol
    - GET Exports a Google Workspace document to the requested MIME type and returns exp
    - GET Lists the labels on a file.
    - POST Modifies the set of labels on a file.
    - permissions
    - revisions
    - POST Subscribes to changes to a file. While you can establish a channel for changes to a fi
    - DEL Permanently deletes a file owned by the user without moving it to the trash. If the fil
    - GET Gets a https://www.googleapis.com/drive/v3/files/{fileId}
    - PATCH Updates a file's metadata and/or content. When calling this method, only populate t
  - GET Lists or searches files.

3. Pass the *fileId* to the DELETE *request* object using a *ConstantValue* object.



4. Right-click on the *API Client* object and select *Preview Output*. You can see that it has returned *HTTPStatusCode*, “204”, which indicates successful execution.



Data Preview for action APIClient. Total Records 1. Records With Errors 0. Duration 00:00:02.096.

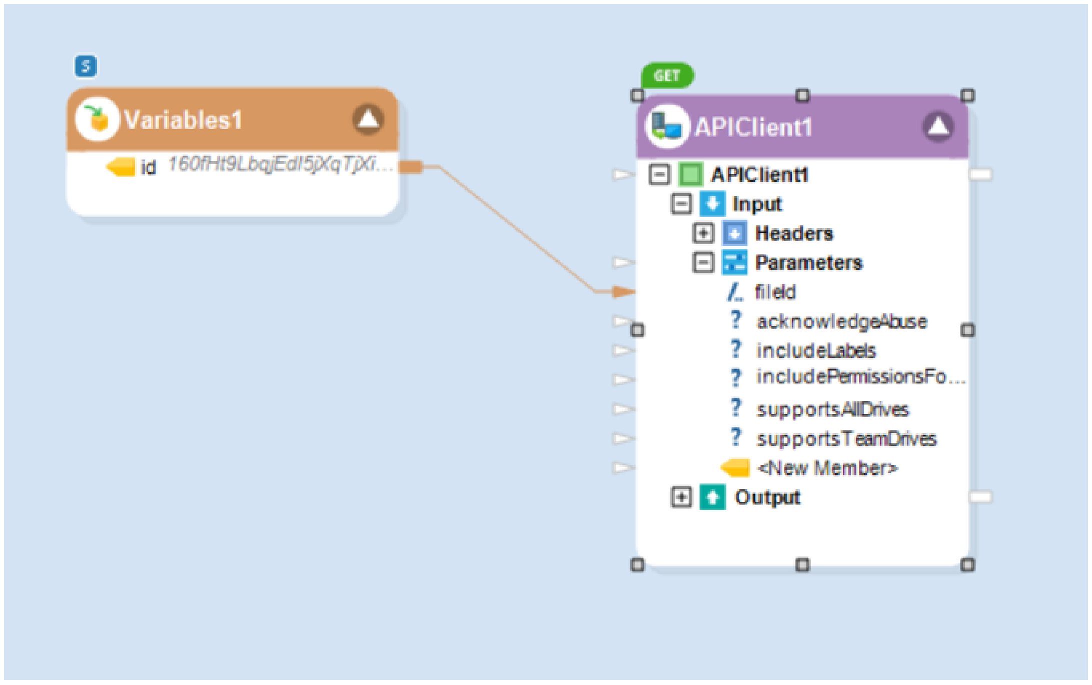
Object Path
APIClient

Object Path
Output

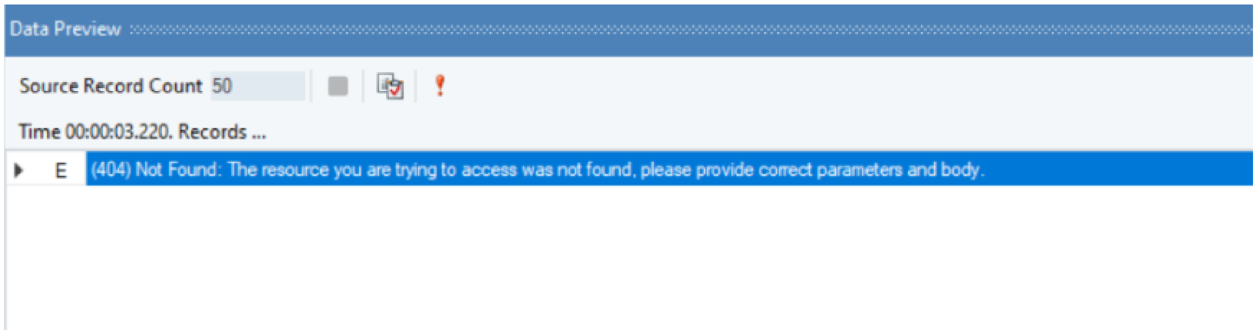
Object Path	ResponseUrl	HttpStatusCode	HttpStatusDescription	Content	ContentType
ResponseInfo	https://www.goog	204	No Content		text/html

Object Path	Name	Value
Headers	Date	Thu, 03 Nov 2022
Headers	Cache-Control	no-store, must-rev
Headers	Pragma	no-cache
Headers	Vary	Origin, X-Origin
Headers	Server	ESF
Headers	X-XSS-Protection	0
Headers	X-Frame-Options	SAMEORIGIN
Headers	X-Content-Type-	nosniff
Headers	Alt-Svc	h3=":443"; ma=25
Headers	Expires	Mon, 01 Jan 1990
Headers	Content-Type	text/html
Headers	Content-Length	0

Let’s verify it by making a GET request again, and check if the fileId, has been deleted.



5. Right-click on the *API Client* object and select *Preview Output*.

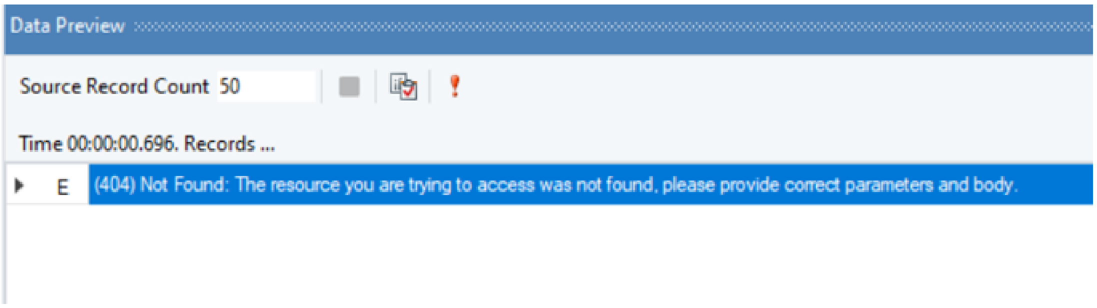

Data Preview

Source Record Count 50

Time 00:00:03.220. Records ...

E (404) Not Found: The resource you are trying to access was not found, please provide correct parameters and body.

You can see that Centerprise has returned error 404 which means that there is no pet found with *PetId*, “5”, and the pet record has been successfully deleted from *petstore* API.


Data Preview

Source Record Count 50

Time 00:00:00.696. Records ...

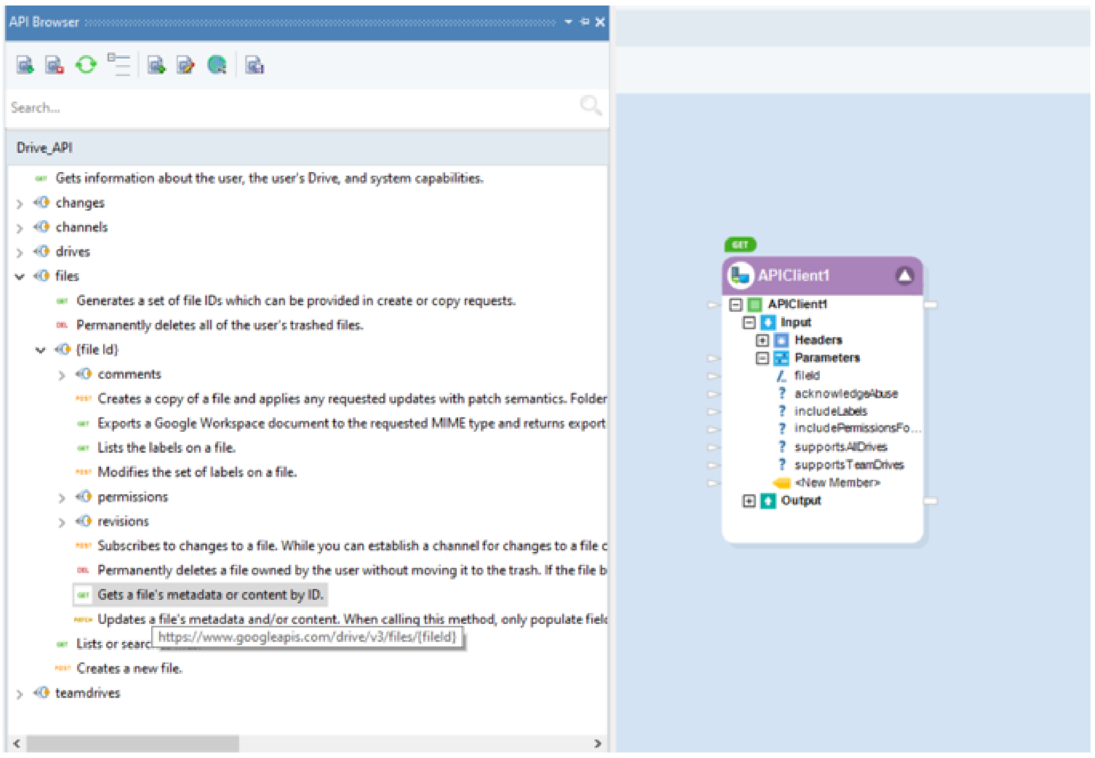
E (404) Not Found: The resource you are trying to access was not found, please provide correct parameters and body.

You can see that API Management has returned error 404 which means that there is no fileId found, and the file record has been successfully deleted from *Google Drive* API.

11.6.4 Making a PUT Request

Let us now look at the PUT *HTTP Method*.

1. Drag and drop the *GET* endpoint from the API Browser onto the dataflow.


API Browser

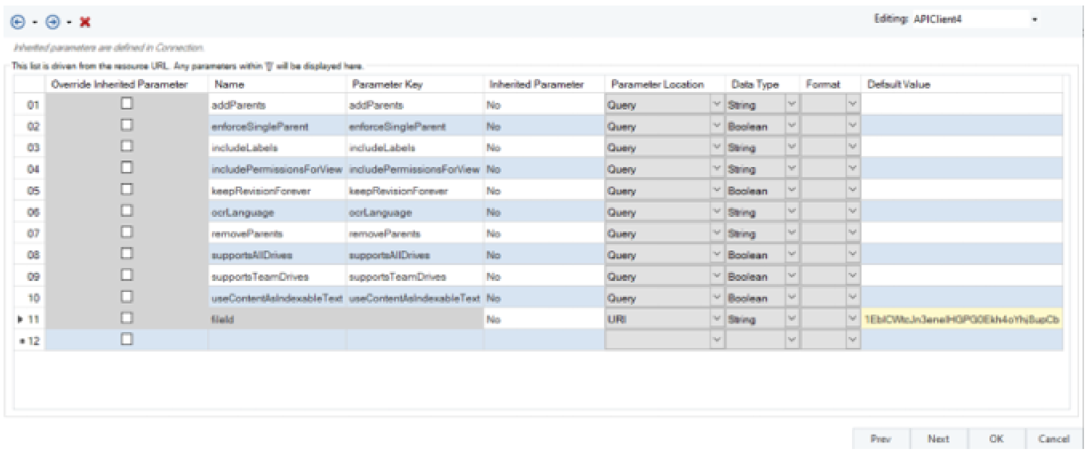
Search...

Drive_API
- Gets information about the user, the user's Drive, and system capabilities.
- changes
- channels
- drives
- files
  - Generates a set of file IDs which can be provided in create or copy requests.
  - Permanently deletes all of the user's trashed files.
  - {file Id}
    - comments
    - Creates a copy of a file and applies any requested updates with patch semantics. Folder
    - Exports a Google Workspace document to the requested MIME type and returns export
    - Lists the labels on a file.
    - Modifies the set of labels on a file.
    - permissions
    - revisions
    - Subscribes to changes to a file. While you can establish a channel for changes to a file c
    - Permanently deletes a file owned by the user without moving it to the trash. If the file b
    - Gets a file's metadata or content by ID.
    - Updates a file's metadata and/or content. When calling this method, only populate fiel
  - Lists or searc https://www.googleapis.com/drive/v3/files/{fileId}
  - Creates a new file.
- teamdrives

2. Right-click on the object and select *Properties* from the context menu.

3. Click *Next,* and the *Parameters* screen will appear.

For this use case, we will update the file with a fileId. Let's define this ID in the *Default Value* field.

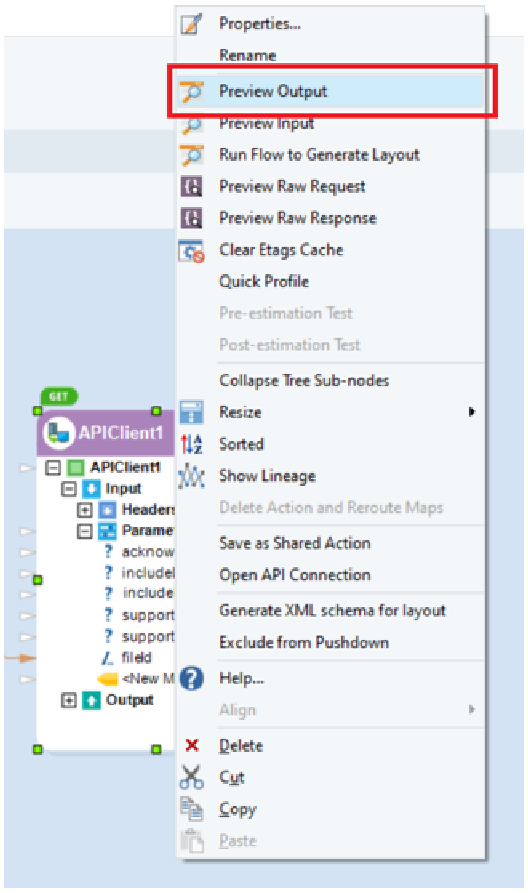

Editing: APIClient4

Inherited parameters are defined in Connection.

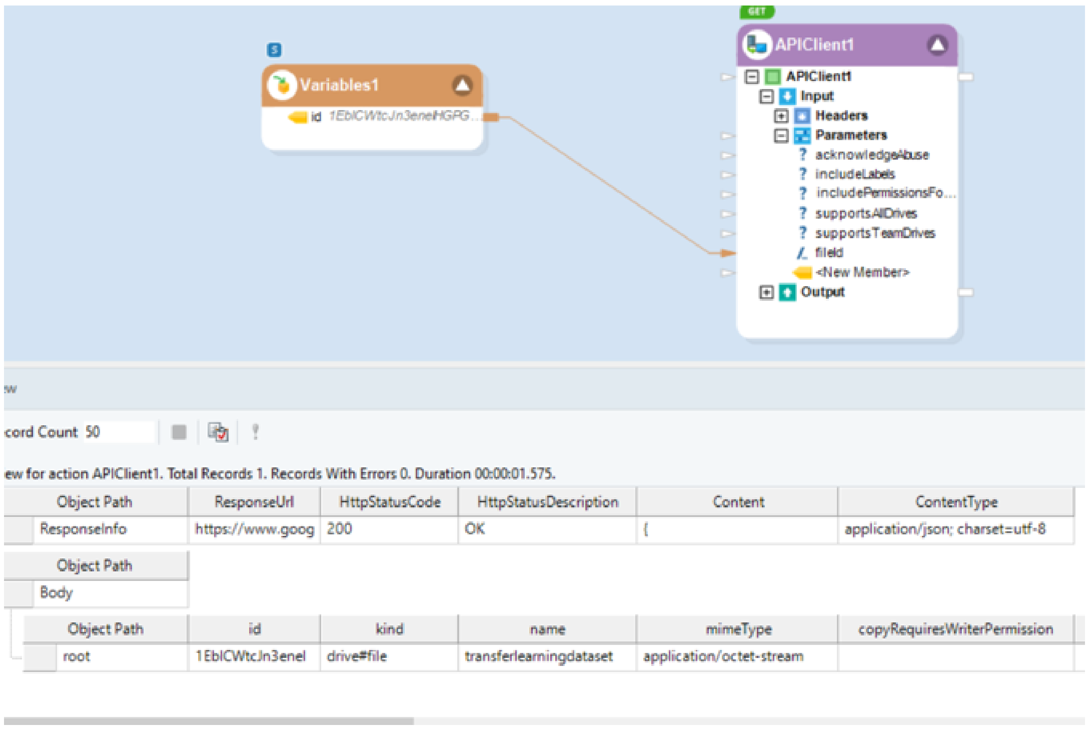
This list is driven from the resource URL. Any parameters within '{}' will be displayed here.

	Override Inherited Parameter	Name	Parameter Key	Inherited Parameter	Parameter Location	Data Type	Format	Default Value
01	☐	addParents	addParents	No	Query	String		
02	☐	enforceSingleParent	enforceSingleParent	No	Query	Boolean		
03	☐	includeLabels	includeLabels	No	Query	String		
04	☐	includePermissionsForView	includePermissionsForView	No	Query	String		
05	☐	keepRevisionForever	keepRevisionForever	No	Query	Boolean		
06	☐	ocrLanguage	ocrLanguage	No	Query	String		
07	☐	removeParents	removeParents	No	Query	String		
08	☐	supportsAllDrives	supportsAllDrives	No	Query	Boolean		
09	☐	supportsTeamDrives	supportsTeamDrives	No	Query	Boolean		
10	☐	useContentAsIndexableText	useContentAsIndexableText	No	Query	Boolean		
11	☐	fileId		No	URI	String		1EbICWtcJn3enelHGPG0Ekh4oYhjSupCb
12	☐							

4. Click *OK* and preview the output by right-clicking on the object and selecting *Preview Output*



As you can see in the preview screen below, the *GET* method has retrieved the file Metadata by ID.



cord Count 50

ew for action APIClient1. Total Records 1. Records With Errors 0. Duration 00:00:01.575.

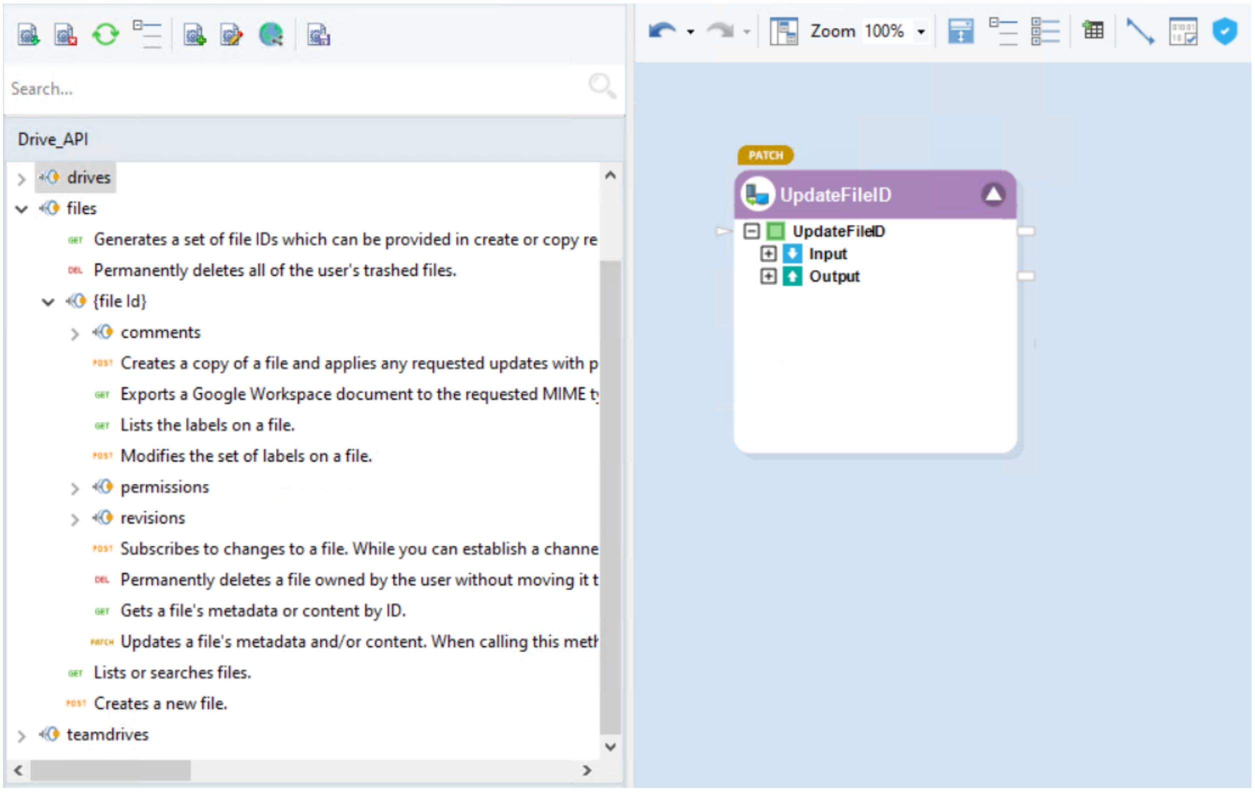
Object Path	ResponseUrl	HttpStatusCode	HttpStatusDescription	Content	ContentType
ResponseInfo	https://www.goog	200	OK	{	application/json; charset=utf-8

Object Path
Body

Object Path	id	kind	name	mimeType	copyRequiresWriterPermission
root	1EbICWtcJn3enel	drive#file	transferlearningdataset	application/octet-stream	

5. Now, drag and drop the relevant endpoint from the API Browser onto the dataflow.

For our use case, we will be using this *Patch* object for the *PUT* method so we can update the ID.


Search...
Drive_API
drives
files
Generates a set of file IDs which can be provided in create or copy re
Permanently deletes all of the user's trashed files.
{file Id}
comments
Creates a copy of a file and applies any requested updates with p
Exports a Google Workspace document to the requested MIME t
Lists the labels on a file.
Modifies the set of labels on a file.
permissions
revisions
Subscribes to changes to a file. While you can establish a channe
Permanently deletes a file owned by the user without moving it t
Gets a file's metadata or content by ID.
Updates a file's metadata and/or content. When calling this meth
Lists or searches files.
Creates a new file.
teamdrives
Zoom 100%
PATCH
UpdateFileID
UpdateFileID
Input
Output

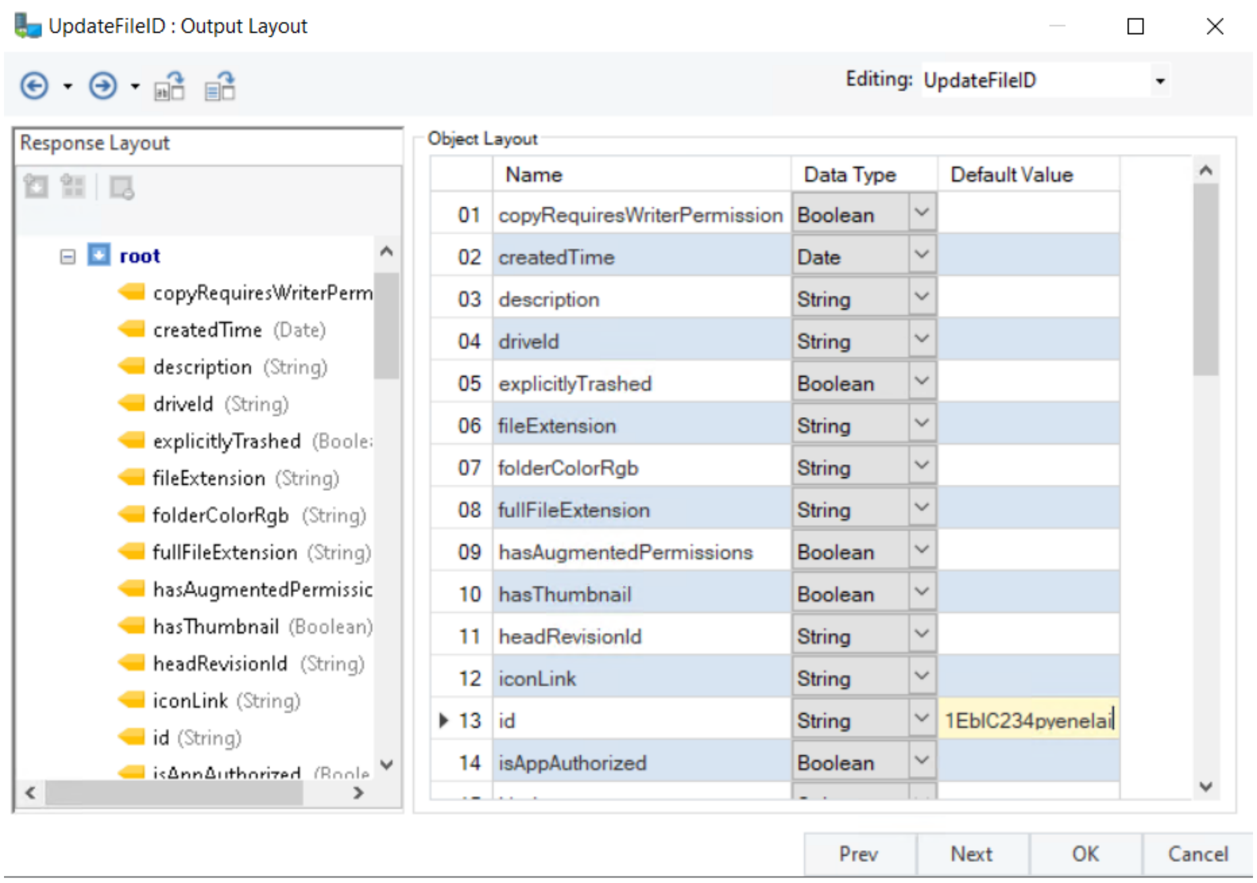
Right-click on the object and select *Properties* from the context menu.

Our *Shared Connection* has already been defined. The *HTTP Method* is *Put*, and the *Resource* to update is a file.

6. Click *Next*, and you will be led to the *Output Layout* screen

We have defined the FileId here that we wish the resource to be updated to,

If required, an output can be generated by running a request using the available option.


UpdateFileID : Output Layout

Editing: UpdateFileID

Response Layout

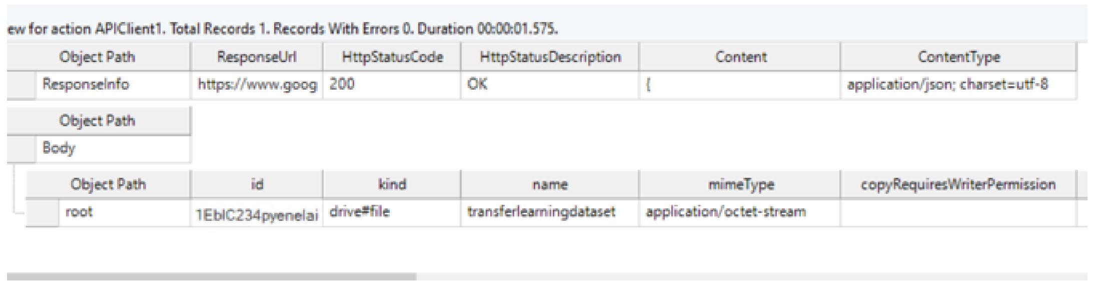
- root
  - copyRequiresWriterPerm
  - createdTime (Date)
  - description (String)
  - driveId (String)
  - explicitlyTrashed (Boole
  - fileExtension (String)
  - folderColorRgb (String)
  - fullFileExtension (String)
  - hasAugmentedPermissic
  - hasThumbnail (Boolean)
  - headRevisionId (String)
  - iconLink (String)
  - id (String)
  - isAppAuthorized (Boole

Object Layout

	Name	Data Type	Default Value
01	copyRequiresWriterPermission	Boolean	
02	createdTime	Date	
03	description	String	
04	driveId	String	
05	explicitlyTrashed	Boolean	
06	fileExtension	String	
07	folderColorRgb	String	
08	fullFileExtension	String	
09	hasAugmentedPermissions	Boolean	
10	hasThumbnail	Boolean	
11	headRevisionId	String	
12	iconLink	String	
13	id	String	1EbIC234pyenelai
14	isAppAuthorized	Boolean	

9. Click *OK*, right-click on the object, and select *Preview Output*.

As you can see here, the fileId has been updated,


ew for action APIClient1. Total Records 1. Records With Errors 0. Duration 00:00:01.575.

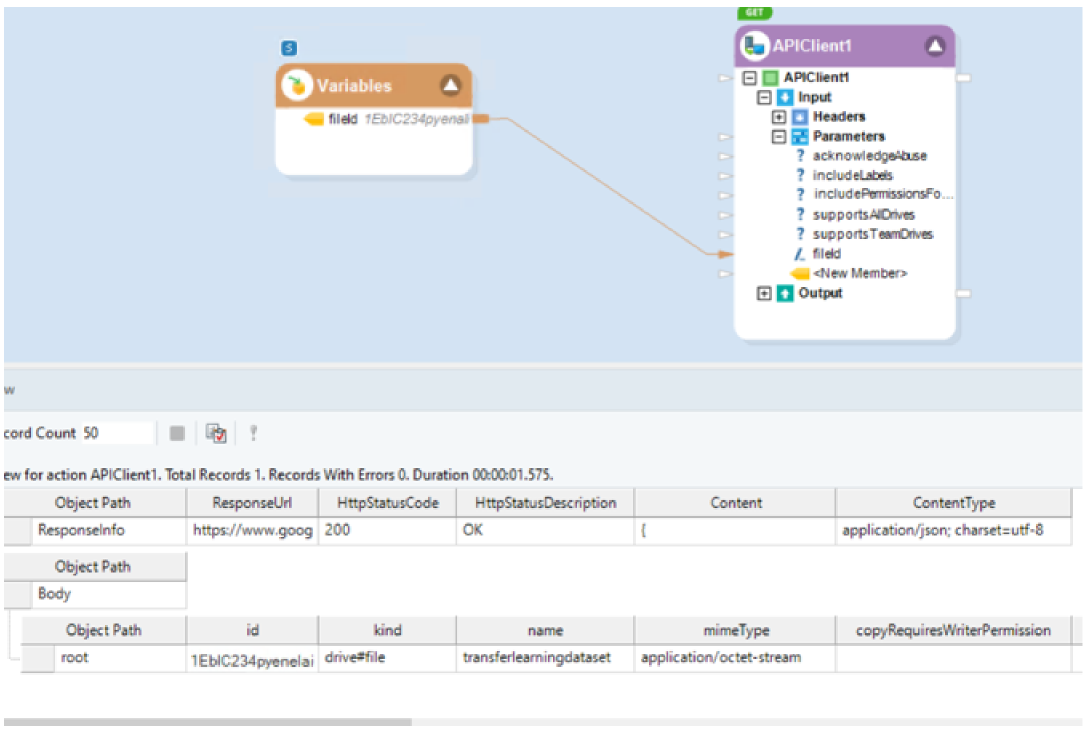
Object Path	ResponseUrl	HttpStatusCode	HttpStatusDescription	Content	ContentType
ResponseInfo	https://www.goog	200	OK	{	application/json; charset=utf-8

Object Path
Body

Object Path	id	kind	name	mimeType	copyRequiresWriterPermission
root	1EbIC234pyenelai	drive#file	transferlearningdataset	application/octet-stream	

10. We will now preview the output of the *GET* object we have configured to verify if the pet status has been updated.

As you can see, the value has been updated.



cord Count 50

ew for action APIClient1. Total Records 1. Records With Errors 0. Duration 00:00:01.575.

Object Path	ResponseUrl	HttpStatusCode	HttpStatusDescription	Content	ContentType
ResponseInfo	https://www.goog	200	OK	{	application/json; charset=utf-8

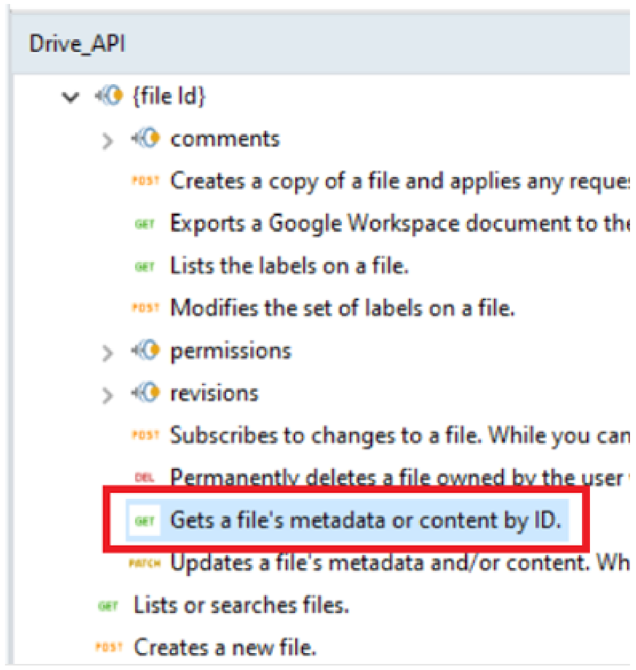
Object Path
Body

Object Path	id	kind	name	mimeType	copyRequiresWriterPermission
root	1EbIC234pyenelai	drive#file	transferlearningdataset	application/octet-stream	

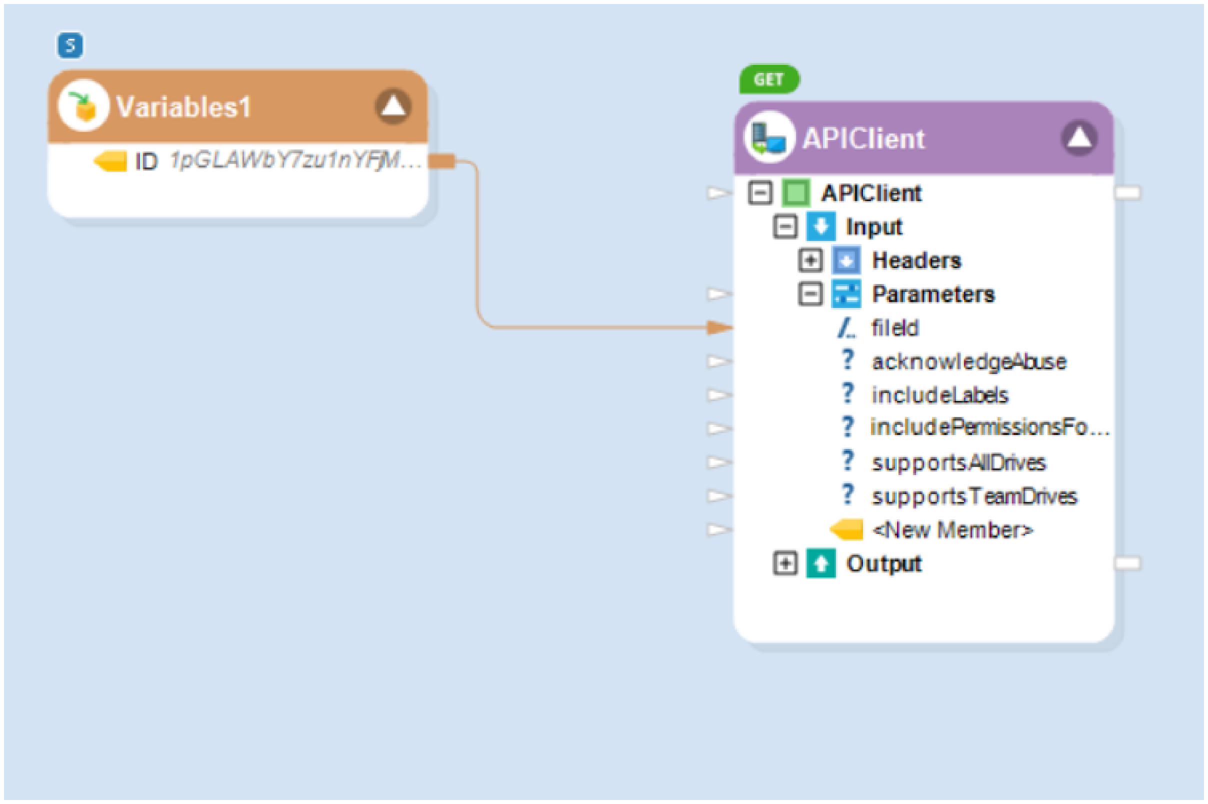
11.6.5 Making a PATCH Request

Let's make a GET request to see what information is there in the File ID where we want to update something.

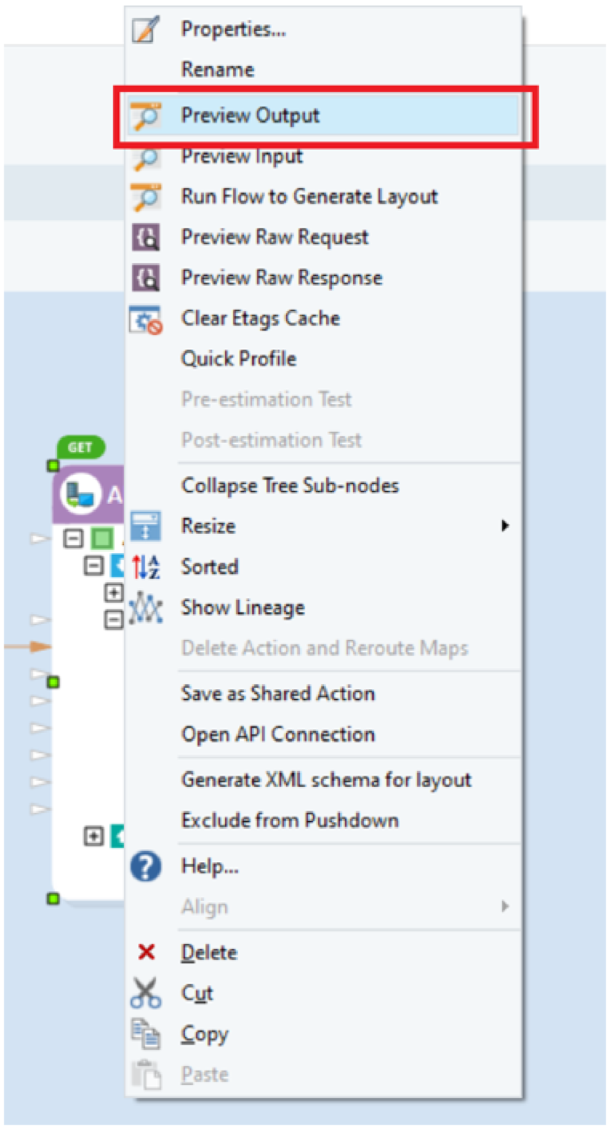
1. To make a GET request, drag-and-drop the GET *API Client* object onto the dataflow.


Drive_API
{file Id}
comments
POST Creates a copy of a file and applies any reque
GET Exports a Google Workspace document to the
GET Lists the labels on a file.
POST Modifies the set of labels on a file.
permissions
revisions
POST Subscribes to changes to a file. While you can
DEL Permanently deletes a file owned by the user
GET Gets a file's metadata or content by ID.
PATCH Updates a file's metadata and/or content. Wh
GET Lists or searches files.
POST Creates a new file.

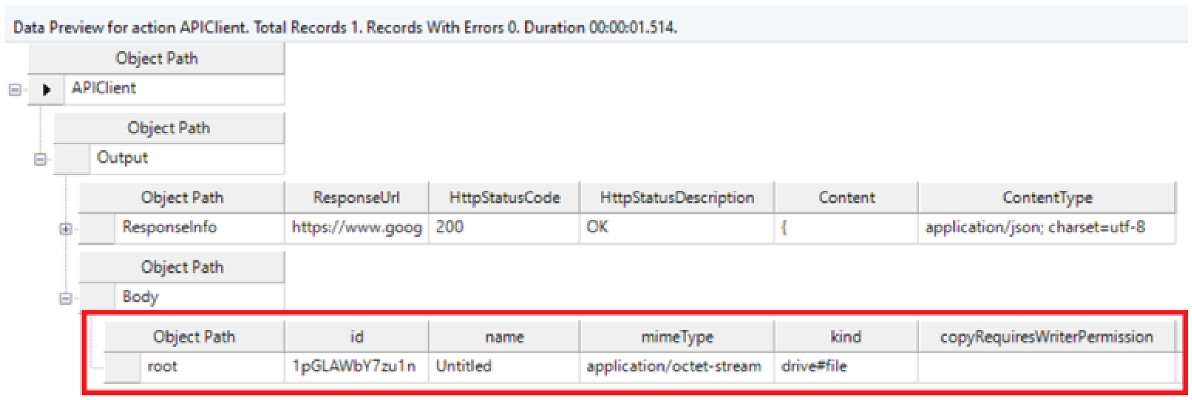
2. Pass userID '1pGLAWbY7zu1nYFjMFB5GmTjVK2kXGHP1' to the *id* under the *Parameters* node in the *API Client* object using the *Variable* transformation object.



3. Right-click the *API Client* object's header and select *Preview Output*.



Here is what the output looks like:



Data Preview for action APIClient. Total Records 1. Records With Errors 0. Duration 00:00:01.514.

Object Path
APIClient

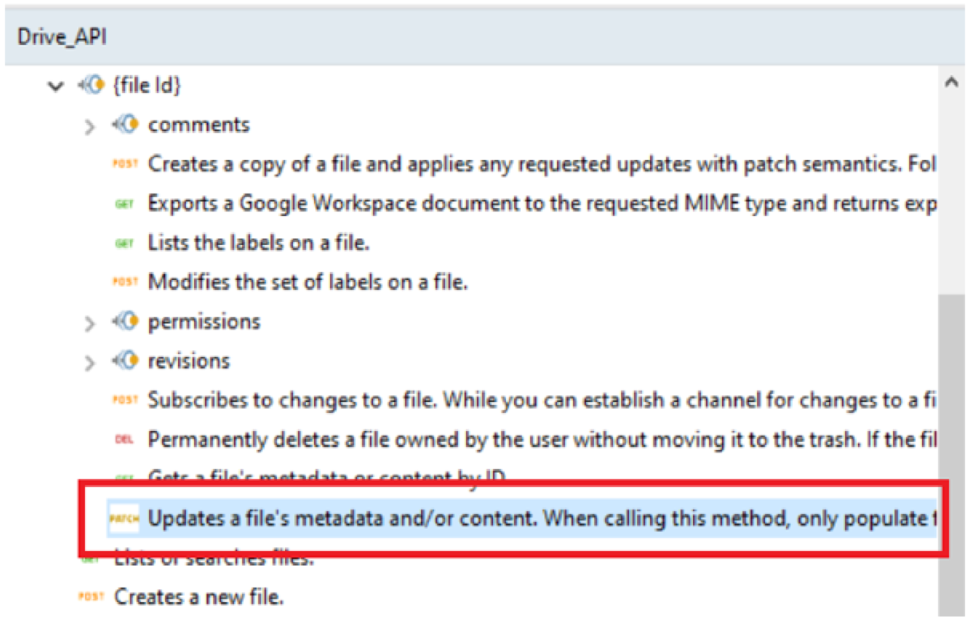
Object Path
Output

Object Path	ResponseUrl	HttpStatusCode	HttpStatusDescription	Content	ContentType
ResponseInfo	https://www.goog	200	OK	{	application/json; charset=utf-8

Object Path
Body

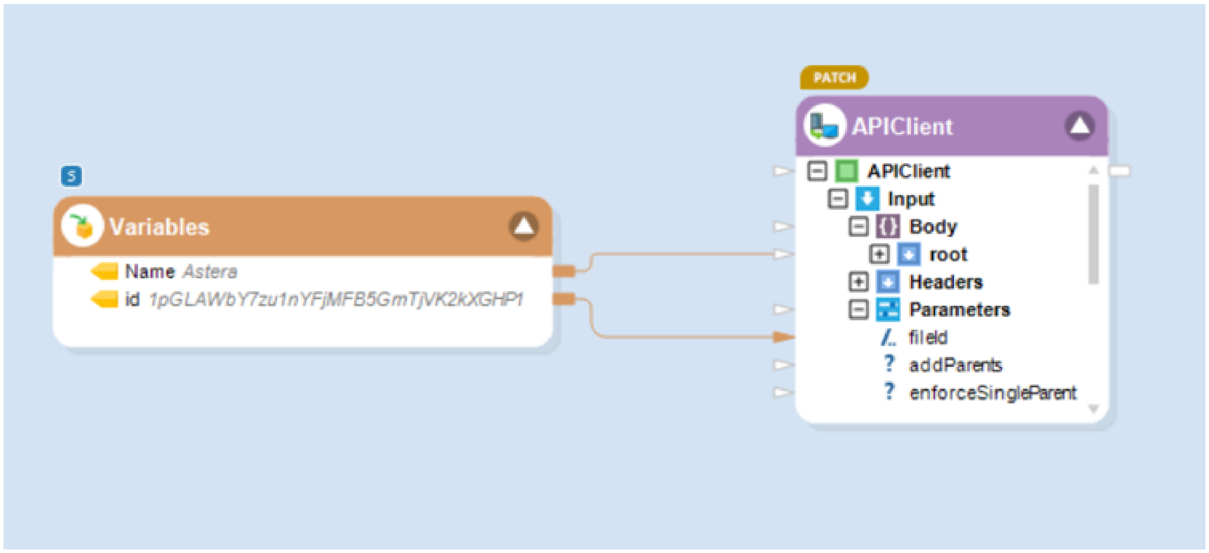
Object Path	id	name	mimeType	kind	copyRequiresWriterPermission
root	1pGLAWbY7zu1n	Untitled	application/octet-stream	drive#file	

4. Drag-and-drop the *Update a file's metadata API Client* object to use the PATCH method.

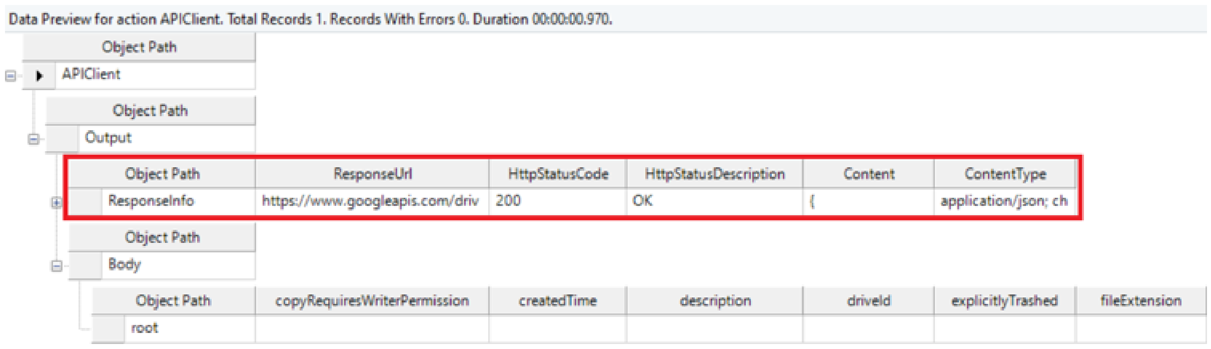

Drive_API

{file Id}
comments
POST Creates a copy of a file and applies any requested updates with patch semantics. Fol
GET Exports a Google Workspace document to the requested MIME type and returns exp
GET Lists the labels on a file.
POST Modifies the set of labels on a file.
permissions
revisions
POST Subscribes to changes to a file. While you can establish a channel for changes to a fi
DEL Permanently deletes a file owned by the user without moving it to the trash. If the fil
GET Gets a file's metadata or content by ID.
PATCH Updates a file's metadata and/or content. When calling this method, only populate f
GET Lists or searches files.
POST Creates a new file.

5. Pass fileId' 1pGLAWbY7zu1nYFjMFB5GmTjVK2kXGHP1', and *name*, "Astera", using a *Variables* resource object.



6. Right-click on the object's header, and select *Preview Output*.


Data Preview for action APIClient. Total Records 1. Records With Errors 0. Duration 00:00:00.970.

Object Path
APIClient

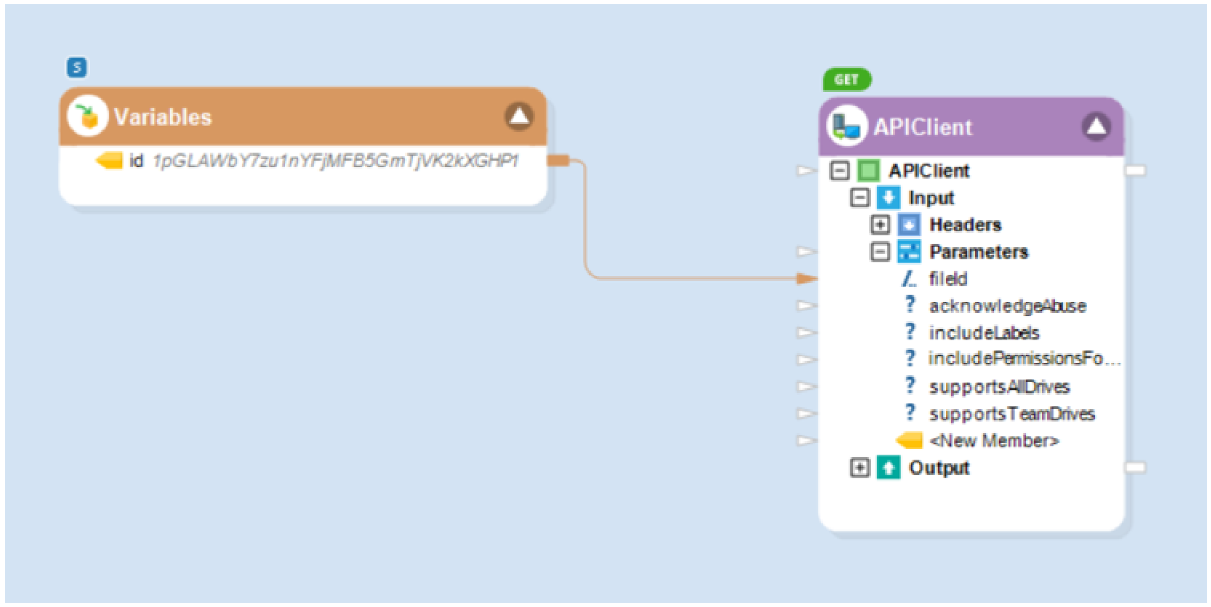
Object Path
Output

Object Path	ResponseUrl	HttpStatusCode	HttpStatusDescription	Content	ContentType
ResponseInfo	https://www.googleapis.com/driv	200	OK	{	application/json; ch

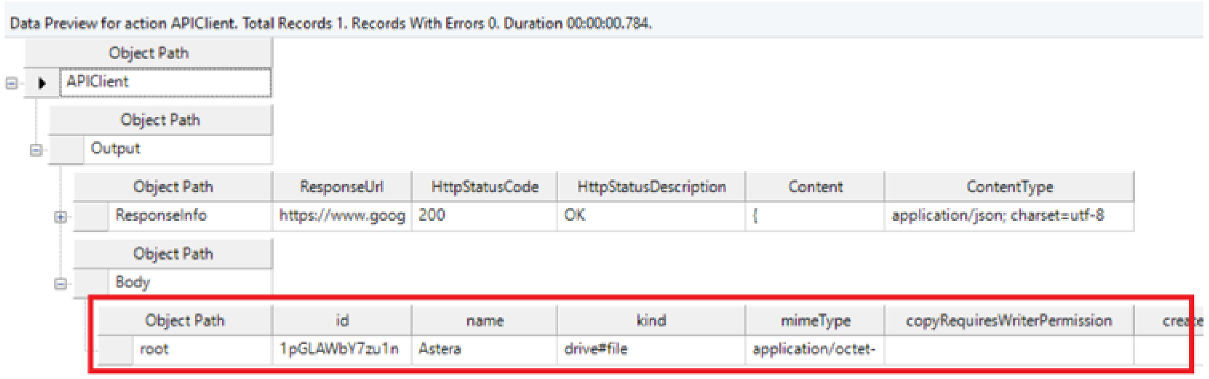
Object Path
Body

Object Path	copyRequiresWriterPermission	createdTime	description	driveId	explicitlyTrashed	fileExtension
root						

You can see that the *HTTPStatusCode* is 200, which means that the API has successfully carried out the PATCH request. Let's verify it by making a GET request for the same *fileId* which we altered.



7. Right-click the *APIClient* object's header and select *Preview Output*.


Data Preview for action APIClient. Total Records 1. Records With Errors 0. Duration 00:00:00.784.

Object Path
APIClient

Object Path
Output

Object Path	ResponseUrl	HttpStatusCode	HttpStatusDescription	Content	ContentType
ResponseInfo	https://www.goog	200	OK	{	application/json; charset=utf-8

Object Path
Body

Object Path	id	name	kind	mimeType	copyRequiresWriterPermission	create
root	1pGLAWbY7zu1n	Astera	drive#file	application/octet-		

As you can see, the request has been successfully carried out and the *email* address has been updated.

This concludes our discussion on the HTTP method operations in Astera API Management.

11.7 Pagination

Pagination refers to managing the traffic of records coming from a source. It divides the records into a discrete number of pages so that they are comprehensible for a user.

Pagination is not supported by all APIs. For those that do support it, Astera offers four types of paginations.

11.7.1 Offset

This type of pagination requires two parameters: A *Limit* and an *Offset* value to be specified by the user. A *Limit* specifies the number of records that you want to fetch in a one-page request, and an *Offset* simply tells the number of records to be skipped before selecting records.

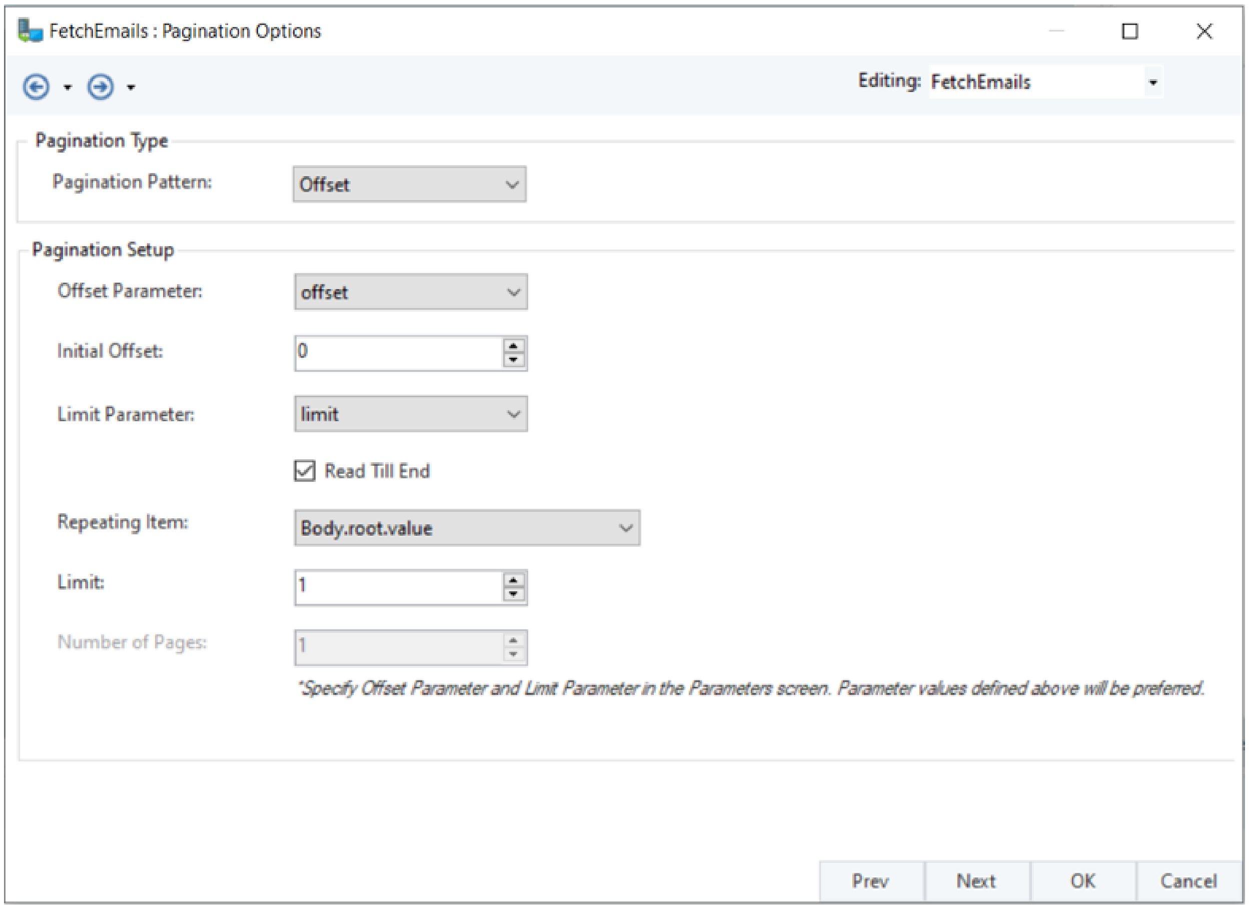
Offset Parameter: Select the offset parameter of the API that you are working with, as specified on the *Parameters* screen.

Initial Offset: The record index from which you want to start your pagination.

Limit Parameter: Select the limit parameter of the API that you are working with, as specified on the *Parameters* screen.

Limit: Number of records on a one-page request.

Number of Pages: The number of pages indicates the number of request iterations which you want to be processed. Each iterative request incrementally adds the respective offset and limit values for the next set of records page.



Pagination Type

Pagination Pattern: Offset

Pagination Setup

Offset Parameter: offset

Initial Offset: 0

Limit Parameter: limit

☑ Read Till End

Repeating Item: Body.root.value

Limit: 1

Number of Pages: 1

*Specify Offset Parameter and Limit Parameter in the Parameters screen. Parameter values defined above will be preferred.

Read till end: Check this option if you want to fetch all the records. Selecting this will disable the 'Number of pages' option and all the records will be returned as requests are sent in a loop till no more data is found.

Repeating item: This option is only enabled when you check the Read till end box. You can choose a repeating item or the collection node of the data from the output layout of the API client object. The repeating item helps the API client recognize the end of records, as whenever an empty response node is returned, the client stops sending further requests, and pagination ends.

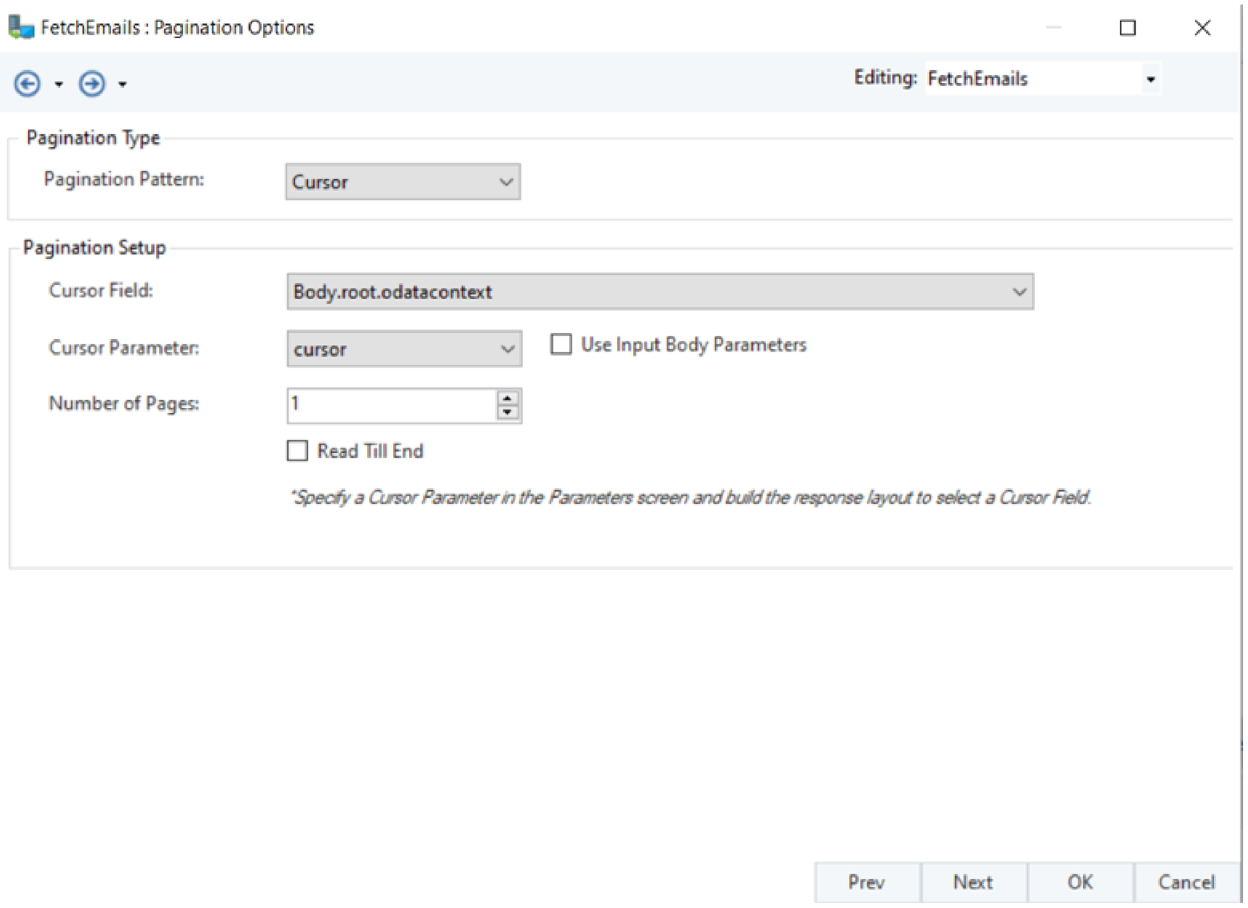
11.7.2 Cursor

This type of pagination generates a token to indicate a pointer for the next page of records. You can set a limit to the number of pages you want to process.

Cursor Field: Here, you can specify the field from the output layout which contains the cursor from the server response.

Cursor Parameter: Here, you can select the parameter to be used to send the cursor value received in the previous request of the API that you are working with, as specified on the *Parameters* screen. Alternatively, you can choose to send the cursor as an input body layout field by selecting the 'Use Input Body Parameters' checkbox.

Number of Pages: Here, you can specify the number of pages or the number of requests to be made iterating over the data set. Additionally, you can simply check the *Read till End* option if you want to fetch all records without specifying the number of pages.



Pagination Type

Pagination Pattern: Cursor

Pagination Setup

Cursor Field: Body.root.odatacontext

Cursor Parameter: cursor

☐ Use Input Body Parameters

Number of Pages: 1

☐ Read Till End

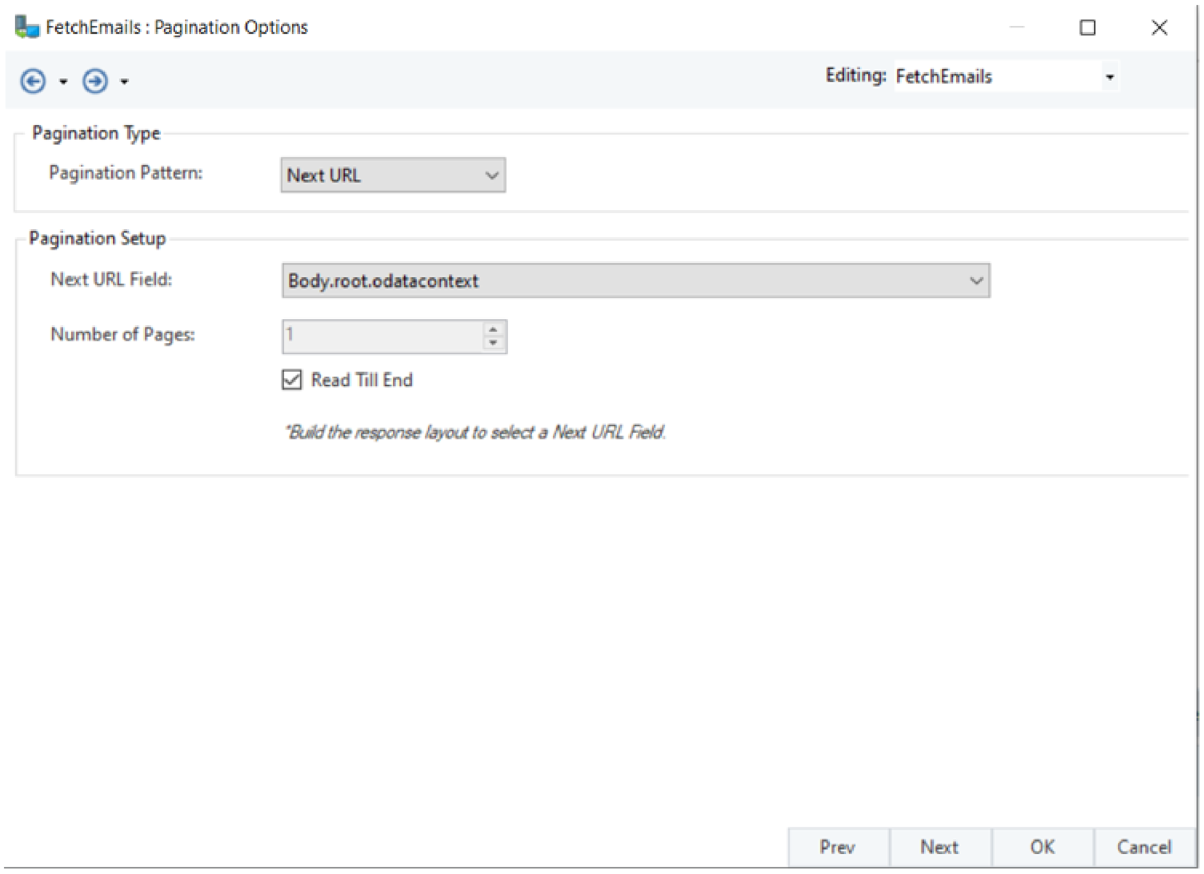
*Specify a Cursor Parameter in the Parameters screen and build the response layout to select a Cursor Field.

11.7.3 Next URL

This type is the same as *Cursor* pagination, except that it generates a URL instead of a token for every subsequent page.

Next URL Field: Here, you can specify the field from the response layout that contains the URl to fetch the next set of records.

Number of Pages: Here, you can specify the number of pages or requests you want to fetch, or you can simply check the *Read Till End* option if you want to fetch all records without specifying a page number limit.


FetchEmails : Pagination Options

Editing: FetchEmails

Pagination Type

Pagination Pattern: Next URL

Pagination Setup

Next URL Field: Body.root.odatacontext

Number of Pages: 1

☑ Read Till End

*Build the response layout to select a Next URL Field.

11.7.4 Page Number

In this type of pagination, you can specify the number of pages you would like to fetch in one go.

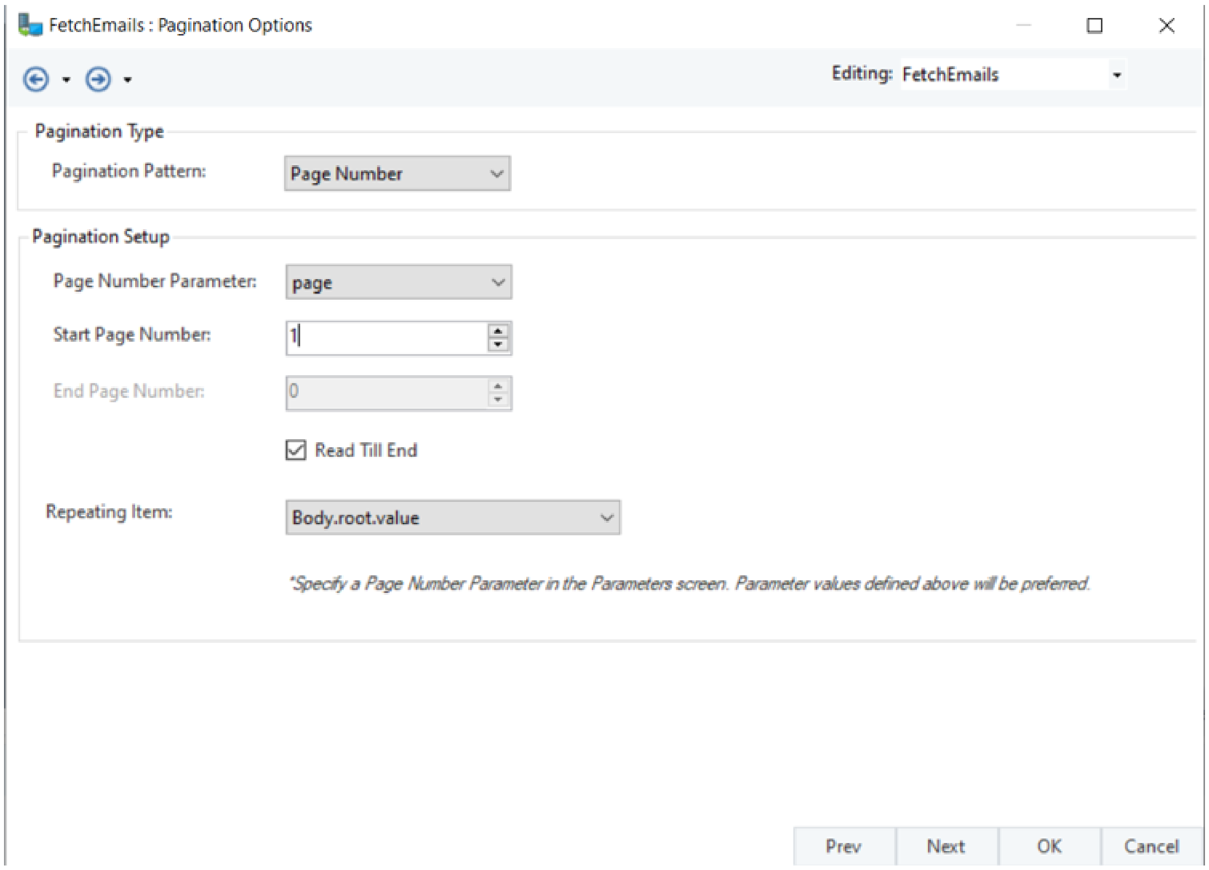
Page Number Parameter: Here, you can specify the page number parameter of the API that you are working with, as specified on the *Parameters* screen.

Start Page Number: The page number from where you want to start fetching your output, or the lower limit.

End Page Number: The page number where you want to end.

Read till end: Check this option if you want to fetch all the available records. Selecting this will disable the End page number option and make requests till no data is returned.

Repeating item: This option is only enabled when you check the Read till end box. You will be required to choose a repeating item, which can be one of the collection nodes from the output layout of the API client object. The repeating item helps the API client recognize the end of records, as whenever an empty response node is returned, the client stops reading the response and the pagination ends.


FetchEmails : Pagination Options

Editing: FetchEmails

Pagination Type

Pagination Pattern: Page Number

Pagination Setup

Page Number Parameter: page

Start Page Number: 1

End Page Number: 0

☑ Read Till End

Repeating Item: Body.root.value

*Specify a Page Number Parameter in the Parameters screen. Parameter values defined above will be preferred.

This concludes our discussion of pagination for APIs in Astera API Management.

11.8 Raw Preview And Copy Curl Command

11.8.1 Raw Preview Request/Response

The raw request and response preview features allow API developers to view the exact request and response payloads being exchanged between clients and servers in their APIs.

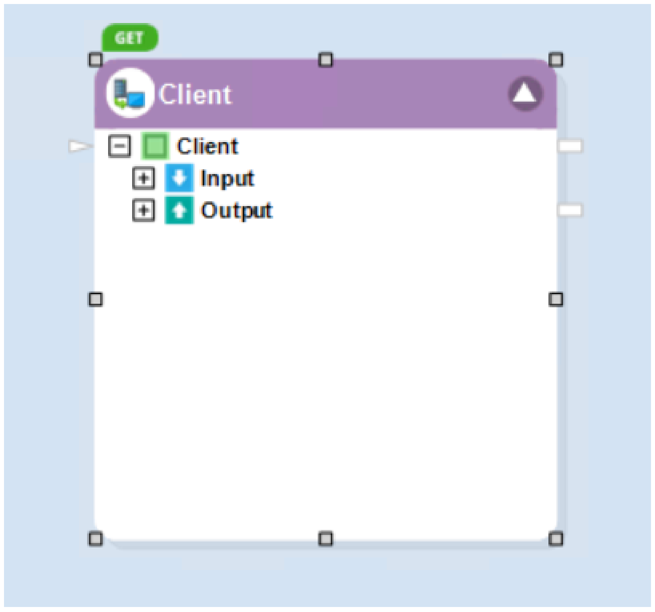
This feature provides a detailed look at the headers, body, parameters, and metadata of the HTTP request and response, which can help API developers debug issues, test APIs, and optimize performance. By using raw preview request and response capabilities, API developers can gain a deeper understanding of how their APIs are being used and troubleshoot issues quickly and efficiently.

Raw Preview in Astera API Management

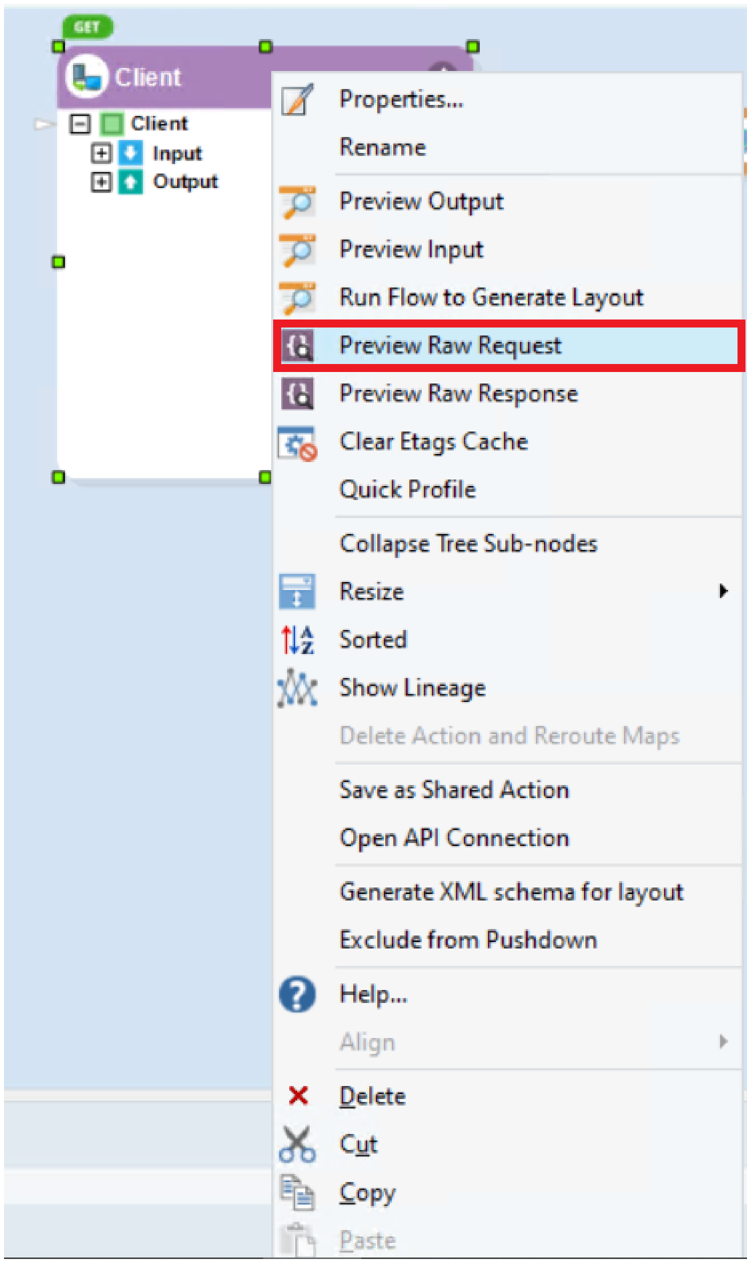
Astera API Management lets the user preview the Raw request and Raw response both from the *API Client* object.

1. Drag and drop an API Client object and configure it.

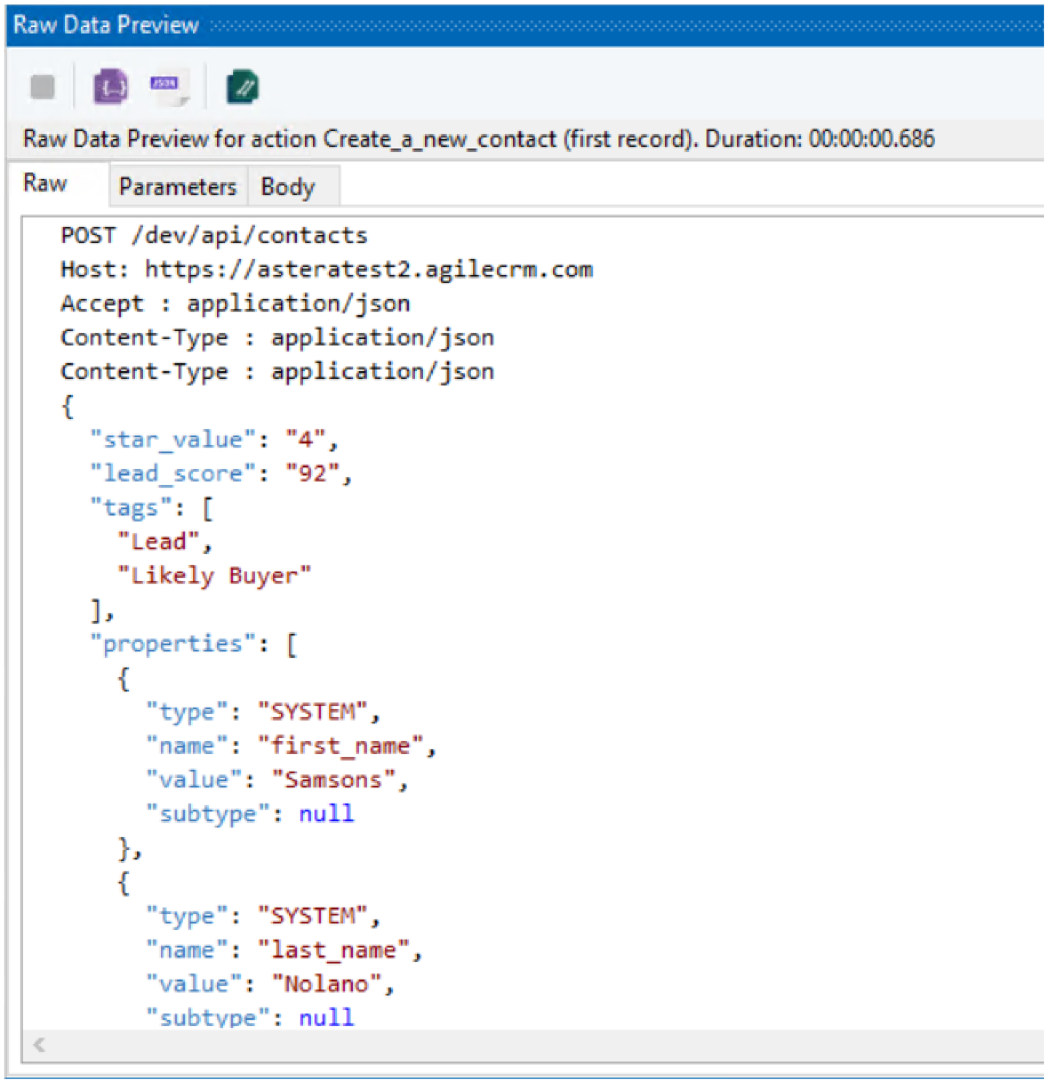
For our use case, we have used an *API Client* making a GET Call to a resource.



2. Right-Click on the object and select *Preview Raw Request*.



This will show the raw request in the Raw Data Preview window.

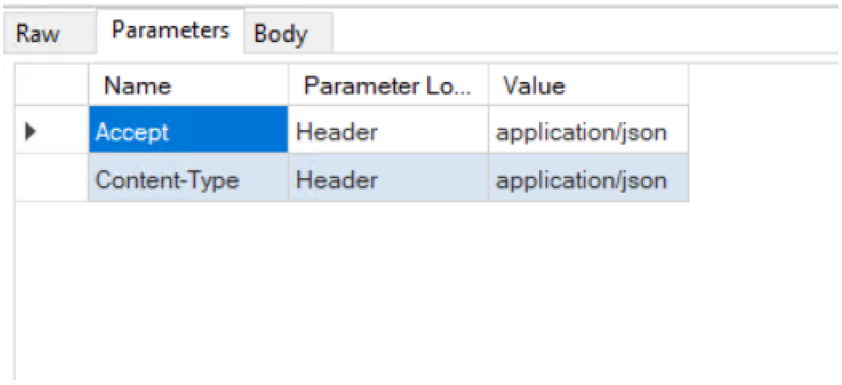


Raw Data Preview for action Create_a_new_contact (first record). Duration: 00:00:00.686

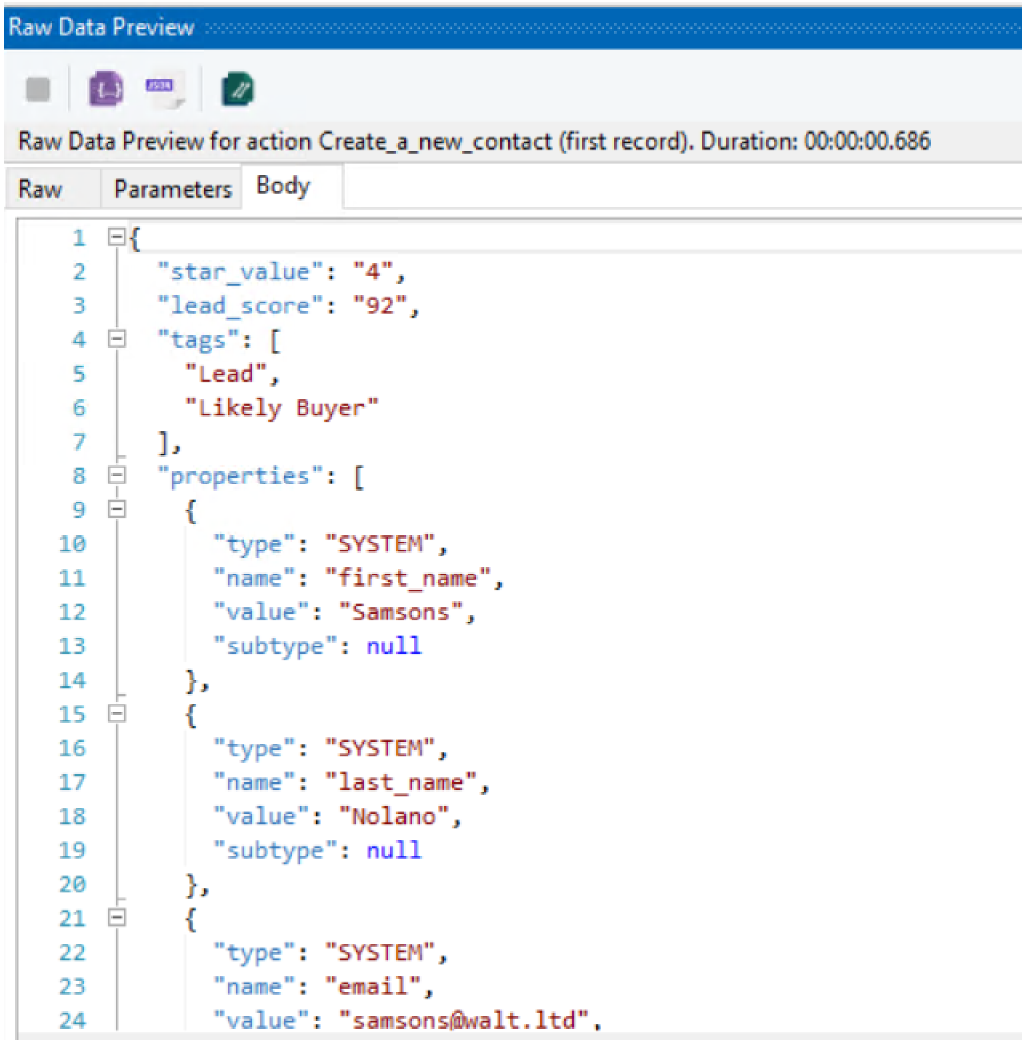
```
POST /dev/api/contacts
Host: https://asteratest2.agilecrm.com
Accept : application/json
Content-Type : application/json
Content-Type : application/json
{
  "star_value": "4",
  "lead_score": "92",
  "tags": [
    "Lead",
    "Likely Buyer"
  ],
  "properties": [
    {
      "type": "SYSTEM",
      "name": "first_name",
      "value": "Samsons",
      "subtype": null
    },
    {
      "type": "SYSTEM",
      "name": "last_name",
      "value": "Nolano",
      "subtype": null
```

As you can see, it has shown the HTTP method as well as the resource, host server details, and the Content-Type of the Request.

It even shows us tabs on the *Request, Parameters,* and *Body*.



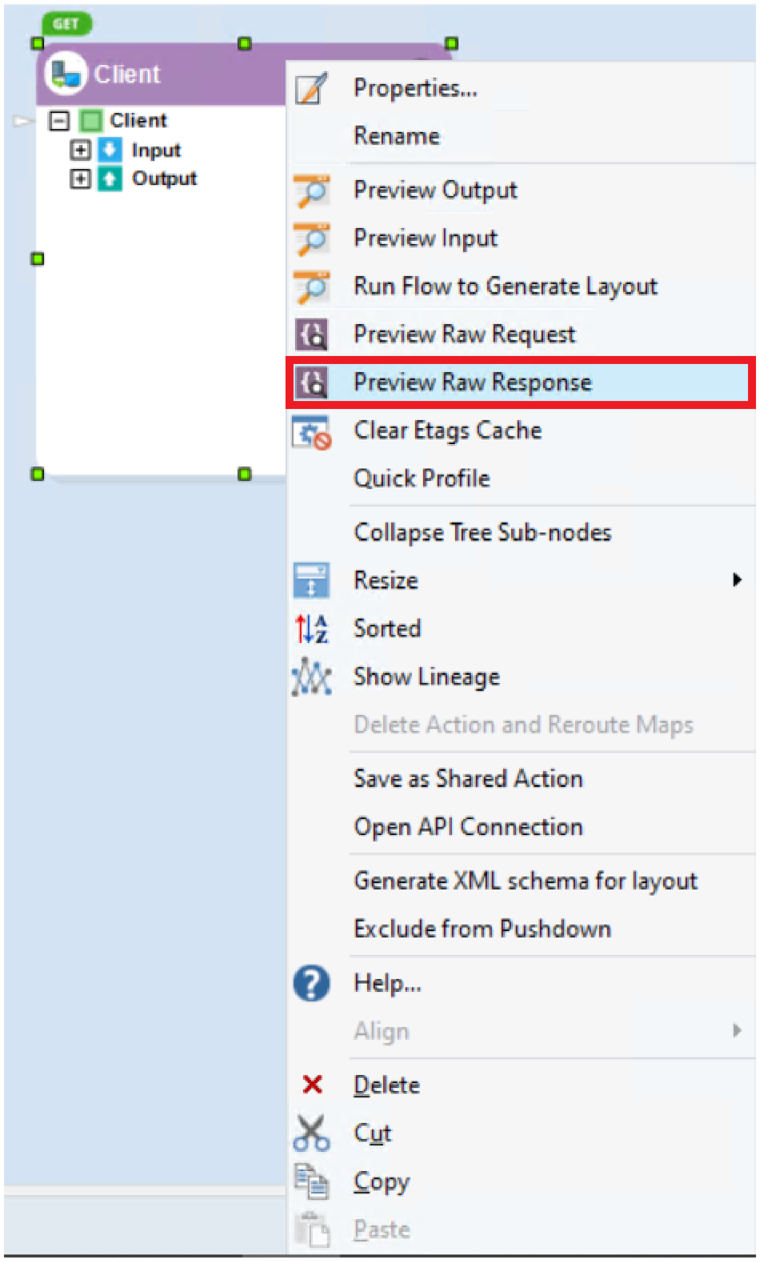
Name	Parameter Lo...	Value
Accept	Header	application/json
Content-Type	Header	application/json



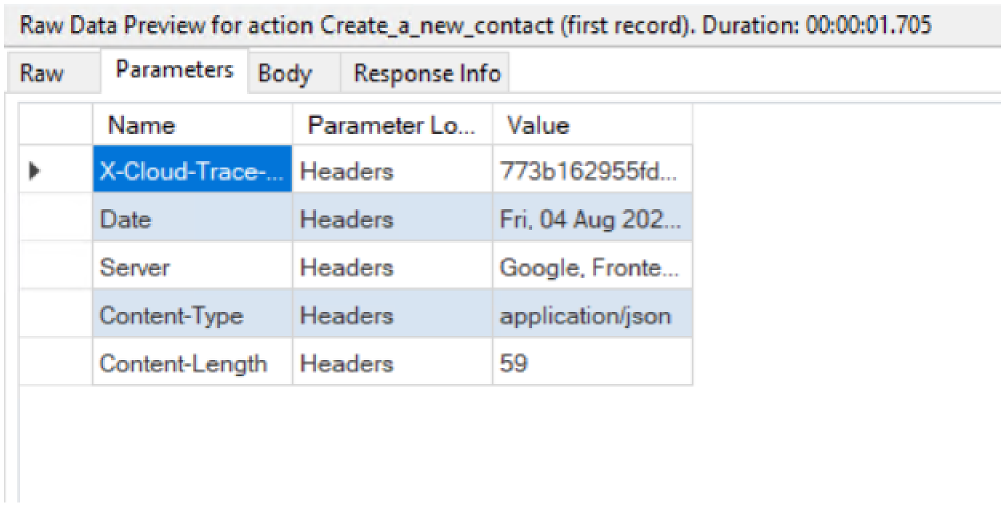
Raw Data Preview for action Create_a_new_contact (first record). Duration: 00:00:00.686

```
{
  "star_value": "4",
  "lead_score": "92",
  "tags": [
    "Lead",
    "Likely Buyer"
  ],
  "properties": [
    {
      "type": "SYSTEM",
      "name": "first_name",
      "value": "Samsons",
      "subtype": null
    },
    {
      "type": "SYSTEM",
      "name": "last_name",
      "value": "Nolano",
      "subtype": null
    },
    {
      "type": "SYSTEM",
      "name": "email",
      "value": "samsons@walt.ltd",
```

3. To preview the raw response, right-click on the *API Client* object and select *Preview Raw Response* from the context menu.

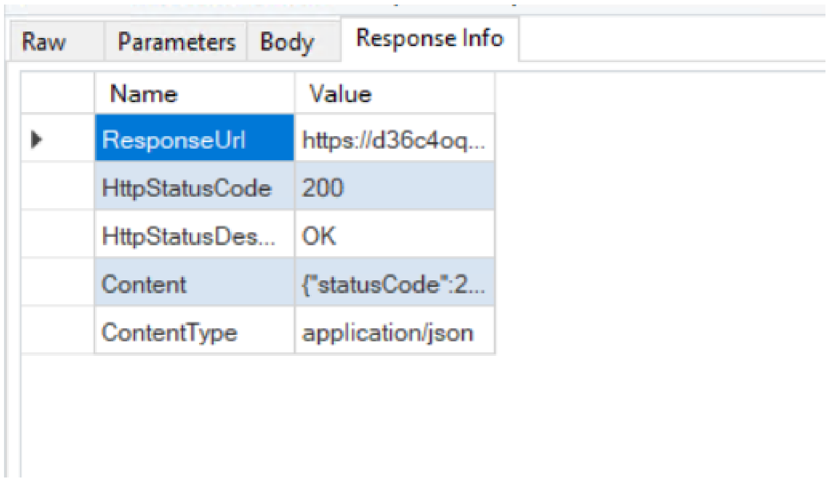


This will generate a raw response in the Raw Data Preview window.


Raw Data Preview for action Create_a_new_contact (first record). Duration: 00:00:01.705

Name	Parameter Lo...	Value
X-Cloud-Trace-...	Headers	773b162955fd...
Date	Headers	Fri, 04 Aug 202...
Server	Headers	Google, Fronte...
Content-Type	Headers	application/json
Content-Length	Headers	59

As you can see above, the raw response has been generated, which shows us the entire HTTP response in raw form. It even has tabs that show us the *Parameters*, *body*, and *response info*.



Name	Value
ResponseUrl	https://d36c4oq...
HttpStatusCode	200
HttpStatusDes...	OK
Content	{"statusCode":2...
ContentType	application/json

11.8.2 CURL Command

Curl is a command-line utility that can be used to send HTTP requests to APIs and retrieve the respective responses.

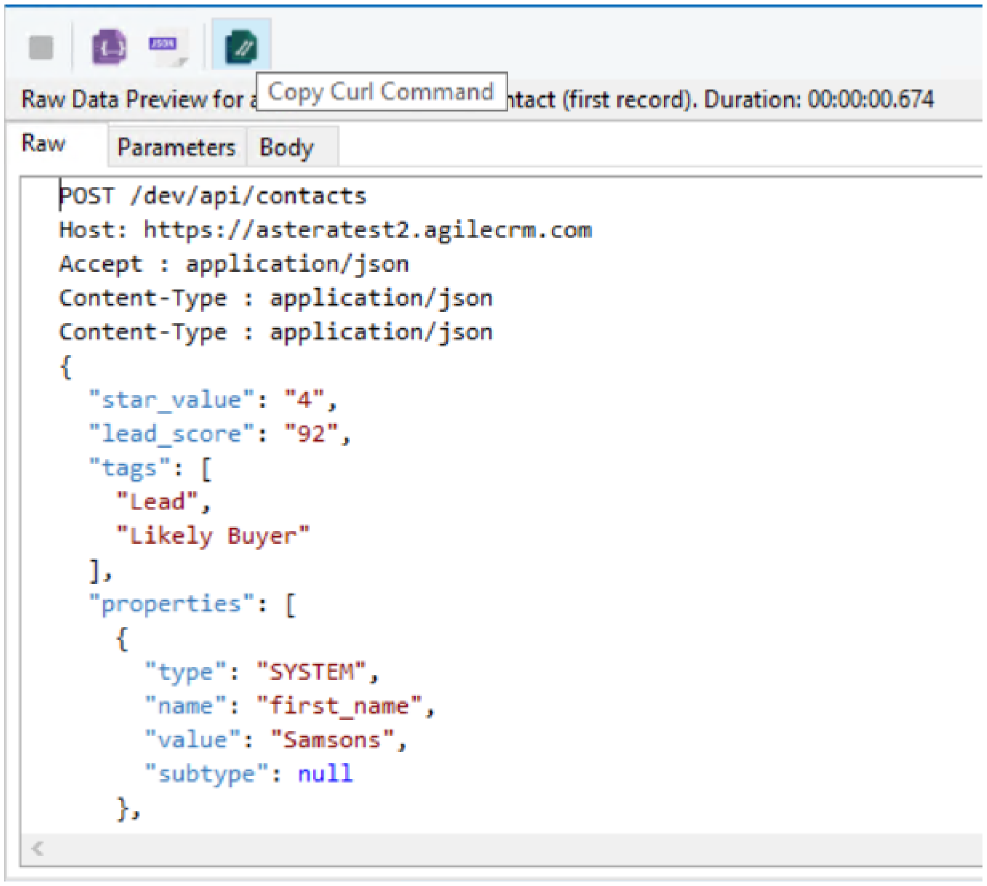
It allows API developers and testers to easily interact with APIs and perform tasks such as testing, debugging, and troubleshooting. Curl supports various HTTP methods such as GET, POST, PUT, and DELETE, and can handle HTTP headers, cookies, and authentication.

It is a simple yet powerful tool that is widely used in API development and management.

Copy CURL in Astera API Management

Astera API Management lets the user copy and view the CURL command from the Raw Data Preview window to help in comparing and debugging results from any external clients such as Windows command prompt or Postman.

Note: The *Copy CURL Command* option is available in the raw request preview.



```
POST /dev/api/contacts
Host: https://asteratest2.agilecrm.com
Accept : application/json
Content-Type : application/json
Content-Type : application/json
{
  "star_value": "4",
  "lead_score": "92",
  "tags": [
    "Lead",
    "Likely Buyer"
  ],
  "properties": [
    {
      "type": "SYSTEM",
      "name": "first_name",
      "value": "Samsons",
      "subtype": null
    },
```

This concludes Raw Preview and Copy CURL in Astera API Management.

11.9 Open APIs – Configuration Details

Note: *Client Secret*, *Access Token* and *API Key* are to be generated by the user, and will be unique for every application. The values specified below are just for example.

11.9.1 Adafruit IO

Authentication Type: API Key

- *Import API*: https://raw.githubusercontent.com/adafruit/io-api/gh-pages/v2.json
- *Authentication:* API-KEY
- *Key:* X-AIO-Key
- *Value*: aio_UTqF73klycqdLWpbp0wLl7RHKV25
- *UserName*: [Enter you user name]

- *FeedKey*: [Enter your feed key]
- *Adafruit Login Page*: https://accounts.adafruit.com/users/sign_in
- *Email:* [Enter your login email]
- *Password*: [Enter your password]

11.9.2 Avaza API

Authentication Type: OAuth 2, Authorization Code

- *Import API*: https://api.avaza.com/swagger/docs/v1
- *Authentication*: oauth2 (Access token will be valid for 1 day)
- *Token URL*: https://any.avaza.com/oauth2/token
- *Auth URL*: https://any.avaza.com/oauth2/authorize
- *ClientId*: [Enter client ID]
- *Client Secret*: c1d4b723790f0e24d0b2df68ebde613e9533
- *Avaza Login Page*: https://any.avaza.com/account/login
- *Email*: [Enter your email]
- *Password:* [Enter your password]

11.9.3 BOX API

Authentication Type: Bearer Token

- *Base URL*: https://api.box.com/2.0
- *Authentication*: Bearer Token (Access token will be valid for 1 hr)
- *Token*: 1IVYyDgfDPyWpoXe9c4RMOt7tmtiB75q
- Steps to generate access token:
- *Page*: https://app.box.com/developers/console/app/984015/configuration
- *Email*: [Enter your login email]
- *Password*: [Enter password]
- Click *Generate Developer Token* to generate access token
- *API Reference*: https://developer.box.com/en/reference

11.9.4 Facebook API

Authentication Type: OAuth 2, Authorization Code

- *Base URL:* https://graph.facebook.com/
- *Auth URL:* https://www.facebook.com/dialog/oauth
- *Access Token URL*: https://graph.facebook.com/oauth/access_token
- *Client ID*: 217423066002

- *Client Secret*: d7d8969c6ea31bf117f04768b63bb
- Credentials to use when using 'Request Token'
- *Email address*: [Enter your email]
- *Password:* [Enter your password]

11.9.5 Google Drive

Authentication Type: Bearer Token

- *Base URL*: https://www.googleapis.com/drive/v3
- *Authentication*: Bearer Token (Token will be valid for an hour)
- *Token*: ya29.Il_AB6CICAcAQD6lKoQCW3K2DO_enBd3be5G2Vvd0hZ3Q8US4eHL-PEOS1qRD7zzSEN3t_qb_eNqWzZS3zsXP_FcAHA9TSoy-tDpsWv0RnWRledPhZqRt79f9X
- *API Reference*: https://developers.google.com/drive/api/v3/reference

Steps to generate access token:

1. Go to https://developers.google.com/oauthplayground/
2. Select the APIs you want to authorize and click *Authorize APIs*.
3. On the next screen, provide your credentials.
4. *Email*: [Enter your login email]
5. *Password*: [Enter your password]
6. Now click *Exchange authorization code for tokens* to generate access token.

11.9.6 netAuth API

Authentication Type: API Key

- *Import API*: https://api.doc.nextauth.com/api/swagger.json
- *Authentication*: API-KEY
- *KEY:* [Enter API Key]
- *VALUE*: J5znqilK_qUt65iQyy9W2Q
- Help link: https://api.doc.nextauth.com/

11.9.7 OMDb API

Authentication Type: API Key

- API key to be passed as a query parameter
- *JSON File*: http://www.omdbapi.com/swagger.json

Steps to generate API Key:

1. Open http://www.omdbapi.com/apikey.aspx?__EVENTTARGET=freeAcct&__EVENTARGUMENT=&__LASTFOCUS=&__V
2. Select *Account Type*, 'FREE.'
3. Enter your email address.

4. Enter your first name and last name.
5. Describe in a few words your purpose of using this service.
6. Click *Submit*.
7. You will get the API Key in your email with a link to activate it. Click on this link and the key will be activated.

11.9.8 Square Connect API

Authentication Type: Bearer Token

- *Import API*: https://raw.githubusercontent.com/square/connect-api-specification/master/api.json
- *Authentication*: Bearer Token
- *Token*: EAAAEPXVtza2Utrx-GJ90Az4sCQ_NLbLYOKANVFmJiPGJ1Z6B-eJgZ-2V1
- Use this API to import: https://raw.githubusercontent.com/

  Note: This looks like an issue with Square Connect's documentation because the '*Import API*' option does not work.

11.9.9 Zendesk API

Authentication Type: Basic Authentication

- *Username*: [Enter username or login email]
- *Password*: [Enter password]

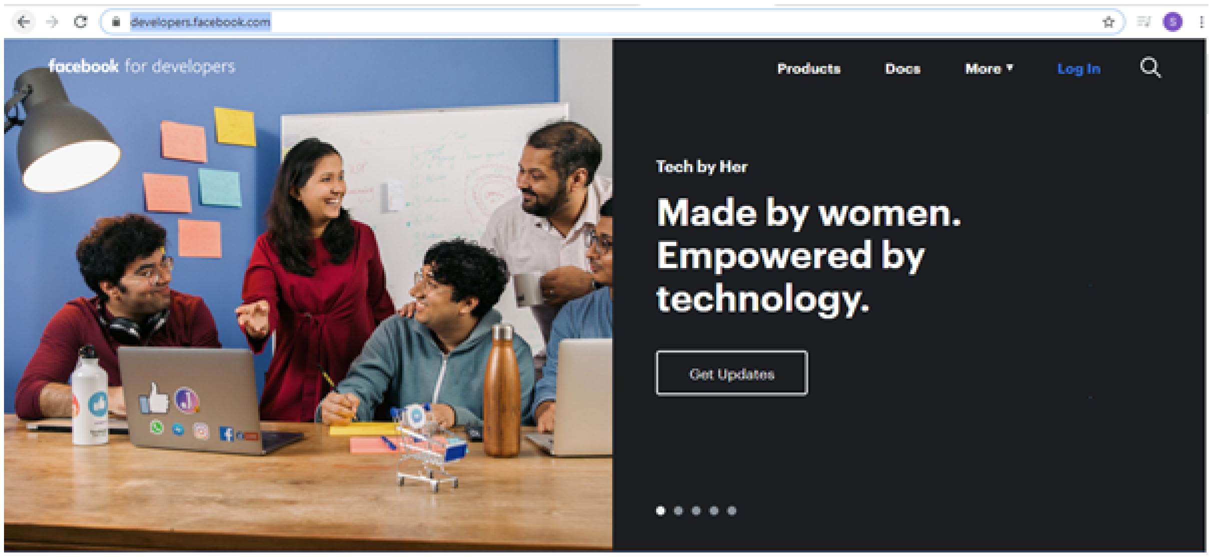
11.10 Authorizing Facebook APIs in Astera Centerprise

Facebook uses HTTP-based APIs that can be utilized to extract or load data, to and from Facebook. You can configure Facebook APIs for use in Astera Centerprise using the 'Custom API' source in the REST API Browser (Beta).

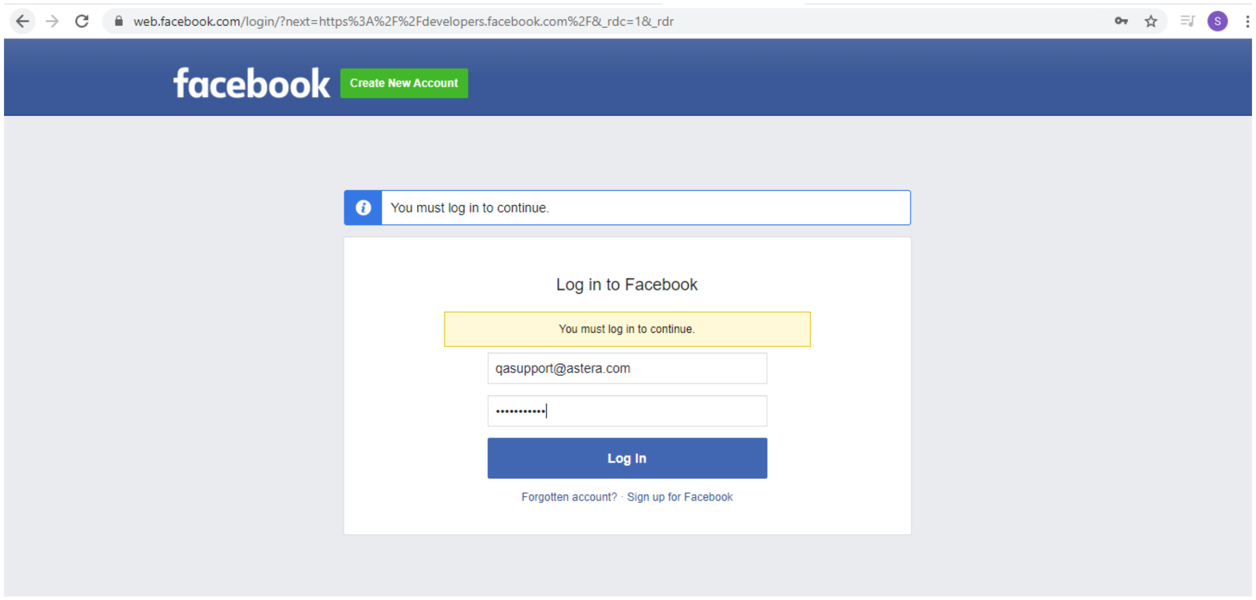
To authorize a Facebook API in Astera Centerprise, follow the steps below.

1. Go to this Url: https://developers.facebook.com/ and log in.

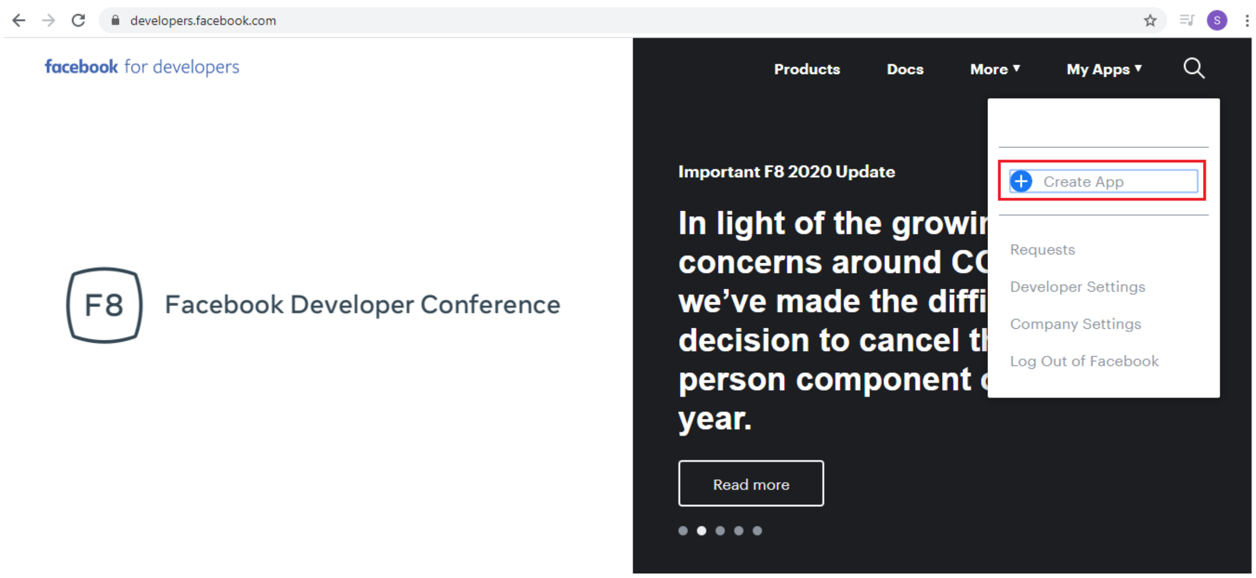
   Note: If you have not created an account yet, you need to create one first after signing in.



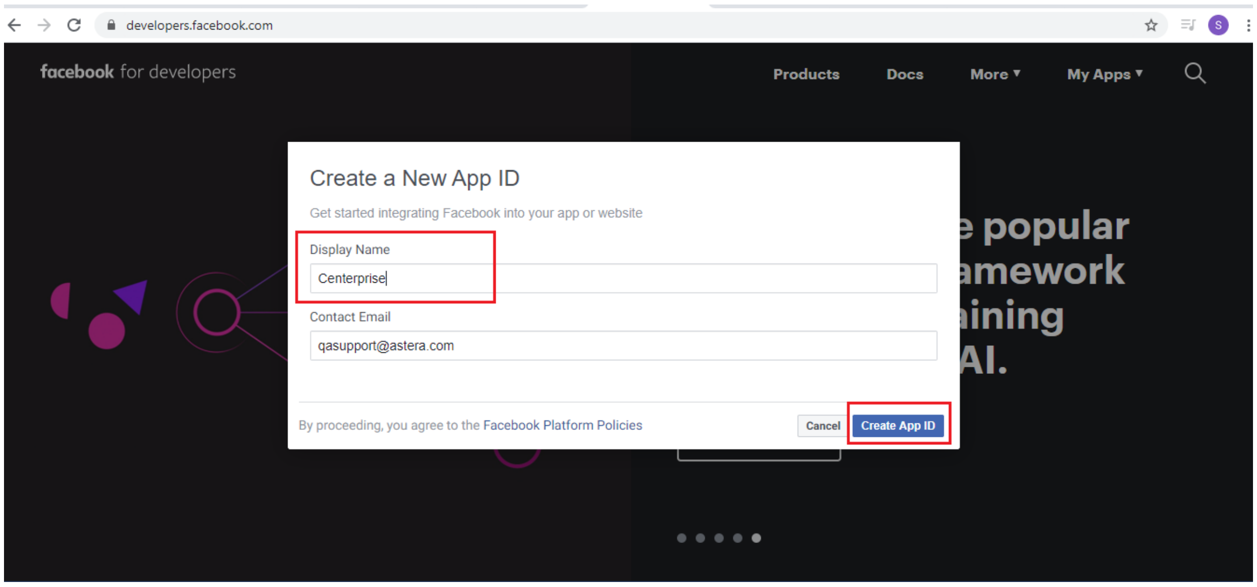
2. Enter your Facebook account credentials to log in.


web.facebook.com/login/?next=https%3A%2F%2Fdevelopers.facebook.com%2F&_rdc=1&_rdr
facebook
Create New Account
You must log in to continue.
Log in to Facebook
You must log in to continue.
qasupport@astera.com
Log In
Forgotten account? · Sign up for Facebook

3. Go to *My Apps* > *Create App* to create an application.



4. Provide the *Display Name* for your application, and click *Create App ID*.



Create a New App ID

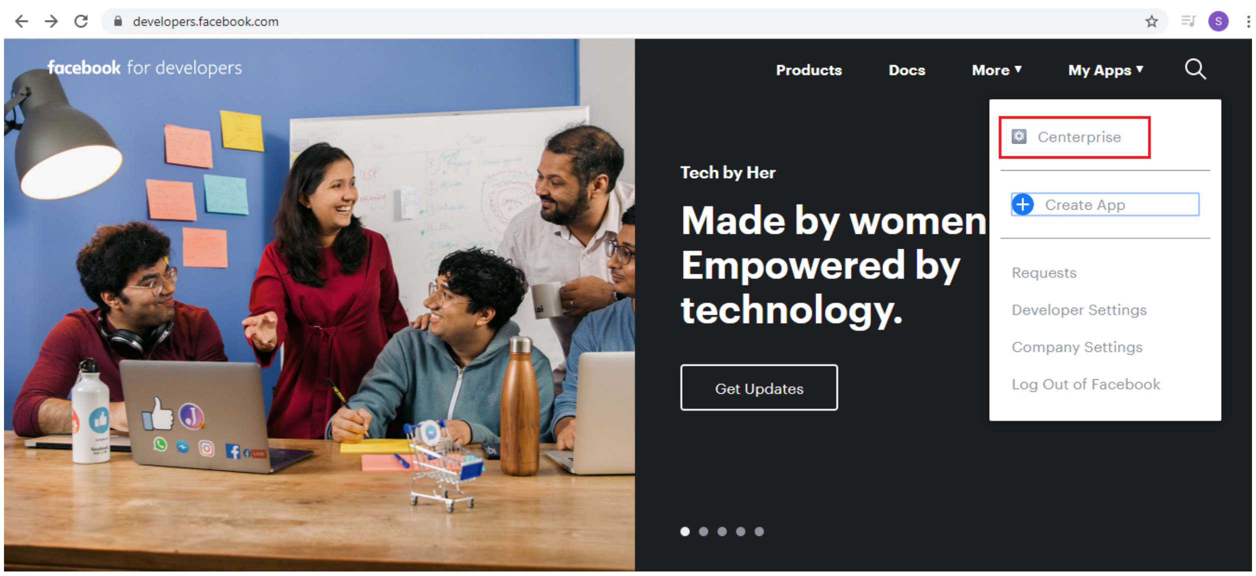
Get started integrating Facebook into your app or website

Display Name: Centerprise

Contact Email: qasupport@astera.com

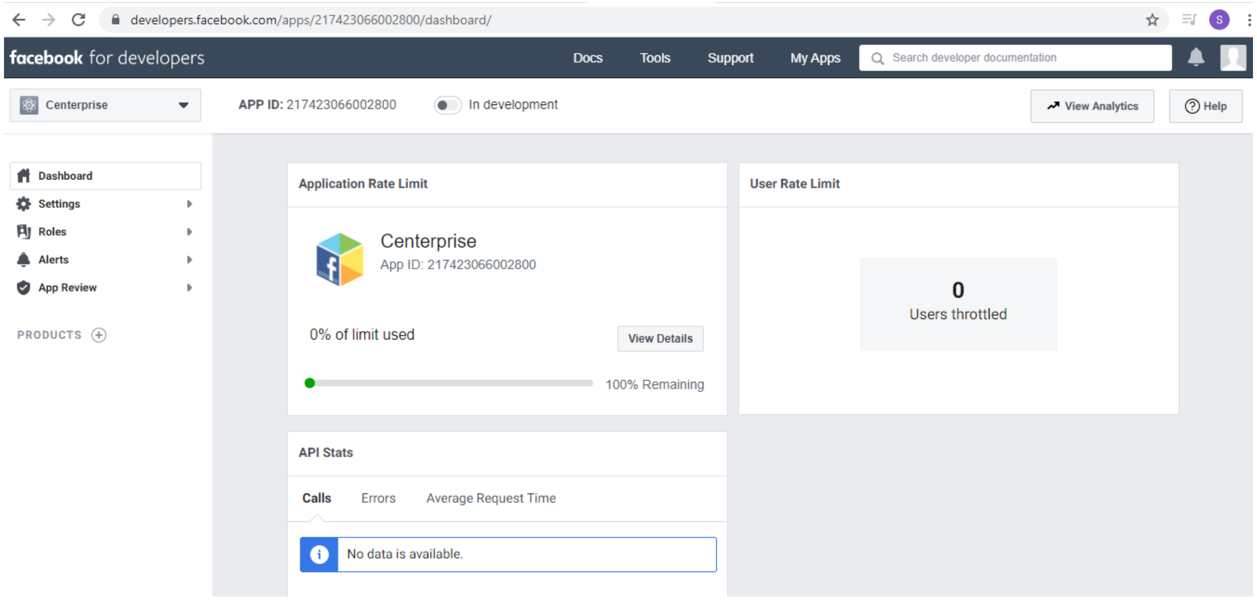
By proceeding, you agree to the Facebook Platform Policies

Once your application is created, it will show under the *My Apps* tab.



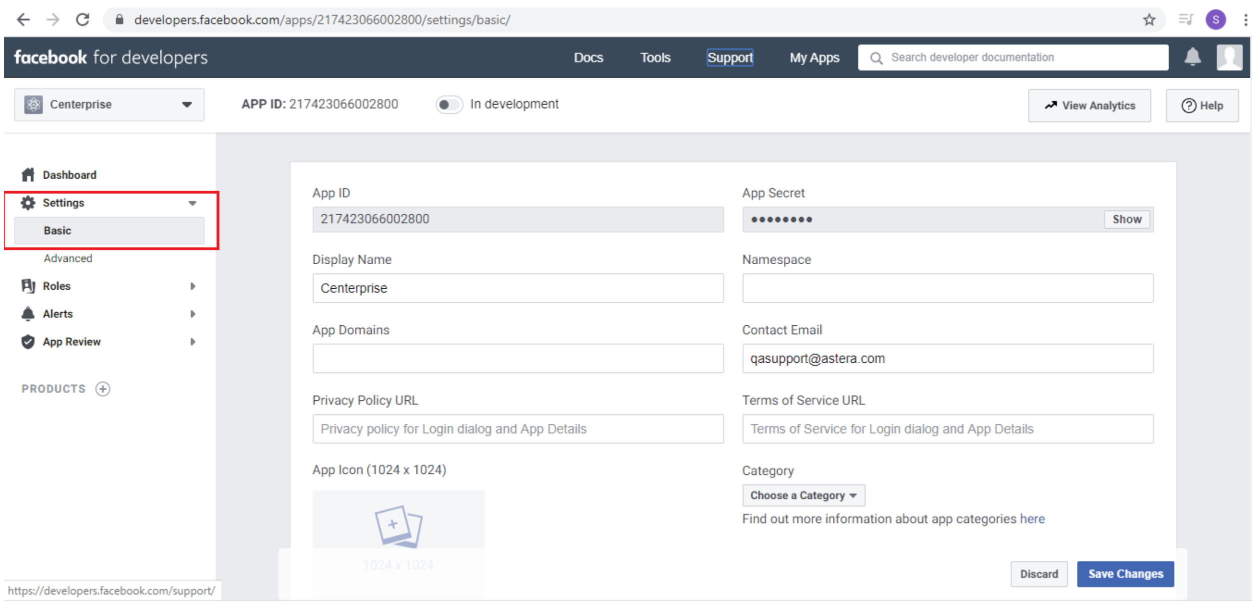
5. Click *Centerprise* to open the dashboard.

Reference Url: https://developers.facebook.com/apps/217423066002800/dashboard/



6. Click on *Settings* > *Basic* to get the relevant credentials.

Reference Url: https://developers.facebook.com/apps/217423066002800/settings/basic/



APP ID: 217423066002800 In development

App ID: 217423066002800

App Secret: ••••••••

Display Name: Centerprise

Namespace:

App Domains:

Contact Email: qasupport@astera.com

Privacy Policy URL: Privacy policy for Login dialog and App Details

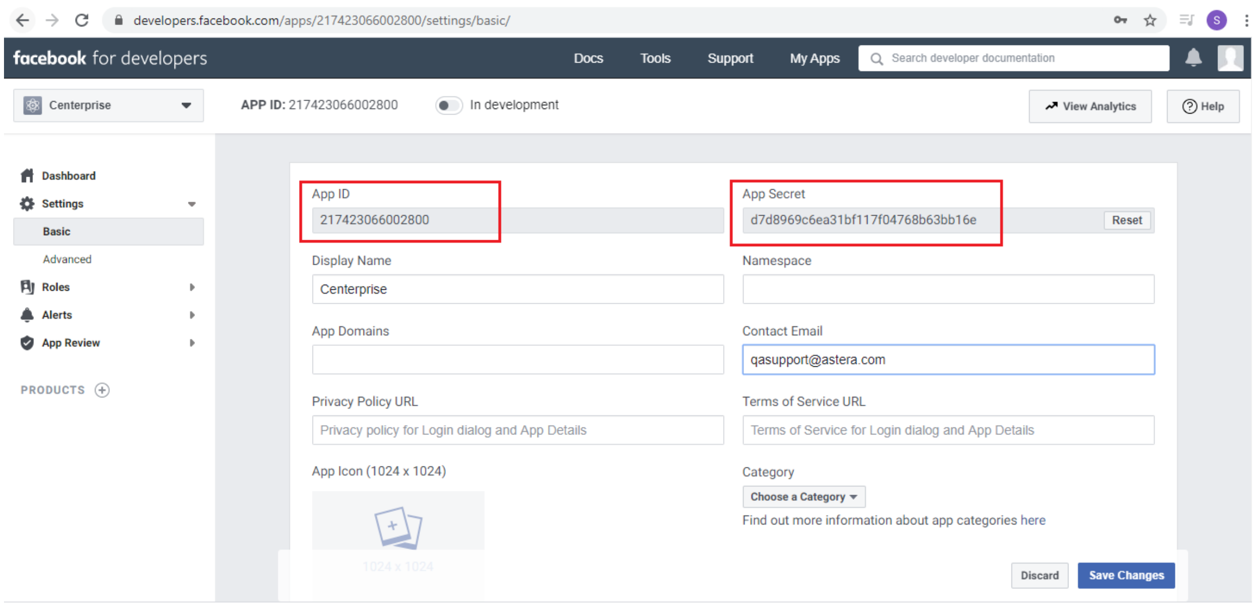
Terms of Service URL: Terms of Service for Login dialog and App Details

App Icon (1024 x 1024)

Category: Choose a Category

Find out more information about app categories here

7. Here you can see the *App ID* and *App Secret*. Save this information to use later for authentication.



APP ID: 217423066002800 In development

App ID: 217423066002800

App Secret: d7d8969c6ea31bf117f04768b63bb16e

Display Name: Centerprise

Namespace:

App Domains:

Contact Email: qasupport@astera.com

Privacy Policy URL: Privacy policy for Login dialog and App Details

Terms of Service URL: Terms of Service for Login dialog and App Details

App Icon (1024 x 1024)

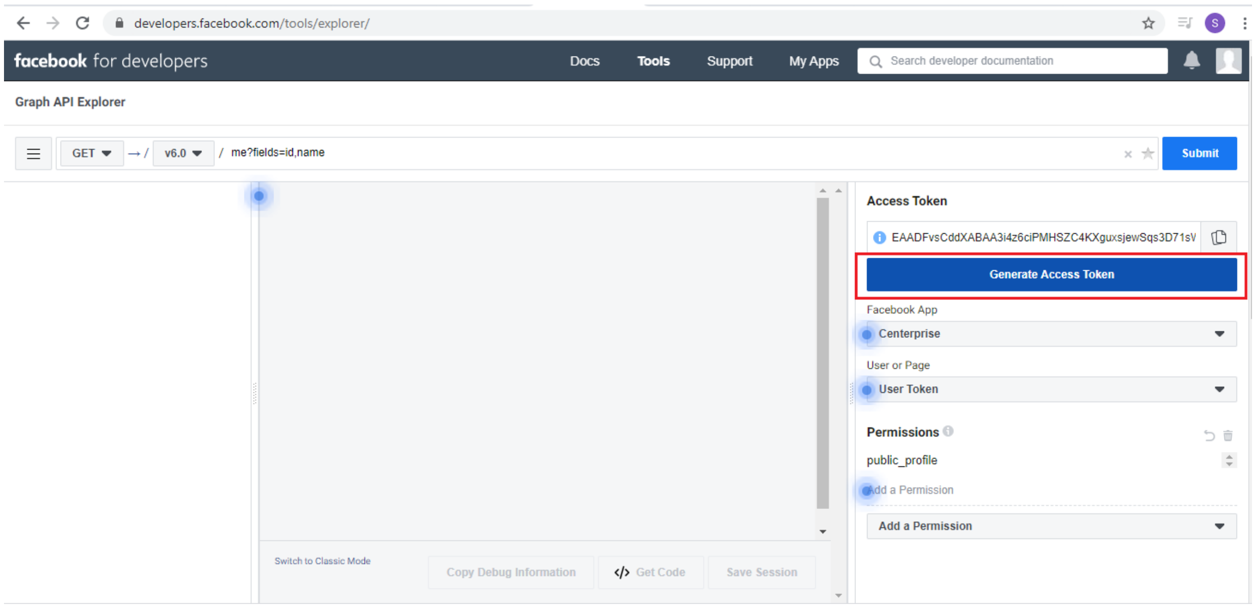
Category: Choose a Category

Find out more information about app categories here

8. To use *Bearer Token* authentication, go to *Tools* > *Graph API Explorer*.

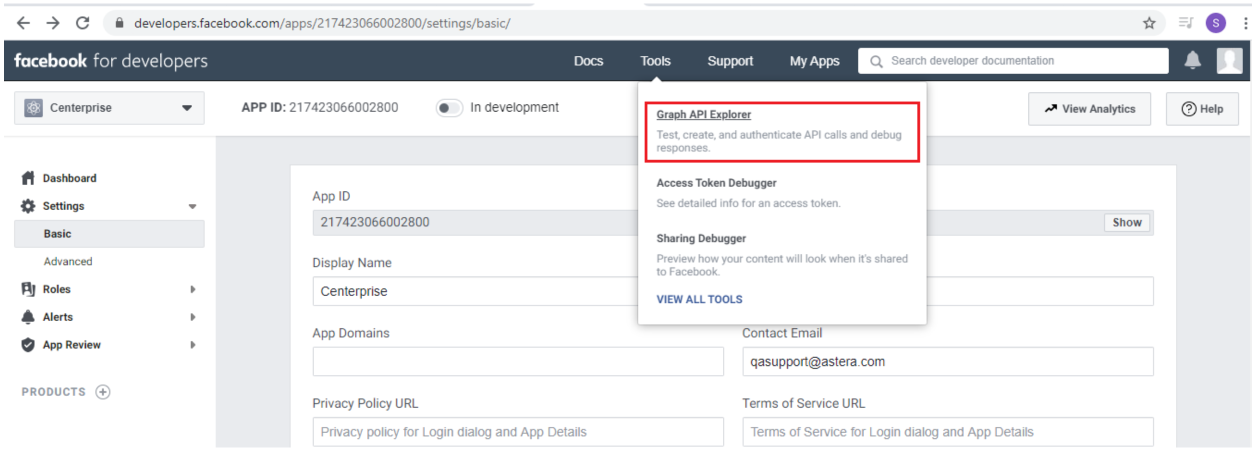
Reference Url: https://developers.facebook.com/tools/explorer/

9. Click *Generate Access Token* and copy the token.

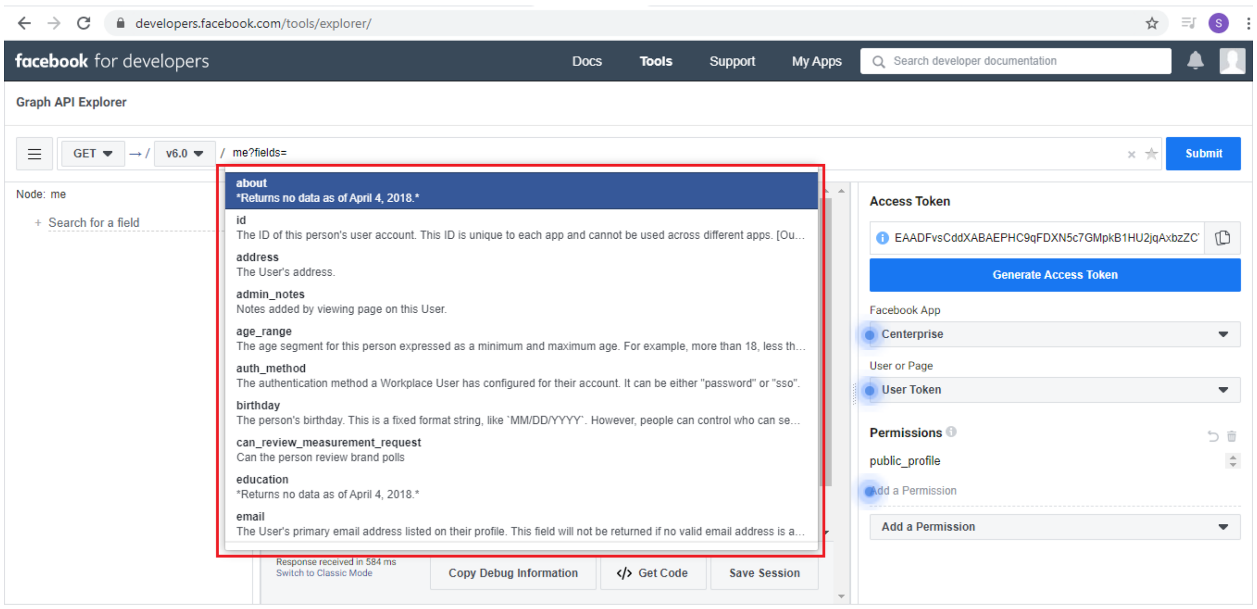


10. To access and try out different APIs, go to *Tools* > *Graph API Explorer*.

Reference Url: https://developers.facebook.com/tools/explorer/



11. Select anything from the drop-down list.

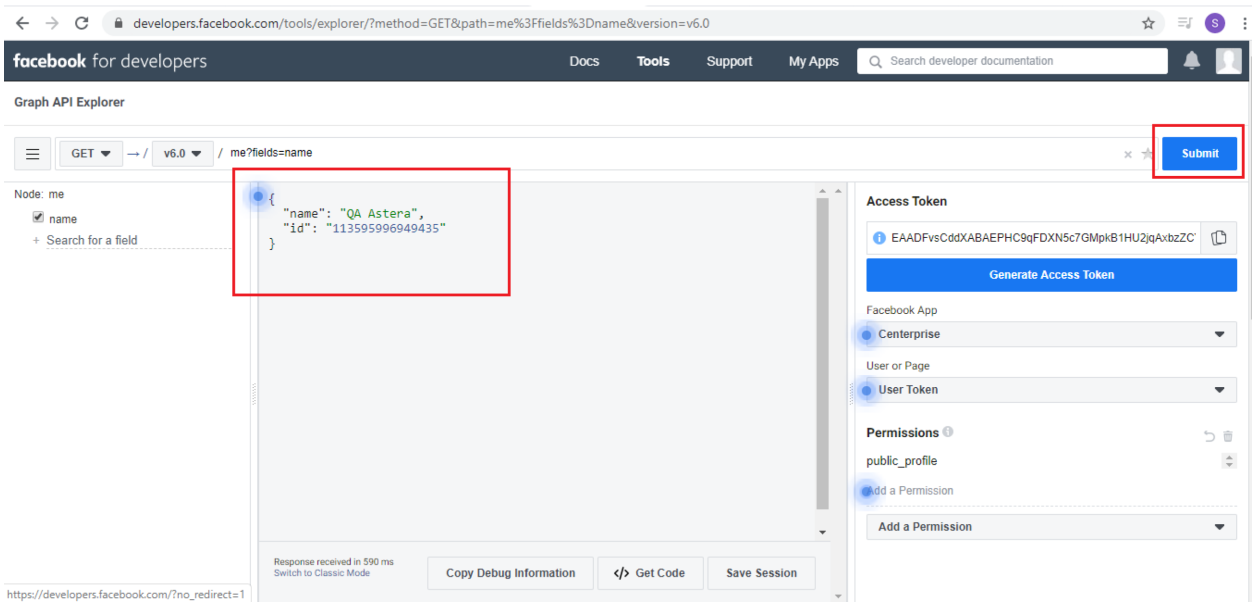


GET → / v6.0 / me?fields=

about
Returns no data as of April 4, 2018.
id
The ID of this person's user account. This ID is unique to each app and cannot be used across different apps. [Ou…
address
The User's address.
admin_notes
Notes added by viewing page on this User.
age_range
The age segment for this person expressed as a minimum and maximum age. For example, more than 18, less th…
auth_method
The authentication method a Workplace User has configured for their account. It can be either "password" or "sso".
birthday
The person's birthday. This is a fixed format string, like `MM/DD/YYYY`. However, people can control who can se…
can_review_measurement_request
Can the person review brand polls
education
Returns no data as of April 4, 2018.
email
The User's primary email address listed on their profile. This field will not be returned if no valid email address is a…

Access Token: EAADFvsCddXABAEPHC9qFDXN5c7GMpkB1HU2jqAxbzZC
Facebook App: Centerprise
User or Page: User Token
Permissions: public_profile

12. Click *Submit*, to see the results.



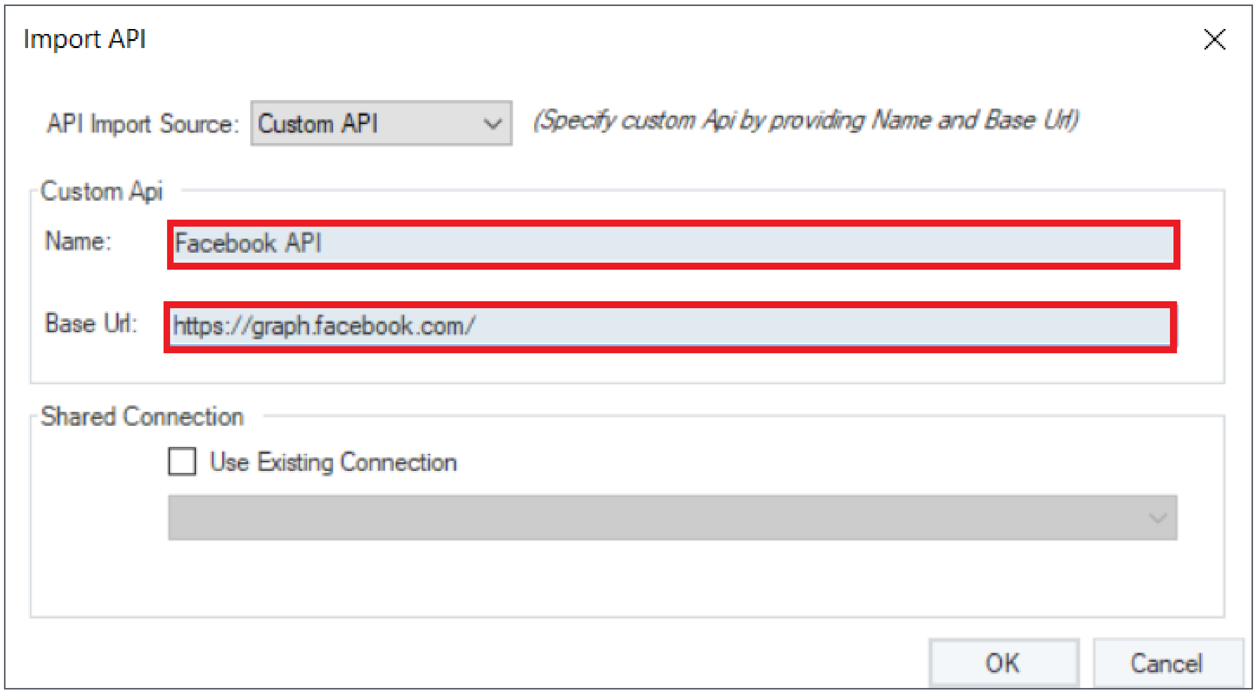
GET → / v6.0 / me?fields=name

```
{
  "name": "QA Astera",
  "id": "113595996949435"
}
```

Access Token: EAADFvsCddXABAEPHC9qFDXN5c7GMpkB1HU2jqAxbzZC
Facebook App: Centerprise
User or Page: User Token
Permissions: public_profile

13. Import the API in Centerprise using the *Import API* option in the REST API Browser (Beta). Select *API Import Source* as *Custom API* by providing *Name* and *Base Url*. To learn more about how to work with custom APIs in Centerprise, click here.

Base Url: https://graph.facebook.com/



Import API

API Import Source: Custom API (Specify custom Api by providing Name and Base Url)

Custom Api

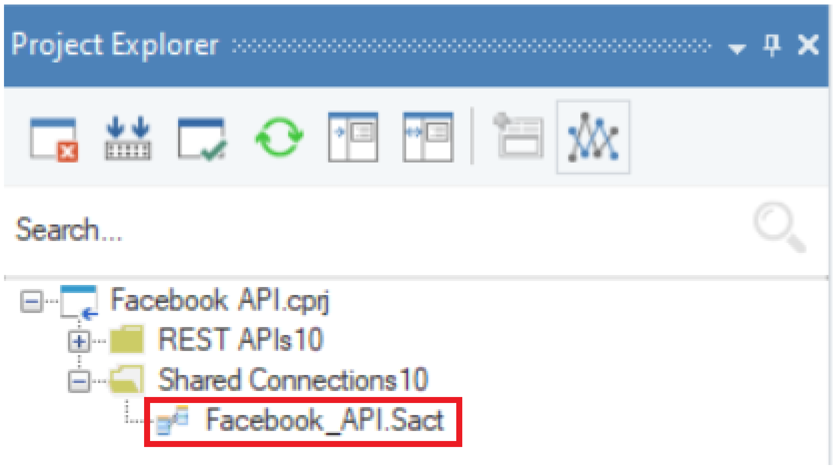
Name: Facebook API

Base Url: https://graph.facebook.com/

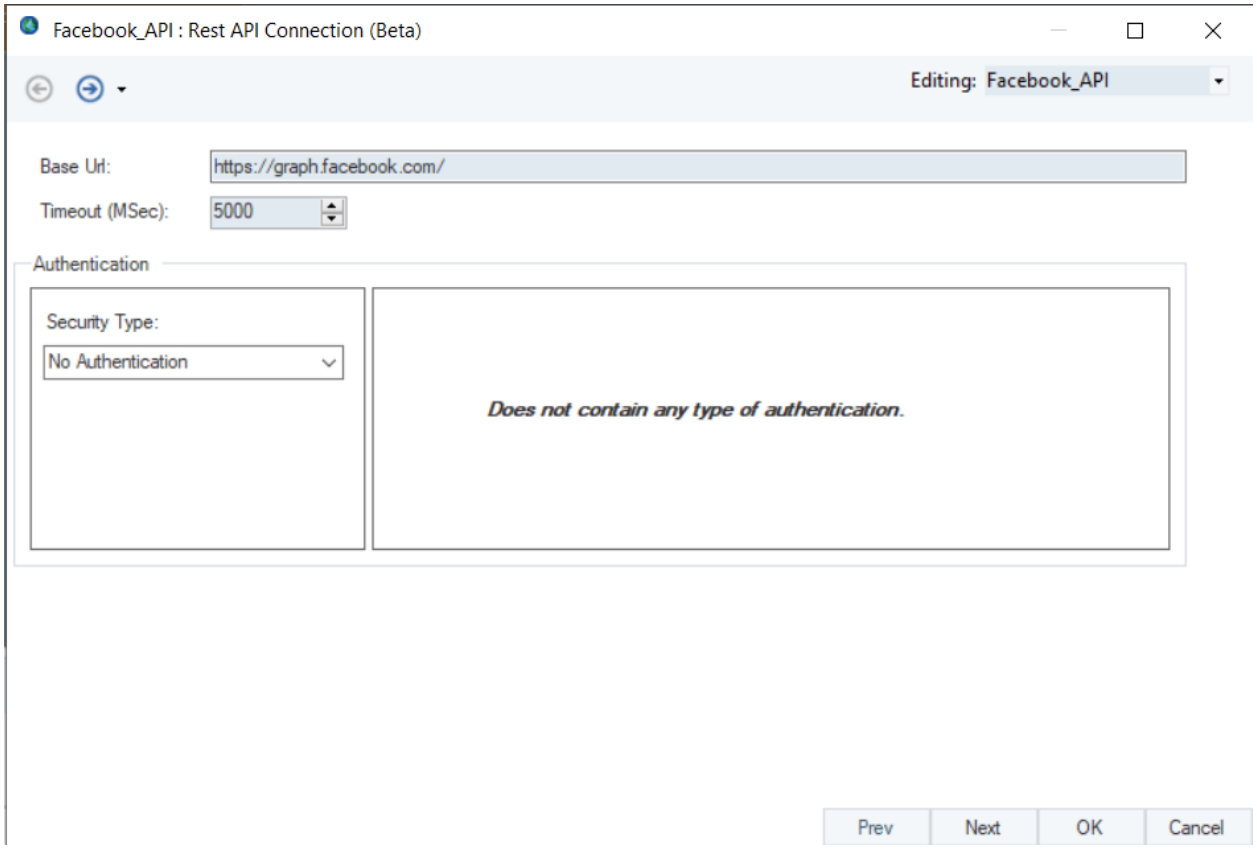
Shared Connection

☐ Use Existing Connection

14. Now, you need to authenticate the Facebook APIs to use them in your dataflow. Without authentication, you will get an error. To authenticate an API, go to the Project Explorer panel and double click on the API's .sact file under the *Shared Connection* node.



Facebook's .sact file will open on the designer. Now, right-click on the shared action file's header and select *Properties*. This will open the *REST API Connection* window, where you can configure the settings to authenticate Facebook's API.



Base Url: https://graph.facebook.com/

Timeout (MSec): 5000

Authentication

Security Type: No Authentication

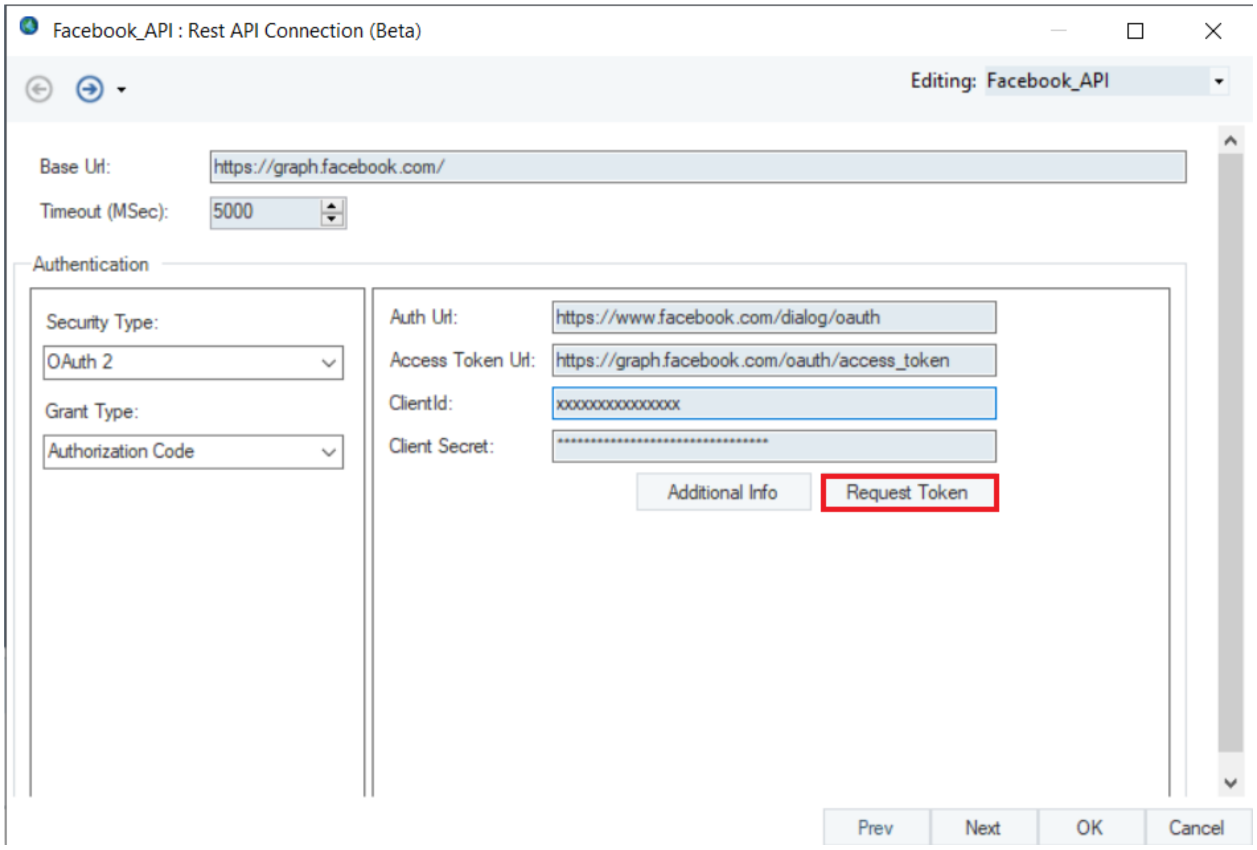
Does not contain any type of authentication.

Facebook uses '*OAuth 2*' authentication with *Grant Type*, '*Authorization Code*'.

Auth Url: https://www.facebook.com/dialog/oauth

Access Token Url: https://graph.facebook.com/oauth/access_token

Provide *ClientID* and *Client Secret* that you had saved earlier, then click on *Request token* to generate the access token for Facebook.


Facebook_API : Rest API Connection (Beta)

Editing: Facebook_API

Base Url: https://graph.facebook.com/

Timeout (MSec): 5000

Authentication

Security Type: OAuth 2

Grant Type: Authorization Code

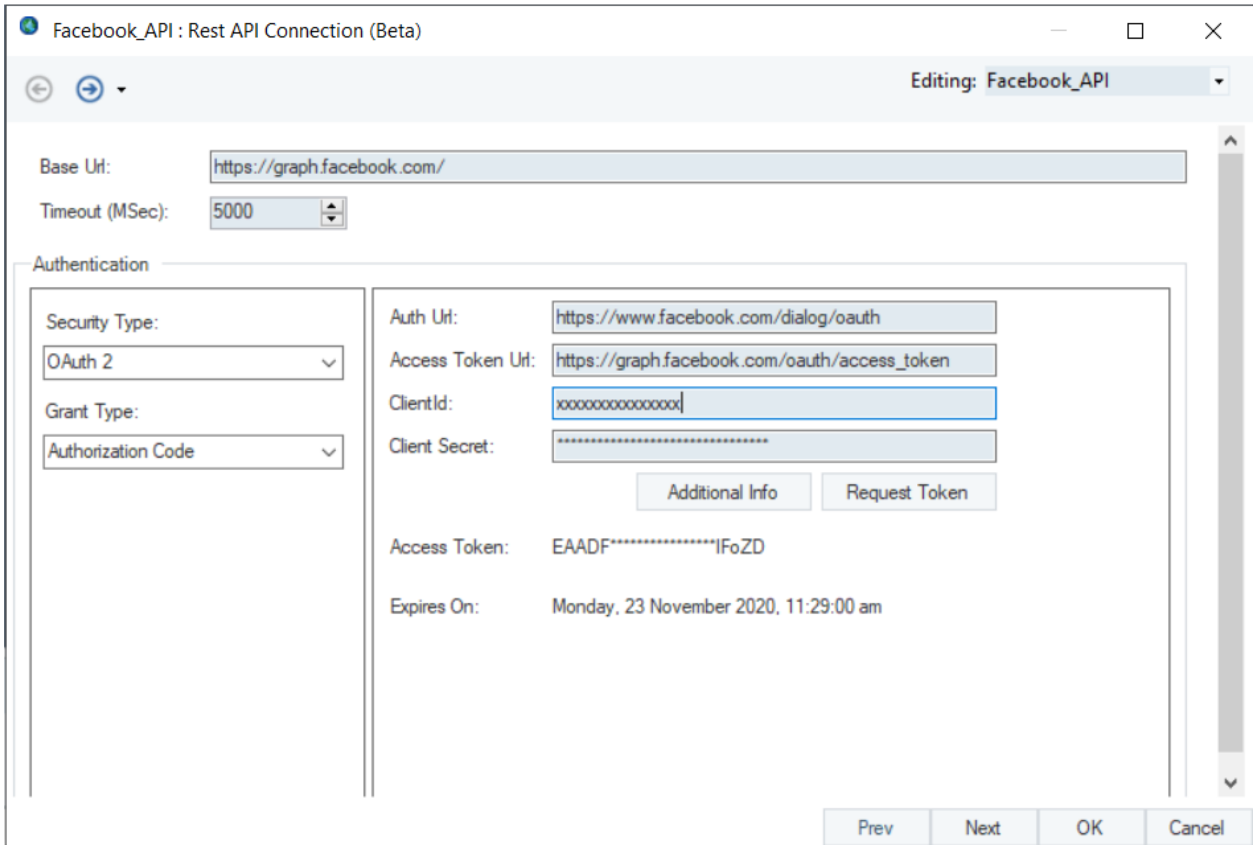
Auth Url: https://www.facebook.com/dialog/oauth

Access Token Url: https://graph.facebook.com/oauth/access_token

ClientId: xxxxxxxxxxxxxxx

Client Secret: ********************************

Note: As you click on *Request Token*, Facebook's login window will open where you will have to provide your credentials to generate the access token to access Facebook API.



Base Url: https://graph.facebook.com/
Timeout (MSec): 5000
Security Type: OAuth 2
Grant Type: Authorization Code
Auth Url: https://www.facebook.com/dialog/oauth
Access Token Url: https://graph.facebook.com/oauth/access_token
ClientId: xxxxxxxxxxxxxxx
Client Secret: ********************************
Access Token: EAADF****************lFoZD
Expires On: Monday, 23 November 2020, 11:29:00 am

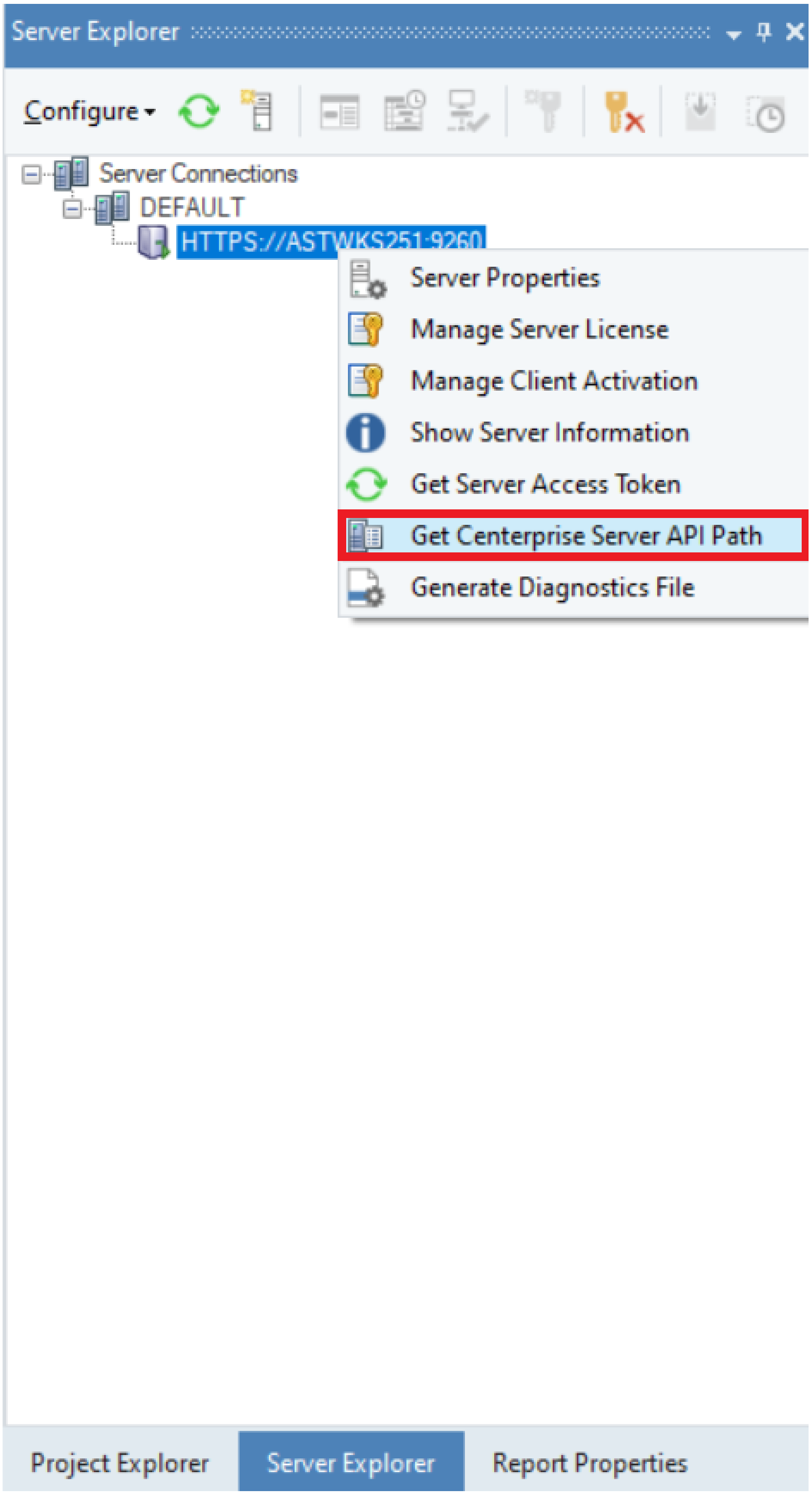
15. Save the shared action file after authentication and you are ready to use Facebook APIs in Centerprise. For more information on how to use a Custom API in Centerprise, click here.

This concludes authenticating the Facebook APIs in Astera Centerprise.

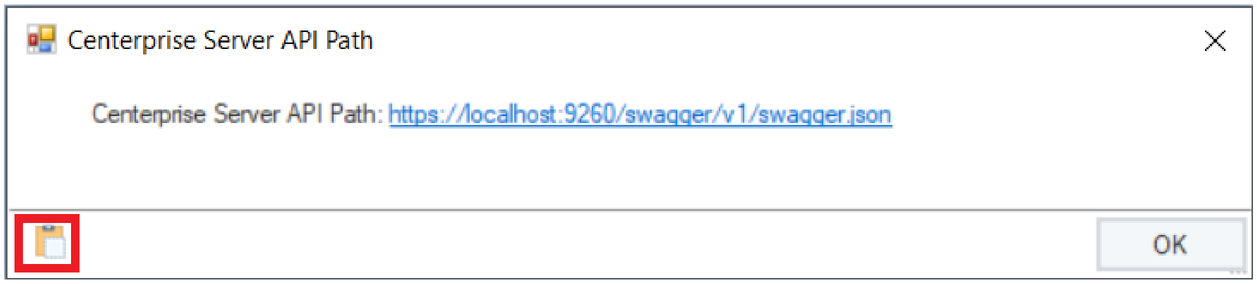
11.11 Authorizing Centerprise's Server APIs

Follow the steps below to learn how to authenticate Centerprise's Server APIs.

1. Right-click on the server name in *Server Explorer* > *Server Connections* > *DEFAULT* > *HTTPS://(ServerName):9260*.

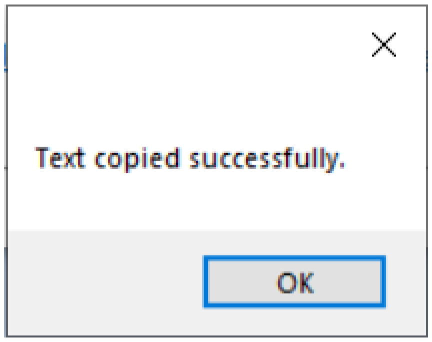

Server Explorer
Configure
Server Connections
DEFAULT
HTTPS://ASTWKS251:9260
Server Properties
Manage Server License
Manage Client Activation
Show Server Information
Get Server Access Token
Get Centerprise Server API Path
Generate Diagnostics File
Project Explorer
Server Explorer
Report Properties

2. A wizard will appear with the *Centerprise Server API Path*. Click on the copy icon located at the bottom-left of the wizard to copy it.



Centerprise Server API Path: https://localhost:9260/swagger/v1/swagger.json

A message will appear to confirm that the text has been copied successfully. Click *OK*.

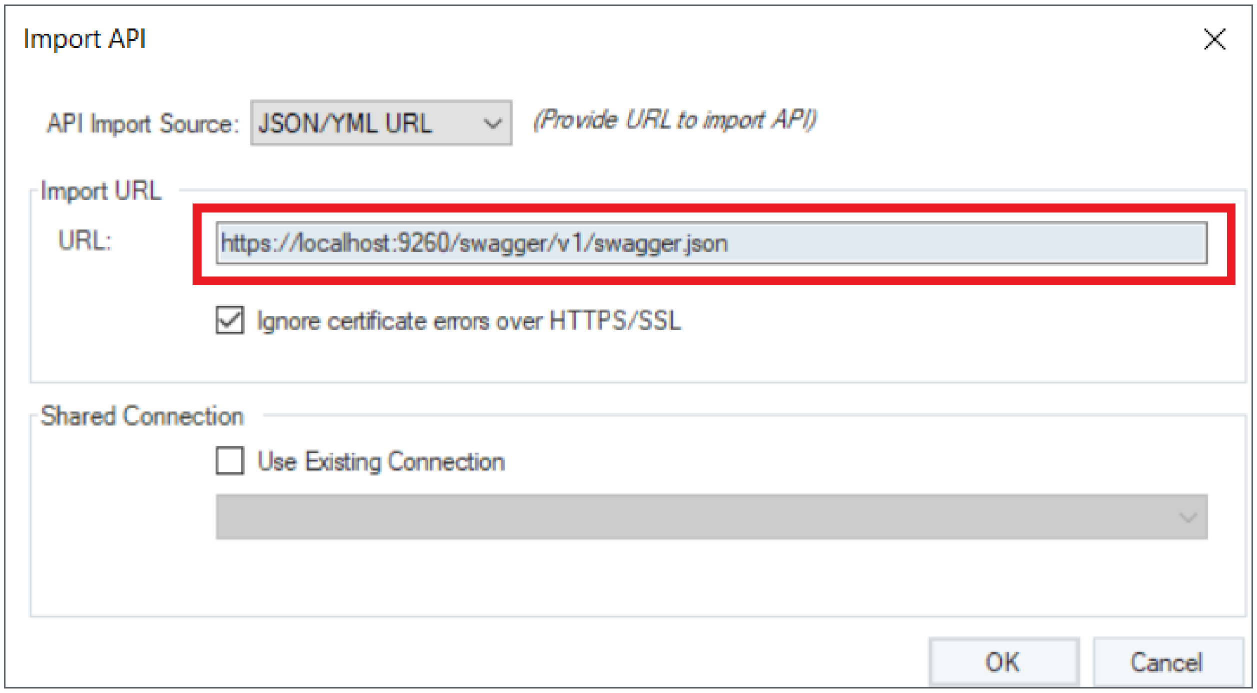


Text copied successfully.

11.11.1 Importing APIs in Centerprise

3. Click the *Import API* option in the REST API Browser and paste the Centerprise Server API path in the *URL* box. Then click *OK*.

Note: Check the "*Ignore certificate errors over HTTP/SSL*" option to avoid any certification barriers.



API Import Source: JSON/YML URL

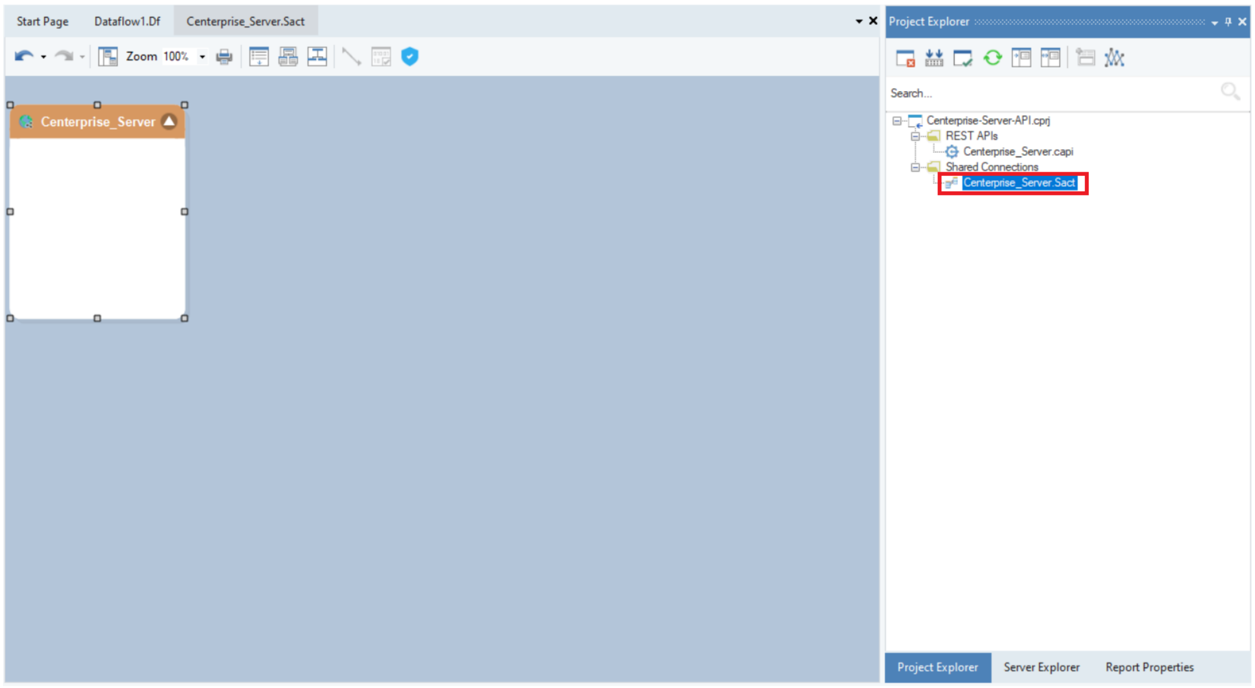
URL: https://localhost:9260/swagger/v1/swagger.json

☑ Ignore certificate errors over HTTPS/SSL

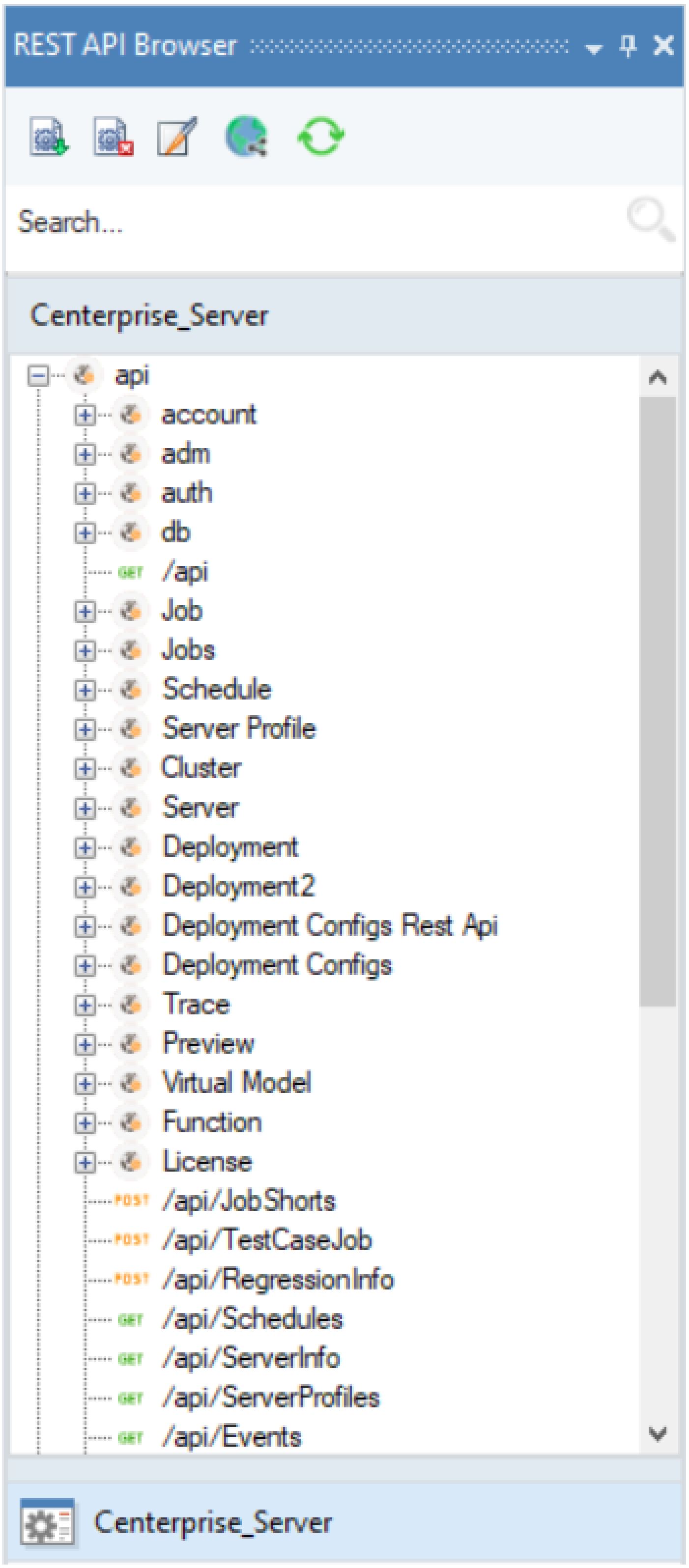
☐ Use Existing Connection

4. A wizard will appear, notifying you about the created shared action file. Click *Yes* to set it up.

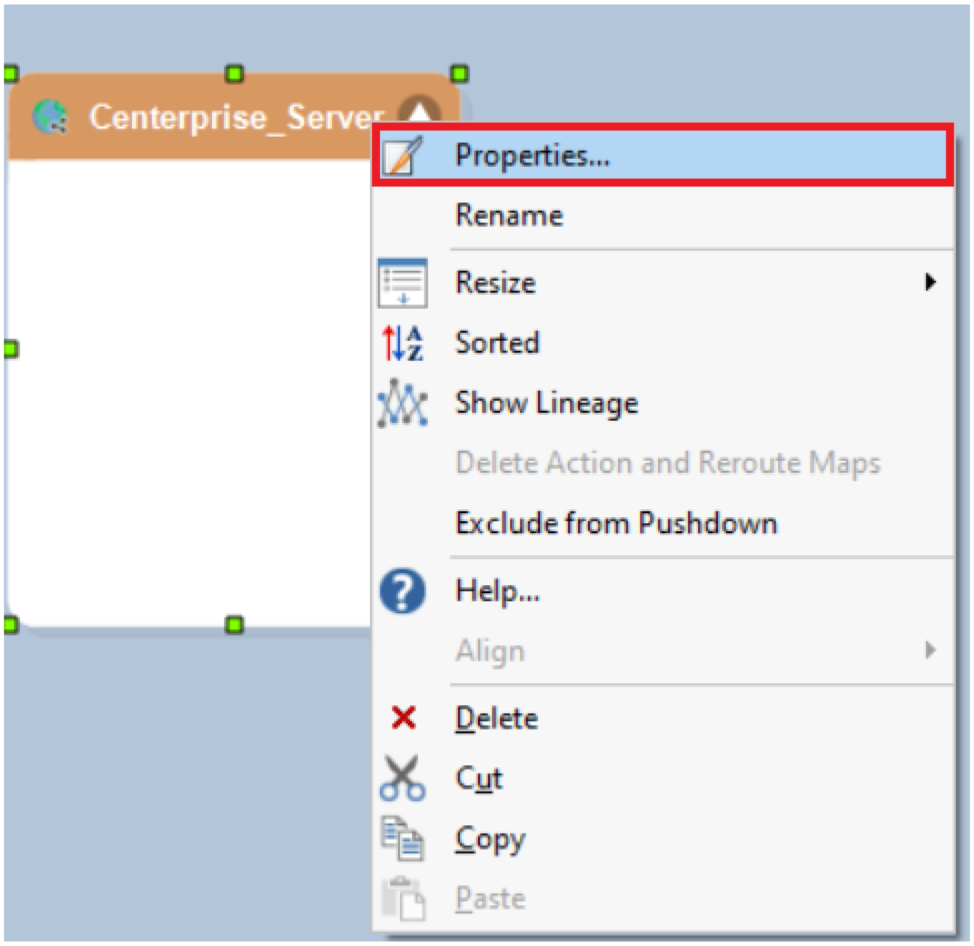
You can also click on the .sact file in Project Explorer to configure the authentication settings.



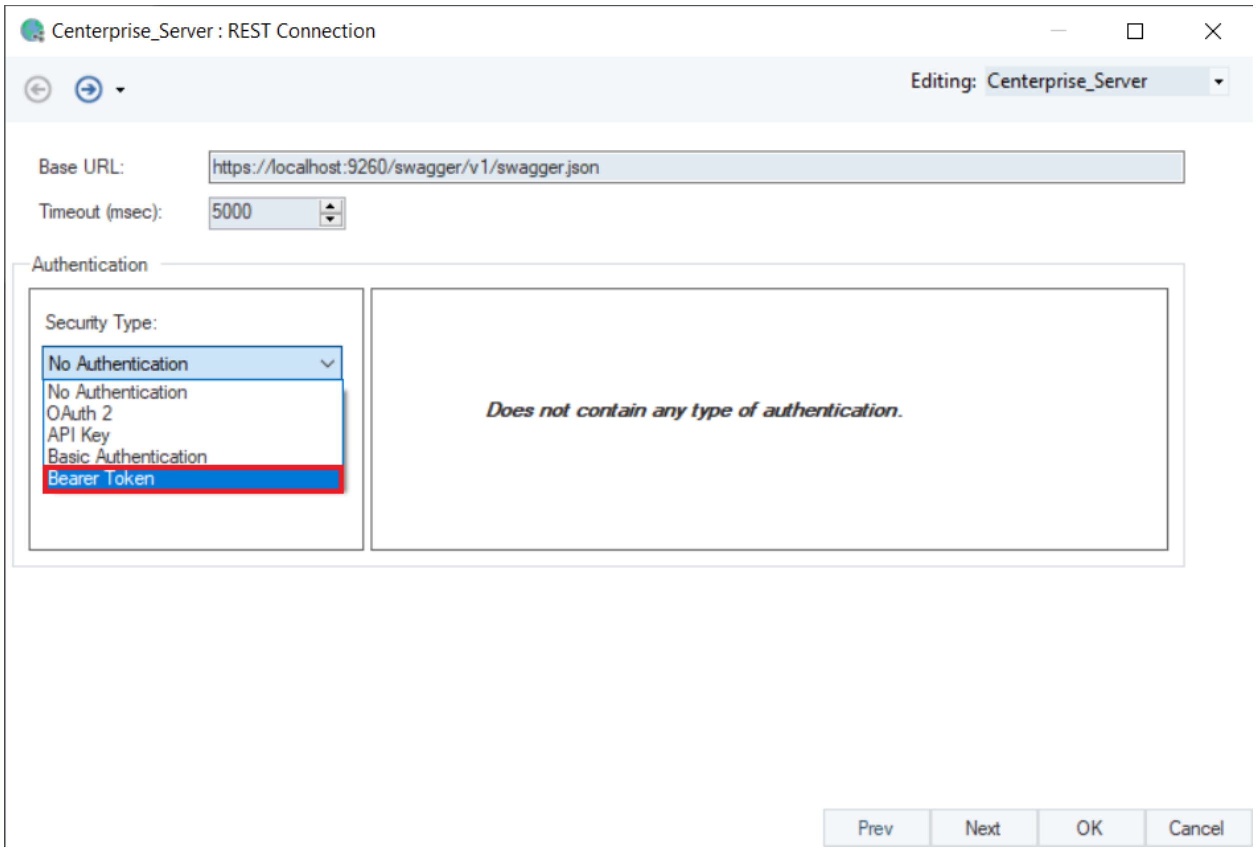
The REST API Browser will be populated with Centerprise's Server APIs, which you can use in your dataflow.



5. Right-click on the *Centerprise_Server* object and select *Properties*.



This will open the *REST Connection* screen. Select the *Security Type* as *Bearer Token*, as Centerprise Server APIs use *Bearer Token* authentication.


Centerprise_Server : REST Connection

Editing: Centerprise_Server

Base URL: https://localhost:9260/swagger/v1/swagger.json

Timeout (msec): 5000

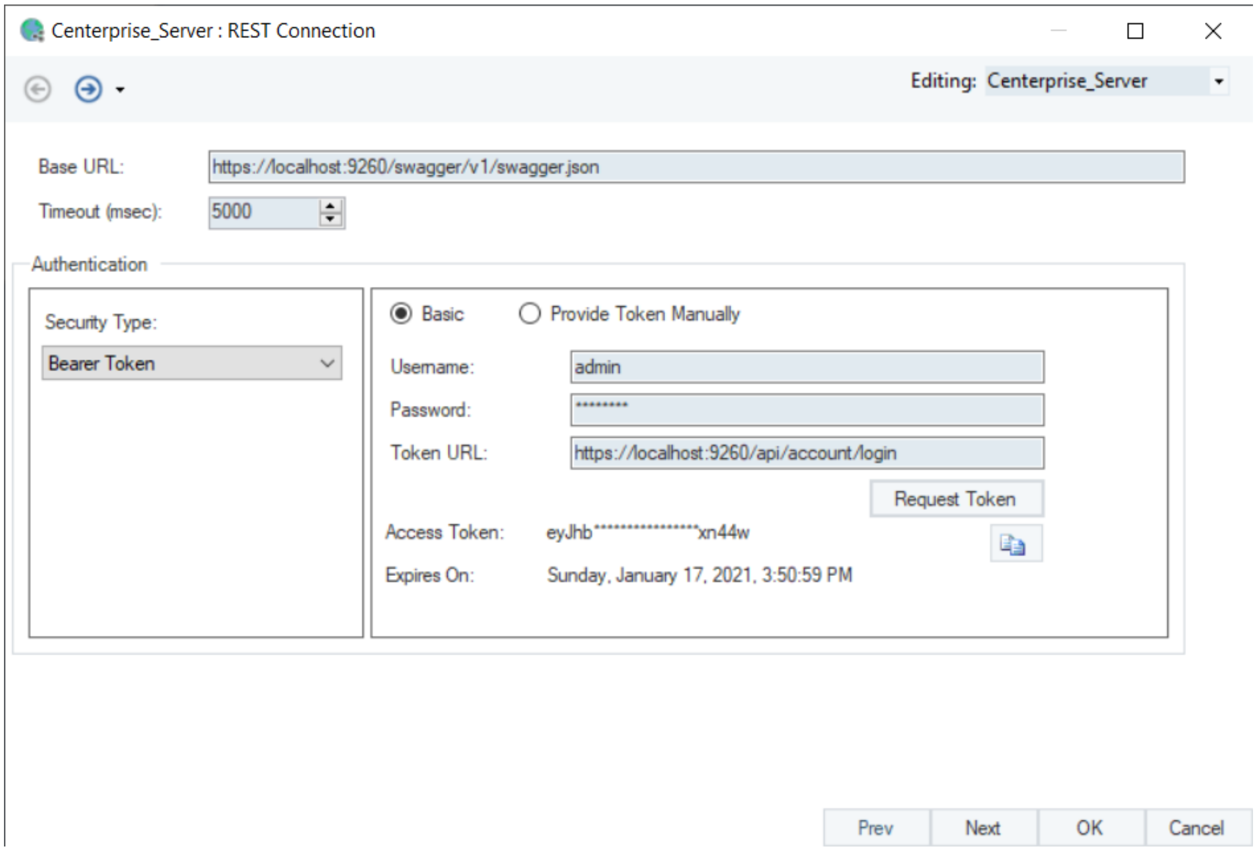
Authentication

Security Type: No Authentication

Does not contain any type of authentication.

Provide the *User Name*, *Password*, and *Token URL* for Bearer Token. Then click *Request Token* to generate a token, and click *OK*. Press Ctrl+S to save changes in the shared action file.

Note: You will have to regenerate the token if the validity period has expired.


Centerprise_Server : REST Connection
Editing: Centerprise_Server
Prev
Next
OK
Cancel
Request Token

Base URL: https://localhost:9260/swagger/v1/swagger.json
Timeout (msec): 5000

Authentication

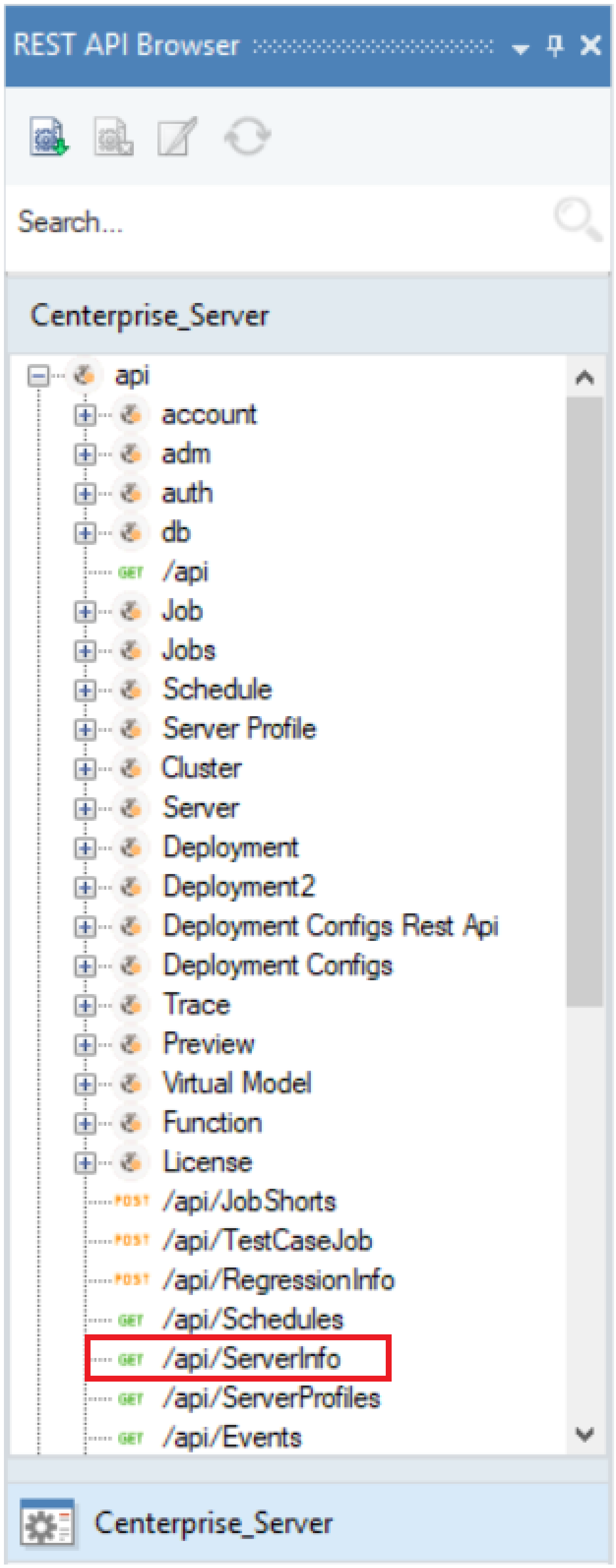
Security Type: Bearer Token

(● Basic) ○ Provide Token Manually

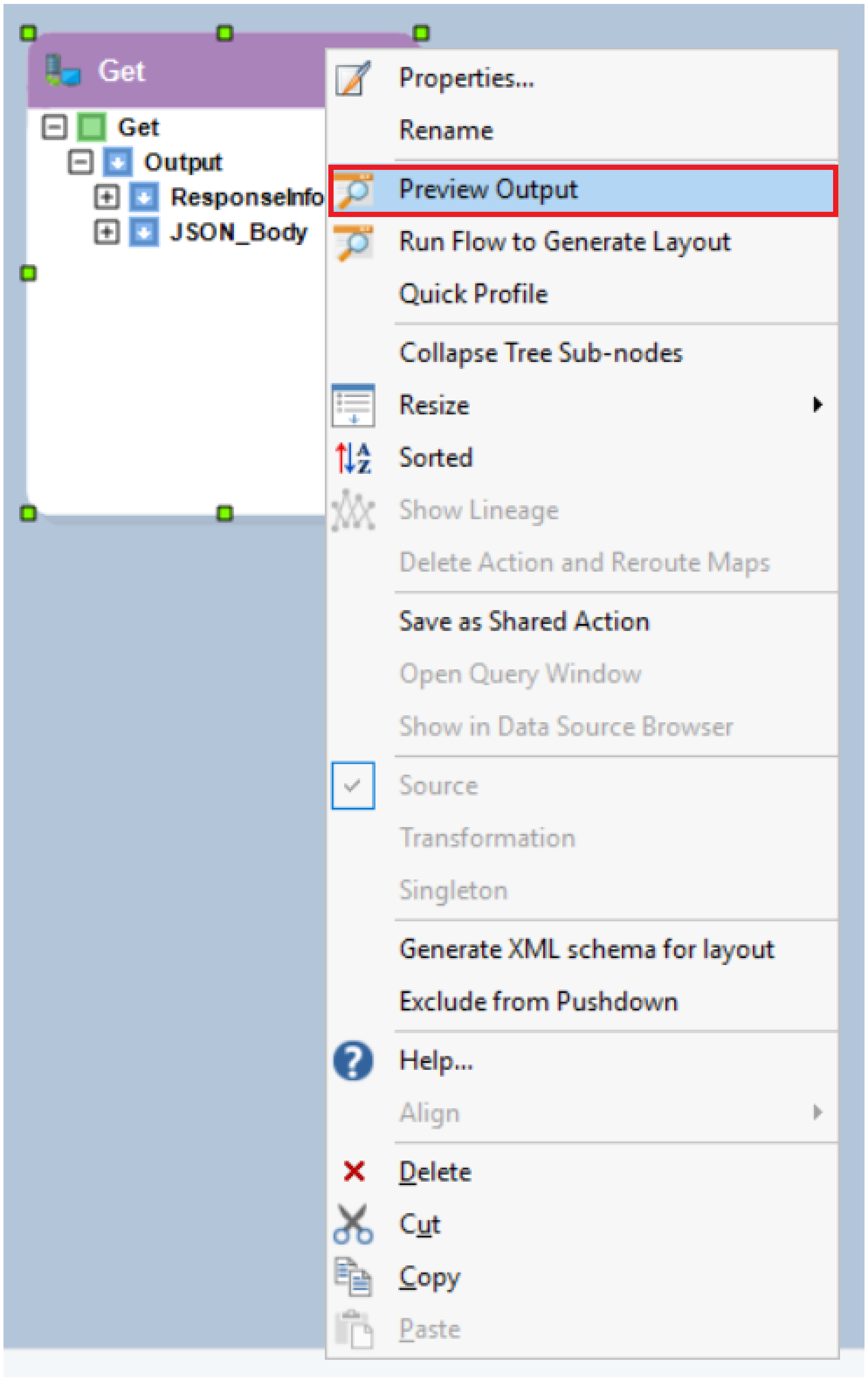
Username: admin
Password: ********
Token URL: https://localhost:9260/api/account/login

Access Token: eyJhb****************xn44w
Expires On: Sunday, January 17, 2021, 3:50:59 PM

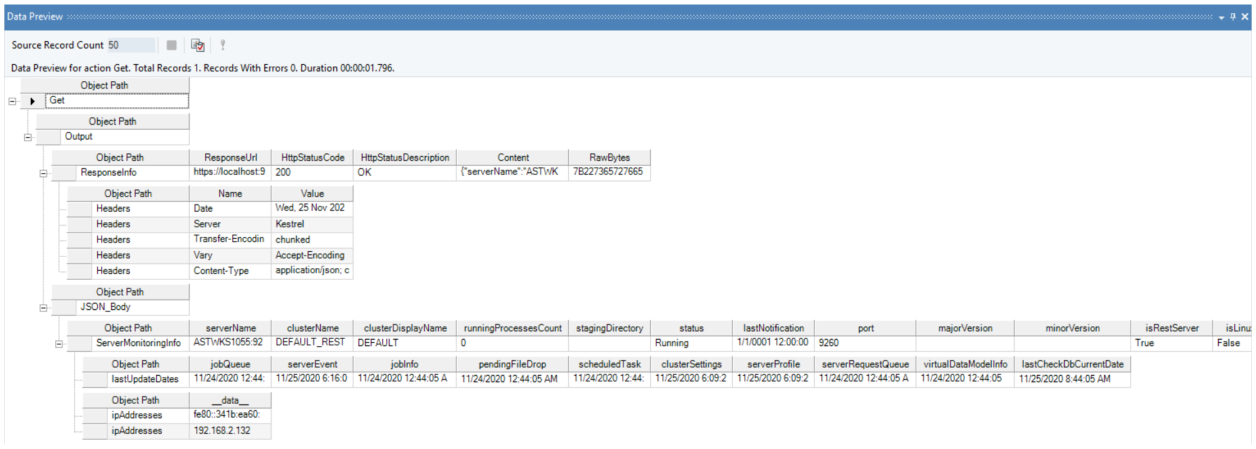
6. Now, drag-and-drop the */api/ServerInfo* from the REST API Browser to make a GET request.



7. Right-click on the object's header and select *Preview Output*.



This is how your output would look like:



Data Preview

Source Record Count 50

Data Preview for action Get. Total Records 1. Records With Errors 0. Duration 00:00:01.796.

Object Path
Get

Object Path
Output

Object Path	ResponseUrl	HttpStatusCode	HttpStatusDescription	Content	RawBytes
ResponseInfo	https://localhost:9	200	OK	{"serverName":"ASTWK	7B227365727665

Object Path	Name	Value
Headers	Date	Wed, 25 Nov 202
Headers	Server	Kestrel
Headers	Transfer-Encodin	chunked
Headers	Vary	Accept-Encoding
Headers	Content-Type	application/json; c

Object Path
JSON_Body

Object Path	serverName	clusterName	clusterDisplayName	runningProcessesCount	stagingDirectory	status	lastNotification	port	majorVersion	minorVersion	isRestServer	isLinu
ServerMonitoringInfo	ASTWKS1055:92	DEFAULT_REST	DEFAULT	0		Running	1/1/0001 12:00:00	9260			True	False

Object Path	jobQueue	serverEvent	jobInfo	pendingFileDrop	scheduledTask	clusterSettings	serverProfile	serverRequestQueue	virtualDataModelInfo	lastCheckDbCurrentDate
lastUpdateDates	11/24/2020 12:44:	11/25/2020 6:16:0	11/24/2020 12:44:05 A	11/24/2020 12:44:05 AM	11/24/2020 12:44:	11/25/2020 6:09:2	11/25/2020 6:09:2	11/24/2020 12:44:05 A	11/24/2020 12:44:05	11/25/2020 8:44:05 AM

Object Path	__data__
ipAddresses	fe80::341b:ea60:
ipAddresses	192.168.2.132

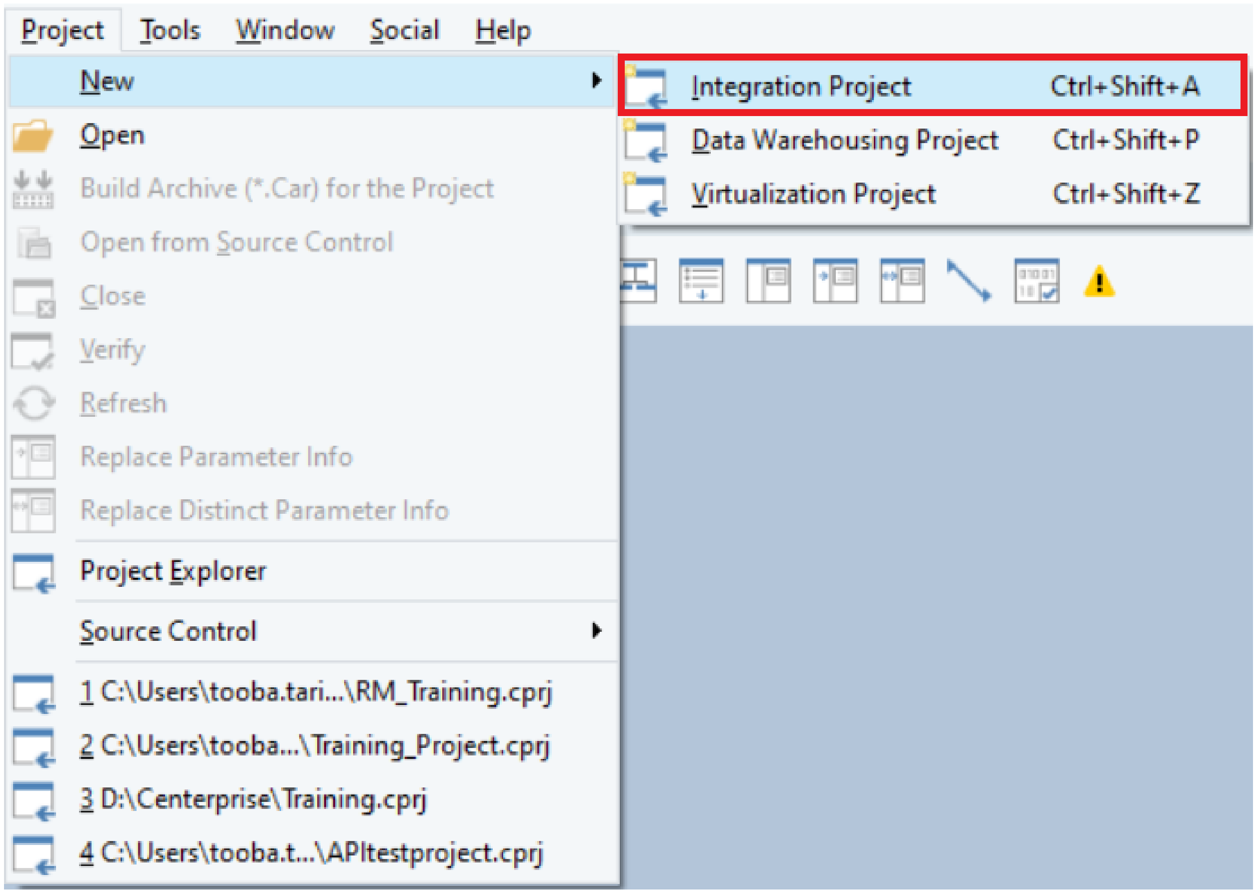
This concludes working with Centerprise's Server API in Astera Centerprise.

11.12 Authorizing Avaza APIs in Astera Centerprise

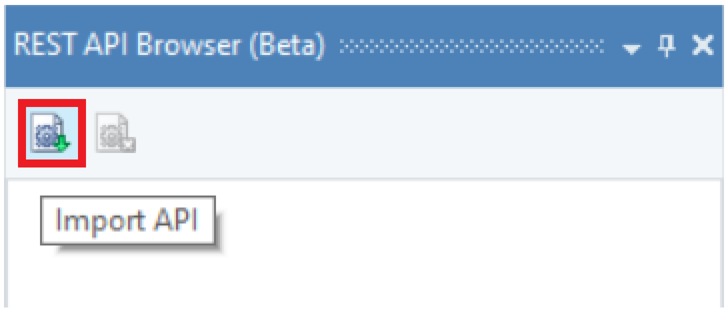
The Avaza API follows REST protocol with '*OAuth2*' authentication. It allows you to access contacts, projects, tasks, invoices and taxes. In Astera Centerprise, you can configure an Avaza API through a swagger definition using the *Import API* option in REST API Browser.

Let's go over how we can authenticate an Avaza API in Astera Centerprise.

1. Create an integration project by going to *Project* > *New* > *Integration* Project.

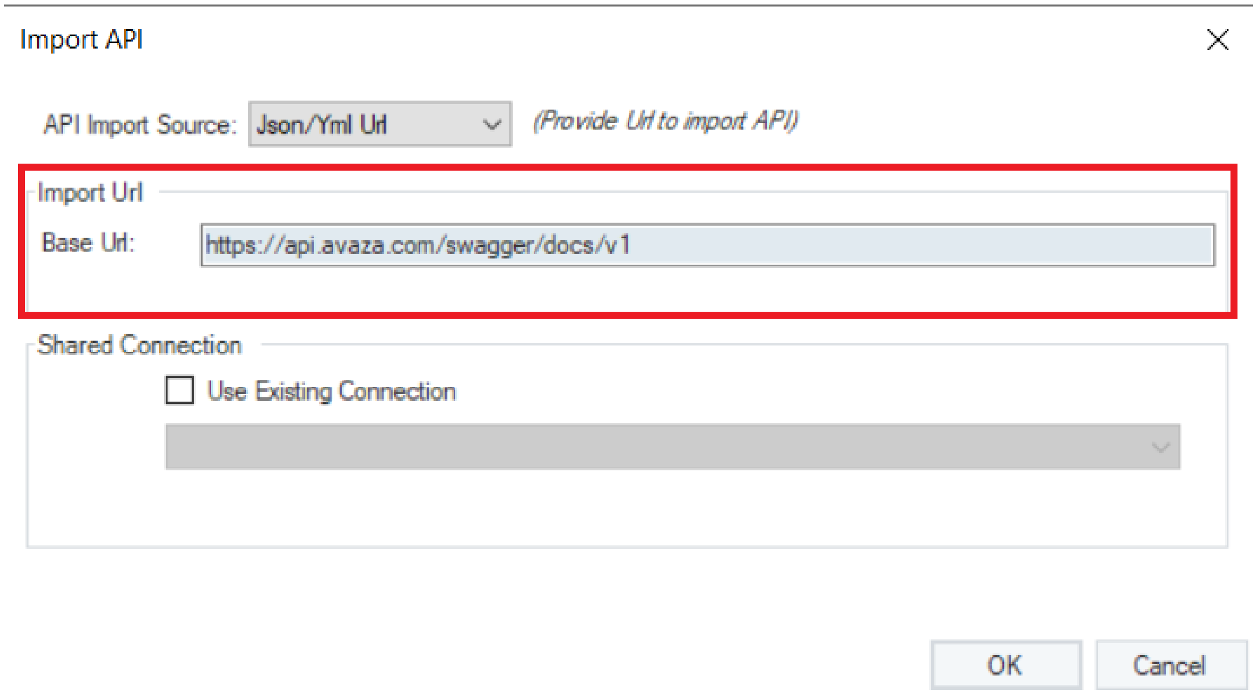

Project
Tools
Window
Social
Help
New
Open
Build Archive (*.Car) for the Project
Open from Source Control
Close
Verify
Refresh
Replace Parameter Info
Replace Distinct Parameter Info
Project Explorer
Source Control
1 C:\Users\tooba.tari...\RM_Training.cprj
2 C:\Users\tooba...\Training_Project.cprj
3 D:\Centerprise\Training.cprj
4 C:\Users\tooba.t...\APItestproject.cprj
Integration Project Ctrl+Shift+A
Data Warehousing Project Ctrl+Shift+P
Virtualization Project Ctrl+Shift+Z

2. To import Avaza API in your Centerprise client, click on the following icon.

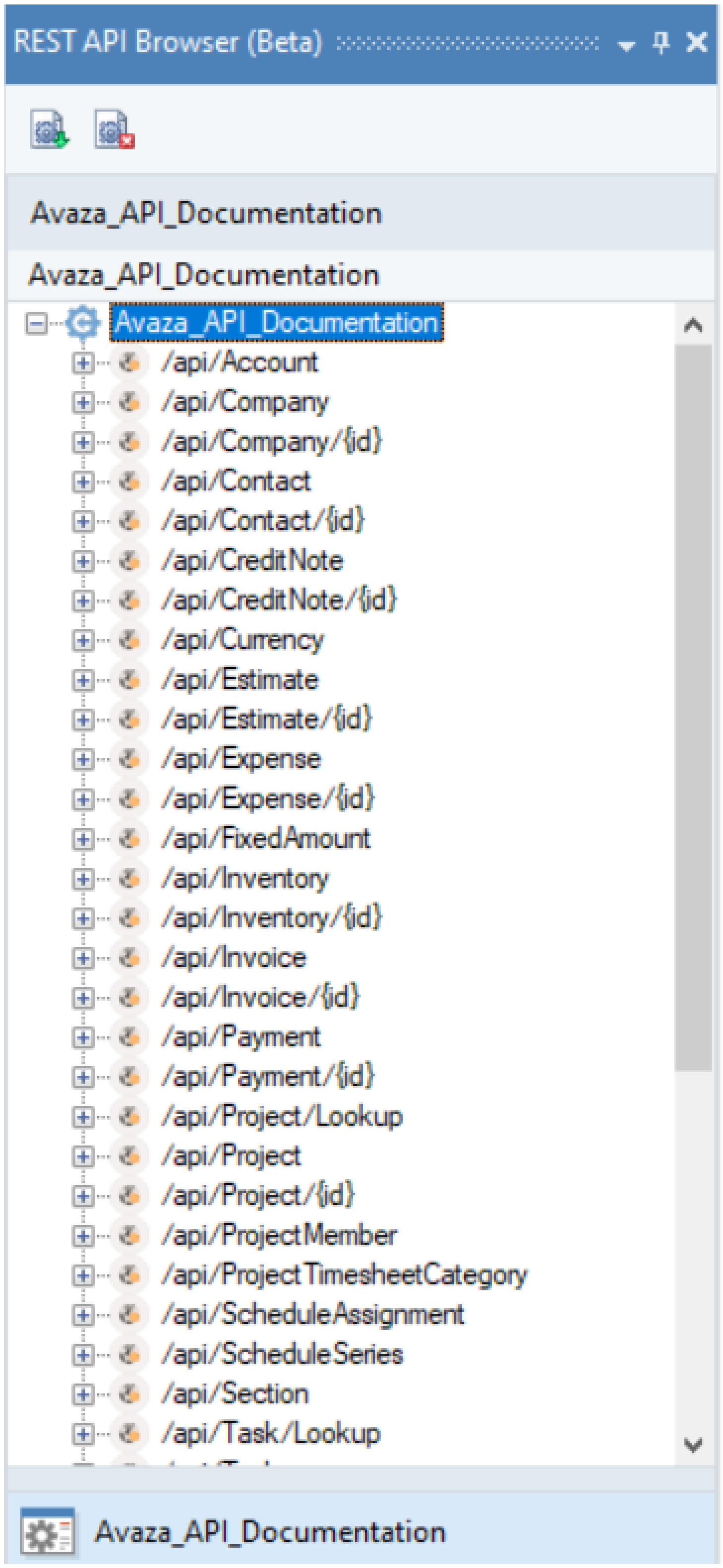


3. An *Import API* window will open. Here you will need to select your relevant import source. In this case, we will import using the *Json/Yml Url* source.

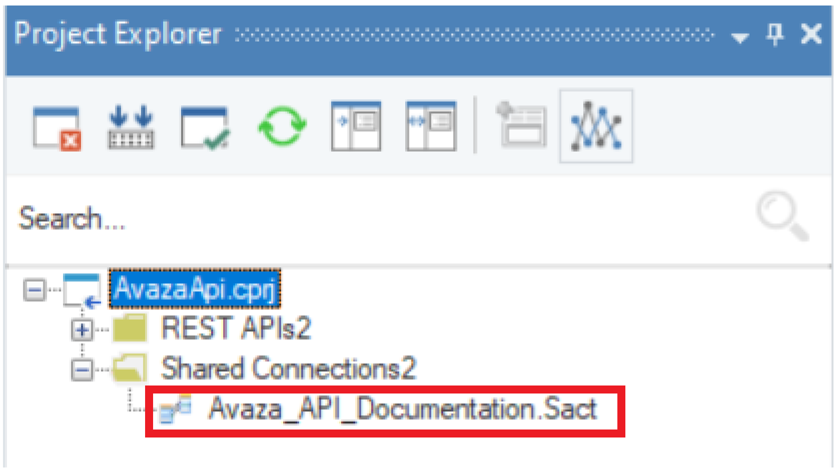
Base URL: https://api.avaza.com/swagger/docs/v1


Import API
API Import Source: Json/Yml Url
(Provide Url to import API)
Import Url
Base Url: https://api.avaza.com/swagger/docs/v1
Shared Connection
Use Existing Connection
OK
Cancel

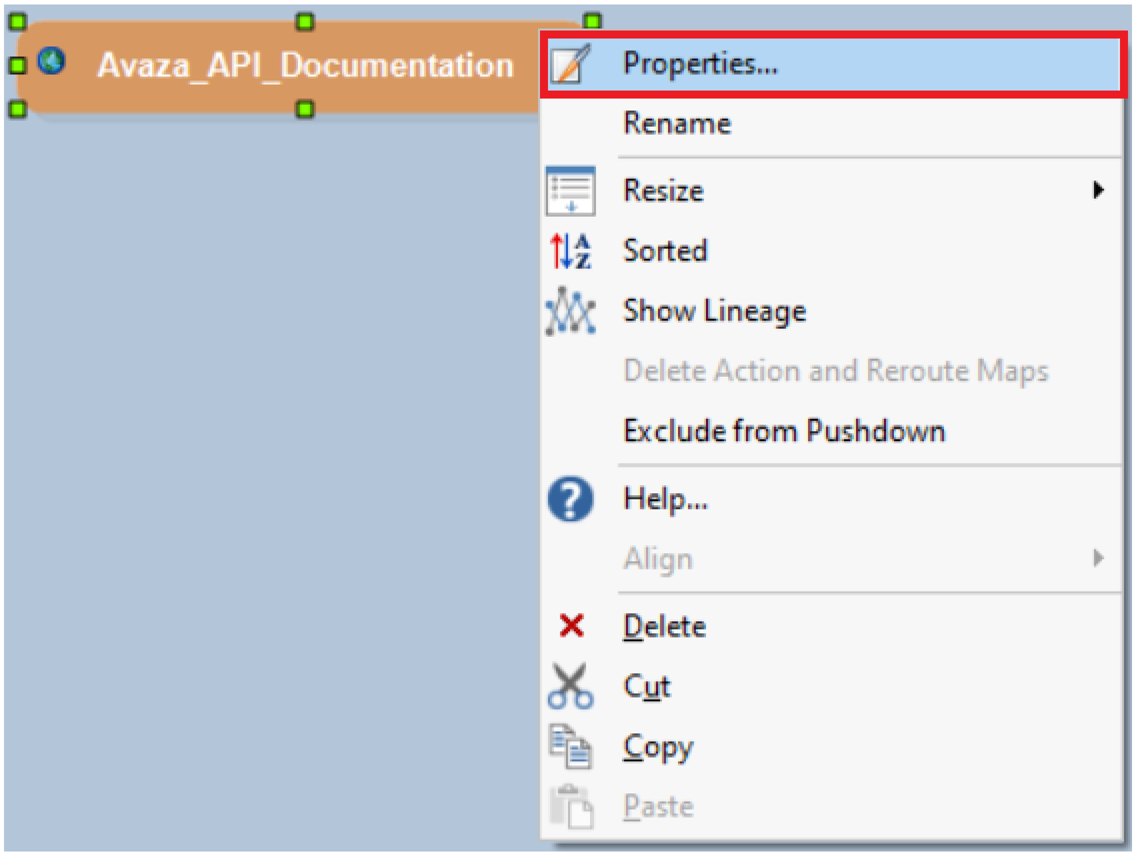
You will see that all the APIs present on Avaza's URL have been populated in the REST API Browser (Beta).



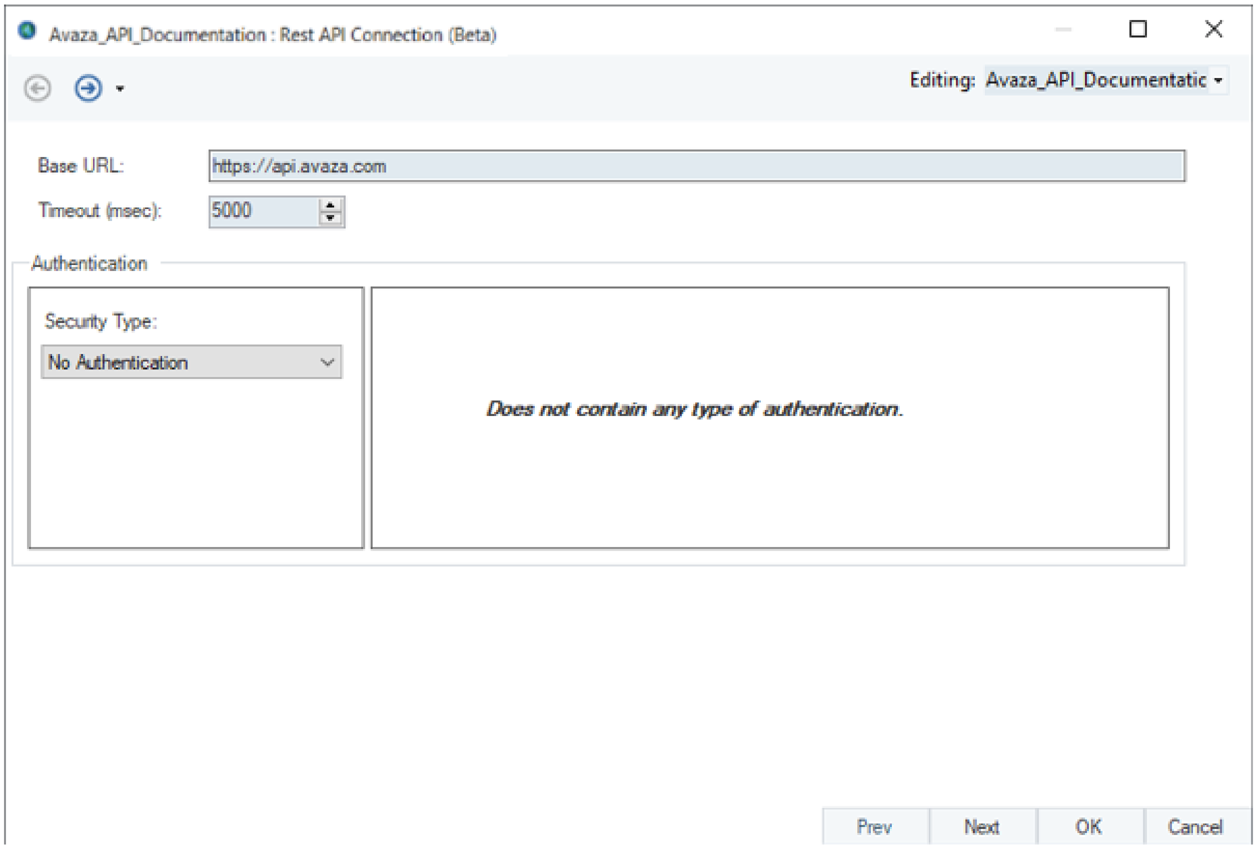
4. Now, you need to authenticate the Avaza APIs to be able to use them in your dataflow. Without authentication, you will get an error. To authenticate an API, go to the Project Explorer and double click on the API's .sact file under the *Shared Connection* node.



The Avaza .sact file will open on the designer. Now, right-click the shared action file's header and select *Properties*.



5. This will open the *REST API Connection* window where you can configure settings to authenticate Avaza API.


Avaza_API_Documentation : Rest API Connection (Beta)
Editing: Avaza_API_Documentatic
Prev
Next
OK
Cancel

Base URL: https://api.avaza.com

Timeout (msec): 5000

Authentication

Security Type: No Authentication

Does not contain any type of authentication.

Avaza uses '*OAuth 2*' authentication. In the '*OAuth 2*' *Security Type*, select one from the following *Grant Type* options:

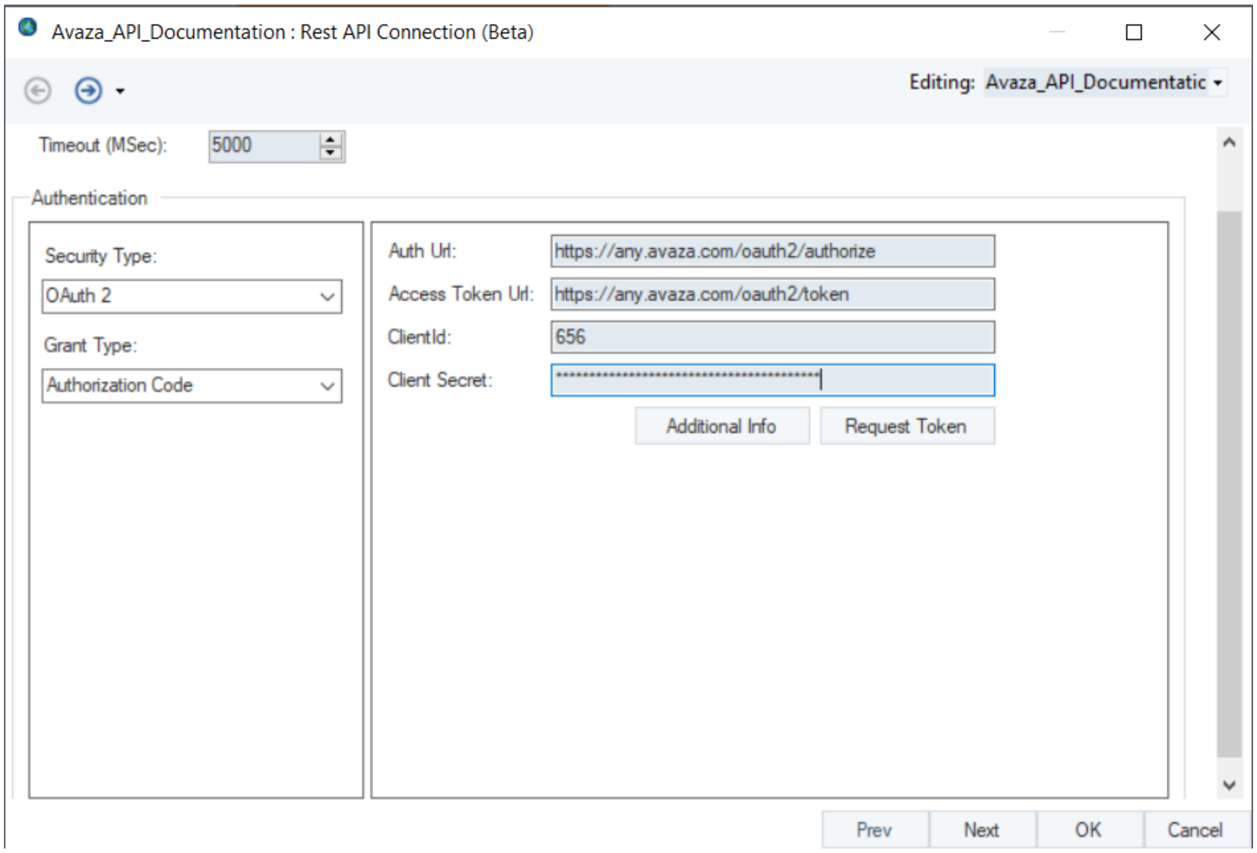
1. *Authorization Code*
2. *Implicit*

In this case, we will be using the *'Authorization Code'*.

> **Note:** Login to your Avaza account and go to *Settings* > *Developer Apps* > *Add OAuth App* to generate the *ClientID* and *Client Secret.*

Auth Url: https://any.avaza.com/oauth2/authorize

Access Token Url: https://any.avaza.com/oauth2/token


Avaza_API_Documentation : Rest API Connection (Beta)

Editing: Avaza_API_Documentatic

Timeout (MSec): 5000

Authentication

Security Type: OAuth 2

Grant Type: Authorization Code

Auth Url: https://any.avaza.com/oauth2/authorize

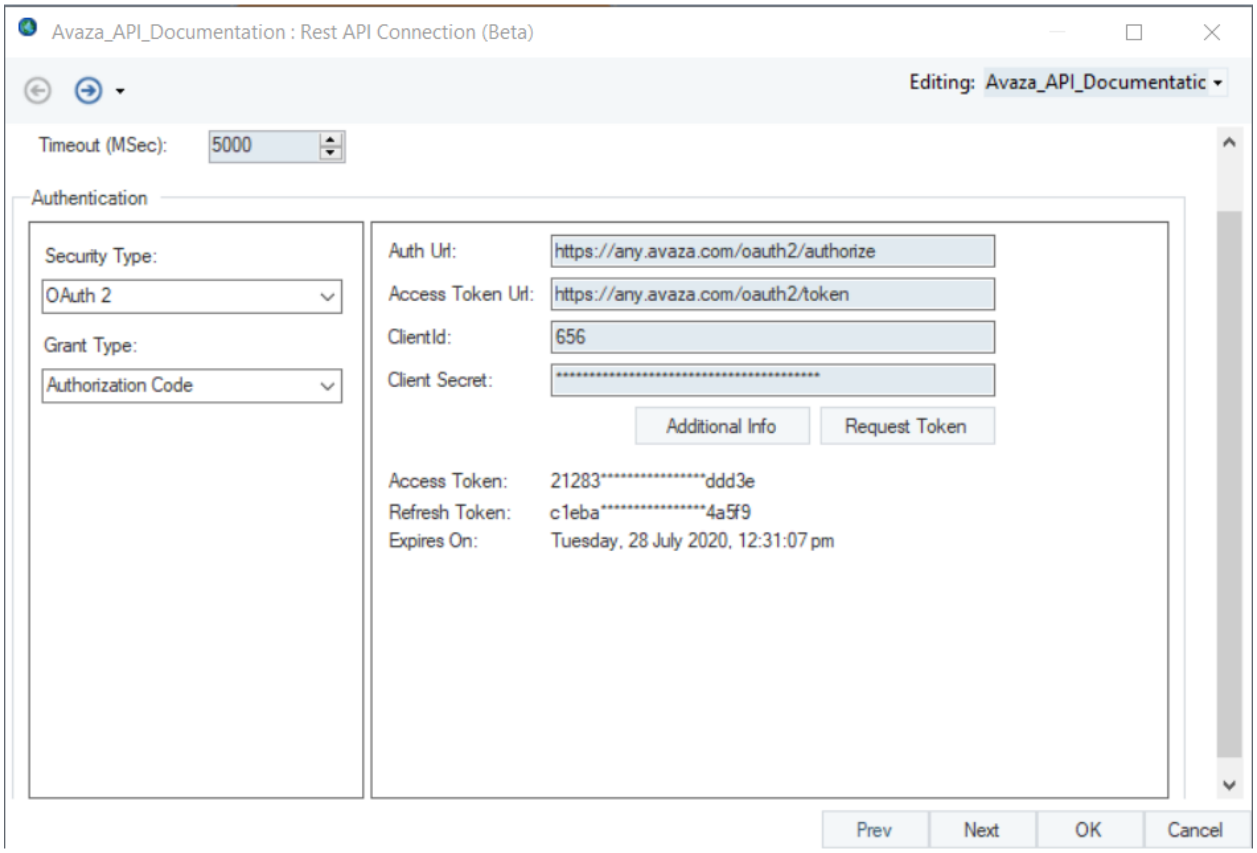
Access Token Url: https://any.avaza.com/oauth2/token

ClientId: 656

Client Secret: **

6. Now, click *Request token* to generate an access token and refresh token for Avaza.

   Note: As you click on *Request Token*, Avaza's authorization app will open where you will be required to provide your credentials to be able to generate access token and refresh token to access Avaza.


Avaza_API_Documentation : Rest API Connection (Beta)

Editing: Avaza_API_Documentatic

Timeout (MSec): 5000

Authentication

Security Type: OAuth 2

Grant Type: Authorization Code

Auth Url: https://any.avaza.com/oauth2/authorize

Access Token Url: https://any.avaza.com/oauth2/token

ClientId: 656

Client Secret: **

Access Token: 21283****************ddd3e

Refresh Token: c1eba****************4a5f9

Expires On: Tuesday, 28 July 2020, 12:31:07 pm

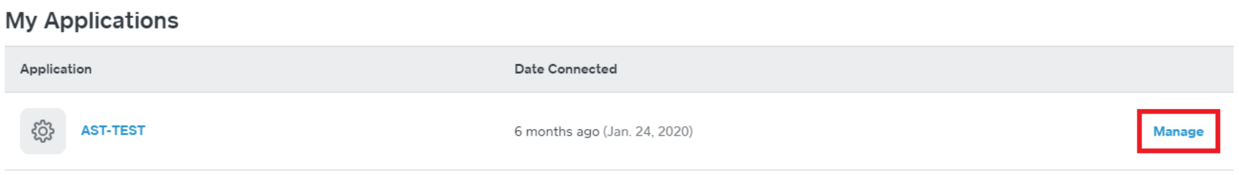
7. After authentication, save the shared action file, and you are ready to use Avaza APIs in Centerprise.

This concludes authenticating the Avaza APIs in Astera Centerprise.

11.13 Authorizing Square API in Astera Centerprise

Square API is an HTTP-based API that follows REST standards. It allows you to manage the resources of your Square account by making requests to URLs representing those resources. You can configure Square API for use in Astera Centerprise by providing its swagger definition using the *Import API* option in the REST API Browser.

1. After you have created the application in Square, go to *Manage Properties*.

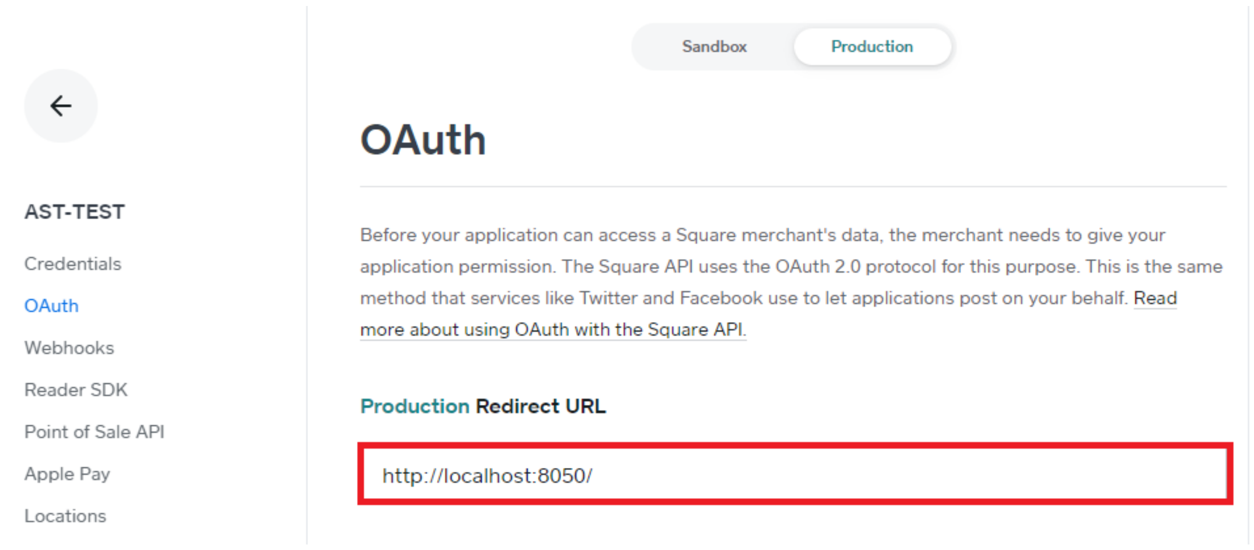

My Applications

Application	Date Connected	
AST-TEST	6 months ago (Jan. 24, 2020)	Manage

2. Now go to *OAuth* properties in *Production* tab. Here, you have to provide the *Redirect URL* for the authorization callback.

Note: Save Applicant ID and secret to use it later for Centerprise authentication.

Reference Link: https://developer.squareup.com/docs/oauth-api/overview



Sandbox Production

OAuth

AST-TEST

Credentials

OAuth

Webhooks

Reader SDK

Point of Sale API

Apple Pay

Locations

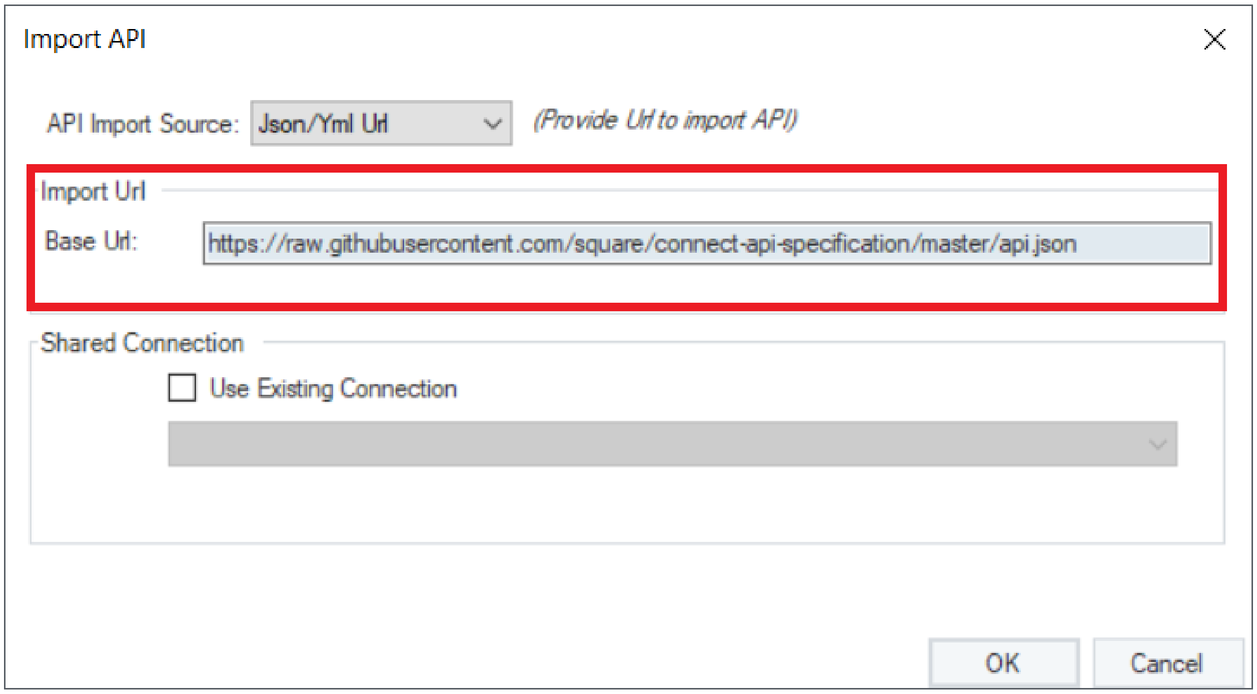
Before your application can access a Square merchant's data, the merchant needs to give your application permission. The Square API uses the OAuth 2.0 protocol for this purpose. This is the same method that services like Twitter and Facebook use to let applications post on your behalf. Read more about using OAuth with the Square API.

Production Redirect URL

http://localhost:8050/

3. Now create an integration project in Centerprise by following the instructions provided in this article. Also, import the following swagger definition in REST API Browser:

Base Url: https://raw.githubusercontent.com/square/connect-api-specification/master/api.json



Import API

API Import Source: Json/Yml Url *(Provide Url to import API)*

Import Url

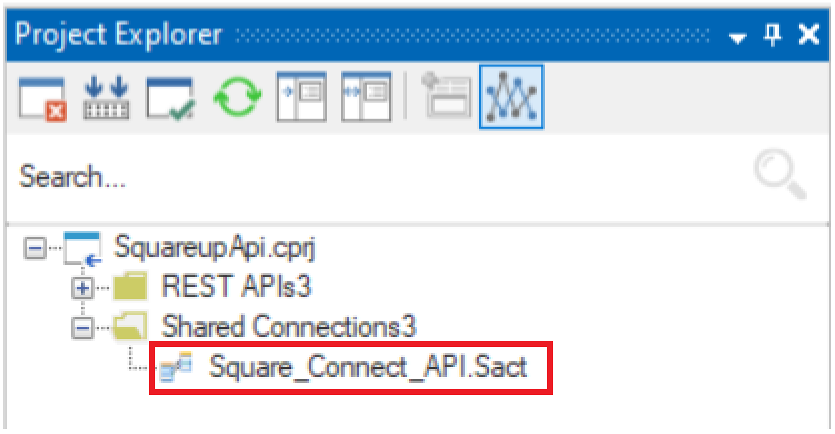
Base Url: https://raw.githubusercontent.com/square/connect-api-specification/master/api.json

Shared Connection

☐ Use Existing Connection

OK Cancel

4. Go to the Square's shared action file's (.sact) properties to authenticate it in Centerprise. Click here to learn more about how to work with APIs that require authentication.



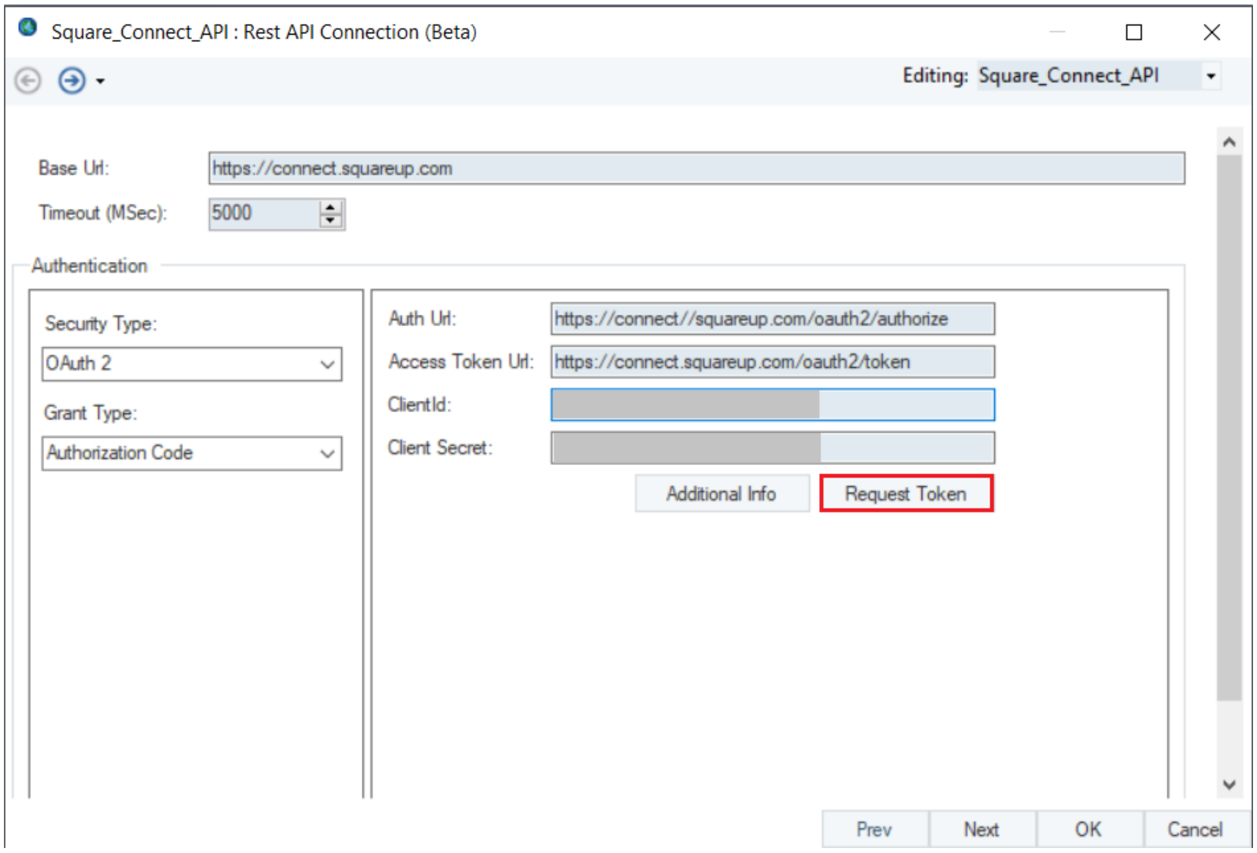
You can authorize Square API by using *Security Type OAuth 2* or *Bearer Token*. In this example, we will be authorizing using *OAuth 2*.

5. Set its *Security Type* as 'OAuth 2' and *Grant Type* as 'Authentication Code'. Provide the application ID and secret that you had saved in step 2.

Click on *Request Token* to get the access token to Square API.

Auth Url: https://connect//squareup.com/oauth2/authorize

Access Token Url: https://connect.squareup.com/oauth2/token


Square_Connect_API : Rest API Connection (Beta)

Editing: Square_Connect_API

Base Url: https://connect.squareup.com

Timeout (MSec): 5000

Authentication

Security Type: OAuth 2

Grant Type: Authorization Code

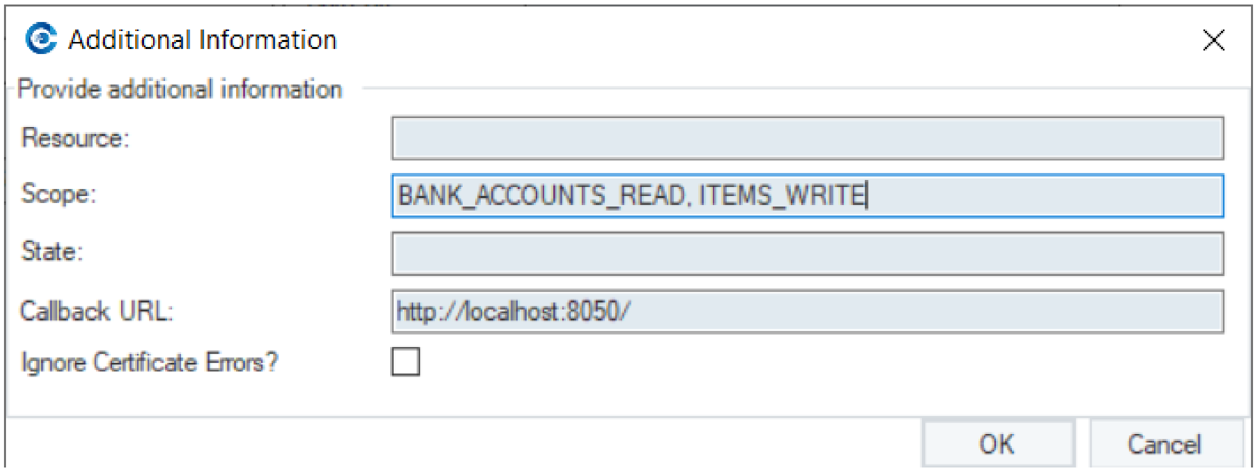
Auth Url: https://connect//squareup.com/oauth2/authorize

Access Token Url: https://connect.squareup.com/oauth2/token

ClientId:

Client Secret:

Additional Info: You can modify your authorization by mentioning names of only those permissions that you want to access from your Square account in Centerprise. In case you want to access all of them, leave the settings at default.


Additional Information

Provide additional information

Resource:

Scope: BANK_ACCOUNTS_READ, ITEMS_WRITE

State:

Callback URL: http://localhost:8050/

Ignore Certificate Errors? ☐

6. Once you get the access token, save the Shared Action file and you are ready to use Square API in Centerprise.

This concludes authenticating the Square API in Astera Centerprise.

11.14 Authorizing ActiveCampaign API in Astera Centerprise

The ActiveCampaign API is structured around REST, HTTP, and JSON. You can make requests by using URL endpoints particular to a specific resource. The resources in ActiveCampaign are represented in JSON following a conventional schema. In Astera Centerprise, you can configure an ActiveCampaign API using the *Import API* option present in the REST API Browser.

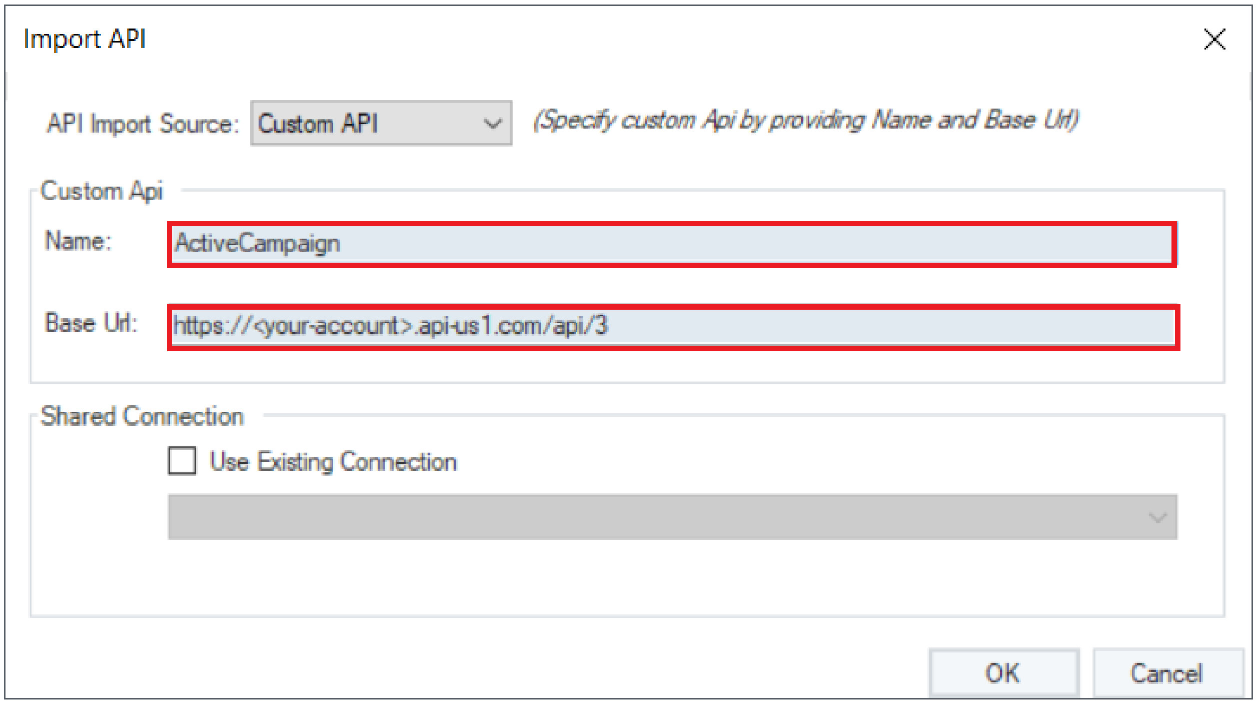
ActiveCampaign does not provide an Open API definition so we will add a request manually by using a Custom API in Centerprise.

To authorize an ActiveCampaign API in Centerprise, follow these steps:

1. Create an integration project in Centerprise by following the instructions provided in this article.

2. Create a *Custom API* and provide *Base Url*.

Reference link for Base Url: https://developers.activecampaign.com/reference#url



Import API

API Import Source: Custom API (Specify custom Api by providing Name and Base Url)

Custom Api

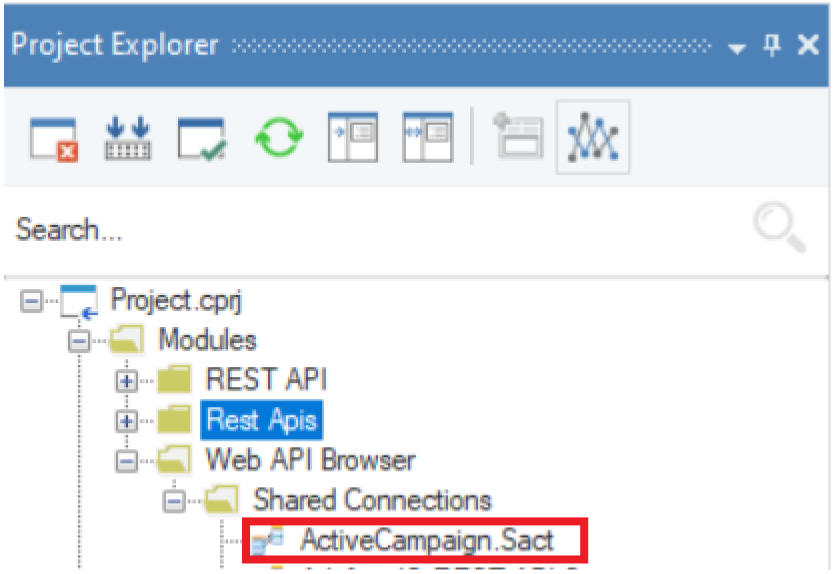
Name: ActiveCampaign

Base Url: https://<your-account>.api-us1.com/api/3

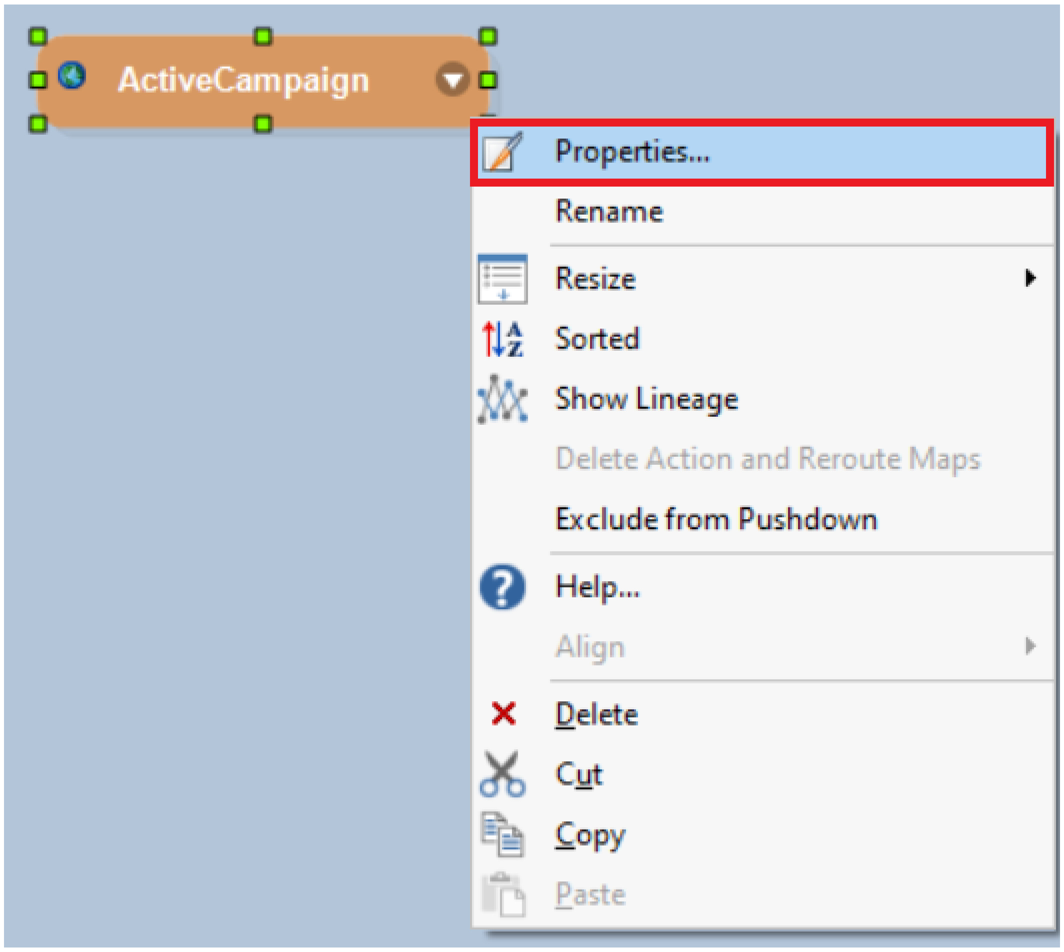
Shared Connection

☐ Use Existing Connection

3. Now, you need to authenticate the ActiveCampaign APIs to use them in your dataflow. Without authentication, you will get an error. To authenticate an API, go to the Project Explorer and double click on the API's .sact file under the *Shared Connection* node.



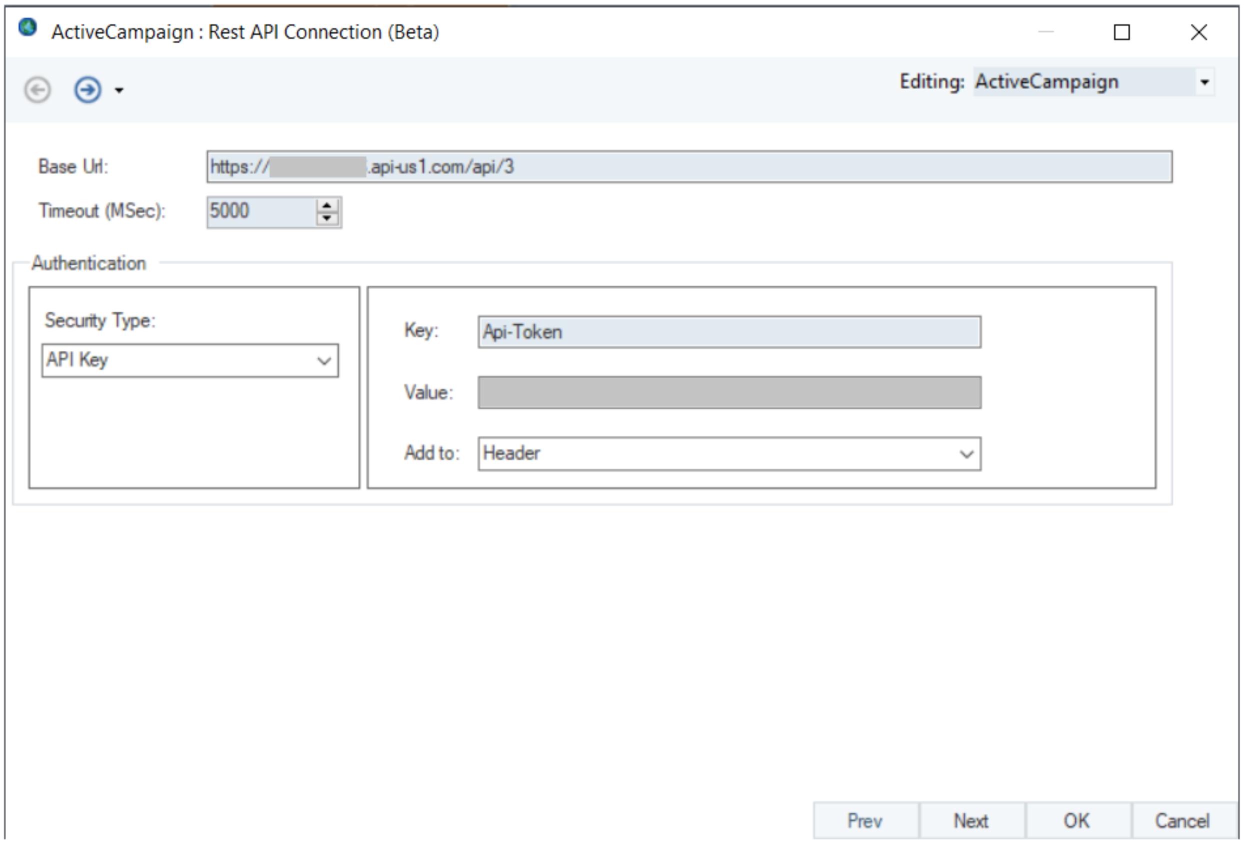
The *ActiveCampaign* .sact file will open in the designer. Now, right-click the shared action file's header and select *Properties*.



4. ActiveCampaign uses an *API Key* as *Security Type.* Specify your *Key* and *Value*.

Key: API-Token

Value: {Token}


ActiveCampaign : Rest API Connection (Beta)

Editing: ActiveCampaign

Base Url: https://[redacted].api-us1.com/api/3

Timeout (MSec): 5000

Authentication

Security Type: API Key

Key: Api-Token

Value: [redacted]

Add to: Header

5. Click *OK*, and save the shared action file (.sact).

6. Add methods in REST API Browser panel which you want to use in Centerprise by adding requests, and you are ready to use the ActiveCampaign API in Centerprise.

This concludes authorizing the ActiveCampaign API in Astera Centerprise.

11.15 Authorizing QuickBooks' API in Astera Centerprise

The QuickBooks API is a RESTful API which allows you to read or write data to and from QuickBooks. It uses '*OAuth 2*' authentication type. You can configure a QuickBooks API in Astera Centerprise by using the *Import API* option present in the REST API Browser.

QuickBooks does not provide Open API definition, so we will add the request manually by using a *Custom API* in Astera Centerprise.

We only need to follow steps from *Development* > *Create and Configure an App* from the following link:

Authentication steps: https://developer.intuit.com/app/developer/qbo/docs/build-your-first-app

Where the *Redirect Url* used in step 7 in the above link for Centerprise would be:

Redirect Url for Centerprise Server: http://{Server_Name}:8050/)

Note: Save ClientID and secret to use it afterwards in Centerprise authentication

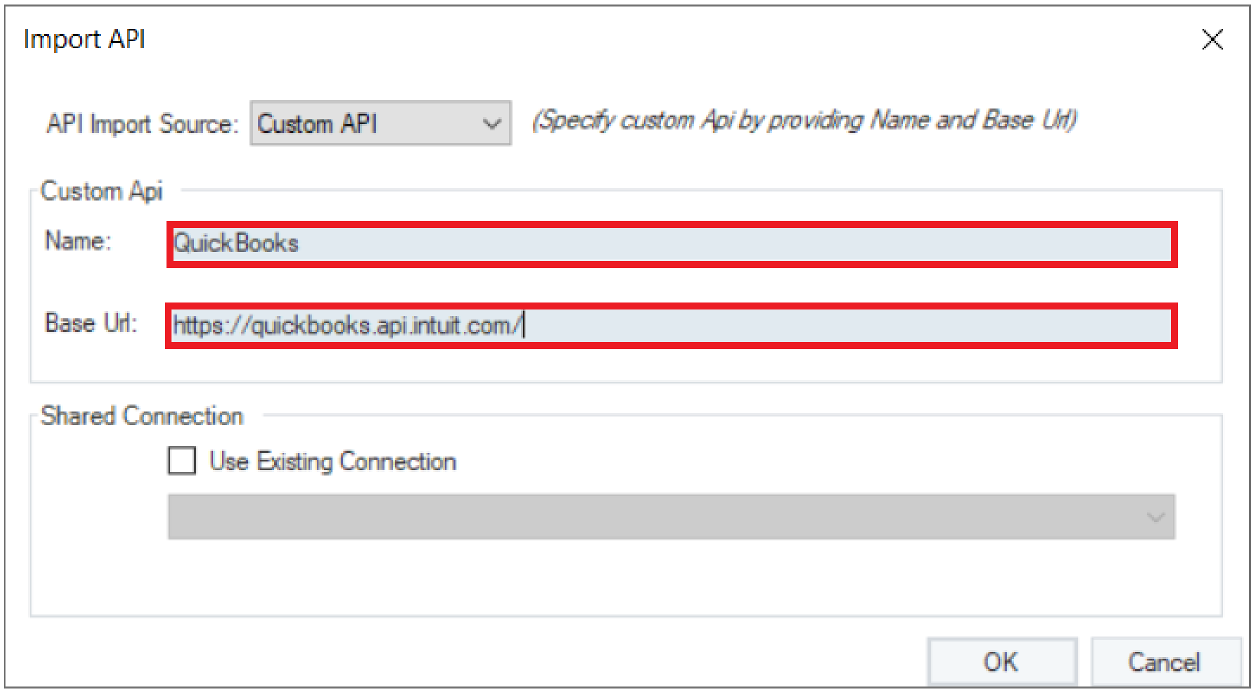
11.15.1 Follow these steps to authorize QuickBooks' API in Astera Centerprise:

1. Create an integration project in Centerprise by following the instructions provided in this article.

2. Create a *Custom API* and provide a *Name* and *Base Url*.

Base Url (Sandbox): https://sandbox-quickbooks.api.intuit.com

Base Url (Production): URL:https://quickbooks.api.intuit.com



Import API

API Import Source: Custom API (Specify custom Api by providing Name and Base Url)

Custom Api

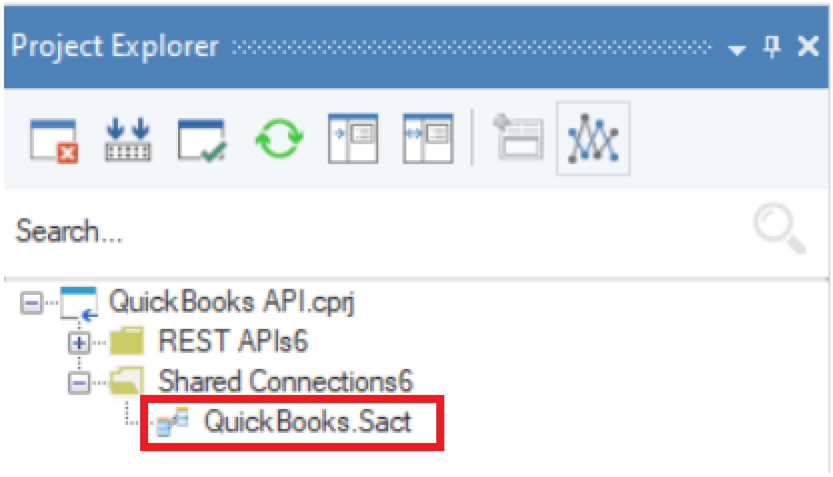
Name: QuickBooks

Base Url: https://quickbooks.api.intuit.com/

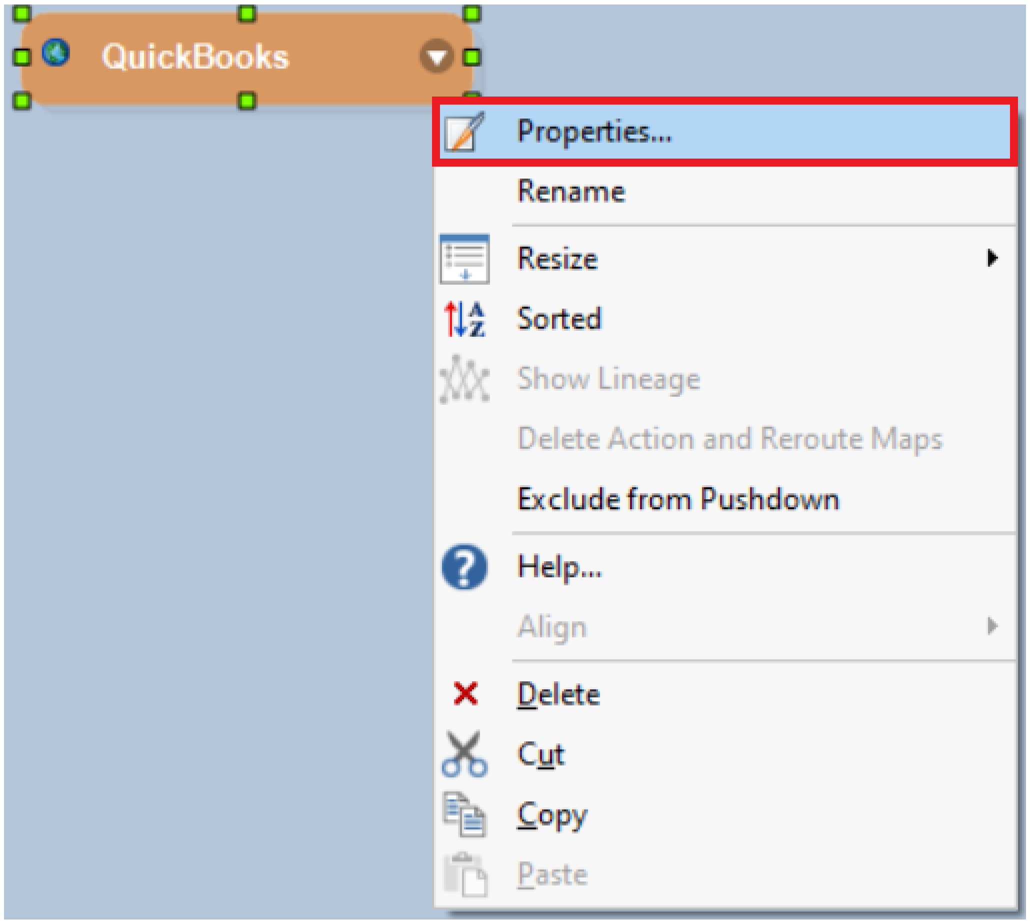
Shared Connection

☐ Use Existing Connection

3. Now, you need to authenticate QuickBooks APIs to be able to use them in your dataflow. Without authentication, you will get an error. To authenticate an API, go to the Project Explorer and double click on the API's .sact file under the *Shared Connection* node.



The *QuickBooks* .sact file will open in the designer. Now, right click on the Shared Action file's header and select *Properties*.



4. QuickBooks uses ‘*OAuth 2*’ *Security Type* with *Grant Type*, ‘*Authentication Code*’.

Auth Url: https://appcenter.intuit.com/connect/oauth2

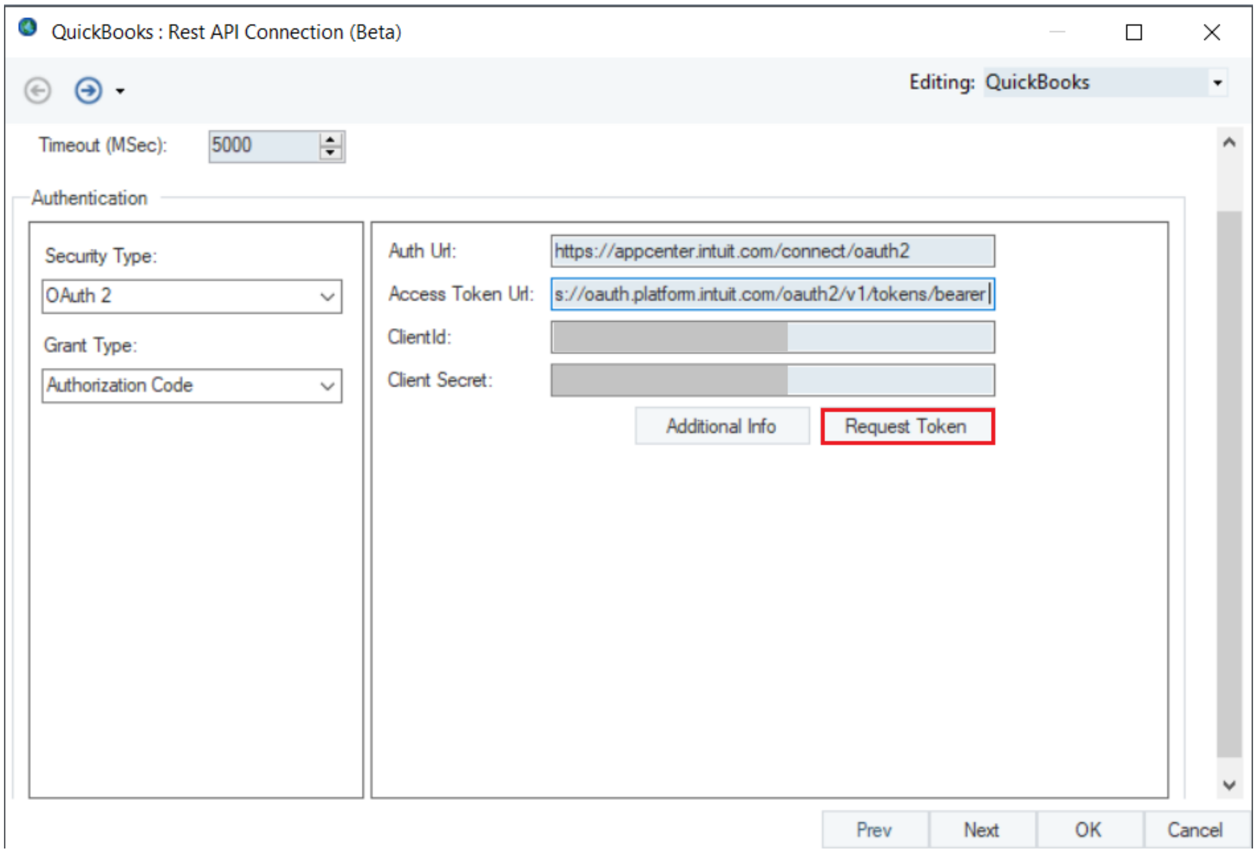
Token Url: https://oauth.platform.intuit.com/oauth2/v1/tokens/bearer

ClientID: {ClientID}

Client Secret: {Client_Secret}

Scope: {Scope}

State: {State}


QuickBooks : Rest API Connection (Beta)

Editing: QuickBooks

Timeout (MSec): 5000

Authentication

Security Type: OAuth 2

Grant Type: Authorization Code

Auth Url: https://appcenter.intuit.com/connect/oauth2

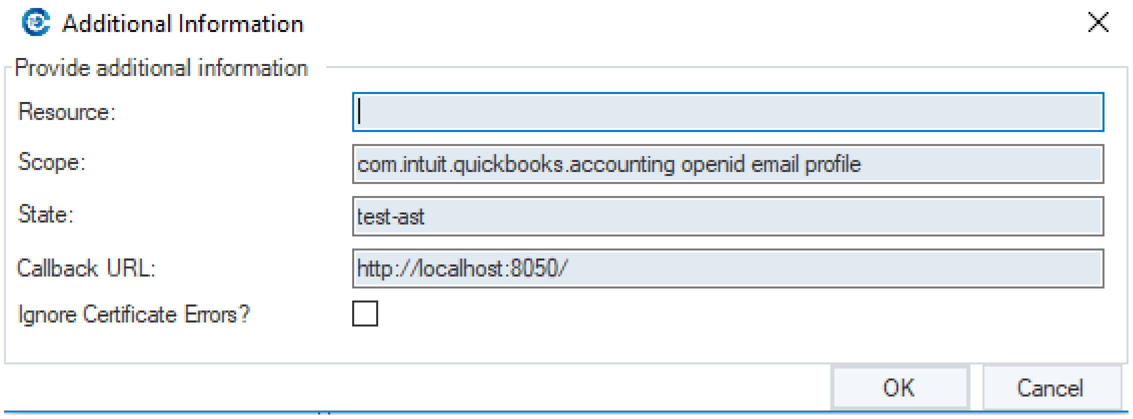
Access Token Url: s://oauth.platform.intuit.com/oauth2/v1/tokens/bearer

ClientId:

Client Secret:

Additional Info - You can modify the authorization by mentioning names of only those permissions that you want to access from QuickBooks in Centerprise.

> **Note:** While working with QuickBooks APIs, it is necessary to specify *Scope* and *State* to generate the access token.


Additional Information

Provide additional information

Resource:

Scope: com.intuit.quickbooks.accounting openid email profile

State: test-ast

Callback URL: http://localhost:8050/

Ignore Certificate Errors? ☐

5. Click *OK*, and save the Shared Action file (.sact).

6. Add methods in the REST API Browser which you want to access in Centerprise by adding requests and you are ready to use QuickBooks APIs in Centerprise.

Reference Link: https://developer.intuit.com/app/developer/qbo/docs/api/accounting/most-commonly-used/account

This concludes authorizing a QuickBooks API in Astera Centerprise.

11.16 Accessing Centerprise's Server APIs Through a Third-Party Tool

Astera Centerprise provides you with the flexibility to execute your jobs through a third-party tool, without using the Centerprise client. Let's learn how to achieve this in the article below.

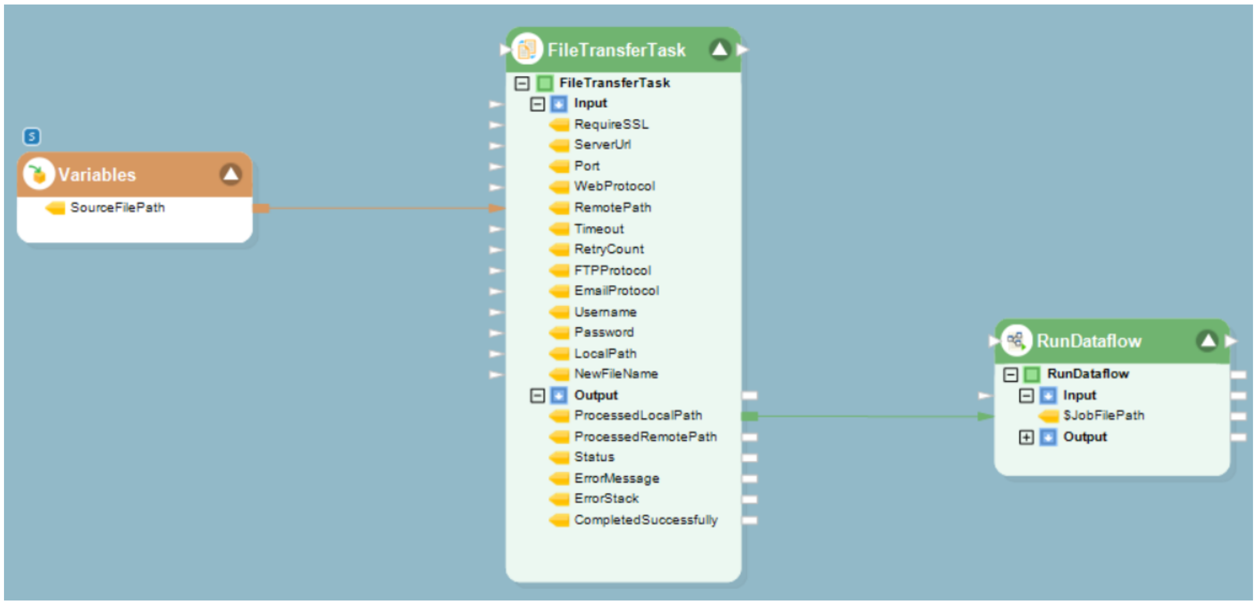
11.16.1 Use Case

In this use case, we have our Centerprise client on a local machine and server installed on a virtual machine. Instead of using Centerprise client, we will use Postman as a third-party tool to send REST requests to the server in order to execute the job.

Workflow in Centerprise

The workflow document in Centerprise consists of a *Variables* object, a *FileTransferTask* object and a *RunDataflow* object.

We will pass the name of the file that we want to download and process to the *FileTransferTask* from the *Variables* object. The *Variables* object takes an input from the REST call sent through Postman, and passes it to FTP to download the file with that name. We then pass the file path of the downloaded file to the *RunDataflow* object.

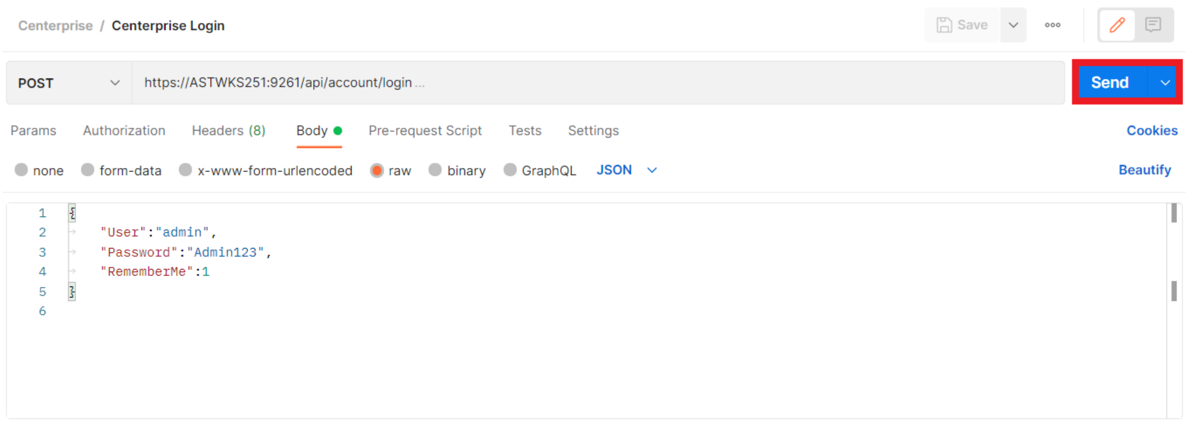


In the following section, we will cover a step-by-step overview of how you can achieve this.

11.16.2 How to Execute a Job Using Postman

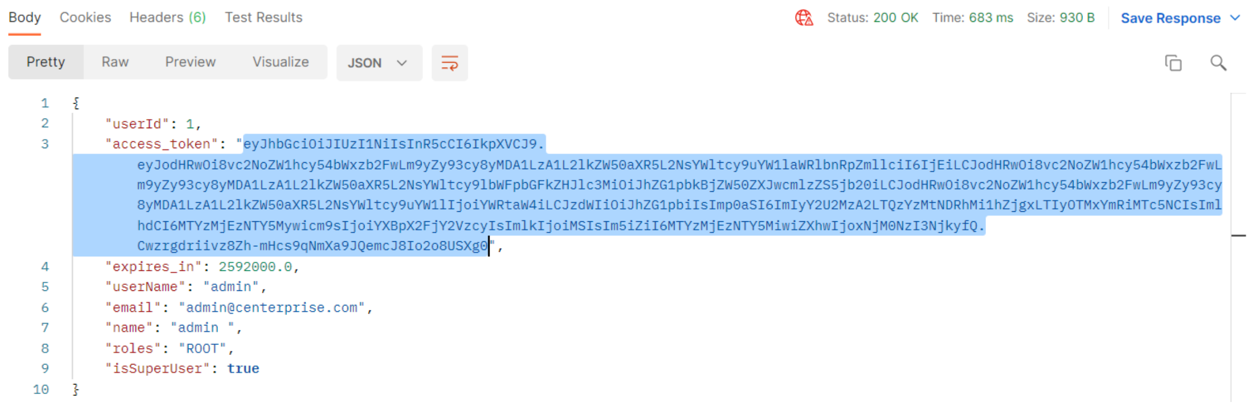
1. We will make the first API call for logging into the Centerprise server to generate an access token. Provide the following credentials in the request body and click on *Send*.

- *User*: admin
- *Password*: Admin123
- *RememberMe*: 1



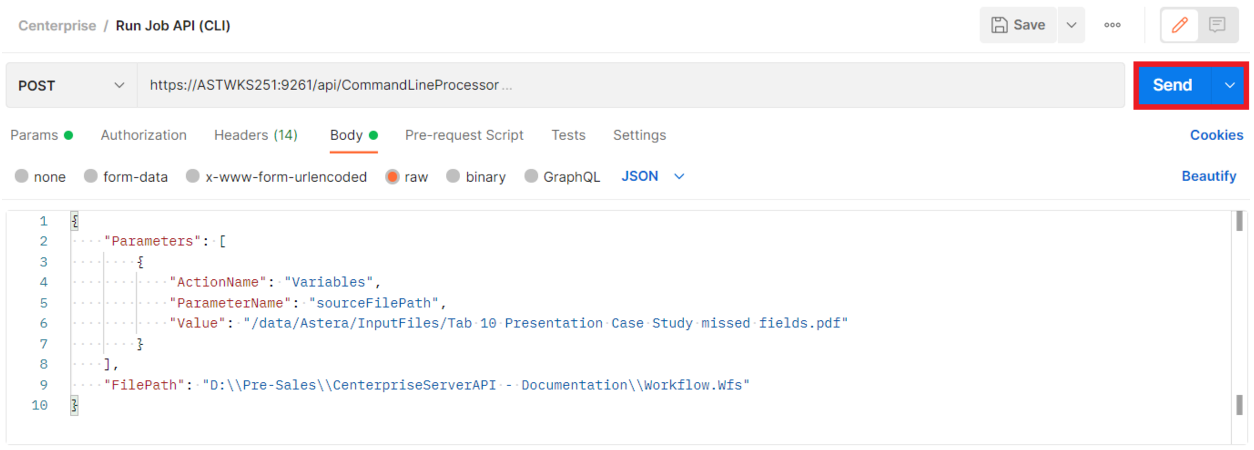
```
{
    "User":"admin",
    "Password":"Admin123",
    "RememberMe":1
}
```

Centerprise server will provide you with an access token in response.



```
{
    "userId": 1,
    "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJodHRwOi8vc2NoZW1hcy54bWxzb2FwLm9yZy93cy8yMDA1LzA1L2lkZW50aXR5L2NsYWltcy9uYW1laWRlbnRpZmllciI6IjEiLCJodHRwOi8vc2NoZW1hcy54bWxzb2FwL
        m9yZy93cy8yMDA1LzA1L2lkZW50aXR5L2NsYWltcy9lbWFpbGFkZHJlc3MiOiJhZG1pbkBjZW50ZXJwcmlzZS5jb20iLCJodHRwOi8vc2NoZW1hcy54bWxzb2FwLm9yZy93cy
        8yMDA1LzA1L2lkZW50aXR5L2NsYWltcy9uYW1lIjoiYWRtaW4iLCJzdWIiOiJhZG1pbiIsImp0aSI6ImIyY2U2MzA2LTQzYzMtNDRhMi1hZjgxLTIyOTMxYmRiMTc5NCIsIml
        hdCI6MTYzMjEzNTY5Mywicm9sIjoiYXBpX2FjY2VzcyIsImlkIjoiMSIsIm5iZiI6MTYzMjEzNTY5MiwiZXhwIjoxNjM0NzI3NjkyfQ.
        Cwzrgdriivz8Zh-mHcs9qNmXa9JQemcJ8Io2o8USXg0",
    "expires_in": 2592000.0,
    "userName": "admin",
    "email": "admin@centerprise.com",
    "name": "admin ",
    "roles": "ROOT",
    "isSuperUser": true
}
```

2. In the second step, we will send the path of the file that we want to download from FTP, in the form of a string, to the *Variables* object.



```
{
    "Parameters": [
        {
            "ActionName": "Variables",
            "ParameterName": "sourceFilePath",
            "Value": "/data/Astera/InputFiles/Tab 10 Presentation Case Study missed fields.pdf"
        }
    ],
    "FilePath": "D:\\Pre-Sales\\CenterpriseServerAPI - Documentation\\Workflow.Wfs"
}
```

In the parameters:

- *ActionName*: Variables

  Name of the object present inside the workflow to which the name of the file will be passed

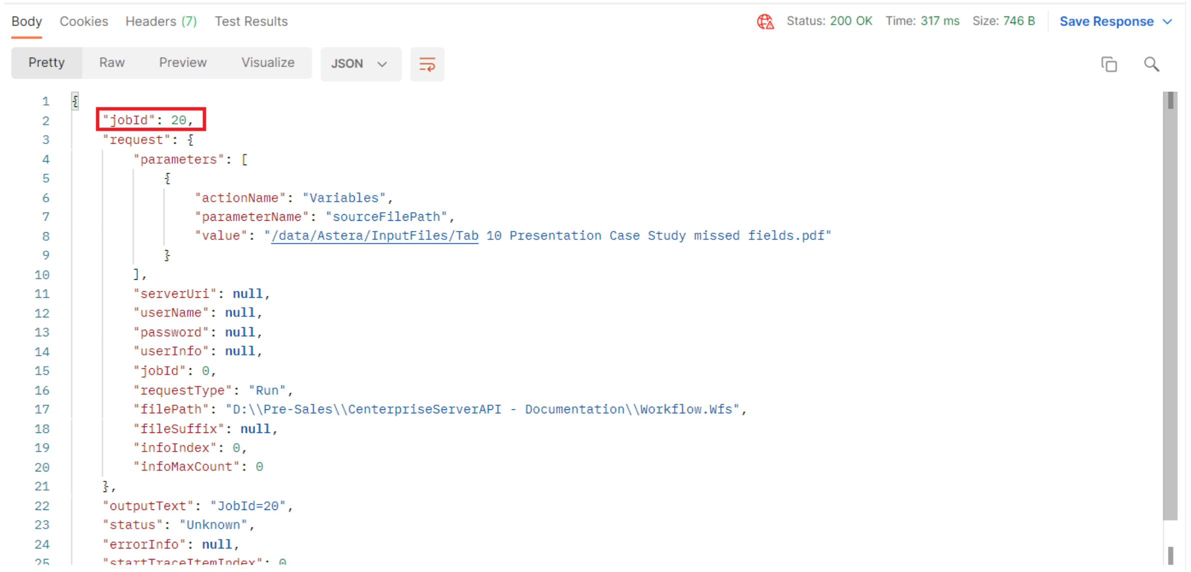
- *Parameters*: sourceFilePath

  The value of the input variable field inside the workflow

- *Value*: [file path of the file that you want to download]

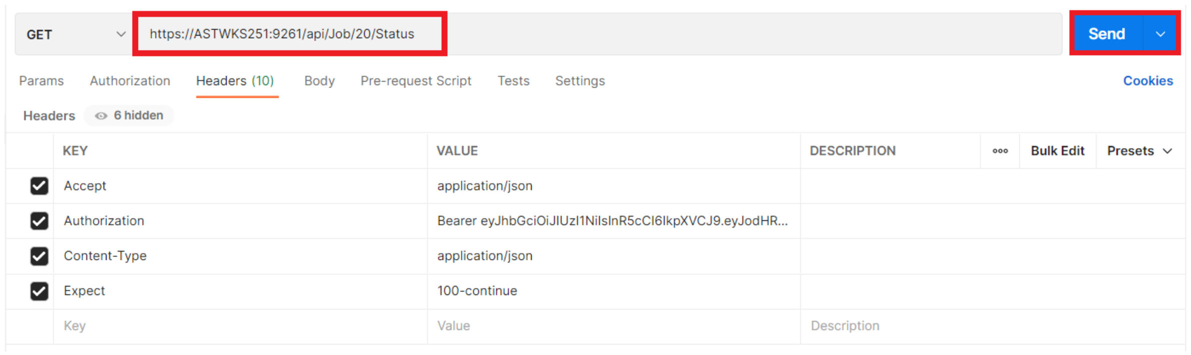
  The value of the input variable field inside the workflow

As soon as you send this API request, Centerprise will provide you with a *jobID* that you can use to get the job status.



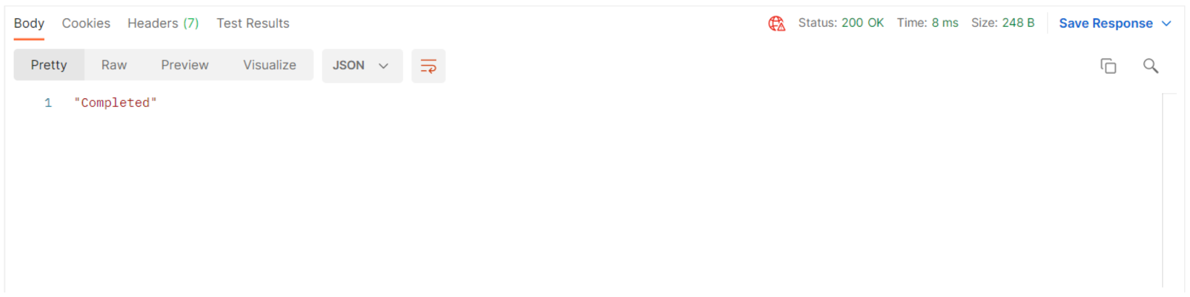
```
{
    "jobId": 20,
    "request": {
        "parameters": [
            {
                "actionName": "Variables",
                "parameterName": "sourceFilePath",
                "value": "/data/Astera/InputFiles/Tab 10 Presentation Case Study missed fields.pdf"
            }
        ],
        "serverUri": null,
        "userName": null,
        "password": null,
        "userInfo": null,
        "jobId": 0,
        "requestType": "Run",
        "filePath": "D:\\Pre-Sales\\CenterpriseServerAPI - Documentation\\Workflow.Wfs",
        "fileSuffix": null,
        "infoIndex": 0,
        "infoMaxCount": 0
    },
    "outputText": "JobId=20",
    "status": "Unknown",
    "errorInfo": null,
```

3. In the third step, we will make a GET call to fetch the job's status by providing the job ID.



	KEY	VALUE	DESCRIPTION
✓	Accept	application/json	
✓	Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJodHR...	
✓	Content-Type	application/json	
✓	Expect	100-continue	
	Key	Value	Description

This is what Centerprise's response would look like.



```
"Completed"
```

This concludes accessing Centerprise's server APIs through a third-party tool.

11.17 Centerprise's Server API Documentation

11.17.1 Authentication

Centerprise's Server APIs use *Bearer Token* authentication. To learn more about authenticating Centerprise's Server APIs, click here.

Resource: Account

Login

Method: POST

Endpoint: https://{servername}:{portno}/api/account/login

In this case: https://LOCALHOST:9261/api/account/login

Resource: /api/account/login

Request Body

Note: The format of our request body is JSON type.

Resource: Job

Status

Method: GET

Endpoint: https://LOCALHOST:9261/api/Job/{jobID}/Status

Resource: /api/Job/{jobID}/Status

Required Parameter

Description: This method fetches the status of a job for the given job ID. A few of the response statuses are given below:

1. Unknown
2. Invalid
3. NotStarted
4. Queued
5. Initializing
6. Running
7. Completed

11.18 NTLM Authentication

NTLM (NT LAN Manager) authentication is a Microsoft proprietary authentication protocol used to authenticate users in a Windows-based network.

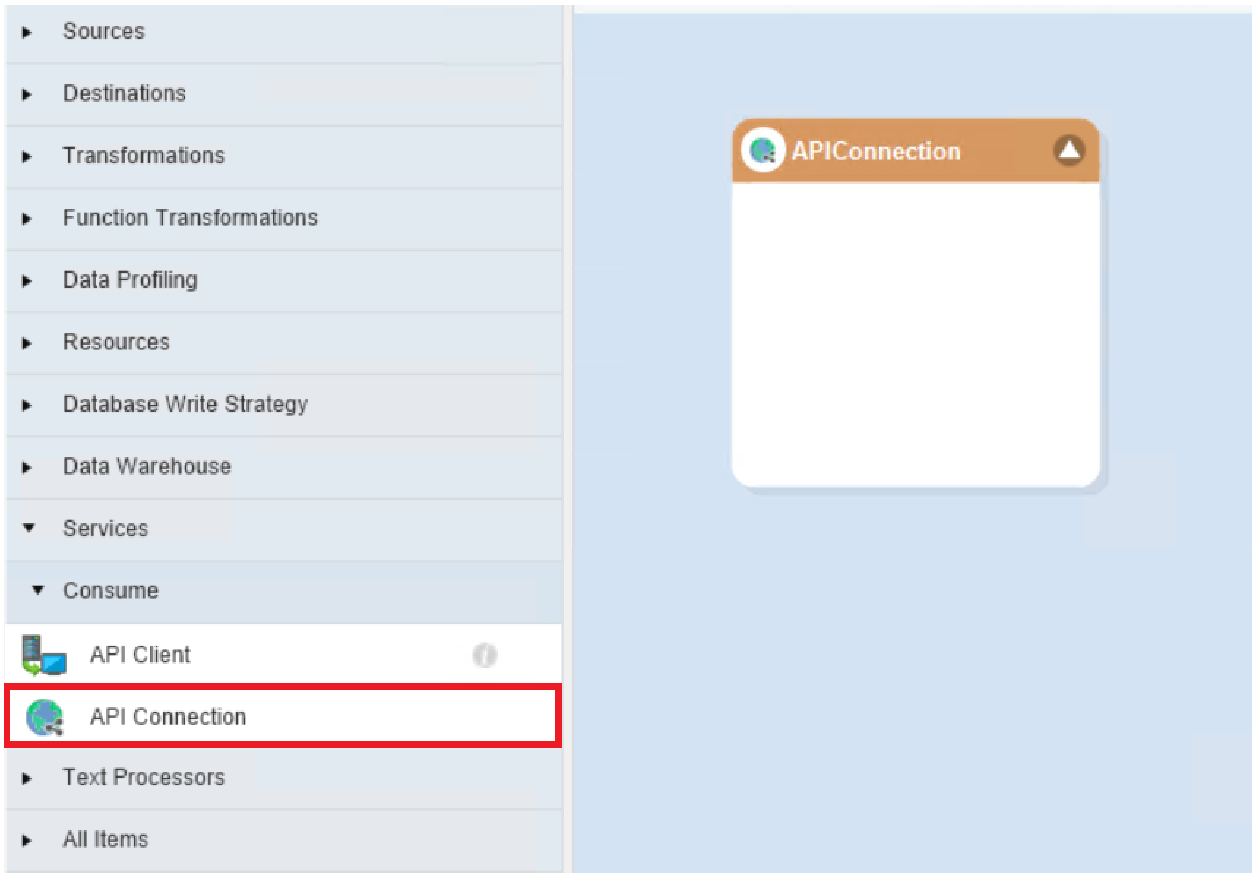
It provides secure authentication by using a challenge-response mechanism, where the server sends a challenge to the client, and the client sends a response that is encrypted using a hash of the user's password.

NTLM authentication is used in various Microsoft products, including Windows, Internet Explorer, and Microsoft Office.

11.18.1 NTLM in Astera API Management

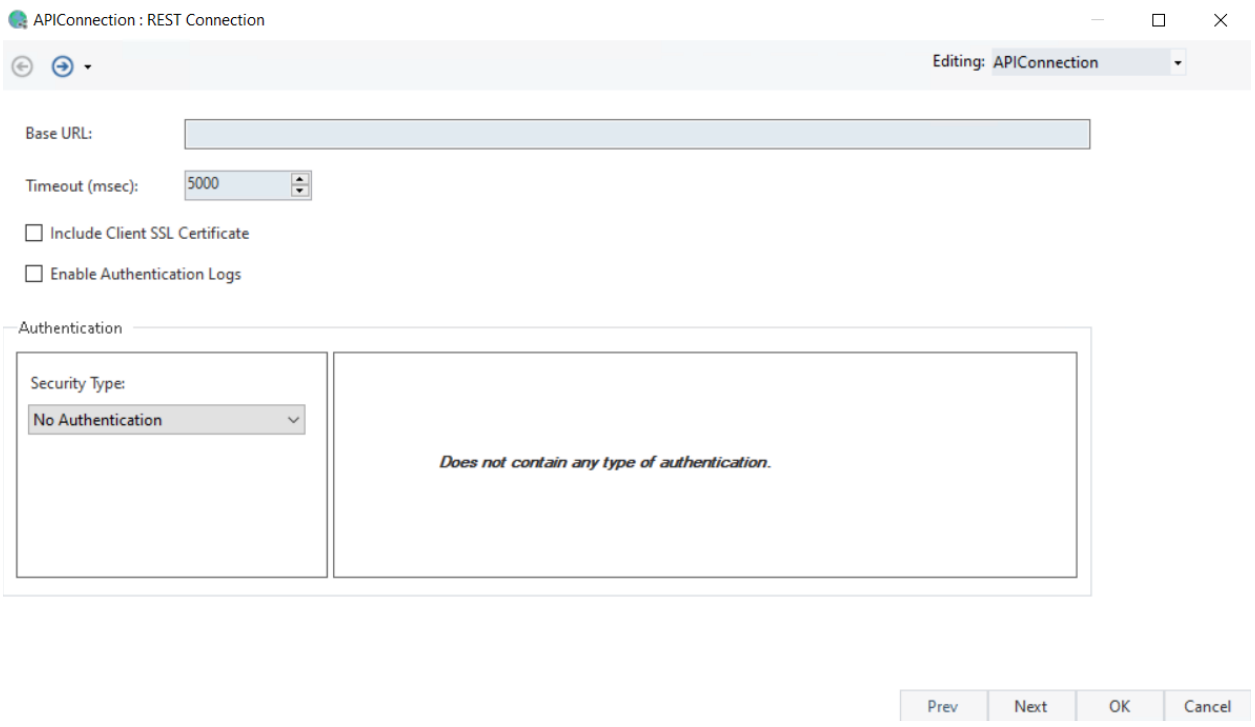
Astera also offers the ability to use NTLM authentication when establishing an API connection.

1. To start, drag and drop the *API Connection* object from the toolbox onto a dataflow.



2. Right-click on the object and select *Properties* from the context menu.

This will open a new window,



Base URL:

Timeout (msec): 5000

☐ Include Client SSL Certificate

☐ Enable Authentication Logs

Authentication

Security Type: No Authentication

Does not contain any type of authentication.

Base URL: Here, you can specify the base URL of the API which will prepend as a common path to all API endpoints sharing this connection. A Base URL usually consists of the scheme hostname and port of the API web address.

Timeout (msec): Specify the duration, in milliseconds, to wait for the API server to respond before giving a timeout error.

Include Client SSL Certificate: Selecting this option is going to include any Client SSL certificate that is needed for authentication.

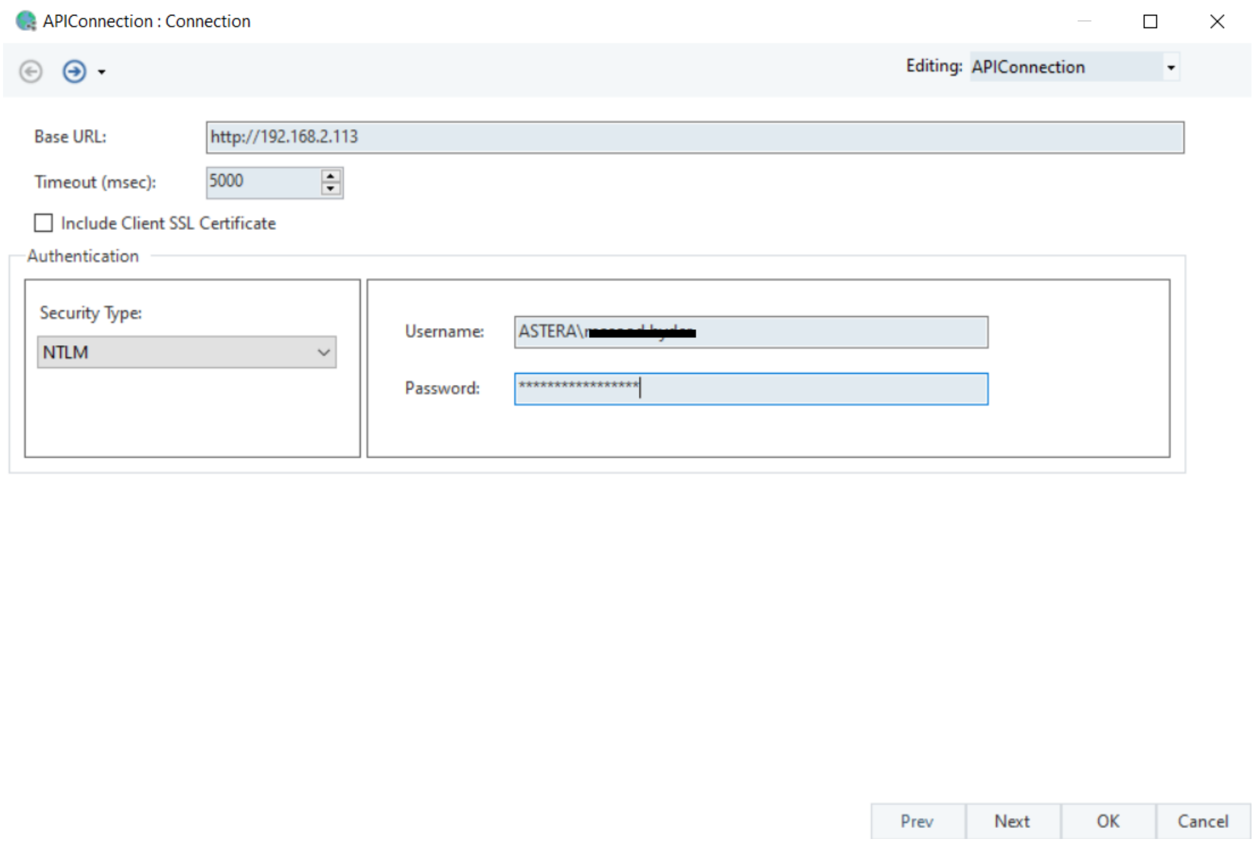
Enable Authentication Logs: Selecting this checkbox will allow the client to generate authentication logs when the API connection has been configured.

3. Fill in the Base URL and open the *Security Type* drop-down menu,

For our use case, we have deployed an API on IIS Manager on another machine, and we will send a request to access that API.

4. Select *NTLM* as the authentication type.

This will give us the following options,



Base URL: http://192.168.2.113

Timeout (msec): 5000

☐ Include Client SSL Certificate

Security Type: NTLM

Username: ASTERA\[redacted]

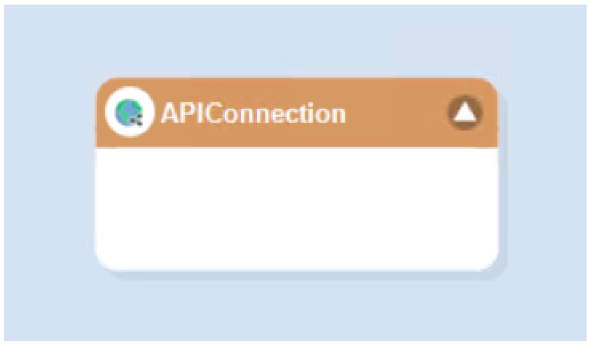
Password: *****************

Username: This field will input the same username that is used to login to Windows.

Password: The password associated with Windows login credentials.

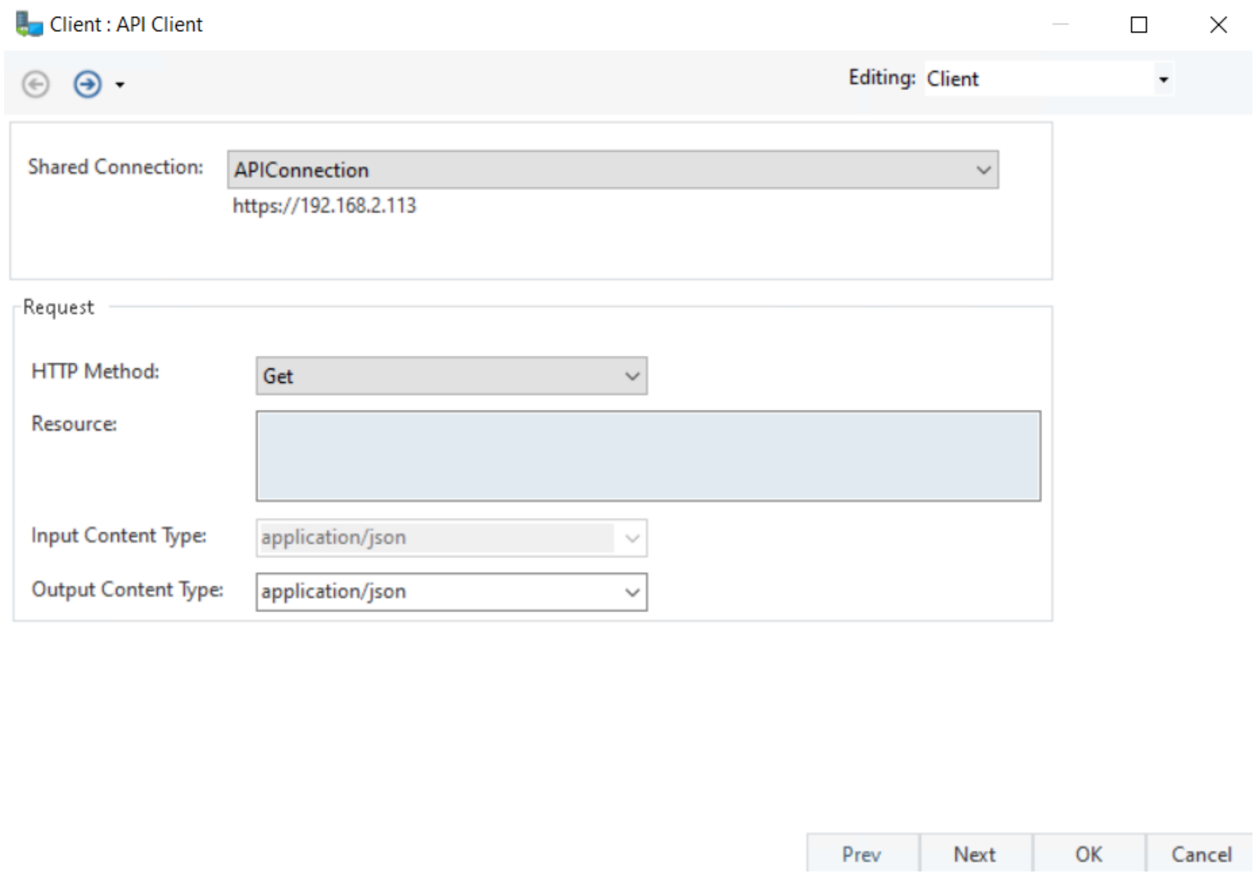
Note: NTLM authentication establishes API connections using a challenge-response mechanism. When sending an API request, Centerprise sends a hashed version of the user's credentials (username and password) to the server, which sends back a random challenge. Centerprise then mixes this challenge with the user's password and sends back a hashed value for verification. Access is granted if the validation is successful.

5. Click *Ok* and the *API Connection* object will be configured with NTLM Authentication.



This *API Connection* can then be used in *API Client* objects to make API calls to the server and receive appropriate responses in return.

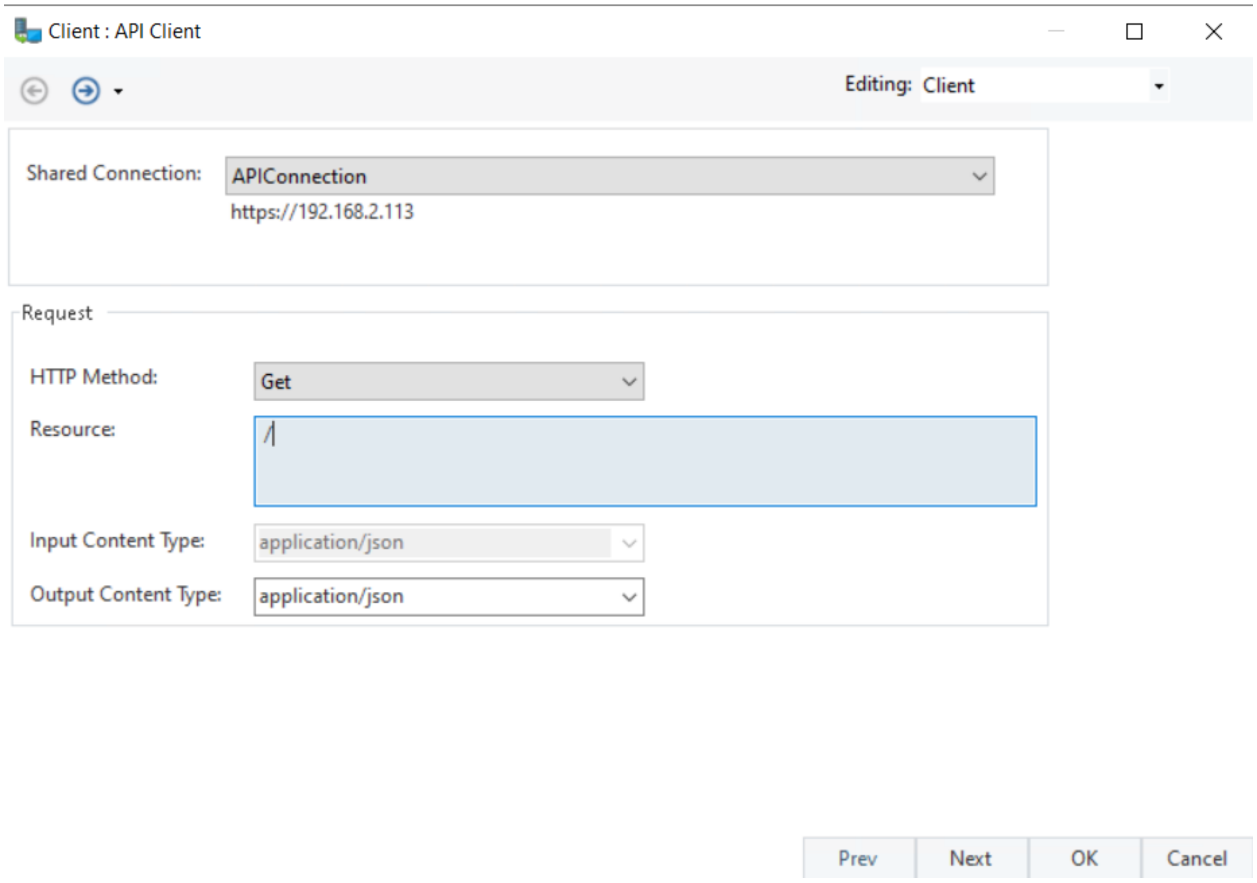
6. Drag and drop an *API Client* object onto the dataflow and select the shared connection that was defined.



Shared Connection: APIConnection
https://192.168.2.113

HTTP Method: Get
Resource:
Input Content Type: application/json
Output Content Type: application/json

Note: The *Resource* will be '/' since our entire address has been defined in the *Base URL*.


Client : API Client

Editing: Client

Shared Connection: APIConnection

https://192.168.2.113

Request

HTTP Method: Get

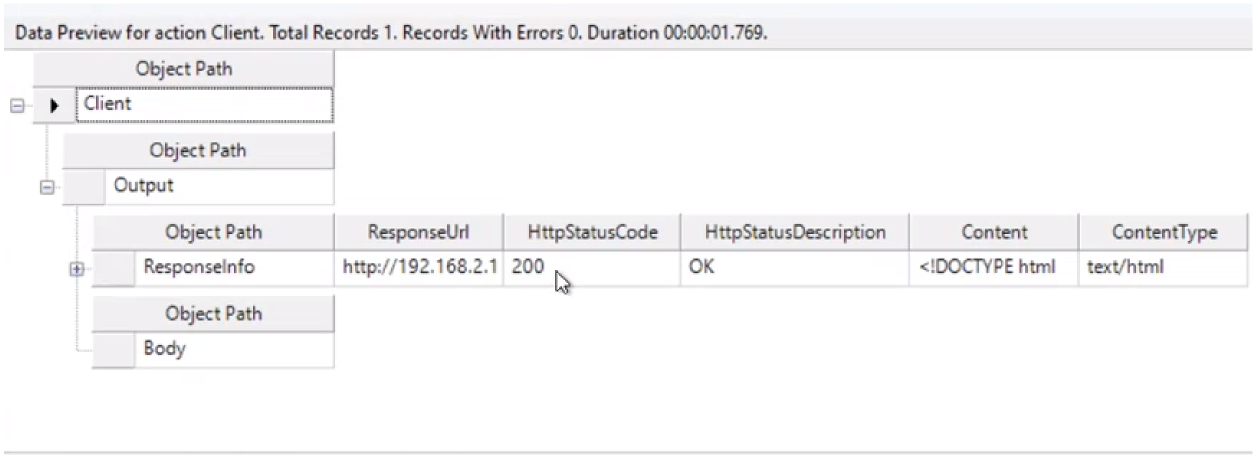
Resource: /

Input Content Type: application/json

Output Content Type: application/json

7. Click *Ok* and preview the output of the *API Client* object.

As we can see in our data preview window, the request has been sent successfully and the response has returned as ‘200 OK’.


Data Preview for action Client. Total Records 1. Records With Errors 0. Duration 00:00:01.769.

Object Path
Client

Object Path
Output

Object Path	ResponseUrl	HttpStatusCode	HttpStatusDescription	Content	ContentType
ResponseInfo	http://192.168.2.1	200	OK	<!DOCTYPE html	text/html

Object Path
Body

This concludes the working and configuring of NTLM Authentication in Astera API Management.

11.19 AWS Signature Authentication

AWS Signature authentication is the process of verifying the authenticity of requests made to Amazon Web Services (AWS) using the AWS Signature method.

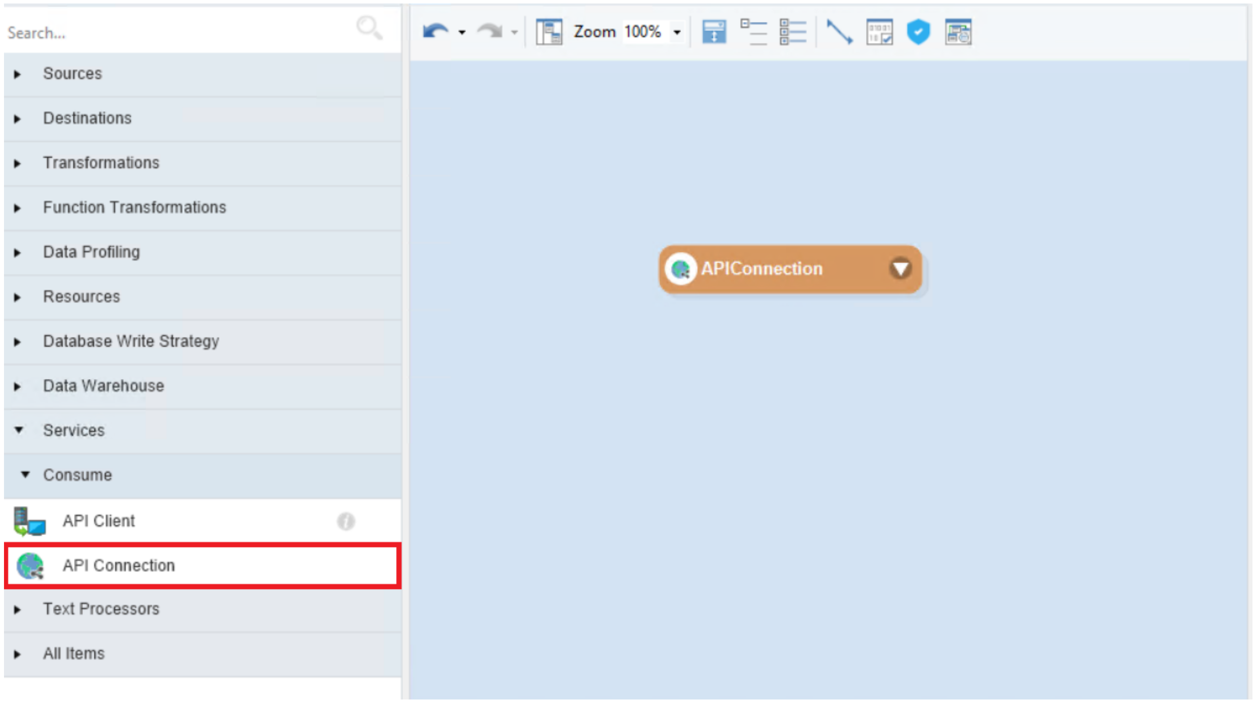
This authentication process involves calculating a digital signature for each request using the requester's access key and secret access key, along with details about the request being made. AWS verifies the signature against the user's access credentials and grants access to the requested resources if the signature is valid.

The AWS Signature authentication method ensures that requests are securely transmitted and that only authorized users can access AWS resources.

11.19.1 AWS Signature Authentication in Astera API Management

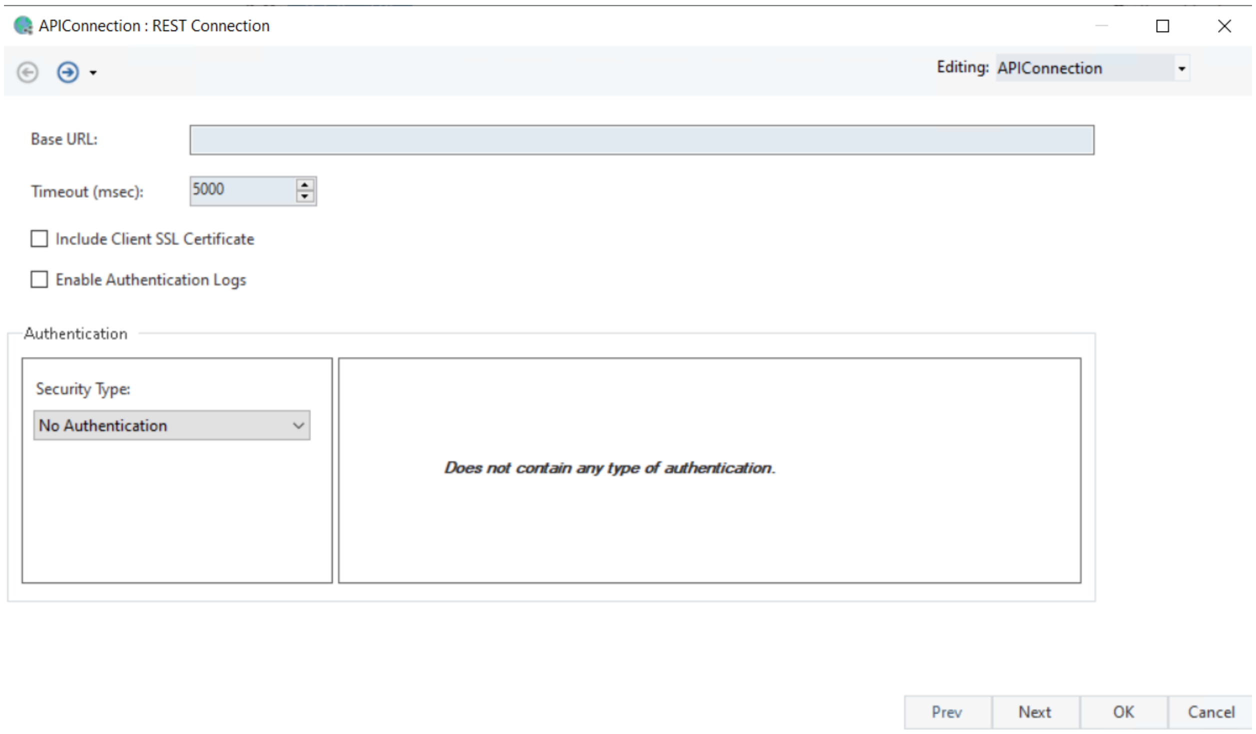
Astera API Management lets the user configure an *API Connection* with AWS Signature as an authentication type.

1. Drag and drop an *API Connection* object from the toolbox onto a dataflow.



2. Right-Click on the object and select *Properties* from the context menu.

This will open a new window,


APIConnection : REST Connection
Editing: APIConnection
Base URL:
Timeout (msec): 5000
Include Client SSL Certificate
Enable Authentication Logs
Authentication
Security Type:
No Authentication
Does not contain any type of authentication.
Prev
Next
OK
Cancel

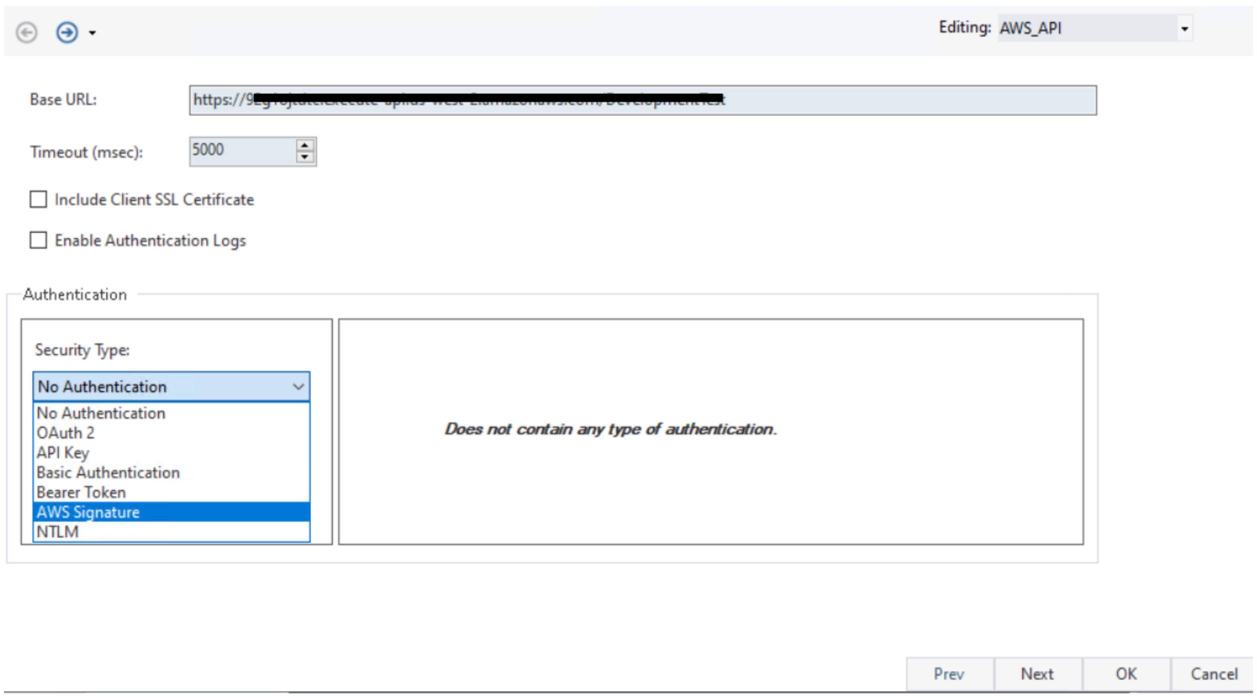
Base URL: Here, you can specify the base URL of the API which will prepend as a common path to all API endpoints sharing this connection. A Base URL usually consists of the scheme hostname and port of the API web address.

Timeout (msec): Specify the duration, in milliseconds, to wait for the API server to respond before giving a timeout error.

Include Client SSL Certificate: Selecting this option is going to include any Client SSL certificate that is needed for authentication.

Enable Authentication Logs: Selecting this checkbox will allow the client to generate authentication logs when the API connection has been configured.

3. Define the *Base URL* and select *AWS Signature* from the security type.



Editing: AWS_API

Base URL: https://9[redacted]

Timeout (msec): 5000

☐ Include Client SSL Certificate

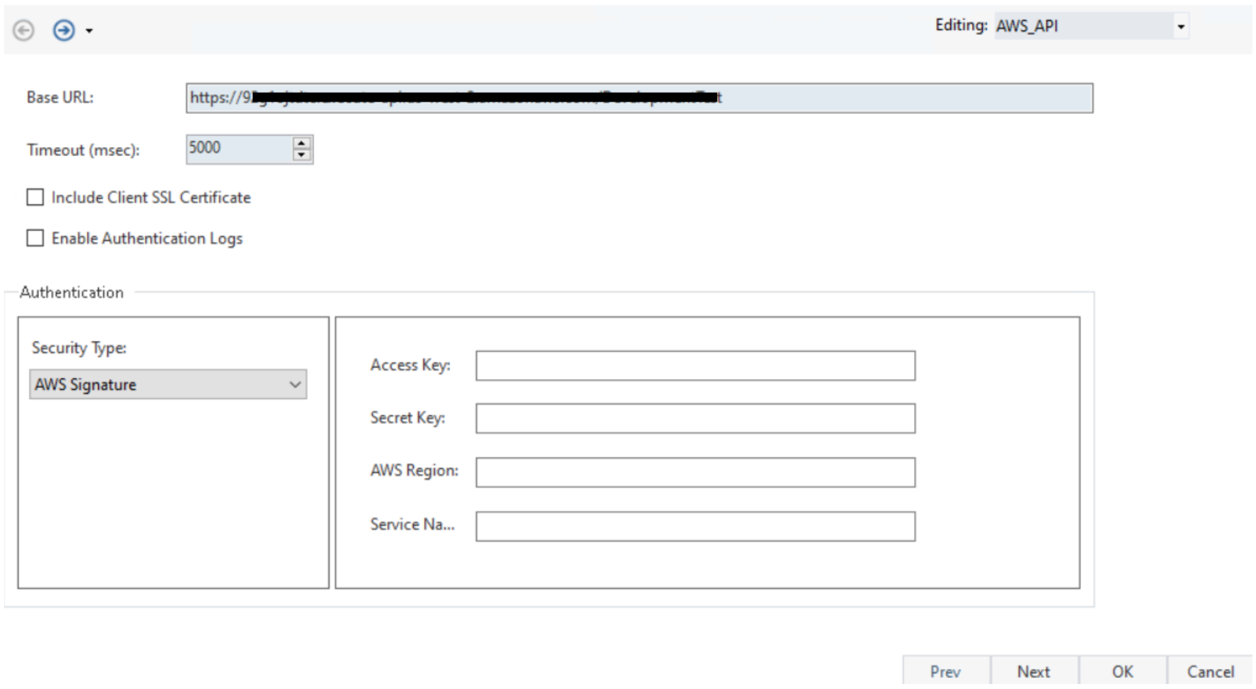
☐ Enable Authentication Logs

Authentication

Security Type: No Authentication

Does not contain any type of authentication.

4. Selecting it will make the following options available.



Editing: AWS_API

Base URL: https://9[redacted]

Timeout (msec): 5000

☐ Include Client SSL Certificate

☐ Enable Authentication Logs

Authentication

Security Type: AWS Signature

Access Key:

Secret Key:

AWS Region:

Service Na...

Access Key: The unique access key provided to the AWS user for authentication.

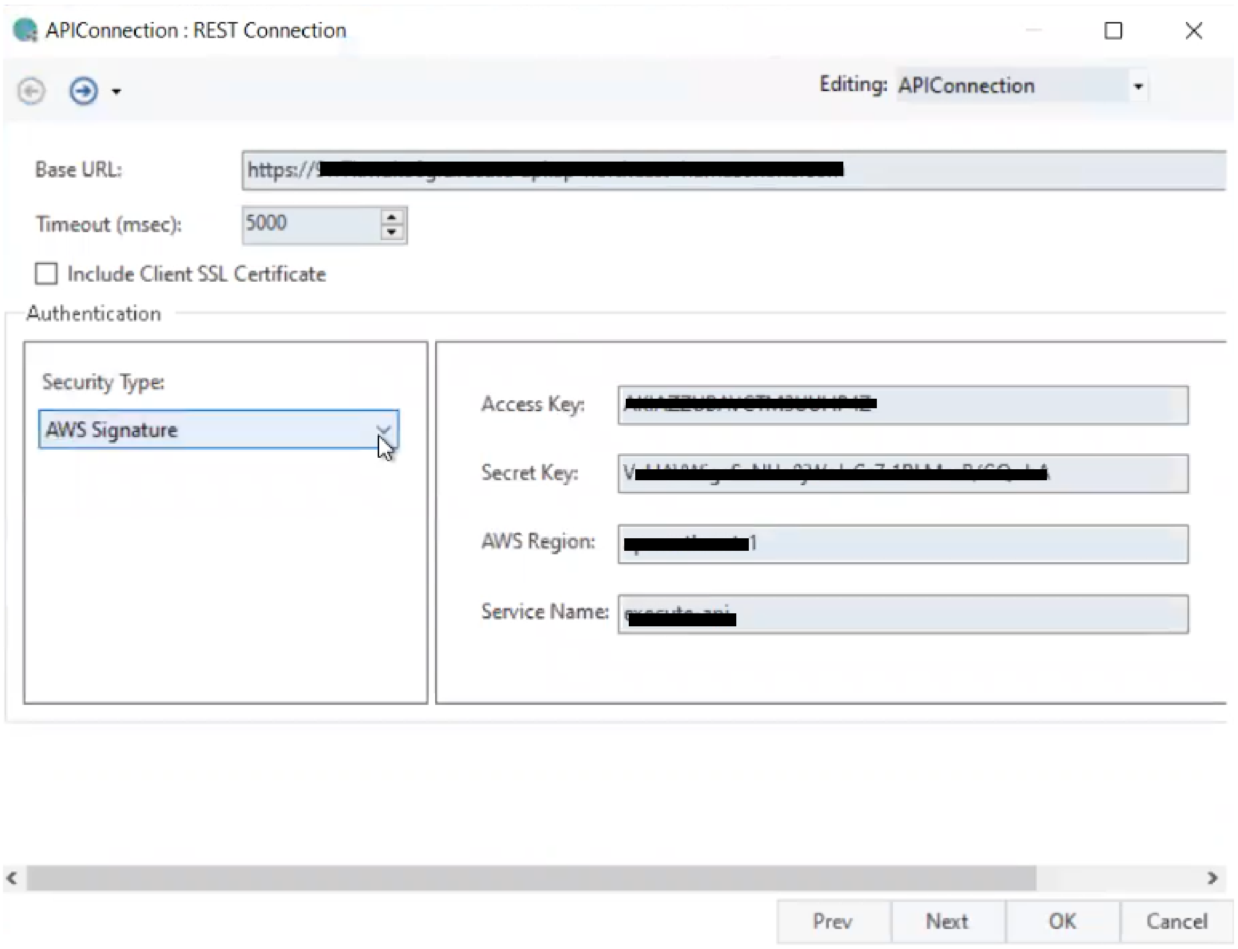
Secret Key: The corresponding unique key provided to the AWS user for authentication

AWS Region: The region from where the *API connection* is being made, set by the admin.

Service Name: The name of the AWS service being used in the *API Connection.*

Note: While the *Access Key* and *Secret Key* are unique to each user, the *AWS Region* and *Service Name* are common among a group of users.

5. Once the fields have been filled, click *OK* and the *API Connection* will be configured.



APIConnection : REST Connection

Editing: APIConnection

Base URL: https://[redacted]

Timeout (msec): 5000

☐ Include Client SSL Certificate

Authentication

Security Type: AWS Signature

Access Key: [redacted]

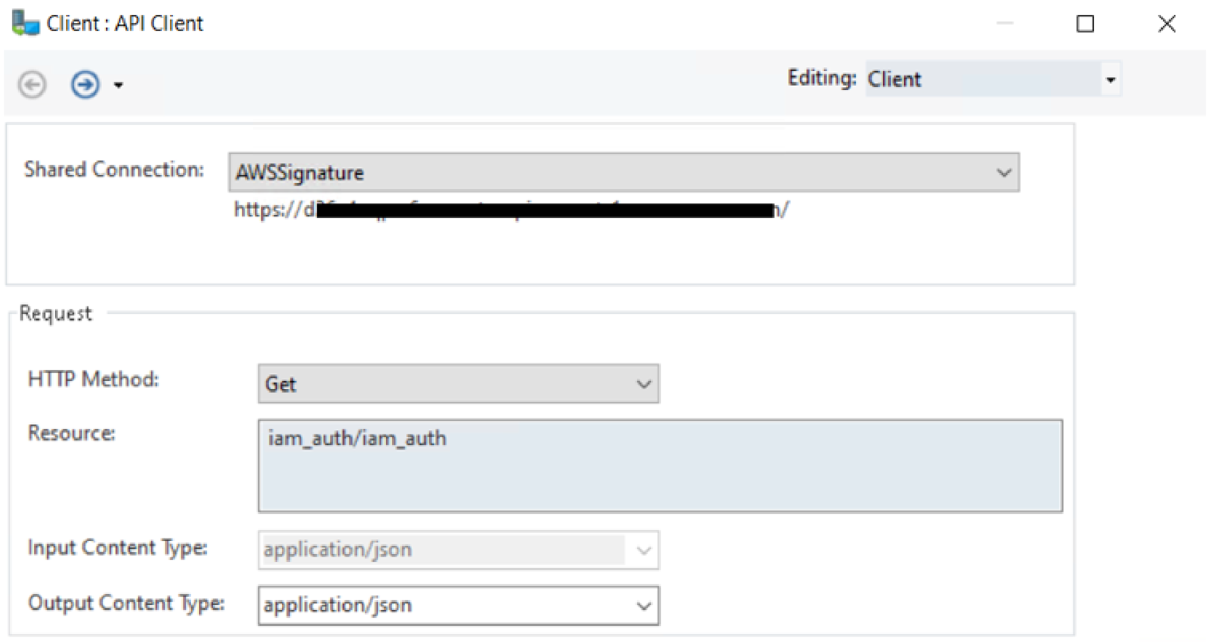
Secret Key: V[redacted]

AWS Region: [redacted]1

Service Name: [redacted]

This *API Connection* can then be used in an *API Client* object to make API Calls to the resource.

6. Drag and drop an *API Client* object and configure it with the *API Connection.*



Client : API Client

Editing: Client

Shared Connection: AWSSignature

https://d[redacted]/

Request

HTTP Method: Get

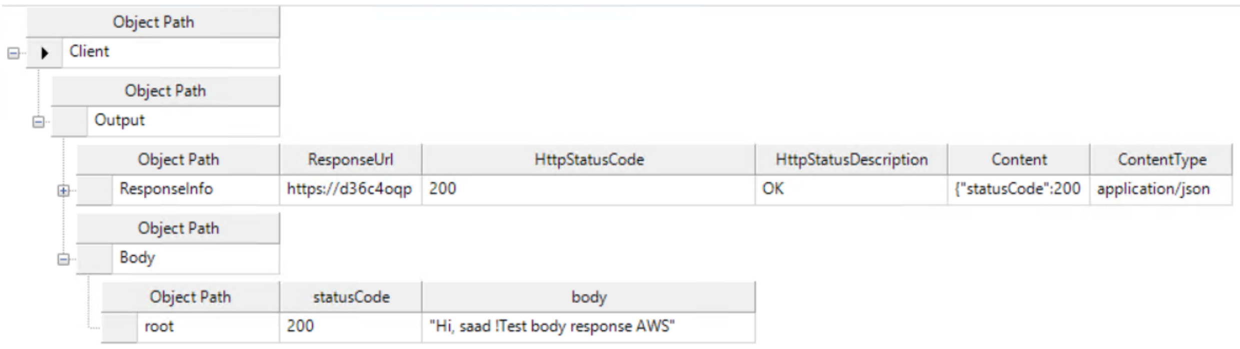
Resource: iam_auth/iam_auth

Input Content Type: application/json

Output Content Type: application/json

7. Preview the output of the *API Client* object.

As you can see, the response has returned a '200 OK' status.



Object Path
Client

Object Path
Output

Object Path	ResponseUrl	HttpStatusCode	HttpStatusDescription	Content	ContentType
ResponseInfo	https://d36c4oqp	200	OK	{"statusCode":200	application/json

Object Path
Body

Object Path	statusCode	body
root	200	"Hi, saad !Test body response AWS"

This concludes the configuration and testing of the AWS Signature Authentication in Astera API Management.